Multi-Cloud Engineering Program

Recorrido de Microsoft Azure

28 clases · 120 horas

La plataforma esencial de Azure y su arquitectura empresarial y operación en producción, con las clases de otras partes que tratan el proveedor.

Extracto del manual integral, generado desde las mismas fuentes versionadas.

Alcance de este cuaderno

Componente	Cobertura
Partes propias	03, 18
Clases de esas partes	24
Clases de otras partes	4
Clases totales	28
Horas estimadas	120

Índice

Parte 03 — Azure: plataforma esencial

037 — Tenant, management groups, suscripciones y resource groups

Parte: 03 — Azure: plataforma esencial Nivel: intermedio · Horas estimadas: 4 Laboratorio: governance · Estado: EXECUTABLECORE

Propósito

Construir la jerarquía de Azure y descubrir en qué se parece y en qué no a la de AWS. El ejercicio central de esta parte no es aprender nombres nuevos: es distinguir qué decisiones de la parte 02 eran de arquitectura —y reaparecen— y cuáles eran de proveedor, que es la pregunta con la que cerró la clase 036.

Resultados de aprendizaje

Al finalizar podrás:

1. Traducir la jerarquía de Azure a los conceptos neutrales de la clase 017 y señalar dónde no hay equivalencia.
2. Explicar por qué el grupo de recursos no es una frontera de aislamiento y qué sí lo es.
3. Escribir una directiva que restrinja regiones sin bloquear los recursos globales.
4. Distinguir cuota de suscripción de límite de servicio y anticipar cuál se agota antes.
5. Diseñar grupos de administración por régimen de gobierno, no por organigrama.

Conceptos centrales

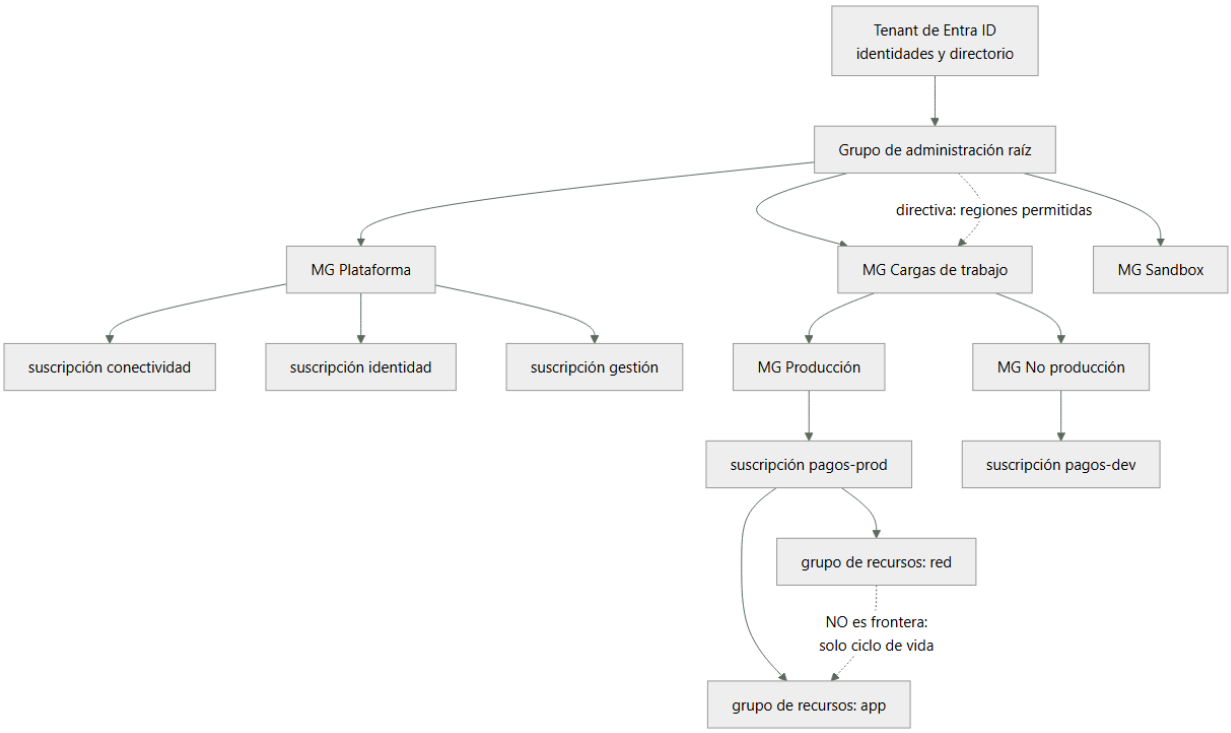
Concepto	Comprensión verificable
tenant	Instancia de Microsoft Entra ID que contiene identidades y suscripciones. Es la raíz de la jerarquía y el equivalente conceptual de la organización, con una diferencia importante: la identidad vive ahí, no en la suscripción.
suscripción	Frontera de aislamiento, facturación y cuota. Es el equivalente funcional de la cuenta de AWS y del proyecto de Google Cloud.
grupo de recursos	Contenedor de recursos con ciclo de vida común dentro de una suscripción. No aísla nada: es unidad de despliegue y de borrado, y confundirlo con frontera de seguridad es el error más frecuente al llegar de AWS.
directiva	Regla que evalúa recursos y puede auditar, denegar, modificar o desplegar. A diferencia de una SCP, puede corregir además de prohibir, y se aplica también a lo que ya existe.
grupo de administración	Agrupador de suscripciones sobre el que se heredan directivas y asignaciones de rol. Admite hasta seis niveles de profundidad bajo la raíz.

Modelo mental

En Azure, identidad y jerarquía organizacional preceden al recurso: tenant, management group, suscripción y resource group determinan el alcance de control.

Aplicado a esta clase, separa siempre cuatro planos: intención (qué necesita el usuario), configuración (qué declaramos), estado observado (qué existe de verdad) y evidencia (cómo sabemos que cumple). Confundirlos produce diseños que se ven correctos en un diagrama pero fallan al operar.

■ Flujo de razonamiento



Desarrollo

1. El mapa de equivalencias, y dónde se rompe

Concepto neutral (clase 017)	AWS	Azure	Google Cloud
Raíz	Organización	Tenant	Organización
Agrupador	Unidad organizativa	Grupo de administración	Carpeta
Frontera de aislamiento	Cuenta	Suscripción	Proyecto
Agrupador interno	—	Grupo de recursos	—
Política heredada	SCP	Azure Policy	Restricción de organización

Dos diferencias cambian el diseño y no son cosméticas:

La identidad vive en el tenant, no en la suscripción. En AWS cada cuenta tiene su propio IAM y el acceso entre cuentas exige asumir roles. En Azure, un usuario del tenant puede tener asignaciones de rol en varias suscripciones sin ningún salto: la frontera de identidad y la de aislamiento no coinciden. Eso simplifica la operación y hace que una identidad comprometida alcance potencialmente más superficie, así que la separación por asignaciones de rol importa más que en AWS.

Existe un nivel intermedio que AWS no tiene. El grupo de recursos agrupa recursos con ciclo de vida común. Su propiedad más útil es que borrarlo borra todo lo que contiene, lo que lo hace excelente para entornos efímeros y peligroso si se confunde con una frontera de seguridad.

Y una asimetría de las directivas frente a las SCP: Azure Policy puede modificar y desplegar, no solo denegar. Una directiva con efecto DeployIfNotExists puede añadir el diagnóstico que falta a un recurso recién creado. Es más potente y exige más cuidado: una directiva de corrección mal escrita cambia recursos en producción sin que nadie lo pida.

2. El grupo de recursos no aísla

Es el error de traducción más común al llegar de AWS. Un grupo de recursos parece una cuenta pequeña y no lo es:

```
grupo de recursos
  ✓ ciclo de vida común: se borra entero
  ✓ ámbito de asignación de roles
  ✓ ámbito de aplicación de directivas
```

```
✗ NO tiene cuotas propias
✗ NO es frontera de facturación
✗ NO impide que un recurso hable con otro de otro grupo
```

Las dos primeras exclusiones son las que producen incidentes. Las cuotas son de la suscripción, así que dos grupos de recursos en la misma suscripción compiten por los mismos núcleos disponibles — exactamente el problema de la clase 017 con otro nombre:

```
$ az vm list-usage --location brazilsouth --query "[?contains(name.value,'cores')].{n:name.value,u:
currentValue,l:limit}" -o table
n                       u     l
----------------------  ----  ----
cores                   186   200
standardDSv3Family      142   150
```

186 de 200 núcleos consumidos en la suscripción, sin importar en cuántos grupos de recursos estén repartidos. Un experimento en el grupo «desarrollo» deja sin capacidad al grupo «producción» si comparten suscripción.

La tercera exclusión también sorprende: no hay aislamiento de red entre grupos de recursos. Dos máquinas en la misma red virtual se hablan aunque estén en grupos distintos; el aislamiento de red lo dan la red virtual y los grupos de seguridad, no la organización de recursos.

La regla de traducción: donde en AWS separarías por cuenta, en Azure separa por suscripción. El grupo de recursos es la unidad de despliegue, no de aislamiento.

3. Directivas: cuatro efectos y cuándo usar cada uno

Azure Policy va más allá de prohibir:

Efecto	Qué hace	Cuándo
Audit	Marca no conforme, no impide	Siempre primero: medir antes de bloquear
Deny	Impide la creación o el cambio	Controles duros, tras medir
Modify	Añade o cambia propiedades	Etiquetas obligatorias
DeployIfNotExists	Despliega un recurso asociado	Diagnóstico, agentes, copias

La disciplina es la misma que con el WAF de la clase 035: empezar en Audit. Una directiva Deny aplicada sin medir bloquea despliegues legítimos, y el equipo la desactiva en el primer incidente en vez de corregirla.

```
$ az policy assignment create --name regiones-permitidas \
    --scope /providers/Microsoft.Management/managementGroups/mg-cargas \
    --policy "e56962a6-4747-49cd-b67b-bf8b01975c4c" \
    --params '{"listOfAllowedLocations":{"value":["brazilsouth","eastus"]}}' \
    --enforcement-mode DoNotEnforce      # equivale a Audit
```

Y la restricción de regiones tiene aquí el mismo problema que en la clase 025, con distinta solución. Varios tipos de recurso son globales y su ubicación declarada es global:

```
Microsoft.Network/trafficManagerProfiles
Microsoft.Network/dnsZones
Microsoft.Cdn/profiles
Microsoft.ManagedIdentity/userAssignedIdentities  (en algunos casos)
```

La directiva integrada de ubicaciones permitidas ya excluye los recursos globales, pero una escrita a mano no. Comprobarlo antes de aplicarla evita bloquear la creación de zonas DNS sin entender por qué.

Y un detalle operativo: las directivas evalúan lo existente, no solo lo nuevo. Tras asignar una, aparece un informe de conformidad de todo lo que ya estaba. Es una ventaja sobre las SCP —que solo actúan sobre peticiones nuevas— y también una fuente de ruido inicial que hay que triar.

4. Cuotas: dos límites que se confunden

Azure tiene dos clases de límite y se agotan por motivos distintos:

```
cuota de suscripción    núcleos por familia y región, IP públicas, redes virtuales
                        → ampliable con solicitud, tarda horas o días

límite de servicio      reglas por grupo de seguridad, tamaño de plantilla,
                        recursos por grupo de recursos
```

```
                          → normalmente NO ampliable
```

La primera es la que aparece en el autoescalado y la segunda en el despliegue. Un caso frecuente del segundo tipo:

```
límite de reglas por grupo de seguridad de red: 1.000
un equipo genera una regla por cliente autorizado
→ el despliegue falla al llegar a 1.000 y NO se puede ampliar
→ la corrección exige rediseñar: grupos de seguridad de aplicación
  o listas de prefijos en vez de reglas individuales
```

La consulta de cuotas debe formar parte del plan de capacidad, no de la respuesta a incidentes:

```
$ az vm list-usage --location brazilsouth -o json \
  | jq -r '.[] | select((.currentValue / .limit) > 0.8)
           | "\(.name.value): \(.currentValue)/\(.limit)"'
standardDSv3Family: 142/150
```

El 80 % como umbral de alerta, igual que en la clase 017. Ampliar una cuota tarda horas o días, así que descubrirla durante un pico significa que ya no es una palanca disponible.

Y una diferencia con AWS que conviene saber: en Azure las cuotas de núcleos son por familia de máquina virtual además de por total. Tener 60 núcleos libres del total no sirve si la familia concreta que necesitas está al límite, y el mensaje de error no siempre lo deja claro.

5. Grupos de administración por gobierno, no por organigrama

El mismo criterio de la clase 017: agrupar por qué directiva se aplica, porque los organigramas se reorganizan y mover suscripciones entre grupos cambia las directivas heredadas.

La estructura de referencia de Microsoft para escala empresarial:

```
raíz
■■■ Plataforma           directivas estrictas, la opera el equipo central
■   ■■■ conectividad     redes virtuales de concentrador, DNS privado
■   ■■■ identidad        controladores de dominio, servicios de identidad
■   ■■■ gestión          registros, copias, automatización
■■■ Cargas de trabajo
■   ■■■ Producción       regiones, cifrado obligatorio, diagnóstico forzado
■   ■■■ No producción    regiones, límite de gasto
■■■ Sandbox              directivas laxas, presupuesto duro, caducidad
■■■ Desmantelado         suscripciones en retirada, solo lectura
```

Dos elementos que AWS no tiene explícitos y aquí conviene aprovechar:

El grupo «Desmantelado» recoge suscripciones que se van a cerrar, con directivas que impiden crear nada nuevo. Evita que una suscripción en retirada siga acumulando recursos durante meses.

La separación de plataforma en tres suscripciones —conectividad, identidad y gestión— responde a que cada una tiene un ciclo de vida y un equipo distintos. Es el equivalente a las cuentas de red, seguridad y auditoría de la clase 025.

Y una restricción práctica: la profundidad máxima es de seis niveles bajo la raíz. Suficiente, pero conviene no gastarlos en reflejar jerarquías organizativas que cambiarán.

Un recurso que no debe alojar cargas: el grupo de administración raíz. Igual que la cuenta de gestión de AWS, es el único punto sin nada por encima, y las asignaciones que se hacen ahí se heredan a todo el tenant.

Ejemplo trabajado

CloudShop despliega en Azure la misma aplicación de la parte 02. El equipo replica la estructura de AWS literalmente: un grupo de recursos por lo que allí era una cuenta. A las tres semanas, un despliegue de pruebas deja producción sin capacidad.

El incidente, idéntico al de la clase 017 con otro vocabulario:

```
$ az vm list-usage --location brazilsouth \
    --query "[?name.value=='standardDSv3Family'].{u:currentValue,l:limit}" -o tsv
148   150
$ az group list --query "[].name" -o tsv
rg-cloudshop-prod
rg-cloudshop-dev
rg-cloudshop-red
```

Tres grupos de recursos en una sola suscripción. El equipo tradujo «cuenta» por «grupo de recursos» y las cuotas no lo respetan:

```
cuota de la familia DSv3      150 núcleos
producción en régimen          88
experimento en rg-dev          60
total                         148 → producción no puede escalar
```

Se verifica que el grupo de recursos tampoco aísla la red:

```
$ az network vnet subnet list -g rg-cloudshop-red --vnet-name vnet-cloudshop \
    --query "[].name" -o tsv
snet-prod
snet-dev
# una VM de rg-dev en snet-dev alcanza una de rg-prod en snet-prod:
$ az network watcher test-ip-flow --vm vm-dev-01 --direction Outbound \
    --local 10.30.2.4:0 --remote 10.30.1.7:5432 --protocol TCP -o tsv --query access
Allow
```

Ni cuotas ni red separadas. La traducción era incorrecta.

Rediseño con la equivalencia correcta:

```
tenant cloudshop
■■■ mg-raiz
    ■■■ mg-plataforma
    ■   ■■■ sub-conectividad      red de concentrador, DNS privado
    ■■■ mg-cargas
    ■   ■■■ mg-produccion
    ■   ■   ■■■ sub-cloudshop-prod    cuota propia
    ■   ■■■ mg-no-produccion
    ■       ■■■ sub-cloudshop-dev     cuota propia
    ■■■ mg-sandbox
        ■■■ sub-lab-*                 presupuesto duro
```

Directiva de regiones, primero en modo auditoría:

```
$ az policy assignment create --name regiones --display-name "Regiones permitidas" \
    --scope /providers/Microsoft.Management/managementGroups/mg-cargas \
    --policy e56962a6-4747-49cd-b67b-bf8b01975c4c \
    --params '{"listOfAllowedLocations":{"value":["brazilsouth","eastus2"]}}' \
    --enforcement-mode DoNotEnforce
```

Tras 7 días, el informe de conformidad revela algo que nadie había notado:

```
$ az policy state summarize --management-group mg-cargas \
    --query "policyAssignments[0].results.{conformes:compliantResources,no:nonCompliantResources}"
{"conformes": 214, "no": 9}
$ az policy state list --management-group mg-cargas --filter "complianceState eq 'NonCompliant'" \
    --query "[].{r:resourceId,loc:resourceLocation}" -o tsv | awk '{print $2}' | sort | uniq -c
      7 westeurope
      2 global
```

Siete recursos en una región no aprobada —creados durante una prueba y olvidados— y dos globales, que son zonas DNS y no deben bloquearse. La directiva integrada ya los excluye; una escrita a mano los habría bloqueado.

Se migran los siete, se comprueba que la conformidad llega al 100 % y solo entonces se pasa a bloqueo:

```
$ az policy assignment update --name regiones --enforcement-mode Default
$ az group create --name rg-prueba --location westeurope
(RequestDisallowedByPolicy) Resource 'rg-prueba' was disallowed by policy   ✓
$ az network dns zone create -g rg-cloudshop-red -n cloudshop.cl
{... "location": "global" ...}                                              ✓ no bloqueado
```

Resultado, con la comparación que interesa para el resto del programa:

```
frontera de aislamiento                cuenta             suscripción
agrupador de gobierno                  unidad organizativa grupo de administración
cuota compartida entre entornos        no                 no (tras el rediseño)
política heredada                      SCP (solo denegar)  Directiva (auditar,
                                                           denegar, modificar,
                                                           desplegar)
identidad                              por cuenta          por tenant  ← DIFIERE
nivel intermedio                       —                   grupo de recursos ← DIFIERE
```

Las dos últimas filas son las decisiones de proveedor; el resto reaparece con otro nombre. Esa distinción es la que pedía la pregunta de cierre de la clase 036, y es la que hará que la parte 13 pueda hablar de portabilidad con criterio.

Laboratorio guiado

Ejecuta desde la raíz:

```
python classes/part-03-azure-core-platform/037-tenant-management-groups-suscripciones-y-resource-groups/lab.py
```

El laboratorio selecciona el motor de práctica governance y produce labresult.json. El escenario, sus comprobaciones y el artefacto esperado corresponden a esta clase; no requiere credenciales y deja explícito qué debe revalidarse en un sandbox real.

1. Lee exercise.steps y formula una predicción antes de ejecutar.
2. Ejecuta la práctica y verifica todos los elementos de checks.
3. Provoca el caso de negativetest y explica la señal observada.
4. Materializa jerarquia-azure en evidence/ usando la plantilla indicada.
5. Para proveedor real, sigue sandbox.requires, registra costo y ejecuta destroy.

Evidencia esperada

El artefacto principal es una jerarquía de política, ownership y evidencia. Además, la entrega debe incluir el comando ejecutado, la salida estructurada y una conclusión que no exceda lo observado.

Reto verificable

Construye jerarquia-azure para el caso CloudShop. Incluye una alternativa descartada, un supuesto que pueda falsarse, una prueba de fallo y una decisión de rollback.

■ Criterio de aceptación

- [ ] lab.py termina con código 0 y genera JSON válido.
- [ ] La entrega conecta al menos tres requisitos con mecanismos verificables.
- [ ] Existe una prueba positiva y una prueba negativa con evidencia.
- [ ] Seguridad, costo y operación aparecen como decisiones, no como anexos.
- [ ] Se declara una limitación y una condición que obligaría a revisar el diseño.
- [ ] Otra persona puede repetir el recorrido sin conocimiento tácito.

■■ Errores frecuentes

Síntoma	Causa probable	Corrección
Un entorno de pruebas deja sin capacidad a producción pese a estar en otro grupo de recursos	Las cuotas son de la suscripción; el grupo de recursos no aísla	Separa por suscripción donde en AWS separarías por cuenta; el grupo de recursos es ciclo de vida.
Una directiva de regiones bloquea la creación de zonas DNS	Los recursos globales declaran ubicación global y una directiva casera no los excluye	Usa la directiva integrada de ubicaciones permitidas, que ya los contempla.
Se aplica una directiva Deny y se paran despliegues legítimos	No se midió la conformidad del estado existente antes de bloquear	Asigna en modo auditoría, revisa el informe una o dos semanas y solo entonces refuerza.
Hay núcleos libres en la suscripción y la creación falla igual	La cuota es por familia de máquina además de por total	Consulta el uso por familia; el total disponible no garantiza capacidad de la familia que necesitas.
Un despliegue falla al superar el número de reglas de un grupo de seguridad	Es un límite de servicio, no una cuota: no se amplía	Rediseña con grupos de seguridad de aplicación o listas de prefijos en vez de reglas individuales.

■ Seguridad, ética y costo

Trabaja con cuentas propias o sandboxes autorizados. No publiques secretos, identificadores reales ni datos personales. Antes de crear recursos pagos define presupuesto, etiquetas y comando de destrucción; después verifica que no queden recursos huérfanos. Los ejemplos locales enseñan contratos, pero no certifican cumplimiento ni disponibilidad de producción.

■ Preguntas de comprobación

1. ¿Cuál es la frontera de aislamiento en Azure y por qué el grupo de recursos no lo es?
2. Nombra las dos diferencias estructurales entre la jerarquía de AWS y la de Azure que sí cambian el diseño.
3. ¿Qué puede hacer una directiva de Azure que una SCP no puede, y qué cuidado adicional exige?
4. Tienes 60 núcleos libres en la suscripción y la creación de una máquina falla. ¿Qué compruebas?
5. ¿Por qué una directiva se asigna primero en modo auditoría, y qué información produce esa fase?

Referencias

- Microsoft (2024). Organize your Azure resources with management groups — jerarquía, herencia y profundidad máxima. <https://learn.microsoft.com/en-us/azure/governance/management-groups/overview>
- Microsoft (2024). Azure Policy: understand policy effects — audit, deny, modify y deployIfNotExists. <https://learn.microsoft.com/en-us/azure/governance/policy/concepts/effects>
- Microsoft (2024). Azure subscription and service limits, quotas, and constraints — cuotas ampliables y límites duros. <https://learn.microsoft.com/en-us/azure/azure-resource-manager/management/azure-subscription-service-limits>
- Microsoft (2024). Cloud Adoption Framework: enterprise-scale landing zones — estructura de grupos de administración de referencia. <https://learn.microsoft.com/en-us/azure/cloud-adoption-framework/ready/landing-zone/>
- Microsoft (2024). Azure Resource Manager: resource groups — ámbito, ciclo de vida y qué no aísla. <https://learn.microsoft.com/en-us/azure/azure-resource-manager/management/manage-resource-groups-portal>
- Documentación oficial vigente del servicio implementado; registra URL y fecha de consulta.

Descargar: Parte 03 en PDF · Recorrido de Azure en PDF · Manual integral

Anterior	Índice	Siguiente
← 036 · Proyecto: aplicación de tres capas en AWS	Parte 03 · Programa	038 · Microsoft Entra ID, RBAC, managed identities y PIM →

Evaluación — 037 Tenant, management groups, suscripciones y resource groups

Preguntas

1. Explica el problema que resuelve esta capacidad sin mencionar un proveedor.
2. Identifica una frontera de responsabilidad y su propietario.
3. Compara dos alternativas mediante confiabilidad, seguridad, costo y operación.
4. ¿Qué demuestra el laboratorio y qué no puede demostrar?
5. Propón un caso límite o una prueba de fallo.

Reto verificable

Entrega jerarquia-azure con diagrama o configuración, evidencia de ejecución, alternativa descartada, riesgo residual y criterio de rollback.

Criterio de aceptación

- [ ] Resultado reproducible por una segunda persona.
- [ ] Evidencia vinculada a cada afirmación técnica.
- [ ] Prueba negativa o de fallo incluida.
- [ ] Costos y permisos expresados con unidades y alcance.
- [ ] Limitaciones declaradas sin presentar la simulación como producción.

Escala

Nivel	Evidencia
A — competente	Cumple todo y defiende trade-offs con datos.
B — en desarrollo	Cumple lo esencial; falta profundidad en una dimensión.
C — inicial	Artefacto parcial o conclusión sin evidencia suficiente.
0	Sin entrega reproducible.

038 — Microsoft Entra ID, RBAC, managed identities y PIM

Parte: 03 — Azure: plataforma esencial Nivel: intermedio · Horas estimadas: 4 Laboratorio: iam · Estado: EXECUTABLECORE

Propósito

Dominar el modelo de identidad de Azure, que difiere del de AWS en algo estructural: el directorio es del tenant y los permisos sobre recursos son otra cosa. Confundir un rol de Entra ID con uno de Azure RBAC es la causa de la mitad de los problemas de acceso al llegar de AWS, y de una parte de las escaladas de privilegio.

Resultados de aprendizaje

Al finalizar podrás:

1. Distinguir roles de Entra ID de roles de Azure RBAC y saber cuál gobierna cada acción.
2. Explicar por qué una denegación de Azure RBAC no es equivalente a un Deny de IAM y qué la sustituye.
3. Sustituir secretos de aplicación por identidades administradas y por federación desde un pipeline.
4. Aplicar acceso con privilegios mínimos en el tiempo mediante activación con caducidad y aprobación.
5. Diagnosticar un acceso denegado sabiendo en qué ámbito y en qué sistema de roles buscar.

Conceptos centrales

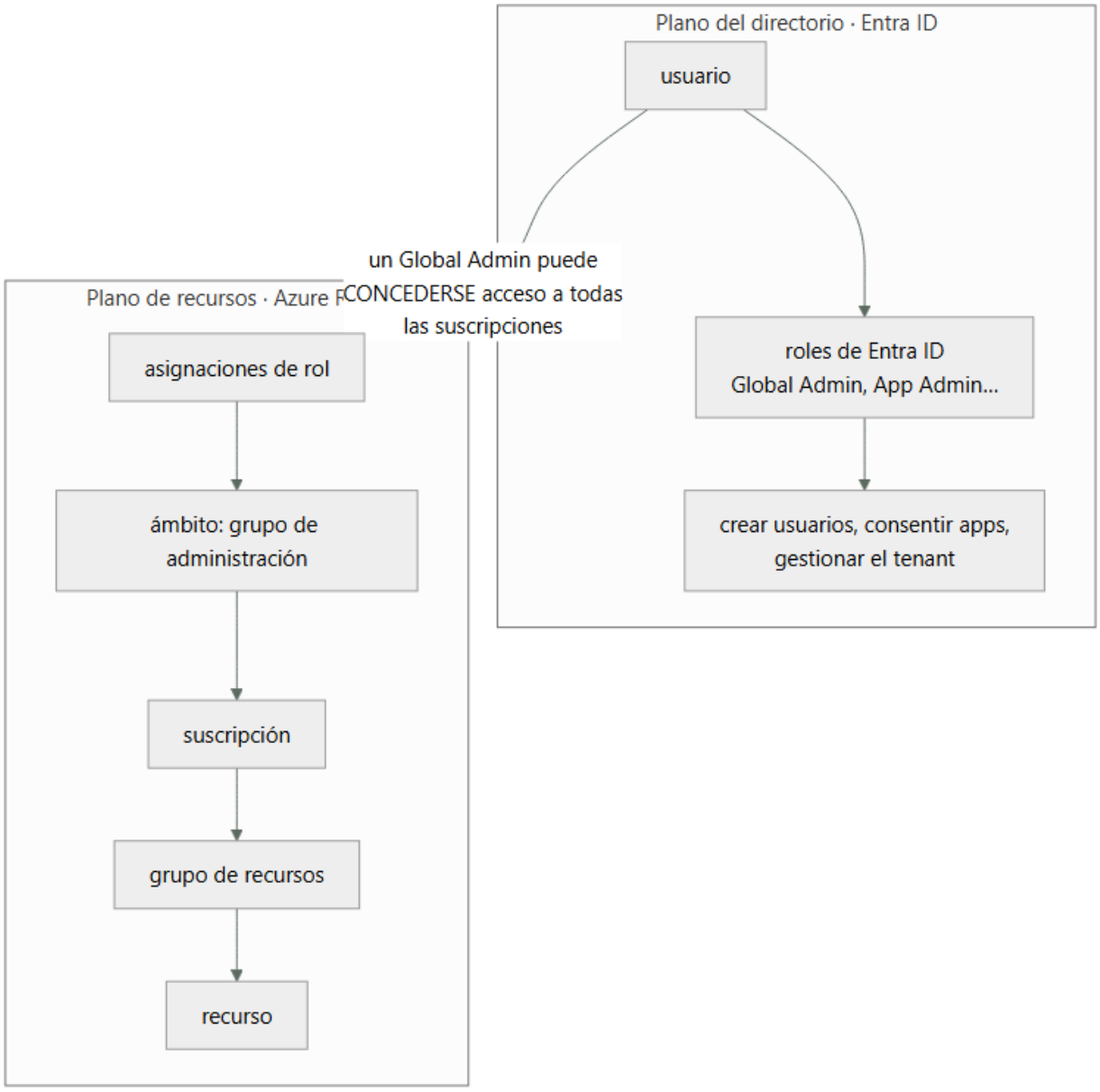
Concepto	Comprensión verificable
rol de Entra ID	Permiso sobre el directorio: crear usuarios, conceder consentimiento a aplicaciones, gestionar el propio tenant. No otorga ningún permiso sobre recursos de Azure.
rol de Azure RBAC	Permiso sobre recursos, asignado en un ámbito —grupo de administración, suscripción, grupo de recursos o recurso—. Se hereda hacia abajo y es acumulativo.
identidad administrada	Identidad que Azure crea y rota por ti, asociada a un recurso. Elimina el secreto: no hay nada que guardar ni que filtrar.
asignación de denegación	Mecanismo que bloquea acciones con independencia de los roles concedidos. A diferencia de IAM, no se puede crear directamente: solo la generan servicios como los blueprints o los entornos gestionados.
acceso con privilegios mínimos en el tiempo	Modelo en el que un rol privilegiado no está activo permanentemente: se activa por un periodo acotado, con justificación y posible aprobación.

Modelo mental

En Azure, identidad y jerarquía organizacional preceden al recurso: tenant, management group, suscripción y resource group determinan el alcance de control.

Aplicado a esta clase, separa siempre cuatro planos: intención (qué necesita el usuario), configuración (qué declaramos), estado observado (qué existe de verdad) y evidencia (cómo sabemos que cumple). Confundirlos produce diseños que se ven correctos en un diagrama pero fallan al operar.

■ Flujo de razonamiento



Desarrollo

1. Dos sistemas de roles que no se solapan

Azure tiene dos planos de autorización independientes, y esa es la diferencia estructural con AWS:

	Roles de Entra ID	Roles de Azure RBAC
Gobiernan	El directorio	Los recursos
Ejemplos	Global Administrator, User Administrator	Owner, Contributor, Reader
Ámbito	Todo el tenant o unidades administrativas	Grupo de administración → recurso
Pregunta que responden	¿Puedes crear un usuario?	¿Puedes crear una máquina virtual?

Un Global Administrator no tiene por defecto ningún permiso sobre recursos. Puede crear usuarios y consentir aplicaciones, y no puede leer una base de datos.

Pero hay una conexión peligrosa que conviene conocer:

```
# Un Global Administrator puede concederse a sí mismo acceso a TODAS las suscripciones
$ az rest --method post --url "https://management.azure.com/providers/Microsoft.Authorization/
elevateAccess?api-version=2016-07-01"
```

Esa llamada le asigna User Access Administrator en el ámbito raíz. Es una función legítima —recuperar el control de una suscripción huérfana— y una escalada completa. Debe alertarse siempre, y el número de Global Administrators debe ser mínimo: entre dos y cuatro, todos con acceso privilegiado en el tiempo.

Al depurar «no tengo permiso», la primera pregunta es en qué plano está la acción. Crear un grupo de recursos es RBAC; invitar a un usuario externo es Entra ID. Añadir permisos en el plano equivocado no cambia nada, igual que añadir permisos de identidad contra un Deny de SCP en la clase 026.

2. RBAC es aditivo y las denegaciones no se crean a mano

El modelo de evaluación de Azure RBAC es más simple que el de AWS y su simplicidad tiene una consecuencia importante:

```
permiso efectivo = unión de todas las asignaciones en todos los ámbitos heredados
                   menos las asignaciones de denegación (que no puedes crear)
```

Es aditivo. No existe un Deny que puedas escribir para excluir una acción concreta de un rol amplio. Si alguien tiene Contributor en la suscripción, tiene todo lo que ese rol incluye sobre todo lo que hay debajo, y no hay forma de restarle una acción con otra asignación.

Las dos formas reales de acotar:

```
1. Rol personalizado con NotActions
   define exactamente lo que incluye y lo que excluye

2. Azure Policy con efecto Deny
   no es autorización, es gobierno: impide la operación aunque el rol la permita
```

La segunda es la que sustituye funcionalmente al Deny de IAM, y opera en otro plano: el rol dice que puedes y la directiva impide que ocurra. Al diagnosticar, el mensaje de error las distingue:

```
AuthorizationFailed              → falta el rol
RequestDisallowedByPolicy        → el rol lo permite y la directiva lo impide
```

Un rol personalizado con exclusiones:

```
{
  "Name": "Colaborador sin borrado",
  "Actions": ["Microsoft.Compute/*", "Microsoft.Network/*", "Microsoft.Storage/*"],
  "NotActions": [
    "Microsoft.Compute/virtualMachines/delete",
    "Microsoft.Storage/storageAccounts/delete"
  ],
  "AssignableScopes": ["/subscriptions/aaaa-bbbb"]
}
```

Y una precisión que evita sorpresas: NotActions no es una denegación. Si otra asignación concede esa misma acción, se permite. Solo excluye la acción de ese rol.

3. Identidades administradas: el secreto que no existe

Es el equivalente al rol de instancia de AWS, con dos variantes:

	Asignada por el sistema	Asignada por el usuario
Ciclo de vida	Ligado al recurso: se borra con él	Independiente
Compartible	No	Sí, entre varios recursos
Cuándo	Un recurso único	Flotas y despliegues azul-verde

La segunda es la que conviene en una flota: si la identidad estuviera ligada al recurso, cada instancia nueva tendría una identidad distinta y habría que asignarle roles al crearla. Con identidad asignada por el usuario, los roles se conceden una vez y todas las instancias los heredan.

El código no maneja ningún secreto:

```
from azure.identity import DefaultAzureCredential
from azure.keyvault.secrets import SecretClient

cred = DefaultAzureCredential()          # detecta la identidad administrada
cliente = SecretClient("https://kv-cloudshop.vault.azure.net", cred)
```

```
secreto = cliente.get_secret("db-password").value
```

DefaultAzureCredential prueba varias fuentes en orden: variables de entorno, identidad administrada, credenciales del desarrollador. Eso hace que el mismo código funcione en local y en producción, y también esconde una trampa: si la identidad administrada falla, puede caer silenciosamente a otra credencial y funcionar por el motivo equivocado. En producción conviene fijar la fuente explícitamente.

Para el pipeline, la equivalencia con la federación OIDC de la clase 026:

```
$ az ad app federated-credential create --id $APP_ID --parameters '{
    "name": "github-produccion",
    "issuer": "https://token.actions.githubusercontent.com",
    "subject": "repo:miorg/cloudshop:environment:produccion",
    "audiences": ["api://AzureADTokenExchange"]
  }'
```

El campo subject cumple exactamente la misma función crítica que en AWS: sin él acotado, cualquier repositorio puede obtener el token. Y aquí hay un detalle propio: el valor es de coincidencia exacta, no admite comodines, lo que elimina el error de usar StringLike con un patrón demasiado amplio.

4. Privilegio mínimo en el tiempo, no solo en el alcance

El privilegio mínimo clásico acota qué puedes hacer. Añadir la dimensión temporal acota cuándo:

```
modelo permanente   el rol está activo 24×7
                    una credencial comprometida a las 3 de la madrugada
                    tiene los mismos permisos que a mediodía

modelo elegible     el rol existe pero está inactivo
                    se activa por 1-8 h, con justificación y a veces aprobación
                    fuera de esa ventana, el permiso no existe
```

La reducción de superficie es aritmética:

```
rol permanente:  720 h/mes de exposición
rol elegible activado 4 h/semana: 16 h/mes
reducción: 97,8 %
```

La configuración añade tres controles a la activación:

```
duración máxima          8 h, no indefinida
justificación            obligatoria y registrada
aprobación               para los roles más sensibles
notificación             a un segundo par al activarse
```

La última es la más rentable y la menos usada: si alguien activa un rol privilegiado y otra persona lo ve al instante, un uso indebido se detecta en minutos en vez de en la revisión trimestral.

Los roles que deberían ser siempre elegibles y nunca permanentes:

```
Global Administrator          (Entra ID)
Privileged Role Administrator (Entra ID)
Owner y User Access Administrator en suscripciones de producción (RBAC)
```

Y una excepción deliberada: conviene mantener dos cuentas de emergencia con acceso permanente, excluidas de acceso condicional y con credenciales en custodia física. Son la salida si el sistema de activación o el proveedor de identidad falla. Su uso debe alertar de inmediato, y su existencia hay que documentarla — porque una cuenta de emergencia olvidada es una puerta trasera.

5. Diagnosticar un acceso denegado

El orden que acota el problema en minutos:

```
# 1. ¿Qué roles tengo y en qué ámbito?
$ az role assignment list --assignee $UPN --all -o table --include-inherited

# 2. ¿La acción concreta está en alguno de esos roles?
$ az role definition list --name "Storage Blob Data Reader" \
    --query "[0].permissions[0].{a:actions,da:dataActions}"

# 3. ¿Lo impide una directiva en vez de faltar el rol?
$ az policy state list --resource $ID --filter "complianceState eq 'NonCompliant'" \
    --query "[].policyDefinitionName"
```

El paso 2 esconde la sutileza que más confunde en Azure: existen actions y dataActions, y son planos distintos.

```
actions      operaciones sobre el recurso: crear la cuenta de almacenamiento,
             leer sus claves, cambiar su configuración
dataActions  operaciones sobre los DATOS: leer un blob concreto
```

Un rol Contributor sobre una cuenta de almacenamiento no permite leer los blobs — permite gestionarla y obtener sus claves, que es otra cosa. Leerlos exige un rol con dataActions, como Storage Blob Data Reader.

Eso produce el error más desconcertante para quien llega de AWS: alguien con Owner sobre la suscripción recibe AuthorizationPermissionMismatch al listar blobs. No es un fallo: es que Owner concede actions y no dataActions sobre datos.

Y el mensaje de error distingue los tres casos:

```
AuthorizationFailed                 falta el rol de gestión
AuthorizationPermissionMismatch     falta el rol de DATOS
RequestDisallowedByPolicy           hay rol y una directiva lo impide
```

Leerlo ahorra la búsqueda en el plano equivocado, igual que en la clase 026.

Ejemplo trabajado

Al desplegar CloudShop en Azure, el equipo replica el rol del pipeline de AWS y se encuentra con tres problemas en dos días.

Problema 1 — el pipeline no puede leer el almacenamiento pese a ser Owner.

```
$ az role assignment list --assignee $APP_ID --all --query "[].{rol:roleDefinitionName,ambito:scope}"
-o tsv
Owner    /subscriptions/aaaa-bbbb
$ az storage blob list --account-name stcloudshop --container-name facturas --auth-mode login
AuthorizationPermissionMismatch
```

Owner sobre la suscripción y no puede listar blobs. La causa es la separación entre actions y dataActions:

```
$ az role definition list --name Owner --query "[0].permissions[0].dataActions" -o tsv
(vacío)
```

Owner no tiene ninguna dataAction. Se asigna el rol de datos, acotado al contenedor:

```
$ az role assignment create --assignee $APP_ID \
    --role "Storage Blob Data Contributor" \
    --scope "/subscriptions/aaaa-bbbb/resourceGroups/rg-prod/providers/\
Microsoft.Storage/storageAccounts/stcloudshop/blobServices/default/containers/facturas"
$ az storage blob list --account-name stcloudshop --container-name facturas --auth-mode login -o tsv
| wc -l
412                                                                    ✓
```

Y se aprovecha para quitar Owner, que estaba de más:

```
antes:   Owner sobre la suscripción entera
después: Storage Blob Data Contributor sobre UN contenedor
         + Website Contributor sobre el grupo de recursos de la app
```

Problema 2 — el pipeline usaba un secreto de aplicación.

```
$ az ad app credential list --id $APP_ID --query "[].{fin:endDateTime}" -o tsv
2027-03-14T10:22:00Z
```

Un secreto con dos años de vigencia, guardado en el repositorio. Se sustituye por federación:

```
$ az ad app federated-credential create --id $APP_ID --parameters '{
    "name":"github-produccion",
    "issuer":"https://token.actions.githubusercontent.com",
    "subject":"repo:miorg/cloudshop:environment:produccion",
    "audiences":["api://AzureADTokenExchange"]}'
$ az ad app credential delete --id $APP_ID --key-id $KEY_ID
```

Pruebas del sujeto, igual que en la clase 026:

```
desde otro repositorio                  AADSTS70021: no matching federated identity  ✓
desde miorg/cloudshop, rama de trabajo  AADSTS70021                                  ✓
desde miorg/cloudshop, entorno produccion token emitido                              ✓
```

Problema 3 — un despliegue legítimo bloqueado.

```
RequestDisallowedByPolicy: Resource 'st-cloudshop-tmp' was disallowed by policy
  'Storage accounts should restrict network access'
```

El mensaje no dice AuthorizationFailed, así que no falta rol: hay rol y una directiva lo impide. Añadir permisos no habría servido de nada.

```
$ az policy assignment show --name red-almacenamiento --query "parameters" -o json
{"effect":{"value":"Deny"}}
```

La cuenta temporal se creaba sin restricción de red. Se corrige en la plantilla, no en la directiva.

Revisión final de la superficie de identidad:

```
$ az role assignment list --all --query "[?roleDefinitionName=='Owner']\
.{p:principalName,a:scope}" -o tsv | wc -l
11
```

Once asignaciones de Owner. Se reducen a dos y el resto pasa a elegible con activación:

```
Owner permanentes                   11            2 (emergencia, documentadas)
Owner elegibles                      0            9 (activación de 8 h con justificación)
Global Administrators                6            3, todos elegibles
secretos de aplicación               1            0
exposición de roles privilegiados  720 h/mes    ~22 h/mes   (−97 %)
```

La lección que traslada esta clase al resto del programa: el modelo de identidad de Azure se parece al de AWS en el objetivo y difiere en la mecánica. Traducir literalmente —«Owner es como AdministratorAccess»— produce a la vez permisos de más en el plano de gestión y permisos de menos en el plano de datos.

Laboratorio guiado

Ejecuta desde la raíz:

```
python classes/part-03-azure-core-platform/038-microsoft-entra-id-rbac-managed-identities-y-pim/lab.py
```

El laboratorio selecciona el motor de práctica iam y produce labresult.json. El escenario, sus comprobaciones y el artefacto esperado corresponden a esta clase; no requiere credenciales y deja explícito qué debe revalidarse en un sandbox real.

1. Lee exercise.steps y formula una predicción antes de ejecutar.
2. Ejecuta la práctica y verifica todos los elementos de checks.
3. Provoca el caso de negativetest y explica la señal observada.
4. Materializa identidad-azure en evidence/ usando la plantilla indicada.
5. Para proveedor real, sigue sandbox.requires, registra costo y ejecuta destroy.

Evidencia esperada

El artefacto principal es una matriz de acceso mínimo con prueba de denegación. Además, la entrega debe incluir el comando ejecutado, la salida estructurada y una conclusión que no exceda lo observado.

Reto verificable

Construye identidad-azure para el caso CloudShop. Incluye una alternativa descartada, un supuesto que pueda falsarse, una prueba de fallo y una decisión de rollback.

■ Criterio de aceptación

- [ ] lab.py termina con código 0 y genera JSON válido.
- [ ] La entrega conecta al menos tres requisitos con mecanismos verificables.
- [ ] Existe una prueba positiva y una prueba negativa con evidencia.
- [ ] Seguridad, costo y operación aparecen como decisiones, no como anexos.
- [ ] Se declara una limitación y una condición que obligaría a revisar el diseño.
- [ ] Otra persona puede repetir el recorrido sin conocimiento tácito.

■■ Errores frecuentes

Síntoma	Causa probable	Corrección
Un principal con Owner recibe AuthorizationPermissionMismatch al leer datos	Owner concede actions y no dataActions: son planos distintos	Asigna un rol de datos como Storage Blob Data Reader, acotado al contenedor.
Se añaden roles y el despliegue sigue fallando	El error es RequestDisallowedByPolicy: hay rol y una directiva lo impide	Lee el código de error; con directivas, la corrección va en la plantilla o en la directiva, no en los roles.
No se puede excluir una acción concreta de un rol amplio	Azure RBAC es aditivo y las asignaciones de denegación no se crean a mano	Usa un rol personalizado con NotActions o una directiva con efecto Deny.
Un Global Administrator obtiene acceso a todas las suscripciones	La elevación de acceso es una función legítima del rol	Minimiza los Global Administrators, hazlos elegibles y alerta siempre sobre elevateAccess.
Cada instancia nueva de una flota necesita asignaciones de rol propias	Se usó identidad asignada por el sistema, ligada al ciclo de vida del recurso	Usa identidad asignada por el usuario: los roles se conceden una vez y las instancias la comparten.

■ Seguridad, ética y costo

Trabaja con cuentas propias o sandboxes autorizados. No publiques secretos, identificadores reales ni datos personales. Antes de crear recursos pagos define presupuesto, etiquetas y comando de destrucción; después verifica que no queden recursos huérfanos. Los ejemplos locales enseñan contratos, pero no certifican cumplimiento ni disponibilidad de producción.

■ Preguntas de comprobación

1. ¿Qué puede hacer un Global Administrator sobre los recursos de una suscripción, y cómo puede cambiar eso?
2. ¿Por qué NotActions en un rol personalizado no equivale a un Deny de IAM?
3. Un principal con Owner no puede listar blobs. ¿Cuál es la causa y cuál el arreglo?
4. ¿Qué distingue AuthorizationFailed de RequestDisallowedByPolicy y qué corrige cada uno?
5. ¿Cuánto se reduce la exposición al pasar un rol de permanente a elegible activado 4 h por semana?

Referencias

- Microsoft (2024). Azure RBAC vs Microsoft Entra roles — los dos planos y qué gobierna cada uno. <https://learn.microsoft.com/en-us/entra/identity/role-based-access-control/custom-overview>
- Microsoft (2024). Understand Azure role definitions — actions, notActions, dataActions y su evaluación. <https://learn.microsoft.com/en-us/azure/role-based-access-control/role-definitions>
- Microsoft (2024). Managed identities for Azure resources — asignadas por el sistema y por el usuario. <https://learn.microsoft.com/en-us/entra/identity/managed-identities-azure-resources/overview>
- Microsoft (2024). Workload identity federation — credenciales federadas y coincidencia exacta del sujeto. <https://learn.microsoft.com/en-us/entra/workload-id/workload-identity-federation>
- Microsoft (2024). Privileged Identity Management — activación con caducidad, justificación y aprobación. <https://learn.microsoft.com/en-us/entra/id-governance/privileged-identity-management/pim-configure>
- Documentación oficial vigente del servicio implementado; registra URL y fecha de consulta.

Descargar: Parte 03 en PDF · Recorrido de Azure en PDF · Manual integral

Anterior	Índice	Siguiente
← 037 · Tenant, management groups, suscripciones y resource groups	Parte 03 · Programa	039 · Virtual Network, subredes, NSG, UDR, peering y Private Link →

Evaluación — 038 Microsoft Entra ID, RBAC, managed identities y PIM

Preguntas

1. Explica el problema que resuelve esta capacidad sin mencionar un proveedor.
2. Identifica una frontera de responsabilidad y su propietario.
3. Compara dos alternativas mediante confiabilidad, seguridad, costo y operación.
4. ¿Qué demuestra el laboratorio y qué no puede demostrar?
5. Propón un caso límite o una prueba de fallo.

Reto verificable

Entrega identidad-azure con diagrama o configuración, evidencia de ejecución, alternativa descartada, riesgo residual y criterio de rollback.

Criterio de aceptación

- [ ] Resultado reproducible por una segunda persona.
- [ ] Evidencia vinculada a cada afirmación técnica.
- [ ] Prueba negativa o de fallo incluida.
- [ ] Costos y permisos expresados con unidades y alcance.
- [ ] Limitaciones declaradas sin presentar la simulación como producción.

Escala

Nivel	Evidencia
A — competente	Cumple todo y defiende trade-offs con datos.
B — en desarrollo	Cumple lo esencial; falta profundidad en una dimensión.
C — inicial	Artefacto parcial o conclusión sin evidencia suficiente.
0	Sin entrega reproducible.

039 — Virtual Network, subredes, NSG, UDR, peering y Private Link

Parte: 03 — Azure: plataforma esencial Nivel: intermedio · Horas estimadas: 4 Laboratorio: network · Estado: EXECUTABLECORE

Propósito

Diseñar la red de Azure sabiendo dónde falla la traducción literal desde una VPC. Tres puntos concretos rompen el modelo mental de la clase 027: no existe la subred privada por defecto, hay dos grupos de seguridad de red evaluándose en cadena y ambos deben permitir, y un punto de conexión privado sin zona DNS no falla — funciona por el camino público mientras crees que es privado. Es la base de las clases 040 a 048 y el origen de la mayoría de incidentes de conectividad de la parte.

Resultados de aprendizaje

Al finalizar podrás:

1. Planificar el espacio de direcciones contando las cinco IP reservadas por subred y las subredes de nombre obligatorio.
2. Explicar por qué una subred de Azure alcanza internet sin ninguna ruta declarada y qué cambió el 30 de septiembre de 2025.
3. Aplicar reglas de NSG sabiendo que subred y NIC se evalúan en cadena y que las reglas por defecto permiten todo dentro de la red virtual.
4. Decidir entre endpoint de servicio y punto de conexión privado con criterio de alcance, origen y costo.
5. Diagnosticar conectividad de abajo hacia arriba distinguiendo ruta, NSG y resolución de nombres.

Conceptos centrales

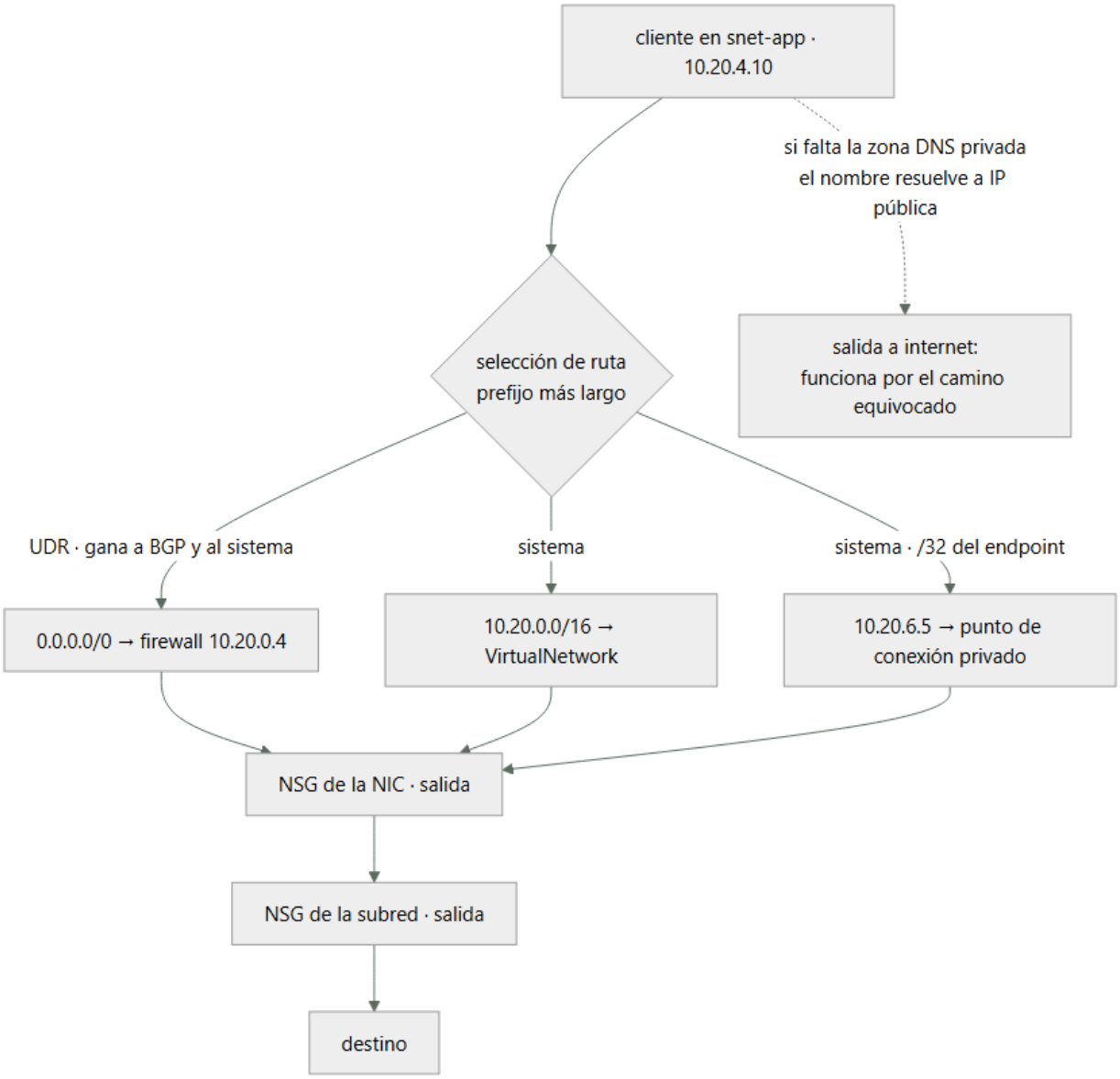
Concepto	Comprensión verificable
red virtual y espacio de direcciones	Ámbito de direccionamiento privado dentro de una región. Azure reserva cinco direcciones por subred —las cuatro primeras y la última—, así que una /24 ofrece 251 utilizables, no 254.
subred de nombre reservado	Subredes cuyo nombre es parte de la API: GatewaySubnet, AzureFirewallSubnet, AzureBastionSubnet. Escrito de otra forma, el servicio simplemente no se puede crear ahí.
ruta del sistema	Ruta que Azure crea sin que la declares, incluida 0.0.0.0/0 → Internet en toda subred. Es la razón de que no exista una subred privada por omisión.
UDR	Tabla de rutas definida por el usuario. Se elige por prefijo más largo y, a igualdad de prefijo, UDR gana a BGP y BGP gana al sistema.
NSG	Filtro con estado que puede aplicarse a la subred y a la NIC a la vez. Ambos se evalúan y ambos deben permitir; la primera regla que coincide decide.
punto de conexión privado	Interfaz de red con IP privada de tu subred que representa a un recurso concreto. Sin la zona DNS privada correspondiente no da error: el nombre sigue resolviendo a la IP pública.

Modelo mental

En Azure, identidad y jerarquía organizacional preceden al recurso: tenant, management group, suscripción y resource group determinan el alcance de control.

Aplicado a esta clase, separa siempre cuatro planos: intención (qué necesita el usuario), configuración (qué declaramos), estado observado (qué existe de verdad) y evidencia (cómo sabemos que cumple). Confundirlos produce diseños que se ven correctos en un diagrama pero fallan al operar.

■ Flujo de razonamiento



Desarrollo

1. Cinco direcciones y tres nombres que no puedes elegir

El direccionamiento se planifica antes de crear nada, por la misma razón que en la clase 027: el emparejamiento entre redes virtuales rechaza espacios solapados, y dos equipos que eligieron 10.0.0.0/16 cada uno por su cuenta ya no pueden conectarse sin rehacer una de las dos.

Azure reserva cinco direcciones por subred, igual que AWS, pero con otros papeles:

```
10.20.1.0    identificador de red
10.20.1.1    puerta de enlace predeterminada
10.20.1.2    servicio DNS de Azure
10.20.1.3    reservada
10.20.1.255  difusión
→ una /24 ofrece 251 direcciones utilizables
```

Hasta aquí, aritmética conocida. Lo que no existe en AWS son las subredes cuyo nombre forma parte del contrato:

Nombre exacto	Tamaño mínimo	Para qué
GatewaySubnet	/27 recomendado	Puertas de enlace de VPN y ExpressRoute

Nombre exacto	Tamaño mínimo	Para qué
AzureFirewallSubnet	/26	Azure Firewall
AzureFirewallManagementSubnet	/26	Plano de gestión del firewall forzado
AzureBastionSubnet	/26	Bastion

El nombre no es una convención, es la API. Una subred llamada snet-gateway con el tamaño correcto no sirve: la puerta de enlace no se puede crear ahí. Y el orden importa, porque estas subredes deben caber en el espacio que ya repartiste: descubrir que hace falta una /26 para el firewall cuando solo quedan /28 libres obliga a ampliar el espacio de direcciones de la red virtual y a recrear emparejamientos.

Un reparto que deja sitio a lo que vendrá en las clases 040 a 048:

```
vnet-hub    10.20.0.0/22
  AzureFirewallSubnet   10.20.0.0/26
  AzureBastionSubnet    10.20.0.64/26
  GatewaySubnet         10.20.0.128/27
  snet-dns              10.20.0.160/28

vnet-spoke-app  10.20.4.0/22
  snet-app              10.20.4.0/24    (251 utilizables)
  snet-datos            10.20.5.0/24
  snet-endpoints        10.20.6.0/26    (puntos de conexión privados)
  snet-integracion      10.20.6.64/26   (delegada a App Service)
```

La última merece una nota: una subred delegada a un servicio —App Service, Container Apps, bases de datos flexibles— queda bajo control de ese servicio y no admite otros recursos. Se decide al crearla y cambiarla implica vaciarla, así que conviene reservarla desde el principio aunque todavía no se use.

2. No existe la subred privada: toda subred sale a internet

Esta es la diferencia estructural con AWS, y la que más incidentes produce al llegar.

En AWS, «pública» y «privada» son propiedades de la tabla de rutas: una subred es pública porque su tabla apunta a una puerta de enlace de internet, y es privada porque no lo hace. En Azure no hay puerta de enlace que adjuntar. Toda subred nace con esta ruta del sistema:

```
0.0.0.0/0  →  Internet
```

Una máquina virtual sin IP pública, sin UDR y sin NAT alcanza internet igualmente, mediante una traducción de direcciones implícita con una IP que elige la plataforma. Se llama acceso saliente predeterminado y tiene tres consecuencias:

```
1. La salida no está controlada: cualquier proceso sale sin que nadie lo declare.
2. La IP de origen no es estable ni tuya: no puede figurar en la lista de
   permitidos de un tercero, porque cambia sin aviso.
3. Los puertos de traducción son escasos y no se pueden dimensionar.
```

Qué cambió. Desde el 30 de septiembre de 2025 el acceso saliente predeterminado está retirado para las redes virtuales nuevas: un despliegue nuevo debe declarar cómo sale. Las redes virtuales creadas antes lo conservan, así que la pregunta útil sobre una plataforma existente no es «¿tenemos salida implícita?» sino «¿de qué fecha es esta red virtual?».

Las tres formas explícitas de salir, y por qué no son intercambiables:

Mecanismo	Puertos de traducción	Cuándo
IP pública en la NIC	64.000 por instancia	Una máquina suelta; no escala ni se audita bien
Regla de salida del balanceador	64.000 repartidos entre las instancias	Flotas pequeñas y estables
NAT Gateway	64.512 por IP, asignados bajo demanda, hasta 16 IP	Recomendado por defecto

El reparto de la segunda opción es la trampa aritmética:

```
64.000 puertos / 100 instancias = 640 puertos por instancia
un servicio que abre 200 conexiones cortas por segundo al mismo destino,
con 240 s de espera antes de reutilizar el puerto → agotamiento en segundos
```

El síntoma es característico y desorienta: conexiones que fallan por tiempo de espera solo hacia un destino concreto, mientras el resto de la red funciona. El NAT Gateway lo elimina porque asigna puertos bajo demanda en vez de repartirlos por adelantado.

Su costo tiene la misma forma que el de AWS, así que la lección de la clase 027 se traslada entera:

```
precio por hora         ~0,045 USD  →  32,85 USD/mes
precio por GB procesado ~0,045 USD
```

Y para conseguir de verdad una subred sin salida hacen falta tres cosas, no una:

```
# 1. desactivar el acceso saliente predeterminado en la subred
$ az network vnet subnet update -g rg-red --vnet-name vnet-spoke-app -n snet-datos \
    --default-outbound-access false

# 2. ninguna UDR que envíe 0.0.0.0/0 a Internet
# 3. NSG que deniegue la salida a la etiqueta Internet
$ az network nsg rule create -g rg-red --nsg-name nsg-datos -n deny-internet-out \
    --priority 4000 --direction Outbound --access Deny \
    --source-address-prefixes '*' --destination-address-prefixes Internet \
    --destination-port-ranges '*' --protocol '*'
```

Saltarse la primera es el error habitual: la regla de NSG bloquea el tráfico, pero la ruta sigue existiendo y basta con que alguien afloje el NSG para que la salida vuelva sin que nadie lo note.

3. Dos NSG en cadena y una regla por defecto que lo permite todo dentro

El NSG tiene estado, en la subred y en la NIC. Esto elimina de golpe la trampa de las ACL de red de AWS —aquellas reglas de vuelta para los puertos efímeros de la clase 027—, y a cambio introduce otra.

```
entrada:  NSG de la subred  →  NSG de la NIC   →  máquina
salida:   NSG de la NIC     →  NSG de la subred →  red
```

Ambos se evalúan y ambos deben permitir. Un NSG de subred impecable no sirve de nada si la NIC tiene otro que deniega, y el mensaje de error no dice cuál de los dos fue. Por eso conviene una regla de equipo simple: NSG en la subred, salvo excepción documentada. Dos capas de filtrado en manos de dos equipos distintos producen incidentes que nadie sabe reproducir.

Las reglas por defecto, que no se pueden borrar y sí sobrescribir con prioridad menor:

```
entrada  65000  AllowVnetInBound              VirtualNetwork → VirtualNetwork
         65001  AllowAzureLoadBalancerInBound AzureLoadBalancer → *
         65500  DenyAllInBound

salida   65000  AllowVnetOutBound             VirtualNetwork → VirtualNetwork
         65001  AllowInternetOutBound         * → Internet
         65500  DenyAllOutBound
```

La primera es la que cambia el modelo mental. VirtualNetwork no significa «esta subred»: incluye toda la red virtual, las redes emparejadas y los rangos anunciados desde la red corporativa. Es decir:

```
AWS    dos instancias con grupos de seguridad distintos NO se hablan
       hasta que una regla lo permita
Azure  dos subredes cualesquiera de la red virtual SÍ se hablan,
       en todos los puertos, desde el primer minuto
```

La segmentación entre subredes en Azure es trabajo explícito, no un estado inicial. Denegar el tráfico lateral y abrir solo lo necesario:

```
$ az network nsg rule create -g rg-red --nsg-name nsg-datos -n allow-app-5432 \
    --priority 100 --direction Inbound --access Allow --protocol Tcp \
    --source-address-prefixes 10.20.4.0/24 --destination-port-ranges 5432

$ az network nsg rule create -g rg-red --nsg-name nsg-datos -n deny-lateral \
    --priority 200 --direction Inbound --access Deny --protocol '*' \
    --source-address-prefixes VirtualNetwork --destination-port-ranges '*'
```

Se evalúa por prioridad ascendente y la primera coincidencia decide: la 100 permite la aplicación y la 200 corta todo lo demás que venga de dentro, incluida la sonda del balanceador si no se tuvo cuidado. Ese es el incidente clásico, porque AllowAzureLoadBalancerInBound vive en la 65001 y cualquier denegación de usuario la sobrescribe:

```
sonda bloqueada → todas las instancias marcadas como no sanas
                → el balanceador deja de enviar tráfico
                → 502 en el frontal, y las máquinas están perfectamente vivas
```

La corrección es una regla de permiso con la etiqueta de servicio AzureLoadBalancer por encima de la denegación. Las etiquetas de servicio son la mejora real frente a mantener listas de IP a mano:

```
VirtualNetwork      la red virtual, las emparejadas y lo anunciado desde on-premises
AzureLoadBalancer   las sondas de estado de la plataforma
Internet            todo lo que no es privado ni de Azure
Storage.WestEurope  los rangos de almacenamiento de una región concreta
AzureMonitor, Sql, AzureKeyVault…
```

Microsoft actualiza los rangos; tus reglas no cambian. Y para el equivalente al grupo de seguridad que referencia a otro grupo de seguridad —que en Azure no existe tal cual— está el grupo de seguridad de aplicación: una etiqueta que se asigna a NIC y se usa como origen o destino en las reglas, en lugar de un CIDR. Permite escribir «los servidores web pueden hablar con los de base de datos» sin depender de cómo se repartieron las subredes.

4. UDR, emparejamiento y las dos formas de dejar la red sin salida

La selección de ruta sigue dos criterios, en este orden:

```
1. prefijo más largo (la ruta más específica gana)
2. a igualdad de prefijo:  UDR  >  BGP  >  sistema
```

Forzar todo el tráfico saliente por un firewall es una UDR de una línea sobre la subred de carga de trabajo:

```
$ az network route-table route create -g rg-red --route-table-name rt-app \
    -n via-firewall --address-prefix 0.0.0.0/0 \
    --next-hop-type VirtualAppliance --next-hop-ip-address 10.20.0.4
```

Y tiene dos formas conocidas de provocar una caída total:

La primera: aplicar esa misma UDR a la subred del firewall. El firewall enruta hacia sí mismo, el tráfico entra en bucle y la salida desaparece para toda la red. La regla es que AzureFirewallSubnet y GatewaySubnet no llevan una ruta por defecto hacia el propio dispositivo; asociar la tabla «a todas las subredes por comodidad» es exactamente cómo ocurre.

La segunda: el enrutamiento asimétrico. El tráfico entra por la IP pública del balanceador y, con la UDR puesta, la respuesta intenta salir por el firewall. El firewall ve un paquete de vuelta de una conexión que nunca vio empezar y lo descarta. El síntoma engaña: la conexión se establece y se queda colgada hasta agotar el tiempo, así que parece un problema de la aplicación. La corrección es una ruta más específica que devuelva ese tráfico por donde entró, o mover la entrada detrás del mismo dispositivo que inspecciona la salida.

Sobre el emparejamiento, tres propiedades que hay que tener presentes a la vez:

```
no es transitivo        radio A ↔ concentrador ↔ radio B  NO da  A ↔ B
es bidireccional        hay que crearlo en los dos sentidos o queda a medias
se factura en los dos   se paga el GB que sale y el GB que entra
```

La no transitividad se resuelve como en AWS, con distinta mecánica: una UDR en cada radio que envíe el rango del otro al firewall del concentrador, más dos interruptores del emparejamiento que casi siempre se olvidan:

```
allowForwardedTraffic   sin él, la red emparejada descarta el tráfico cuyo
                        origen no le pertenece — es decir, todo lo reenviado
allowGatewayTransit /   permiten que los radios usen la puerta de enlace
useRemoteGateways       del concentrador en vez de tener una cada uno
```

El coste conviene calcularlo antes de dibujar el diagrama, porque un concentrador cobra el tránsito dos veces. Para 900 GB mensuales entre dos radios de la misma región:

```
emparejamiento radio A ↔ concentrador
  900 GB salida × 0,01 + 900 GB entrada × 0,01   18,00
emparejamiento concentrador ↔ radio B
  900 GB salida × 0,01 + 900 GB entrada × 0,01   18,00
proceso de datos del firewall  900 × 0,016       14,40
                                                 ■■■■■■■
                                                  50,40

emparejamiento directo A ↔ B                      18,00
```

Inspeccionar ese tráfico cuesta 32,40 USD al mes. No es una cifra que descalifique el diseño; es una cifra que hay que poder decir en voz alta. La decisión defendible es la que elige el concentrador porque el tráfico entre radios debe inspeccionarse, no la que lo elige porque el diagrama de referencia lo dibujaba así.

5. Endpoint de servicio y punto de conexión privado: el que falla es el DNS

Dos mecanismos para llegar a un servicio de plataforma sin pasar por internet, con nombres parecidos y comportamientos distintos:

	Endpoint de servicio	Punto de conexión privado
Qué hace	Añade una ruta optimizada y presenta la identidad de la subred al servicio	Crea una NIC con IP privada de tu subred que representa al recurso
Dirección de destino	La IP pública del servicio	Una IP privada tuya
DNS	Sin cambios	Hay que cambiarlo
Alcance	El servicio en la región, acotado por el cortafuegos del recurso	Un recurso concreto, e incluso un subrecurso (blob, file, table)
Desde on-premises o red emparejada	No funciona	Sí
Costo	Gratuito	0,01 USD/h + 0,01 USD/GB procesado

La equivalencia con la clase 027 es casi exacta: el endpoint de servicio se comporta como el endpoint de tipo gateway de AWS —gratuito, basado en ruta, regional, inútil desde la red corporativa— y el punto de conexión privado como el endpoint de interfaz: una interfaz de red con IP privada, con costo por hora y por GB, alcanzable desde fuera de la red virtual.

Y aquí está el fallo más peligroso de esta clase, porque no se manifiesta como un error.

Crear el punto de conexión privado no cambia nada para el cliente. El nombre público sigue existiendo y sigue resolviendo hacia fuera:

```
stcloudshop.blob.core.windows.net
  → CNAME stcloudshop.privatelink.blob.core.windows.net
    → 20.60.x.x        (IP pública, si no hay zona privada)
    → 10.20.6.5        (IP del endpoint, si la zona privada está enlazada)
```

Sin la zona DNS privada, la aplicación sigue funcionando: sale a internet por el NAT, llega al almacenamiento por su IP pública y devuelve datos correctos. El punto de conexión privado está creado, facturado y sin usar. No hay alerta, no hay excepción, no hay traza roja en ningún panel. La única señal es una consulta:

```
$ nslookup stcloudshop.blob.core.windows.net      # desde dentro de snet-app
Address: 10.20.6.5                                 ✓
```

Los tres pasos completos, en orden:

```
# 1. la zona privada, con el nombre EXACTO que corresponde al servicio
$ az network private-dns zone create -g rg-red -n privatelink.blob.core.windows.net

# 2. enlazarla a cada red virtual que deba resolverlo
$ az network private-dns link vnet create -g rg-red -n link-spoke-app \
    -z privatelink.blob.core.windows.net -v vnet-spoke-app -e false

# 3. cerrar la puerta pública del recurso
$ az storage account update -n stcloudshop --public-network-access Disabled
```

El tercero es el que convierte esto en un control. Con la puerta pública abierta, el camino privado es solo un camino más: bonito en el diagrama e irrelevante para el auditor, porque cualquiera con la clave sigue entrando desde internet.

Dos detalles que ahorran una tarde de depuración:

Las zonas se centralizan. Una zona por nombre de servicio, alojada en la suscripción de conectividad y enlazada a todas las redes virtuales. Si cada equipo crea la suya, dos endpoints del mismo servicio en redes distintas producen resoluciones incoherentes según desde dónde preguntes. Lo sostenible es una directiva de Azure que registre automáticamente cada punto de conexión nuevo en la zona correcta — el mismo patrón de gobierno de la clase 037: la regla se aplica sola en vez de recordarse.

El NSG sobre el endpoint puede estar sin efecto. La subred tiene un interruptor, privateEndpointNetworkPolicies, que decide si los NSG y las UDR se aplican al punto de conexión. Si está deshabilitado, tus reglas se ignoran en silencio y el panel las muestra igualmente. Comprobarlo forma parte de la evidencia:

```
$ az network vnet subnet show -g rg-red --vnet-name vnet-spoke-app -n snet-endpoints \
    --query privateEndpointNetworkPolicies -o tsv
Enabled                                              ✓
```

Ejemplo trabajado

CloudShop traslada a Azure la VPC diseñada en la clase 027. El plano se copia en una tarde y produce cuatro incidentes en tres semanas — cada uno en uno de los puntos donde la traducción no era válida.

El punto de partida, traducido literalmente:

```
vnet-spoke-app 10.20.4.0/22
  snet-app     10.20.4.0/24   sin IP pública  → «privada»
  snet-datos   10.20.5.0/24   sin IP pública  → «privada»
```

Incidente 1 — las subredes privadas no eran privadas.

Una revisión de dependencias detecta que el servicio de catálogo descarga paquetes durante el arranque. No debería tener salida.

```
$ ssh app-01 'curl -s ifconfig.io'
20.103.44.187
```

Hay salida, y con una IP que nadie declaró. Es el acceso saliente predeterminado: la red virtual se creó en 2024 y lo conserva. Se corrige con NAT Gateway explícito y se cierra la subred de datos:

```
$ az network nat gateway create -g rg-red -n natgw-spoke-app \
    --public-ip-addresses pip-nat --idle-timeout 10
$ az network vnet subnet update -g rg-red --vnet-name vnet-spoke-app -n snet-app \
    --nat-gateway natgw-spoke-app
$ az network vnet subnet update -g rg-red --vnet-name vnet-spoke-app -n snet-datos \
    --default-outbound-access false
$ ssh datos-01 'curl -s --max-time 5 ifconfig.io; echo rc=$?'
rc=28                                                                    ✓
```

El tráfico de salida medido es de 3,2 TB/mes, de los cuales 2,4 TB son lecturas y escrituras contra el almacenamiento:

```
NAT Gateway  32,85 + 3.200 × 0,045              176,85
```

Incidente 2 — 502 en el frontal con todas las máquinas vivas.

Al aplicar segmentación se añadió una denegación del tráfico lateral:

```
200  deny-lateral   Inbound  VirtualNetwork → *  Deny
```

A los diez minutos, el balanceador devuelve 502 y las cuatro instancias figuran como no sanas. Ninguna se ha caído.

```
$ az network watcher test-ip-flow --vm app-01 --direction Inbound --protocol TCP \
    --local 10.20.4.10:8080 --remote 168.63.129.16:60000
Access: Deny   RuleName: deny-lateral
```

La sonda de estado llega desde AzureLoadBalancer, permitida por defecto en la prioridad 65001 — y la regla 200 la sobrescribe. Se antepone el permiso explícito:

```
$ az network nsg rule create -g rg-red --nsg-name nsg-app -n allow-lb-probe \
    --priority 100 --direction Inbound --access Allow --protocol '*' \
    --source-address-prefixes AzureLoadBalancer --destination-port-ranges '*'
$ az network lb show -g rg-red -n lb-app --query "probes[0].name" -o tsv && \
  curl -s -o /dev/null -w '%{http_code}\n' https://cloudshop.example
200                                                                      ✓
```

Incidente 3 — dos radios que no se ven, y la factura de arreglarlo.

El servicio de pedidos, en vnet-spoke-app, debe llamar al de facturación, en vnet-spoke-fin. Ambos están emparejados con el concentrador y no se alcanzan entre sí: el emparejamiento no es transitivo. Se añaden las UDR hacia el firewall y sigue sin funcionar.

```
$ az network vnet peering show -g rg-red --vnet-name vnet-spoke-app -n to-hub \
    --query "{fwd:allowForwardedTraffic}" -o tsv
False
```

El concentrador descartaba el tráfico reenviado. Con el interruptor activado en los cuatro emparejamientos —dos por sentido— la ruta se completa. El volumen entre radios es de 900 GB/mes:

```
vía concentrador con inspección                        50,40
emparejamiento directo entre radios, sin inspección    18,00
```

```
diferencia: el precio de inspeccionar                    32,40
```

Se deja el paso por el concentrador y se anota el motivo: son llamadas que cruzan una frontera de datos de pago y deben registrarse. La cifra queda escrita en la decisión, no descubierta en la factura.

Incidente 4 — el punto de conexión privado que llevaba cuatro meses sin usarse.

Una auditoría pide evidencia de que el acceso al almacenamiento no cruza internet. El endpoint existe desde el primer despliegue y la aplicación funciona sin fallos.

```
$ ssh app-01 'nslookup stcloudshop.blob.core.windows.net | tail -2'
Address: 20.60.191.132
```

Una IP pública. Nunca se creó la zona DNS privada, así que el tráfico salía por el NAT y volvía por la puerta pública del almacenamiento. Todo funcionaba, y ninguna de las dos afirmaciones del diagrama era cierta.

```
$ az network private-dns zone create -g rg-hub -n privatelink.blob.core.windows.net
$ az network private-dns link vnet create -g rg-hub -n link-spoke-app \
    -z privatelink.blob.core.windows.net -v vnet-spoke-app -e false
$ az storage account update -n stcloudshop --public-network-access Disabled

$ ssh app-01 'nslookup stcloudshop.blob.core.windows.net | tail -2'
Address: 10.20.6.5                                                    ✓
$ curl -s -o /dev/null -w '%{http_code}\n' \
    https://stcloudshop.blob.core.windows.net/facturas   # desde fuera de la red
403                                                                   ✓
```

La prueba negativa es la mitad que faltaba: que resuelva a una IP privada demuestra que el camino existe; que desde fuera devuelva 403 demuestra que es el único.

Y se recuperan 2,4 TB del NAT:

```
tráfico por NAT Gateway              3.200 GB      800 GB
costo del NAT   32,85 + GB × 0,045    176,85       68,85
punto de conexión privado (7,30 + 2.400 × 0,01)     0,00 → ya facturado    31,30
                                    ■■■■■■■■■■    ■■■■■■■■■■
                                     176,85       100,15   (−43 %)
```

El punto de conexión ya se estaba pagando desde el primer día. Enlazar la zona DNS no añadió costo: hizo que el gasto sirviera para algo.

Resumen del rediseño:

```
subredes con salida no declarada             2               0
camino de datos al almacenamiento          internet          privado
acceso público al almacenamiento           habilitado        deshabilitado
tráfico lateral permitido por defecto      todos los puertos  5432 desde snet-app
salida mensual por NAT                     3,2 TB            0,8 TB
costo mensual de red                       227,25 USD        150,55 USD
```

La lección que esta clase traslada al resto de la parte: en Azure, la conectividad es el estado por defecto y el aislamiento es trabajo explícito. Al revisar un diseño heredado, la pregunta útil no es «¿qué hemos abierto?» sino «¿qué hemos cerrado, y cómo lo demostramos?».

Laboratorio guiado

Ejecuta desde la raíz:

```
python classes/part-03-azure-core-platform/039-virtual-network-subredes-nsg-udr-peering-y-private-link/lab.py
```

El laboratorio selecciona el motor de práctica network y produce labresult.json. El escenario, sus comprobaciones y el artefacto esperado corresponden a esta clase; no requiere credenciales y deja explícito qué debe revalidarse en un sandbox real.

1. Lee exercise.steps y formula una predicción antes de ejecutar.
2. Ejecuta la práctica y verifica todos los elementos de checks.
3. Provoca el caso de negativetest y explica la señal observada.
4. Materializa vnet-azure en evidence/ usando la plantilla indicada.

5. Para proveedor real, sigue sandbox.requires, registra costo y ejecuta destroy.

Evidencia esperada

El artefacto principal es una topología con rutas, puertos y controles. Además, la entrega debe incluir el comando ejecutado, la salida estructurada y una conclusión que no exceda lo observado.

Reto verificable

Construye vnet-azure para el caso CloudShop. Incluye una alternativa descartada, un supuesto que pueda falsarse, una prueba de fallo y una decisión de rollback.

■ Criterio de aceptación

- [ ] lab.py termina con código 0 y genera JSON válido.
- [ ] La entrega conecta al menos tres requisitos con mecanismos verificables.
- [ ] Existe una prueba positiva y una prueba negativa con evidencia.
- [ ] Seguridad, costo y operación aparecen como decisiones, no como anexos.
- [ ] Se declara una limitación y una condición que obligaría a revisar el diseño.
- [ ] Otra persona puede repetir el recorrido sin conocimiento tácito.

■■ Errores frecuentes

Síntoma	Causa probable	Corrección
Una máquina sin IP pública en una subred «privada» descarga desde internet	La ruta del sistema 0.0.0.0/0 → Internet existe en toda subred y el acceso saliente predeterminado la resuelve	Declara la salida con NAT Gateway y desactiva defaultOutboundAccess en las subredes que no deben salir; el NSG por sí solo no basta.
El balanceador devuelve 502 y todas las instancias figuran como no sanas, pero están vivas	Una regla de denegación de usuario sobrescribe AllowAzureLoadBalancerInBound, que vive en la prioridad 65001	Añade un permiso explícito para la etiqueta AzureLoadBalancer con prioridad menor que la denegación.
Dos subredes se hablan en todos los puertos sin que nadie lo haya permitido	AllowVnetInBound permite por defecto todo el tráfico de la red virtual, las emparejadas y lo anunciado desde on-premises	La segmentación es explícita: permite lo necesario y deniega el resto del origen VirtualNetwork.
El punto de conexión privado existe, la aplicación funciona y el tráfico sigue saliendo a internet	Falta la zona DNS privada enlazada, así que el nombre resuelve a la IP pública del servicio	Crea y enlaza privatelink.<servicio>, verifica con nslookup desde dentro y deshabilita el acceso público del recurso.
La conexión se establece y se queda colgada hasta agotar el tiempo de espera	Enrutamiento asimétrico: la entrada llega por el balanceador y la UDR devuelve la respuesta por el firewall, que nunca vio empezar la conexión	Añade una ruta más específica que devuelva ese tráfico por donde entró, o unifica entrada y salida en el mismo dispositivo.
Se pierde la salida de toda la red al aplicar una tabla de rutas	La UDR con 0.0.0.0/0 → dispositivo virtual se asoció también a AzureFirewallSubnet o a GatewaySubnet	Asocia las tablas de rutas subred por subred; esas dos nunca llevan la ruta por defecto hacia el propio dispositivo.

■ Seguridad, ética y costo

Trabaja con cuentas propias o sandboxes autorizados. No publiques secretos, identificadores reales ni datos personales. Antes de crear recursos pagos define presupuesto, etiquetas y comando de destrucción; después verifica que no queden recursos huérfanos. Los ejemplos locales enseñan contratos, pero no certifican cumplimiento ni disponibilidad de producción.

■ Preguntas de comprobación

1. ¿Por qué una máquina virtual sin IP pública y sin UDR alcanza internet, y qué tres cambios hacen falta para impedirlo?
2. Un NSG de subred permite el tráfico y la conexión falla. ¿Dónde más hay que mirar y por qué el error no lo indica?
3. ¿Qué permite AllowVnetInBound exactamente, y en qué se diferencia del comportamiento por defecto entre grupos de seguridad de AWS?
4. ¿Cuándo elegirías un endpoint de servicio en vez de un punto de conexión privado, y qué requisito descarta al primero?
5. El nslookup de una cuenta de almacenamiento devuelve una IP pública desde dentro de la red virtual y la aplicación funciona. ¿Qué está ocurriendo y qué evidencia demuestra la corrección?

Referencias

• Microsoft (2025). Default outbound access in Azure — la salida implícita y su retirada para redes virtuales nuevas. <https://learn.microsoft.com/en-us/azure/virtual-network/ip-services/default-outbound-access>

• Microsoft (2025). Network security groups — reglas por defecto, evaluación en subred y NIC, y etiquetas de servicio. <https://learn.microsoft.com/en-us/azure/virtual-network/network-security-groups-overview>

• Microsoft (2025). Virtual network traffic routing — rutas del sistema y precedencia entre UDR, BGP y sistema. <https://learn.microsoft.com/en-us/azure/virtual-network/virtual-networks-udr-overview>

• Microsoft (2025). Private endpoint DNS configuration — zonas privatelink, enlaces a redes virtuales y resolución desde on-premises. <https://learn.microsoft.com/en-us/azure/private-link/private-endpoint-dns>

• Microsoft (2025). Hub-spoke network topology — transitividad, tráfico reenviado y tránsito de puerta de enlace. <https://learn.microsoft.com/en-us/azure/architecture/networking/architecture/hub-spoke>

• Documentación oficial vigente del servicio implementado; registra URL y fecha de consulta.

Descargar: Parte 03 en PDF · Recorrido de Azure en PDF · Manual integral

Anterior	Índice	Siguiente
← 038 · Microsoft Entra ID, RBAC, managed identities y PIM	Parte 03 · Programa	040 · Virtual Machines, Scale Sets y Load Balancer →

Evaluación — 039 Virtual Network, subredes, NSG, UDR, peering y Private Link

Preguntas

1. Explica el problema que resuelve esta capacidad sin mencionar un proveedor.
2. Identifica una frontera de responsabilidad y su propietario.
3. Compara dos alternativas mediante confiabilidad, seguridad, costo y operación.
4. ¿Qué demuestra el laboratorio y qué no puede demostrar?
5. Propón un caso límite o una prueba de fallo.

Reto verificable

Entrega vnet-azure con diagrama o configuración, evidencia de ejecución, alternativa descartada, riesgo residual y criterio de rollback.

Criterio de aceptación

• [ ] Resultado reproducible por una segunda persona.

- [ ] Evidencia vinculada a cada afirmación técnica.
- [ ] Prueba negativa o de fallo incluida.
- [ ] Costos y permisos expresados con unidades y alcance.
- [ ] Limitaciones declaradas sin presentar la simulación como producción.

Escala

Nivel	Evidencia
A — competente	Cumple todo y defiende trade-offs con datos.
B — en desarrollo	Cumple lo esencial; falta profundidad en una dimensión.
C — inicial	Artefacto parcial o conclusión sin evidencia suficiente.
0	Sin entrega reproducible.

040 — Virtual Machines, Scale Sets y Load Balancer

Parte: 03 — Azure: plataforma esencial Nivel: intermedio · Horas estimadas: 4 Laboratorio: compute · Estado: EXECUTABLECORE

Propósito

Dimensionar cómputo en Azure sabiendo que hay dos límites de disco y casi siempre manda el que no compraste: el de la máquina. Las clases 028 y 029 dejaron el criterio —familia, IOPS, elasticidad, comprobación de estado—; aquí cambian las piezas y aparecen decisiones que en AWS no existen: los dominios de actualización, las dos orquestaciones de un conjunto de escalado, cuatro balanceadores con un reparto distinto y un aviso de desalojo de 30 segundos en vez de dos minutos.

Resultados de aprendizaje

Al finalizar podrás:

1. Leer el nombre de un tamaño de máquina virtual como una especificación y detectar qué falta cuando no lleva s ni d.
2. Distinguir el límite de IOPS del disco del límite de la máquina y localizar cuál es el cuello de botella real.
3. Elegir entre conjunto de disponibilidad, zonas y conjunto de escalado según el fallo que se quiere sobrevivir.
4. Configurar un conjunto de escalado con orquestación flexible, mezcla de prioridades y política de reducción explícita.
5. Seleccionar entre los cuatro balanceadores de Azure a partir del ámbito, la capa y el tiempo de conmutación aceptable.

Conceptos centrales

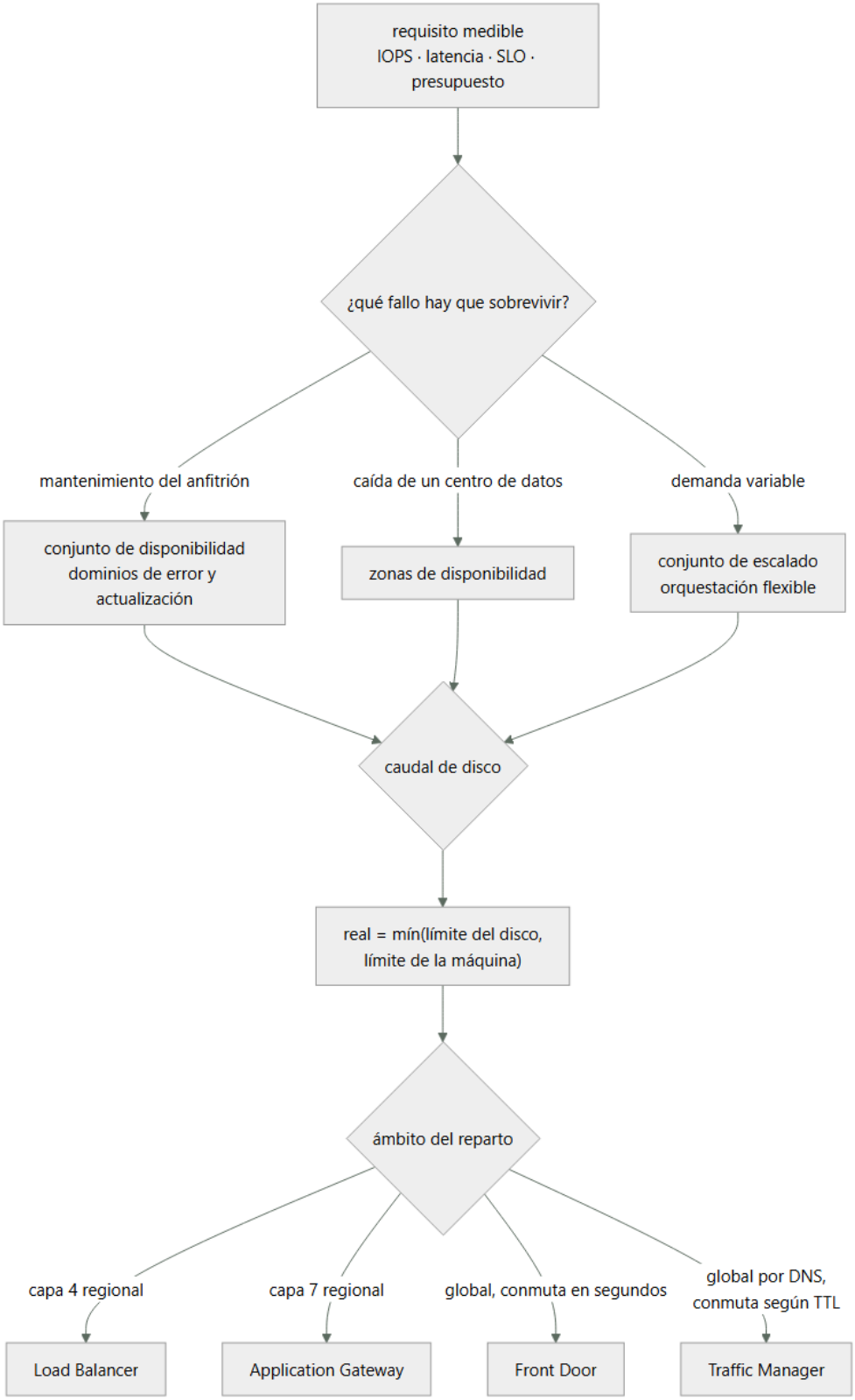
Concepto	Comprensión verificable
tamaño de máquina virtual	Cadena con estructura: familia, número de vCPU y letras aditivas. s habilita discos premium, d añade disco temporal local, a indica AMD, p indica ARM. Sin la letra, la capacidad no existe.
límite de la máquina frente al del disco	Cada tamaño tiene un tope propio de IOPS y de rendimiento de disco, distinto del que ofrece el disco. El caudal real es el menor de los dos.
almacenamiento en caché del host	Modo de caché del disco en el anfitrión: None, ReadOnly o ReadWrite. Cambia el límite aplicable y, en discos de registro de transacciones, ReadWrite puede costar datos.
dominio de actualización	Grupo de máquinas que el mantenimiento planificado de la plataforma reinicia a la vez. Un conjunto de disponibilidad con cinco garantiza que como mucho una quinta parte se reinicie de golpe.
orquestación flexible	Modo de conjunto de escalado en el que las instancias son máquinas virtuales reales, gestionables una a una y con tamaños o prioridades mezcladas. Es el modo por defecto.
aviso de desalojo	Notificación de eventos programados que anuncia el desalojo de una máquina de excedente. En Azure son 30 segundos, no dos minutos: solo sirve para abortar, no para drenar.

Modelo mental

En Azure, identidad y jerarquía organizacional preceden al recurso: tenant, management group, suscripción y resource group determinan el alcance de control.

Aplicado a esta clase, separa siempre cuatro planos: intención (qué necesita el usuario), configuración (qué declaramos), estado observado (qué existe de verdad) y evidencia (cómo sabemos que cumple). Confundirlos produce diseños que se ven correctos en un diagrama pero fallan al operar.

Flujo de razonamiento



Desarrollo

1. Las letras del tamaño son la especificación

La clase 028 dejó establecido que el nombre de una instancia es una especificación, no una etiqueta. En Azure el alfabeto es otro y el orden importa:

```
Standard_D8ads_v6
          ■ ■■■ ■■■ generación
          ■ ■■■■■■■■ s : admite discos premium
          ■ ■■■■■■■■ d : incluye disco temporal local
          ■ ■■■■■■■■ a : procesador AMD
          ■■■■■■■■■■ D8 : familia de propósito general, 8 vCPU
```

Las familias, en una línea cada una:

```
B   ráfaga con créditos      cargas ociosas la mayor parte del tiempo
D   propósito general        4 GB de memoria por vCPU
E   memoria                  8 GB por vCPU — bases de datos, cachés
F   cómputo                  2 GB por vCPU — cálculo, compilación
L   almacenamiento           NVMe local grande y muy rápido
M   memoria extrema          cargas SAP y bases de datos monolíticas
N   GPU
```

Las dos letras que más problemas dan son las que no están:

Sin s, el tamaño no admite discos premium. Se descubre al desplegar, con un error de compatibilidad que no menciona la letra, y obliga a cambiar el tamaño —y por tanto a reiniciar— con la plantilla ya en producción.

Sin d, no hay disco temporal local. Y aquí está el error silencioso: mucha automatización heredada asume que existe /mnt en Linux o D: en Windows porque «siempre estuvo ahí». En un D8sv5 no está. El servicio arranca, escribe en la raíz del disco del sistema operativo, y semanas después el disco se llena o el rendimiento cae, porque el disco del sistema no está dimensionado para eso.

Y una advertencia sobre el disco temporal cuando sí existe: es efímero de verdad. Se borra al desasignar la máquina y al moverla de anfitrión, sin aviso y sin copia. Sirve para archivos de intercambio, cachés reconstruibles y ficheros temporales de compilación. Poner ahí datos de una base de datos es la versión de Azure del problema que cerró la clase 028: el estado local es el enemigo de la elasticidad, y en este caso ni siquiera hay que escalar para perderlo.

Sobre la familia B basta con una nota, porque el mecanismo de créditos ya se trabajó en 028: es el mismo desplome, con distinta contabilidad. La diferencia práctica es que la señal se llama CPU Credits Remaining y hay que graficarla junto al porcentaje de CPU. Sin ella, un servicio que agota créditos aparece en el panel como una CPU al 20 % —el límite base— mientras las latencias se disparan, exactamente el falso negativo que describía aquella clase.

2. Dos límites de disco, y casi siempre manda el que no compraste

Este es el punto donde Azure se aparta más de AWS, y produce la conversación de «duplicamos el disco y no cambió nada».

En el SSD premium clásico, el rendimiento se compra comprando tamaño. No hay un parámetro de IOPS que ajustar: cada escalón de tamaño trae su cifra:

```
P10    128 GiB      500 IOPS     100 MB/s
P20    512 GiB    2.300 IOPS     150 MB/s
P30  1.024 GiB    5.000 IOPS     200 MB/s
P40  2.048 GiB    7.500 IOPS     250 MB/s
```

Quien viene de gp3 —donde IOPS y tamaño se compran por separado— compra tamaño que no necesita para conseguir IOPS que sí. Las alternativas modernas lo desacoplan: SSD premium v2 y Ultra Disk permiten fijar tamaño, IOPS y rendimiento de forma independiente, que es el comportamiento esperado; a cambio, no admiten caché del host y v2 tiene restricciones de zona.

Pero la cifra del disco es solo la mitad. Cada tamaño de máquina tiene su propio tope de IOPS y de MB/s, y el caudal real es el menor de los dos:

```
Standard_D2s_v5         3.750            85
Standard_D4s_v5         6.400           145
Standard_D8s_v5        12.800           290
Standard_D16s_v5       25.600           600
```

```
P40 (7.500 IOPS) en un D2s_v5 (3.750)  →  real: 3.750
cambiar a P50 no mueve la cifra
cambiar a D8s_v5 la sube a 7.500
```

La regla operativa es corta: antes de comprar disco, mira la fila de la máquina. Y para diagnosticar, la señal que lo distingue es la profundidad de cola junto al porcentaje de uso del disco: si el disco no está saturado y la cola crece, el techo es de la máquina.

La caché del host añade un tercer factor y una decisión con consecuencias:

```
None       lecturas y escrituras van al disco
           → discos de registro de transacciones y cargas de escritura intensa
ReadOnly   lecturas desde la memoria del anfitrión, escrituras al disco
           → discos de datos con lectura predominante; suele ser la mejor opción
ReadWrite  escrituras confirmadas en la caché del anfitrión
           → solo el disco del sistema operativo
```

ReadWrite en un disco de datos o de registro es un error con consecuencias reales: si el anfitrión se reinicia de forma no ordenada, las escrituras confirmadas a la aplicación y aún no bajadas al disco se pierden, y un motor transaccional se encuentra con un registro incoherente. El motivo por el que ocurre es prosaico: ReadWrite es el valor por defecto del disco del sistema, y quien copia la plantilla lo arrastra al resto.

Y un detalle contable: la caché cambia el límite aplicable. Un D8sv5 tiene un tope con caché mayor que sin ella, pero ese tope solo aplica a lo que la caché sirve. Contar con el número grande para dimensionar una carga de escritura es contar con capacidad que no se va a usar.

3. Tres formas de sobrevivir y no todas se combinan

Azure separa en tres construcciones lo que AWS resuelve casi todo con zonas:

	Qué fallo sobrevive	Ámbito	SLA orientativo
Máquina única con disco premium	Ninguno estructural	Un anfitrión	99,9 %
Conjunto de disponibilidad	Fallo de bastidor y mantenimiento planificado	Un centro de datos	99,95 %
Zonas de disponibilidad	Caída de un centro de datos completo	Una región	99,99 %

El conjunto de disponibilidad no tiene equivalente directo en AWS y aporta algo que las zonas no dan: los dominios de actualización. Son la unidad del mantenimiento planificado de la plataforma:

```
dominios de error        (hasta 3)  bastidores distintos: alimentación y red
dominios de actualización (hasta 20) grupos que se reinician por separado
                                     al actualizar el anfitrión
```

Con cinco dominios de actualización y diez máquinas, un mantenimiento reinicia como mucho dos a la vez. Sin conjunto de disponibilidad, nada impide que la plataforma reinicie las diez.

La restricción que hay que conocer antes de dibujar: conjunto de disponibilidad y zonas son excluyentes. Una máquina está en uno o en otras, y cambiar de decisión implica recrearla. En una región con zonas, la elección casi siempre es zonas; el conjunto de disponibilidad queda para regiones sin ellas.

El conjunto de escalado es la pieza que integra ambas cosas con elasticidad, y tiene dos modos:

	Uniforme	Flexible
Las instancias son	Objetos del conjunto	Máquinas virtuales reales
Tamaños mezclados	No	Sí
Excedente y bajo demanda mezclados	No	Sí
Gestión individual	Limitada	az vm funciona sobre ellas
Reparto por dominios de error	Implícito	Explícito y configurable

Flexible es el modo por defecto y el que conviene salvo que se necesite escalar a miles de instancias idénticas muy rápido. La razón práctica: permite mezclar prioridades —una base bajo demanda y el pico en excedente— y deja diagnosticar una instancia concreta con las mismas herramientas que cualquier otra máquina.

El autoescalado vive en Azure Monitor, no en el conjunto. Los cuatro parámetros y el efecto de la histéresis ya se trabajaron en la clase 029; lo que cambia aquí son dos cosas concretas:

```
política de reducción    Default | NewestVM | OldestVM
                         decide a quién apagar; con Default, el reparto
                         entre zonas se mantiene equilibrado
perfiles                 por defecto + perfiles con horario
                         para carga predecible, un perfil horario evita
                         escalar por reacción
```

Y la trampa compartida con AWS, que conviene repetir porque cuesta una noche: la memoria no es una métrica del anfitrión. El porcentaje de CPU se publica sin instalar nada; la memoria y el espacio en disco exigen el agente de Azure Monitor. Una regla de escalado por memoria configurada sin agente no falla: no dispara nunca.

4. Cuatro balanceadores y la conmutación que depende del TTL

El reparto de Azure no coincide con el de AWS, y traducir «esto es el ALB» produce elecciones equivocadas:

Servicio	Capa	Ámbito	Conmuta en	Equivalente aproximado
Load Balancer	4	Regional	Segundos	Network Load Balancer
Application Gateway	7	Regional	Segundos	Application Load Balancer + WAF
Front Door	7	Global, anycast	Segundos	CloudFront + enrutamiento global
Traffic Manager	DNS	Global	Lo que dure el TTL y la caché del cliente	Políticas de enrutamiento de Route 53

La fila que hay que leer dos veces es la última. Traffic Manager no ve el tráfico: responde consultas de DNS. Cuando una región cae, la conmutación no ocurre hasta que cada cliente vuelve a resolver el nombre, y los clientes no respetan el TTL de forma uniforme:

```
TTL de 60 s + caché del resolutor + caché de la JVM o del navegador
→ conmutación observada de varios minutos, con una cola larga
   de clientes que siguen yendo a la región caída
```

Para un requisito de conmutación en segundos, la respuesta es Front Door: el cliente conecta siempre a la misma dirección anycast y el borde decide a qué origen enviar. Traffic Manager sigue siendo útil para lo que sí resuelve bien: dirigir por proximidad geográfica, repartir entre destinos que no son HTTP y encaminar hacia recursos fuera de Azure.

Sobre el Load Balancer estándar hay cuatro hechos que producen incidentes:

Es denegar por defecto. El SKU básico se retiró el 30 de septiembre de 2025, y el estándar no acepta tráfico entrante si un NSG no lo permite explícitamente — el básico sí lo aceptaba. Una migración literal deja el servicio inalcanzable con toda la configuración aparentemente correcta.

No da salida por tener entrada. Una regla de balanceo entrante no proporciona conectividad saliente. Unido a la retirada del acceso saliente predeterminado de la clase 039, un despliegue nuevo se queda literalmente sin salida hasta que se declara un NAT Gateway o una regla de salida.

Las sondas llegan desde 168.63.129.16. Es una dirección virtual de la plataforma, la misma que sirve DHCP, DNS y el canal del agente de la máquina. Bloquearla no solo tumba las sondas: deja al agente sin comunicación, así que las extensiones dejan de responder y las operaciones desde el portal se quedan colgadas. Es un fallo con dos síntomas que parecen no tener relación.

La sonda debe reflejar disponibilidad real. Vale aquí íntegra la lección de la clase 029: un / que devuelve 200 mientras la base de datos no responde mantiene en rotación a una instancia que no puede servir. La forma correcta es un punto de comprobación que verifique las dependencias críticas, con un umbral de fallos consecutivos que tolere un parpadeo.

Un detalle propio de Azure que aparece en migraciones de bases de datos con clúster: la IP flotante —retorno directo desde el servidor— hace que el destino vea la dirección del frontal en lugar de la suya. Es imprescindible para grupos de disponibilidad de SQL Server y rompe cualquier servicio que no espere ese comportamiento. Y la regla de puertos de alta disponibilidad, que balancea todos los puertos y protocolos a la vez, existe para poner dispositivos virtuales de red en alta disponibilidad detrás de un balanceador interno: es la pieza que faltaba en el diseño de concentrador y radio de la clase 039.

5. Excedente: 30 segundos cambian el diseño

Las máquinas de excedente cuestan entre un 60 % y un 90 % menos y se retiran cuando Azure necesita la capacidad. La clase 028 fijó el criterio de los modelos de compra; lo que cambia aquí es un número:

```
AWS   aviso de interrupción de excedente   ~120 s
Azure aviso de desalojo                     ~30 s
```

Treinta segundos no dan para drenar conexiones, terminar una petición larga ni volcar un estado grande. Dan para abortar de forma ordenada: dejar de aceptar trabajo, confirmar o descartar la unidad en curso y salir. Cualquier diseño que dependa de un drenado educado sobre excedente en Azure está apostando a que el aviso llegue antes de lo prometido.

El aviso se consulta contra el servicio de metadatos, no llega solo:

```
$ curl -s -H Metadata:true \
    "http://169.254.169.254/metadata/scheduledevents?api-version=2020-07-01" | jq -r \
    '.Events[] | "\(.EventType) \(.NotBefore)"'
Preempt  Mon, 03 Aug 2026 11:42:07 GMT
```

Y la política de desalojo decide qué queda después:

```
Deallocate   la máquina se detiene y se conserva; sigue pagando el disco
             y puede volver a arrancar cuando haya capacidad
Delete       se elimina; no queda nada que pagar ni que recuperar
```

Para un conjunto de escalado, Delete es casi siempre lo correcto: máquinas desasignadas que nadie recuerda son una partida silenciosa en la factura y una lista de recursos huérfanos. Deallocate tiene sentido en una estación de trabajo con estado en su disco.

La forma sensata de usar excedente es mezclarlo, y para eso hace falta orquestación flexible:

```
$ az vmss create -g rg-app -n vmss-catalogo --orchestration-mode Flexible \
    --instance-count 4 --zones 1 2 3 \
    --priority Spot --eviction-policy Delete --max-price -1
```

```
base bajo demanda   capacidad mínima que sostiene el SLO aunque se vaya todo el excedente
pico en excedente   el resto
```

El dimensionado de la base no es una preferencia: es la carga que el servicio debe seguir atendiendo si el excedente desaparece entero en el mismo minuto, que es un escenario real cuando la región se queda sin capacidad. Y `--max-price -1` significa «acepto pagar hasta el precio bajo demanda», que evita un desalojo por precio además del desalojo por capacidad: fijar un máximo bajo añade una segunda causa de interrupción que casi nunca compensa.

Ejemplo trabajado

CloudShop traslada la capa de cómputo a Azure con la arquitectura de las clases 028 y 029. La traducción es correcta en el papel y produce cuatro problemas medibles.

Punto de partida:

```
base de datos   Standard_D2s_v5  + disco P40 (2 TiB, 7.500 IOPS), caché ReadWrite
catálogo        4 × Standard_D4s_v5 en conjunto de escalado uniforme
reparto         Load Balancer básico + Traffic Manager entre dos regiones
```

Problema 1 — se duplicó el disco y el rendimiento no se movió.

La base de datos se queda en 3.750 IOPS bajo carga, con la cola de disco creciendo.

```
$ az monitor metrics list --resource $VM_ID --metric "Data Disk IOPS Consumed Percentage" \
    --aggregation Maximum --interval PT1M --query "value[0].timeseries[0].data[-3:].maximum"
[49.8, 50.1, 49.9]
```

El disco está al 50 %. No es el disco. La fila de la máquina lo explica:

```
P40             7.500 IOPS
D2s_v5          3.750 IOPS sin caché   ← el techo real
```

Se corrige por el lado correcto y se aprovecha para dejar de pagar tamaño que no se usa:

```
máquina           D2s_v5 (3.750 IOPS)   D8s_v5 (12.800 IOPS)
disco             P40 · 2 TiB · ~259     P30 · 1 TiB · ~135
IOPS reales             3.750                 5.000
costo de disco     259 USD/mes            135 USD/mes
```

El volumen ocupado eran 640 GiB: la P40 se había comprado por sus IOPS, que la máquina nunca dejó alcanzar.

Problema 2 — un reinicio del anfitrión deja el registro de transacciones incoherente.

Durante un mantenimiento no planificado, el motor no arranca y reporta escrituras confirmadas que no están en disco.

```
$ az vm show -g rg-datos -n vm-db-01 \
    --query "storageProfile.dataDisks[].{lun:lun,cache:caching}" -o tsv
0   ReadWrite
1   ReadWrite
```

Ambos discos de datos con ReadWrite, heredado de la plantilla del disco del sistema. Se corrige por función:

```
$ az vm update -g rg-datos -n vm-db-01 --disk-caching 0=ReadOnly 1=None
```

```
LUN 0  datos      ReadOnly   lectura predominante, la caché ayuda
LUN 1  registro   None       ninguna escritura confirmada fuera del disco
```

Y se añade la consecuencia operativa que faltaba: la restauración se prueba, no se supone. La prueba de recuperación pasa a ejecutarse mensualmente con evidencia, siguiendo el criterio de la clase 030 — replicación no es copia de seguridad.

Problema 3 — la migración del balanceador deja el servicio inalcanzable.

Al sustituir el balanceador básico por el estándar, el frontal deja de responder. La configuración es idéntica.

```
$ az network watcher test-ip-flow --vm cat-01 --direction Inbound --protocol TCP \
    --local 10.20.4.11:8080 --remote 168.63.129.16:62000
Access: Deny   RuleName: defaultSecurityRules/DenyAllInBound
```

Dos cosas a la vez: el estándar es denegar por defecto y no había NSG que permitiera nada, porque el básico nunca lo había exigido. Además, no había salida.

```
$ az network nsg rule create -g rg-red --nsg-name nsg-app -n allow-lb-probe \
    --priority 100 --direction Inbound --access Allow --protocol '*' \
    --source-address-prefixes AzureLoadBalancer --destination-port-ranges '*'
$ az network nsg rule create -g rg-red --nsg-name nsg-app -n allow-appgw-8080 \
    --priority 110 --direction Inbound --access Allow --protocol Tcp \
    --source-address-prefixes 10.20.4.128/26 --destination-port-ranges 8080
$ az network vnet subnet update -g rg-red --vnet-name vnet-spoke-app -n snet-app \
    --nat-gateway natgw-spoke-app
```

Y se corrige la sonda, que apuntaba a /:

```
antes    GET /          200 mientras la base de datos no respondía
después  GET /readyz    comprueba base de datos y caché;
                        3 fallos consecutivos para retirar de rotación
```

Problema 4 — una conmutación de región que tardó nueve minutos.

Un simulacro corta la región primaria. Traffic Manager marca el destino como degradado en 40 s; el tráfico tarda mucho más en moverse.

```
t+0:00   se corta la región primaria
t+0:40   Traffic Manager marca el destino como degradado
t+1:10   el DNS ya devuelve la región secundaria
t+9:20   el último cliente deja de intentar contra la región caída
```

Los nueve minutos no son del servicio: son de la caché de DNS de los clientes, que no respetan el TTL de 60 s. Se antepone Front Door para el tráfico HTTP y se conserva Traffic Manager solo para el enrutamiento geográfico de un servicio que no es HTTP:

```
simulacro repetido con Front Door
t+0:00   se corta la región primaria
t+0:35   el borde deja de enviar al origen caído
t+0:41   error observado por el cliente: ninguno tras el reintento
```

Y una revisión de costo que no estaba en el plan. El conjunto de escalado uniforme no permitía mezclar prioridades. Al pasarlo a flexible:

```
orquestación                uniforme             flexible
capacidad base          4 × D4s_v5 bajo demanda   2 × D4s_v5 bajo demanda
pico                    escala en bajo demanda    hasta 6 × D4s_v5 excedente
costo mensual de cómputo      ~702 USD              ~412 USD
disco de base de datos         259 USD               135 USD
                          ■■■■■■■■■■            ■■■■■■■■■■
```

```
                961 USD                547 USD   (−43 %)
```

La base de dos instancias bajo demanda no es un número redondeado a ojo: es la capacidad que sostiene el percentil 95 de tráfico observado si todo el excedente desaparece en el mismo minuto. Se comprobó apagando las seis instancias de excedente a la vez y midiendo la latencia resultante:

```
con 2 bajo demanda y 0 excedente, a carga de p95
p99 de latencia   340 ms  (SLO: 500 ms)                                  ✓
```

La lección que esta clase traslada al resto de la parte: en Azure el cuello de botella rara vez está donde se compró la capacidad. El disco, la máquina y la caché tienen topes distintos, y el balanceador y el DNS tienen tiempos de reacción distintos. La cifra que importa siempre es el mínimo de la cadena, y es la que hay que medir antes de firmar un SLO.

Laboratorio guiado

Ejecuta desde la raíz:

```
python classes/part-03-azure-core-platform/040-virtual-machines-scale-sets-y-load-balancer/lab.py
```

El laboratorio selecciona el motor de práctica compute y produce labresult.json. El escenario, sus comprobaciones y el artefacto esperado corresponden a esta clase; no requiere credenciales y deja explícito qué debe revalidarse en un sandbox real.

1. Lee exercise.steps y formula una predicción antes de ejecutar.
2. Ejecuta la práctica y verifica todos los elementos de checks.
3. Provoca el caso de negativetest y explica la señal observada.
4. Materializa servicio-elastico-azure en evidence/ usando la plantilla indicada.
5. Para proveedor real, sigue sandbox.requires, registra costo y ejecuta destroy.

Evidencia esperada

El artefacto principal es una selección de capacidad justificada y observable. Además, la entrega debe incluir el comando ejecutado, la salida estructurada y una conclusión que no exceda lo observado.

Reto verificable

Construye servicio-elastico-azure para el caso CloudShop. Incluye una alternativa descartada, un supuesto que pueda falsarse, una prueba de fallo y una decisión de rollback.

■ Criterio de aceptación

- [ ] lab.py termina con código 0 y genera JSON válido.
- [ ] La entrega conecta al menos tres requisitos con mecanismos verificables.
- [ ] Existe una prueba positiva y una prueba negativa con evidencia.
- [ ] Seguridad, costo y operación aparecen como decisiones, no como anexos.
- [ ] Se declara una limitación y una condición que obligaría a revisar el diseño.
- [ ] Otra persona puede repetir el recorrido sin conocimiento tácito.

■■ Errores frecuentes

Síntoma	Causa probable	Corrección
Se aumenta el tamaño del disco y el rendimiento no cambia	El tope efectivo es el de la máquina virtual, no el del disco	Consulta las IOPS y MB/s sin caché del tamaño elegido; el caudal real es el mínimo de los dos y se corrige cambiando de tamaño.
La automatización falla porque /mnt o D: no existe	El tamaño elegido no lleva la letra d, así que no tiene disco temporal local	Elige un tamaño con d o retira la dependencia del disco temporal; nunca guardes ahí datos que deban sobrevivir.
Tras un reinicio no ordenado del anfitrión, el motor transaccional reporta escrituras perdidas	El disco de registro tenía caché ReadWrite, heredada de la plantilla del disco del sistema	None en el disco de registro, ReadOnly en los de datos con lectura predominante; ReadWrite solo en el disco del sistema.
Al migrar del balanceador básico al estándar, el servicio queda inalcanzable	El estándar deniega el tráfico entrante salvo que un NSG lo permita, y no proporciona salida por tener reglas de entrada	Añade reglas de NSG para la etiqueta AzureLoadBalancer y para el origen real, y declara la salida con NAT Gateway.
La regla de autoescalado por memoria nunca dispara	La memoria no es una métrica del anfitrión: requiere el agente de Azure Monitor	Instala el agente y verifica que la métrica llega antes de confiar la elasticidad a esa regla.
La conmutación entre regiones tarda minutos pese a que la sonda detecta el fallo en segundos	Traffic Manager conmuta por DNS y los clientes cachean más allá del TTL	Usa Front Door para el tráfico HTTP; reserva Traffic Manager para enrutamiento geográfico y destinos que no son HTTP.

■ Seguridad, ética y costo

Trabaja con cuentas propias o sandboxes autorizados. No publiques secretos, identificadores reales ni datos personales. Antes de crear recursos pagos define presupuesto, etiquetas y comando de destrucción; después verifica que no queden recursos huérfanos. Los ejemplos locales enseñan contratos, pero no certifican cumplimiento ni disponibilidad de producción.

■ Preguntas de comprobación

1. ¿Qué capacidades pierdes con un tamaño sin s y sin d, y cuándo lo descubrirías en cada caso?
2. Un disco P40 rinde 3.750 IOPS. ¿Qué medirías para decidir si el problema es el disco o la máquina?
3. ¿Qué aporta un conjunto de disponibilidad que las zonas no dan, y por qué no se pueden combinar?
4. ¿Por qué la orquestación flexible permite una estrategia de costo que la uniforme no?
5. El aviso de desalojo de excedente es de 30 segundos. ¿Qué diseños quedan descartados y cómo se dimensiona la capacidad base?

Referencias

- Microsoft (2025). Sizes for virtual machines in Azure — nomenclatura, letras aditivas y límites por tamaño. <https://learn.microsoft.com/en-us/azure/virtual-machines/sizes/overview>
- Microsoft (2025). Azure managed disk types — escalones de rendimiento, SSD premium v2 y Ultra Disk. <https://learn.microsoft.com/en-us/azure/virtual-machines/disks-types>
- Microsoft (2025). Virtual Machine Scale Sets orchestration modes — flexible frente a uniforme. <https://learn.microsoft.com/en-us/azure/virtual-machine-scale-sets/virtual-machine-scale-sets-orchestration-modes>
- Microsoft (2025). Load-balancing options — Load Balancer, Application Gateway, Front Door y Traffic Manager. <https://learn.microsoft.com/en-us/azure/architecture/guide/technology-choices/load-balancing-overview>
- Microsoft (2025). Azure Spot Virtual Machines — aviso de desalojo, políticas y precio máximo. <https://learn.microsoft.com/en-us/azure/virtual-machines/spot-vms>
- Documentación oficial vigente del servicio implementado; registra URL y fecha de consulta.

Descargar: Parte 03 en PDF · Recorrido de Azure en PDF · Manual integral

Evaluación — 040 Virtual Machines, Scale Sets y Load Balancer

Preguntas

1. Explica el problema que resuelve esta capacidad sin mencionar un proveedor.
2. Identifica una frontera de responsabilidad y su propietario.
3. Compara dos alternativas mediante confiabilidad, seguridad, costo y operación.
4. ¿Qué demuestra el laboratorio y qué no puede demostrar?
5. Propón un caso límite o una prueba de fallo.

Reto verificable

Entrega servicio-elastico-azure con diagrama o configuración, evidencia de ejecución, alternativa descartada, riesgo residual y criterio de rollback.

Criterio de aceptación

- [ ] Resultado reproducible por una segunda persona.
- [ ] Evidencia vinculada a cada afirmación técnica.
- [ ] Prueba negativa o de fallo incluida.
- [ ] Costos y permisos expresados con unidades y alcance.
- [ ] Limitaciones declaradas sin presentar la simulación como producción.

Escala

Nivel	Evidencia
A — competente	Cumple todo y defiende trade-offs con datos.
B — en desarrollo	Cumple lo esencial; falta profundidad en una dimensión.
C — inicial	Artefacto parcial o conclusión sin evidencia suficiente.
0	Sin entrega reproducible.

041 — Blob Storage, Files, redundancia y lifecycle

Parte: 03 — Azure: plataforma esencial Nivel: intermedio · Horas estimadas: 4 Laboratorio: storage · Estado: EXECUTABLECORE

Propósito

Operar el almacenamiento de Azure sabiendo que la unidad de configuración no es el contenedor sino la cuenta, y que casi todo lo que protege los datos son interruptores separados que hay que activar uno a uno. La clase 030 dejó el criterio —versionado, clases, público bloqueado, replicación no es copia—; aquí cambia dónde vive cada control, y aparecen tres decisiones que en AWS no existen: la región emparejada no se elige, la conmutación la inicias tú, y borrar un contenedor no es lo mismo que borrar sus blobs.

Resultados de aprendizaje

Al finalizar podrás:

1. Dimensionar cuentas de almacenamiento sabiendo qué límites son de la cuenta y qué configuraciones arrastra todo lo que hay dentro.
2. Elegir redundancia distinguiendo lo que protege de lo que cuesta, y sabiendo qué ocurre con la cuenta después de una conmutación.
3. Calcular el punto de equilibrio de un nivel de acceso frío antes de mover datos a él.
4. Configurar los interruptores de protección —versionado, eliminación temporal de blob y de contenedor, inmutabilidad— y demostrar cada uno con una prueba de recuperación.
5. Sustituir claves de cuenta y firmas irrevocables por identidad y firmas de delegación de usuario.

Conceptos centrales

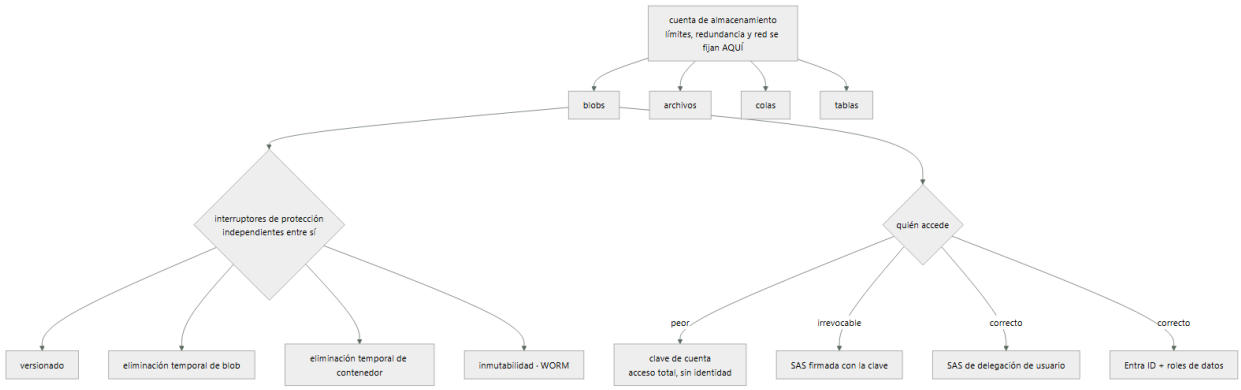
Concepto	Comprensión verificable
cuenta de almacenamiento	Unidad de configuración, facturación y límites. Contiene blobs, archivos, colas y tablas a la vez, y su nombre es una etiqueta DNS global: 3 a 24 caracteres, minúsculas y dígitos, sin guiones.
región emparejada	Destino de la replicación geográfica, asignado por Microsoft y no elegible. Si el dato no puede salir de una jurisdicción, esa asignación puede descartar la redundancia geográfica.
conmutación de cuenta	Cambio de la región principal a la secundaria. La inicia el cliente, pierde lo no replicado y deja la cuenta como LRS, lo que obliga a reconfigurar la redundancia después.
eliminación temporal	Dos interruptores independientes: uno para blobs y otro para contenedores. Activar el primero no protege de borrar el contenedor entero.
firma de delegación de usuario	SAS firmada con una clave derivada de Entra ID en vez de con la clave de la cuenta. Caduca en 7 días como máximo y se puede revocar sin romper el resto.
regla de ciclo de vida	Directiva JSON de la cuenta que mueve o borra por antigüedad. Si solo apunta a baseBlob, las versiones e instantáneas se quedan y la factura no baja.

Modelo mental

En Azure, identidad y jerarquía organizacional preceden al recurso: tenant, management group, suscripción y resource group determinan el alcance de control.

Aplicado a esta clase, separa siempre cuatro planos: intención (qué necesita el usuario), configuración (qué declaramos), estado observado (qué existe de verdad) y evidencia (cómo sabemos que cumple). Confundirlos produce diseños que se ven correctos en un diagrama pero fallan al operar.

■ Flujo de razonamiento



Desarrollo

1. La cuenta es la unidad, y arrastra todo lo que hay dentro

En AWS el bucket es la unidad: sus políticas, su cifrado y su bloqueo de acceso público valen para él y para nadie más. En Azure la unidad es la cuenta de almacenamiento, y dentro de ella conviven cuatro servicios:

```
cuenta stcloudshopprod
  ■■ blobs      objetos
  ■■ archivos   recursos compartidos SMB y NFS
  ■■ colas      mensajería simple
  ■■ tablas     clave-valor
```

Lo que se decide en la cuenta y no se puede matizar por contenedor:

```
redundancia (LRS, ZRS, GRS, GZRS)
reglas de red y punto de conexión privado
versión mínima de TLS
acceso con clave compartida habilitado o no
acceso público anónimo permitido o no
nivel de acceso por defecto
cifrado y clave gestionada por el cliente
```

Esa lista es la razón por la que una cuenta para todo es un error de diseño, aunque parezca ordenado. Datos con requisitos distintos de residencia, de retención o de exposición terminan compartiendo la misma configuración, y el único modo de separarlos después es copiar terabytes.

Además la cuenta tiene límites propios que un bucket no tiene:

```
capacidad                    5 PiB (ampliable por solicitud)
frecuencia de solicitudes    ~20.000 por segundo
salida en la región primaria  límite por cuenta, no por contenedor
```

Un servicio ruidoso puede limitar a los demás inquilinos de la misma cuenta. El síntoma es un 503 ServerBusy intermitente en un servicio que no cambió nada, causado por otro que sí. La respuesta a x-ms-request-id y el contador Throttling Error lo identifican en un minuto; el arreglo es repartir en varias cuentas, no reintentar más fuerte.

Y el nombre importa más de lo que parece: es una etiqueta DNS global, de 3 a 24 caracteres, minúsculas y dígitos, sin guiones ni puntos. Una convención de nombres que incluya guiones —muy común al llegar de AWS— no se puede aplicar aquí, así que conviene fijarla antes de crear la primera:

```
st  cloudshop  prod  weu  01
■■  ■■■■■■■■  ■■■■  ■■■  ■■ secuencia
■■  ■■■■■■■■  ■■■■  ■■■■■■■■■ región
■■  ■■■■■■■■  ■■■■■■■■■■■■■■■ entorno
■■  ■■■■■■■■■■■■■■■■■■■■■■■■■■■■ sistema
■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■ tipo de recurso
```

2. Redundancia: la región no se elige y la conmutación la inicias tú

Cinco opciones, dos ejes: cuántas copias y dónde.

	Copias	Sobrevive a	Lectura en la secundaria
LRS	3 en un centro de datos	Fallo de disco o bastidor	—
ZRS	3 en tres zonas	Caída de un centro de datos	—
GRS	3 locales + 3 remotas	Caída de la región	No
GZRS	3 zonas + 3 remotas	Caída de la región, con zonas en la principal	No
RA-GRS / RA-GZRS	Igual	Igual	Sí, con un nombre -secondary

Tres hechos cambian cómo se diseña con esto, y ninguno tiene equivalente directo en la replicación entre regiones de AWS:

La región emparejada la asigna Microsoft. No se elige el destino. Para una carga sujeta a residencia de datos, eso puede descalificar la redundancia geográfica entera: si el par asignado está fuera de la jurisdicción aprobada, la opción correcta es ZRS más una replicación de objetos explícita hacia una cuenta en la región que sí lo está. Es más trabajo y es la única que se puede documentar ante un auditor.

La replicación es asíncrona y su retraso no es un compromiso de servicio. El indicador es Last Sync Time, y es la única fuente honesta del RPO real:

```
$ az storage account show -n stcloudshopprod -g rg-datos \
    --expand geoReplicationStats --query "geoReplicationStats.lastSyncTime" -o tsv
2026-08-01T09:47:12Z
```

Todo lo escrito después de esa marca no está en la secundaria. Un RPO declarado de 15 minutos que no se contrasta con esta métrica es una aspiración, no un dato.

La conmutación no es automática y deja la cuenta en LRS. Nadie la ejecuta por ti: es una operación que inicia el cliente, y después de completarse la cuenta queda como LRS en la que era región secundaria. Es decir:

```
antes del incidente   GRS   principal weu, secundaria nue
durante la conmutación      se pierde lo posterior a lastSyncTime
después                LRS   una sola región, sin redundancia geográfica
acción pendiente             reconfigurar a GRS y esperar la replicación inicial
```

Ese último renglón es el que sorprende: justo después de un incidente regional, la plataforma está menos protegida que antes, y volver a estarlo tarda lo que tarde copiar todo el volumen. Un plan de recuperación que no incluya ese paso está incompleto.

Y el cambio entre modelos tampoco es un interruptor: pasar de LRS a ZRS exige una conversión —solicitada o mediante copia manual—, con ventana y sin garantía de instantaneidad. Conviene elegir bien al crear la cuenta, porque corregirlo después cuesta tiempo y tráfico.

3. Niveles de acceso: calcula el punto de equilibrio antes de mover nada

Cuatro niveles, con mínimos de permanencia que son la mitad de la decisión:

	Almacenamiento aprox.	Permanencia mínima	Recuperación
Caliente	0,0184 USD/GiB/mes	—	Inmediata
Frío (cool)	0,0100	30 días	Inmediata, con cargo por GiB
Muy frío (cold)	0,0036	90 días	Inmediata, con cargo mayor
Archivo	0,0010	180 días	Hay que rehidratar: hasta 15 h

La clase 030 ya estableció que el ahorro por nivel a veces no lo es. Aquí está la aritmética con las cifras de Azure, para 1 TiB:

```
ahorro mensual al pasar de caliente a frío
  (0,0184 − 0,0100) × 1.024 GiB = 8,60 USD/mes

coste de leer ese TiB una vez al mes
  1.024 GiB × 0,01 USD/GiB de recuperación = 10,24 USD
```

Leer el conjunto entero una sola vez al mes ya destruye el ahorro. El punto de equilibrio está en releer menos del 84 % del volumen cada mes; por encima, el nivel frío sale más caro que el caliente además de más lento.

Y encima está la penalización por eliminación temprana, que se cobra prorrateada: borrar a los 10 días un blob en nivel frío factura los 20 días restantes de los 30 mínimos. En un flujo con mucha rotación —archivos temporales, informes diarios que se sustituyen— mover al nivel frío puede multiplicar la factura en lugar de reducirla.

El nivel archivo es una decisión distinta, no un escalón más: el blob no se puede leer. Hay que rehidratarlo, y eso tiene tiempo y precio:

```
prioridad estándar   hasta 15 h
prioridad alta       menos de 1 h para blobs de menos de 10 GiB, con sobrecoste
```

Quince horas es incompatible con casi cualquier RTO de servicio y perfectamente compatible con una obligación legal de conservar siete años. Esa es la frontera: archivo es para lo que hay que guardar, no para lo que hay que poder leer.

Las reglas de ciclo de vida automatizan el movimiento, y tienen dos comportamientos que hay que anticipar:

```
{
  "rules": [{
    "name": "facturas",
    "type": "Lifecycle",
    "definition": {
      "filters": {"blobTypes": ["blockBlob"], "prefixMatch": ["facturas/"]},
      "actions": {
        "baseBlob": {
          "tierToCool":    {"daysAfterModificationGreaterThan": 30},
          "tierToArchive": {"daysAfterModificationGreaterThan": 365}
        },
        "version":  {"delete": {"daysAfterCreationGreaterThan": 90}},
        "snapshot": {"delete": {"daysAfterCreationGreaterThan": 90}}
      }
    }
  }]
}
```

Primero: las secciones version y snapshot son las que casi nadie escribe, y son la causa de que la factura no baje después de una limpieza. Con versionado activo, borrar un blob no libera nada: crea una versión. Es el mismo mecanismo que el marcador de borrado de la clase 030, con otro nombre y la misma consecuencia contable.

Segundo: la directiva se evalúa una vez al día y la primera ejecución puede tardar hasta 48 horas. No es un fallo de configuración, es su cadencia; comprobarla a los diez minutos y volver a escribirla es la forma habitual de perder una tarde.

Y daysAfterLastAccessTimeGreaterThan —mover por último acceso en vez de por última modificación— exige activar el seguimiento de acceso, que tiene su propio costo por transacción. En un contenedor con lecturas muy frecuentes puede costar más que lo que ahorra el cambio de nivel.

4. Cuatro interruptores para proteger lo mismo, y ninguno cubre al otro

Aquí está la trampa más cara de esta clase. Los mecanismos de protección son independientes, y activar el evidente deja el hueco:

```
versionado                        conserva una versión por cada sobrescritura o borrado
eliminación temporal de blob      recupera blobs borrados durante N días
eliminación temporal de contenedor recupera CONTENEDORES borrados durante N días
restauración a un punto anterior  devuelve un rango de blobs a un instante
inmutabilidad                     impide modificar o borrar durante un plazo
```

La eliminación temporal de blob no protege de borrar el contenedor. Es literal: con eliminación temporal de blob a 7 días y eliminación temporal de contenedor desactivada, un az storage container delete se lleva todo el contenido de forma irreversible. El interruptor que parecía cubrirlo cubría otra cosa.

La configuración completa, y su dependencia entre piezas:

```
$ az storage account blob-service-properties update -n stcloudshopprod -g rg-datos \
    --enable-versioning true \
    --enable-delete-retention true --delete-retention-days 30 \
    --enable-container-delete-retention true --container-delete-retention-days 30 \
    --enable-change-feed true \
    --enable-restore-policy true --restore-days 29
```

La restauración a un punto anterior exige versionado, fuente de cambios y eliminación temporal de blob, y su ventana debe ser menor que la de retención. Activarla suelta da un error que no explica cuál de las tres falta.

La inmutabilidad es otra cosa y responde a otro requisito. Se aplica a contenedor o a versión, en dos formas:

```
retención por tiempo   ningún borrado ni modificación durante N días
retención legal        indefinida hasta que se retire la etiqueta
```

Y el matiz que la hace útil ante un auditor: mientras la directiva está bloqueada, el plazo se puede alargar y no se puede acortar ni eliminar — tampoco por el propietario de la suscripción, tampoco por un administrador global. Es la diferencia entre una política y un control: la política la cambia quien tiene permiso, el control no lo cambia nadie. La contrapartida hay que aceptarla de antemano: la cuenta no se puede borrar mientras haya datos bajo retención, ni siquiera si se creó por error con un plazo de siete años.

Una última pieza que la clase 030 dejó dicha y aquí tiene dos interruptores en vez de uno: el acceso público anónimo se controla en la cuenta y en el contenedor, y basta con que el de la cuenta esté abierto y alguien marque un contenedor como público para exponerlo:

```
$ az storage account update -n stcloudshopprod -g rg-datos \
    --allow-blob-public-access false
```

Con eso, ningún contenedor puede ser público aunque su configuración diga que lo es. Es el equivalente al bloqueo en el nivel de la cuenta, y es el que hay que auditar: comprobar contenedor por contenedor es trabajo que se deshace solo.

5. Claves, firmas y la que no se puede revocar

Cuatro formas de acceder, ordenadas de peor a mejor:

Clave de cuenta. Dos claves, rotables. Conceden acceso total a los cuatro servicios de la cuenta y no llevan identidad: el registro dice qué se hizo, no quién. Una clave filtrada es la cuenta entera. La instrucción es desactivarlas:

```
$ az storage account update -n stcloudshopprod -g rg-datos --allow-shared-key-access false
```

SAS firmada con la clave de cuenta. Es cómoda y tiene un defecto que define la decisión: no se puede revocar. Una firma emitida con vencimiento en 2029 sigue siendo válida hasta 2029, y la única forma de invalidarla es rotar la clave con la que se firmó — lo que rompe a la vez a todos los demás que la usan. El repositorio se limpia, la firma sigue funcionando.

Un paliativo parcial es la directiva de acceso almacenada: la firma referencia una directiva del contenedor y borrar la directiva invalida las firmas asociadas. Requiere haberlo previsto al emitirlas.

SAS de delegación de usuario. Se firma con una clave derivada de Entra ID, no con la clave de cuenta:

```
$ az storage blob generate-sas --account-name stcloudshopprod \
    --container-name facturas --name 2026-07.pdf \
    --permissions r --expiry 2026-08-01T18:00Z --as-user --auth-mode login -o tsv
```

Caduca en 7 días como máximo, hereda los permisos del principal que la emite —no puede conceder más de lo que ese principal tiene— y se revoca retirando el rol o revocando las claves de delegación, sin tocar nada más. Es la opción correcta para dar acceso temporal a un objeto concreto.

Entra ID con roles de datos. Es la vía por defecto para servicios. Y aquí vuelve, con consecuencias, la distinción de la clase 038:

```
Storage Account Contributor       gestiona la cuenta y LEE SUS CLAVES
                                  → sin dataActions, pero con la clave se accede a todo
Storage Blob Data Reader          lee los datos, no gestiona la cuenta
Storage Blob Data Contributor     lee y escribe los datos
```

La primera fila es una escalada disfrazada de permiso administrativo: quien puede leer las claves puede hacer todo lo que hacen las claves, sin necesitar ningún rol de datos y sin dejar rastro de identidad. Con allowSharedKeyAccess desactivado, ese camino se cierra: la clave existe y no sirve para autenticarse.

Y para Azure Files conviene anticipar una decisión de identidad, porque no es simétrica con los blobs:

	Estándar	Premium
Soporte	HDD, cobro por transacción	SSD, capacidad aprovisionada

	Estándar	Premium
Rendimiento	Variable	IOPS en función de los GiB aprovisionados
NFS	No	Sí, con punto de conexión privado
Se factura	Lo usado	Lo aprovisionado

La última fila es la que produce facturas inesperadas: un recurso compartido premium de 5 TiB aprovisionado y 300 GiB usados factura 5 TiB. Y para SMB con identidad de dominio hace falta integrar con AD DS o con Entra Domain Services; sin eso, el acceso es con clave de cuenta, que es justo lo que se acaba de desactivar. Conviene resolverlo antes de migrar un recurso compartido de archivos, no después.

Ejemplo trabajado

CloudShop lleva a Azure el almacenamiento diseñado en la clase 030. La configuración inicial parece completa: versionado activo, replicación geográfica y niveles de acceso. Cuatro sucesos demuestran que ninguna de las tres cosas hacía lo que el equipo creía.

Estado inicial:

```
cuenta stcloudshopprod  GRS  ·  una sola cuenta para facturas, medios y registros
versionado                     activo
eliminación temporal de blob   7 días
eliminación temporal de contenedor  desactivada
acceso con clave compartida    habilitado
```

Suceso 1 — un contenedor borrado por error se lleva 412 GiB.

Un script de limpieza mal parametrizado borra facturas en lugar de facturas-tmp.

```
$ az storage container restore -n facturas --account-name stcloudshopprod
ERROR: Container soft delete is not enabled for this account.
```

La eliminación temporal de blob estaba activa y no cubría esto. No hay recuperación. Se reconstruye desde el sistema de facturación —tres días de trabajo— y se cierran los cuatro interruptores, no uno:

```
$ az storage account blob-service-properties update -n stcloudshopprod -g rg-datos \
    --enable-versioning true \
    --enable-delete-retention true --delete-retention-days 30 \
    --enable-container-delete-retention true --container-delete-retention-days 30 \
    --enable-change-feed true --enable-restore-policy true --restore-days 29
```

Y se prueba la recuperación en lugar de suponerla:

```
$ az storage container delete -n prueba-recuperacion --account-name stcloudshopprod
$ az storage container restore -n prueba-recuperacion --account-name stcloudshopprod
$ az storage blob list -c prueba-recuperacion --account-name stcloudshopprod -o tsv | wc -l
50                                                                              ✓
```

Además, las facturas tienen obligación legal de siete años, así que se añade inmutabilidad bloqueada — con la consecuencia aceptada por escrito de que la cuenta ya no se podrá eliminar durante ese plazo.

Suceso 2 — se borran 3 TiB y la factura no baja.

```
datos visibles      1,8 TiB
facturado           4,9 TiB
```

```
$ az storage account show -n stcloudshopprod -g rg-datos \
    --query "{v:blobServiceProperties.isVersioningEnabled}"
$ az storage account management-policy show --account-name stcloudshopprod -g rg-datos \
    --query "policy.rules[0].definition.actions" -o json
{"baseBlob": {"tierToCool": {"daysAfterModificationGreaterThan": 30}}}
```

La regla solo tocaba baseBlob. Con versionado activo, cada borrado había creado una versión que nadie retiraba nunca. Es el marcador de borrado de la clase 030 con otro nombre. Se completa la directiva con version y snapshot, y se espera a su cadencia diaria en lugar de reescribirla:

```
facturado                4,9 TiB     1,9 TiB
costo mensual de blobs   92,20 USD   35,80 USD
```

Suceso 3 — una firma en una aplicación móvil, válida hasta 2029.

Una revisión de seguridad encuentra una SAS incrustada en el paquete de la aplicación.

```
se=2029-01-01T00%3A00%3A00Z&sp=racwdl&sr=c&sig=…
```

Permisos de lectura, escritura, borrado y listado sobre el contenedor entero, durante tres años. Firmada con la clave de la cuenta, así que revocarla exige rotar la clave, y la clave la usan seis servicios más.

La secuencia, en este orden y no en otro:

```
1. migrar los seis servicios a identidad administrada + roles de datos
2. verificar que ninguno usa ya la clave  (métrica de autenticación por clave = 0)
3. rotar las dos claves  → la firma de 2029 deja de valer
4. desactivar el acceso con clave compartida
5. la aplicación móvil pasa a pedir una SAS de delegación de usuario
   al backend, de 15 minutos y para un solo blob
```

```
$ az storage account update -n stcloudshopprod -g rg-datos --allow-shared-key-access false
$ az storage blob list -c facturas --account-name stcloudshopprod \
    --account-key $CLAVE_ANTIGUA -o tsv
KeyBasedAuthenticationNotPermitted                                  ✓
```

Suceso 4 — el simulacro de región revela dos supuestos falsos.

El plan decía «GRS conmuta a la región emparejada». El simulacro mide otra cosa:

```
$ az storage account show -n stcloudshopprod -g rg-datos --expand geoReplicationStats \
    --query "geoReplicationStats.{estado:status,ultima:lastSyncTime}" -o tsv
Live   2026-08-01T09:36:00Z
```

Con la hora del corte a las 09:47, el RPO real de ese momento era de 11 minutos, no de cero. Y la conmutación no ocurre sola: hay que ejecutarla, y al terminar la cuenta queda como LRS.

```
conmutación                          automática      manual, ~1 h
RPO                                    0 min          11 min
redundancia después de conmutar         GRS            LRS
```

Además, la región emparejada asignada estaba fuera de la jurisdicción aprobada para los datos de facturación. La corrección separa lo que nunca debió compartir cuenta:

```
facturas   ■                              GZRS + replicación de objetos
medios     ■■ una cuenta GRS                a una región elegida y aprobada
registros  ■                  medios:      GRS, ciclo de vida a frío a 30 días
                              registros:   LRS, borrado a 90 días
```

Resumen del rediseño:

```
cuentas de almacenamiento                    1            3
interruptores de protección activos          2 de 5       5 de 5
recuperación de contenedor probada           no           sí, mensual
acceso con clave compartida                  sí           no
firmas irrevocables en circulación           1            0
RPO documentado y medido                     no           sí, 11 min
datos facturados                           4,9 TiB      1,9 TiB
costo mensual de almacenamiento           92,20 USD    41,60 USD
```

La lección que esta clase traslada al resto de la parte: en Azure, la protección de datos no es una propiedad que se activa sino una lista de interruptores independientes, y ninguno de ellos cubre el hueco del de al lado. La única forma honesta de saber cuáles están activos es borrar algo a propósito y recuperarlo.

Laboratorio guiado

Ejecuta desde la raíz:

```
python classes/part-03-azure-core-platform/041-blob-storage-files-redundancia-y-lifecycle/lab.py
```

El laboratorio selecciona el motor de práctica storage y produce labresult.json. El escenario, sus comprobaciones y el artefacto esperado corresponden a esta clase; no requiere credenciales y deja explícito qué debe revalidarse en un sandbox real.

1. Lee exercise.steps y formula una predicción antes de ejecutar.
2. Ejecuta la práctica y verifica todos los elementos de checks.
3. Provoca el caso de negativetest y explica la señal observada.

4. Materializa storage-gobernado-azure en evidence/ usando la plantilla indicada.
5. Para proveedor real, sigue sandbox.requires, registra costo y ejecuta destroy.

Evidencia esperada

El artefacto principal es una política de durabilidad, acceso, retención y costo. Además, la entrega debe incluir el comando ejecutado, la salida estructurada y una conclusión que no exceda lo observado.

Reto verificable

Construye storage-gobernado-azure para el caso CloudShop. Incluye una alternativa descartada, un supuesto que pueda falsarse, una prueba de fallo y una decisión de rollback.

■ Criterio de aceptación

- [ ] lab.py termina con código 0 y genera JSON válido.
- [ ] La entrega conecta al menos tres requisitos con mecanismos verificables.
- [ ] Existe una prueba positiva y una prueba negativa con evidencia.
- [ ] Seguridad, costo y operación aparecen como decisiones, no como anexos.
- [ ] Se declara una limitación y una condición que obligaría a revisar el diseño.
- [ ] Otra persona puede repetir el recorrido sin conocimiento tácito.

■■ Errores frecuentes

Síntoma	Causa probable	Corrección
Se borra un contenedor y no hay forma de recuperarlo pese a tener eliminación temporal	La eliminación temporal de blob y la de contenedor son interruptores distintos	Activa ambos y demuéstralo borrando y restaurando un contenedor de prueba cada mes.
Se liberan terabytes y la factura no cambia	Con versionado activo, borrar crea una versión, y la regla de ciclo de vida solo apuntaba a baseBlob	Añade las secciones version y snapshot a la directiva y espera a su evaluación diaria.
Una SAS filtrada no se puede revocar sin romper otros servicios	Se firmó con la clave de la cuenta, así que solo se invalida rotando esa clave	Migra los servicios a identidad, rota las claves, desactiva el acceso con clave compartida y emite SAS de delegación de usuario.
Tras una conmutación regional la cuenta ya no tiene redundancia geográfica	La conmutación deja la cuenta como LRS en la región secundaria	Incluye en el plan de recuperación el paso de reconfigurar la redundancia y el tiempo de la replicación inicial.
Mover datos al nivel frío aumenta el costo en vez de reducirlo	Los cargos por recuperación y la penalización por eliminación temprana superan el ahorro de almacenamiento	Calcula el punto de equilibrio con el volumen releído al mes antes de aplicar la regla.
Un principal sin roles de datos accede a todos los blobs	Tenía un rol de gestión que permite leer las claves de la cuenta	Desactiva allowSharedKeyAccess: la clave sigue existiendo y deja de servir para autenticarse.

■ Seguridad, ética y costo

Trabaja con cuentas propias o sandboxes autorizados. No publiques secretos, identificadores reales ni datos personales. Antes de crear recursos pagos define presupuesto, etiquetas y comando de destrucción; después verifica que no queden recursos huérfanos. Los ejemplos locales enseñan contratos, pero no certifican cumplimiento ni disponibilidad de producción.

■ Preguntas de comprobación

1. ¿Qué configuraciones se deciden en la cuenta y no se pueden matizar por contenedor, y qué implica eso para el diseño?

2. ¿Por qué la redundancia geográfica puede quedar descartada por un requisito de residencia de datos?
3. Con 1 TiB, ¿a partir de qué volumen releído al mes deja de compensar el nivel frío?
4. Tienes versionado y eliminación temporal de blob activos. ¿De qué pérdida NO estás protegido?
5. ¿Qué diferencia operativa concreta hay entre una SAS firmada con la clave de cuenta y una de delegación de usuario?

Referencias

• Microsoft (2025). Azure Storage redundancy — LRS a GZRS, región emparejada y lectura en la secundaria. <https://learn.microsoft.com/en-us/azure/storage/common/storage-redundancy>

• Microsoft (2025). Disaster recovery and storage account failover — conmutación iniciada por el cliente, lastSyncTime y estado posterior. <https://learn.microsoft.com/en-us/azure/storage/common/storage-disaster-recovery-guidance>

• Microsoft (2025). Access tiers for blob data — niveles, permanencia mínima y rehidratación desde archivo. <https://learn.microsoft.com/en-us/azure/storage/blobs/access-tiers-overview>

• Microsoft (2025). Data protection overview — versionado, eliminación temporal, restauración a un punto anterior e inmutabilidad. <https://learn.microsoft.com/en-us/azure/storage/blobs/data-protection-overview>

• Microsoft (2025). Grant limited access with shared access signatures — tipos de SAS y delegación de usuario. <https://learn.microsoft.com/en-us/azure/storage/common/storage-sas-overview>

• Documentación oficial vigente del servicio implementado; registra URL y fecha de consulta.

Descargar: Parte 03 en PDF · Recorrido de Azure en PDF · Manual integral

Anterior	Índice	Siguiente
← 040 · Virtual Machines, Scale Sets y Load Balancer	Parte 03 · Programa	042 · Azure SQL, Cosmos DB y Azure Cache for Redis →

Evaluación — 041 Blob Storage, Files, redundancia y lifecycle

Preguntas

1. Explica el problema que resuelve esta capacidad sin mencionar un proveedor.
2. Identifica una frontera de responsabilidad y su propietario.
3. Compara dos alternativas mediante confiabilidad, seguridad, costo y operación.
4. ¿Qué demuestra el laboratorio y qué no puede demostrar?
5. Propón un caso límite o una prueba de fallo.

Reto verificable

Entrega storage-gobernado-azure con diagrama o configuración, evidencia de ejecución, alternativa descartada, riesgo residual y criterio de rollback.

Criterio de aceptación

• [ ] Resultado reproducible por una segunda persona.

• [ ] Evidencia vinculada a cada afirmación técnica.

• [ ] Prueba negativa o de fallo incluida.

• [ ] Costos y permisos expresados con unidades y alcance.

• [ ] Limitaciones declaradas sin presentar la simulación como producción.

Escala

Nivel	Evidencia
A — competente	Cumple todo y defiende trade-offs con datos.
B — en desarrollo	Cumple lo esencial; falta profundidad en una dimensión.
C — inicial	Artefacto parcial o conclusión sin evidencia suficiente.
0	Sin entrega reproducible.

042 — Azure SQL, Cosmos DB y Azure Cache for Redis

Parte: 03 — Azure: plataforma esencial Nivel: intermedio · Horas estimadas: 4 Laboratorio: data · Estado: EXECUTABLECORE

Propósito

Elegir y operar los motores de datos de Azure con la disciplina de la clase 031 —los patrones de acceso primero— y con tres precisiones que solo aparecen aquí: el modelo de compra decide si vas a poder diagnosticar, la unidad de solicitud de Cosmos DB pone precio a cada consulta en la cabecera de la respuesta, y dos decisiones de esta clase son puertas de un solo sentido que no se corrigen editando una plantilla.

Resultados de aprendizaje

Al finalizar podrás:

1. Diagnosticar una base de datos lenta identificando cuál de los cinco límites se agotó, y no solo la CPU.
2. Elegir modelo de compra y nivel de servicio a partir de la latencia de E/S exigida y de la capacidad de diagnóstico.
3. Diseñar una clave de partición sabiendo que es inmutable y que la partición lógica tiene un tope duro de 20 GB.
4. Calcular cuándo el escalado automático de Cosmos DB cuesta más que aprovisionar el pico.
5. Configurar una caché con política de expulsión y reserva de memoria que no se bloquee al llenarse.

Conceptos centrales

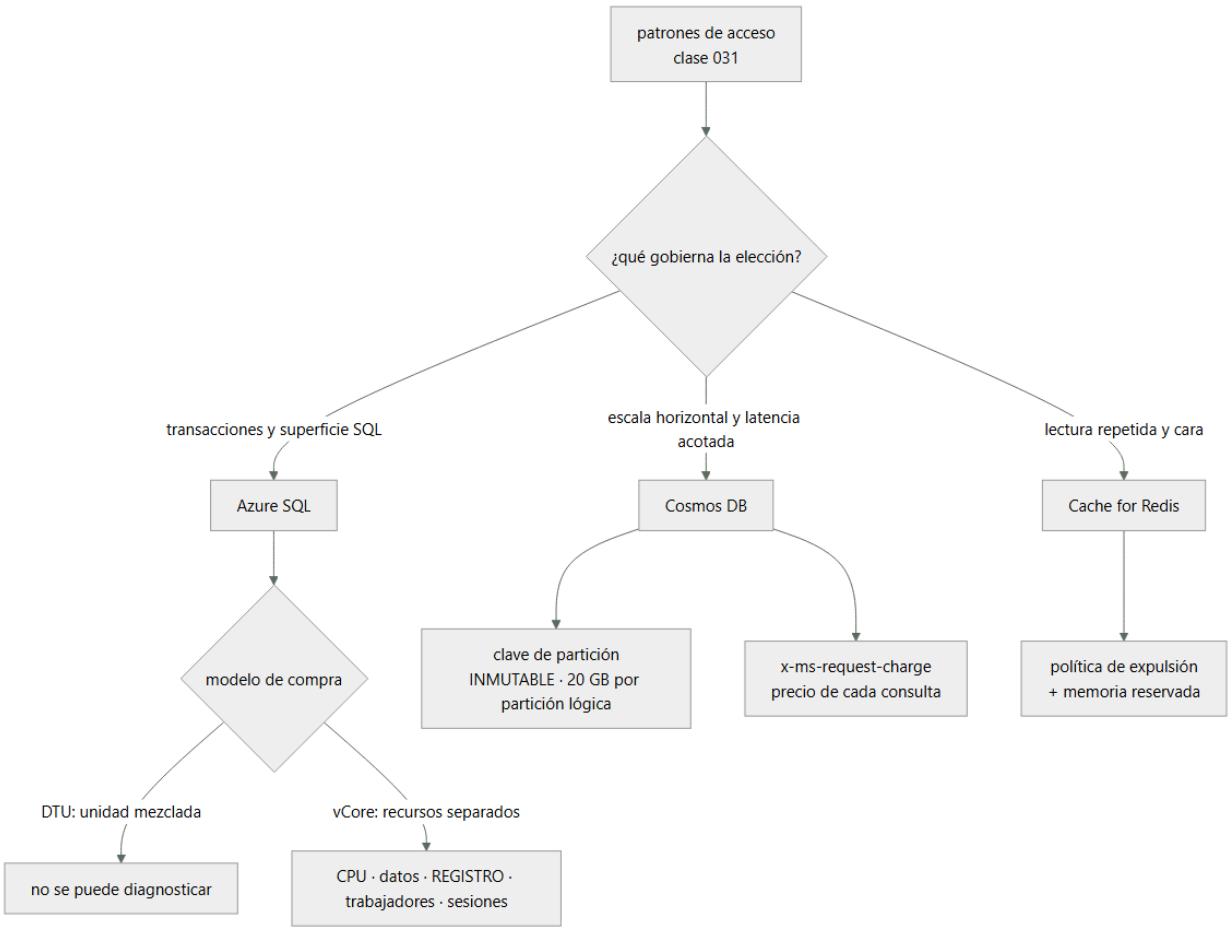
Concepto	Comprensión verificable
DTU frente a vCore	La DTU es una unidad mezclada de CPU, memoria y E/S: cuando se agota, no dice cuál. El modelo vCore expone los recursos por separado y es el único que permite diagnosticar.
gobernador de velocidad de registro	Límite de MB/s de escritura en el registro de transacciones, independiente de la CPU. Es el techo que más se alcanza y el que menos se mira.
unidad de solicitud (RU)	Moneda única de Cosmos DB: toda lectura, escritura y consulta cuesta RU. La cabecera x-ms-request-charge devuelve el precio exacto de cada operación.
partición lógica	Conjunto de documentos con la misma clave de partición. Tiene un tope duro de 20 GB: al alcanzarlo las escrituras fallan, no se ralentizan.
coherencia de sesión	Nivel por defecto de Cosmos DB: garantiza leer lo que tú escribiste mientras se propague el testigo de sesión. Si el cliente se comparte entre usuarios sin propagarlo, la garantía desaparece en silencio.
política de expulsión	Regla que decide qué claves se descartan cuando la caché se llena. volatile-lru —la de por defecto— solo expulsa claves con vencimiento: sin TTL, la caché se llena y deja de aceptar escrituras.

Modelo mental

En Azure, identidad y jerarquía organizacional preceden al recurso: tenant, management group, suscripción y resource group determinan el alcance de control.

Aplicado a esta clase, separa siempre cuatro planos: intención (qué necesita el usuario), configuración (qué declaramos), estado observado (qué existe de verdad) y evidencia (cómo sabemos que cumple). Confundirlos produce diseños que se ven correctos en un diagrama pero fallan al operar.

■ Flujo de razonamiento



Desarrollo

1. El modelo de compra decide si vas a poder diagnosticar

Azure SQL se compra de dos formas, y la elección no es de precio sino de observabilidad.

La DTU es una unidad mezclada de CPU, memoria y E/S. Cuando una base de datos S3 «está al 100 %», el panel no dice al 100 % de qué. Se sube de nivel, mejora un poco, y a las tres semanas vuelve el mismo síntoma: se compró más de todo para resolver la escasez de uno.

El modelo vCore expone los recursos por separado, y con él la vista que resuelve la mayoría de los casos:

```
SELECT TOP 5 end_time,
       avg_cpu_percent, avg_data_io_percent, avg_log_write_percent,
       max_worker_percent, max_session_percent
FROM sys.dm_db_resource_stats ORDER BY end_time DESC;
```

```
end_time   cpu   data_io   log_write   worker   session
21:14:00   18,2    31,0      100,0       12,4      3,1
```

CPU al 18 % y registro al 100 %. Ese es el patrón que casi nadie busca. El gobernador de velocidad de registro limita los MB/s que pueden confirmarse en el registro de transacciones, y es independiente del resto. Un proceso de importación por lotes lo satura mientras todo lo demás parece ocioso, y el tipo de espera lo confirma:

```
SELECT wait_type, wait_time_ms FROM sys.dm_os_wait_stats
WHERE wait_type LIKE 'LOG_RATE_GOVERNOR' OR wait_type = 'WRITELOG';
```

Las correcciones no pasan por más CPU: reducir el registro generado —cargas mínimamente registradas, lotes más pequeños con menos transacciones abiertas, índices que no se reconstruyan durante la ventana— o elegir un nivel con más caudal de registro.

Las cinco columnas cubren cinco agotamientos distintos:

```
avg_cpu_percent        consultas caras o falta de índices
avg_data_io_percent    lecturas que no caben en memoria
```

```
avg_log_write_percent  escritura por lotes, reconstrucción de índices
max_worker_percent     demasiadas conexiones activas: falta agrupación
max_session_percent    conexiones abiertas sin cerrar
```

Las dos últimas son la firma de un problema de aplicación, no de base de datos, y subir de nivel las empeora porque permite abrir más.

La elección de nivel de servicio es una decisión de latencia, y su precio es conocido:

	Almacenamiento	Latencia de E/S	Réplica de lectura	Costo relativo
Propósito general	Remoto	5-10 ms	No	1×
Crítico para la empresa	SSD local + réplicas	1-2 ms	Sí, incluida	2,7×
Hiperescala	Servidores de páginas	Variable	Hasta 4	Intermedio

Y dos avisos sobre las opciones que parecen atractivas:

Sin servidor pausa la base tras un periodo de inactividad y cobra por vCore-segundo. Reanudarla tarda cerca de un minuto: la primera petición después de la pausa agota su tiempo de espera. Es excelente para desarrollo e intermitencia real, y una fuente de incidentes si se pone delante de un servicio con tráfico esporádico y usuarios reales.

Hiperescala escala a 100 TB y restaura en minutos en vez de horas. Es una puerta de un solo sentido en la práctica: entrar es sencillo y volver no. Se decide con el mismo criterio que cualquier decisión irreversible de la clase 026 — con evidencia, no por si acaso.

Y una elección que no es de tamaño sino de compatibilidad: la instancia administrada existe para lo que la base de datos única no tiene —agente SQL, consultas entre bases de datos, Service Broker, CLR—. Una migración heredada que falla lo hace por ausencia de esas funciones, no por rendimiento; comprobarlo antes ahorra rehacer el plan a medias.

Un último punto sobre las copias de seguridad, que aquí son automáticas y crean una falsa sensación de seguridad: la restauración a un punto anterior cubre de 1 a 35 días y la retención a largo plazo llega a 10 años. Pero las copias cuelgan del servidor lógico: eliminar una base de datos conserva sus copias hasta que expire la retención, y eliminar el servidor se lleva las de todas sus bases a la vez. Es el borrado que no tiene vuelta atrás.

2. Cosmos DB: la cabecera que pone precio a cada consulta

Cosmos DB tiene una virtud pedagógica que ningún otro motor del programa ofrece: devuelve el costo de cada operación en la respuesta.

```
x-ms-request-charge: 2.87        lectura puntual con clave de partición
x-ms-request-charge: 1240.15     la misma consulta sin clave de partición
```

La clase 031 argumentaba que los patrones de acceso van primero. Aquí ese argumento deja de ser una recomendación y se convierte en una factura por línea: una consulta que abanica sobre todas las particiones cuesta 432 veces más que la lectura equivalente dirigida. Instrumentar esa cabecera en el registro de la aplicación es la mejor inversión de esta clase, porque convierte una discusión de diseño en un número que cualquiera puede ver.

La clave de partición concentra las dos decisiones irreversibles:

```
tope duro por partición lógica     20 GB
tope de rendimiento por partición  10.000 RU/s
al alcanzar el tope de tamaño      las escrituras FALLAN con 403
                                   (no se ralentizan: fallan)
cambiar la clave de partición      IMPOSIBLE: hay que crear otro contenedor
                                   y migrar los documentos
```

El fallo con 403 es cualitativamente distinto de la limitación por throughput: no es un 429 que el SDK reintenta, es un rechazo definitivo de la escritura mientras esa partición siga llena. Y como el tope es por partición lógica y no por contenedor, un contenedor de 4 TB puede estar rechazando escrituras porque una sola clave concentra 20 GB.

Los criterios para elegirla, en orden:

```
1. cardinalidad alta          muchos valores distintos, ninguno dominante
2. reparto uniforme           de escrituras y de almacenamiento
3. presente en las consultas  si no, todas abanican
```

Cuando ninguna propiedad natural cumple las tres, la salida es una clave sintética: concatenar dos campos, o añadir un sufijo calculado que reparta. Se decide al crear el contenedor, con los patrones de acceso escritos delante, porque después el arreglo cuesta una migración.

El modelo de rendimiento tiene tres opciones y una aritmética que conviene hacer:

```
aprovisionado manual   se paga el pico declarado, 24 h al día
escalado automático    escala entre el 10 % y el 100 % del máximo,
                       y se factura a 1,5× la tarifa por RU
sin servidor           por operación; tope de 5.000 RU/s y 50 GB por contenedor
```

El factor 1,5 fija el punto de equilibrio: el escalado automático solo compensa por debajo del 66 % de utilización media. Con un contenedor que sostiene 8.200 RU/s de forma estable y un máximo de 10.000:

```
escalado automático   8.200/100 × 0,012 USD/h = 0,984 → ~718 USD/mes
aprovisionado manual 10.000/100 × 0,008 USD/h = 0,800 → ~584 USD/mes
diferencia por pagar elasticidad que no se usa:      ~134 USD/mes
```

Y una lectura contraria a la intuición sobre los errores: en Cosmos DB, un 429 no es un fallo. Es la señal de control de flujo, viene con x-ms-retry-after-ms y el SDK reintenta solo. Una tasa de 429 sostenida en cero significa que se está pagando capacidad ociosa. Lo que sí hay que alertar es la tasa de 429 que el SDK no consigue resolver dentro del presupuesto de reintentos, porque esa sí llega al usuario.

Los cinco niveles de coherencia son el diferenciador del servicio y admiten un resumen operativo corto:

```
fuerte              lectura siempre actualizada; NO disponible con escritura
                    multirregión; lecturas al doble de RU
obsolescencia       retraso acotado por operaciones o por tiempo
  acotada           lecturas al doble de RU
sesión (por defecto) lees lo que tú escribiste, dentro de tu sesión
prefijo coherente   nunca ves escrituras desordenadas
eventual            lo más barato y lo más rápido
```

La trampa está en la de sesión, que es la que casi todo el mundo usa: la garantía viaja en un testigo de sesión. Si el cliente es un objeto compartido entre peticiones de usuarios distintos y el testigo no se propaga, un usuario puede no leer lo que acaba de escribir. No hay error, no hay alerta: solo un comportamiento intermitente que en desarrollo nunca se reproduce porque hay una sola instancia.

Y con escritura multirregión, los conflictos se resuelven por defecto con el último que escribe gana, comparando la marca de tiempo. Es determinista y pierde datos en silencio. Si esa pérdida no es aceptable —dos almacenes actualizando el mismo inventario—, hay que escribir un procedimiento de resolución o renunciar a la escritura multirregión.

3. Redis: la caché que se llena y deja de aceptar escrituras

La clase 031 dejó el criterio de cuándo una caché aporta y cómo se invalida. Lo que se añade aquí son tres decisiones de plataforma que producen incidentes con un patrón reconocible.

El nivel decide si hay SLA. El nivel básico es un nodo único sin acuerdo de nivel de servicio y sin réplica: un reinicio de mantenimiento vacía la caché y corta las conexiones. Es perfectamente válido para desarrollo y es el error de producción más repetido, porque funciona durante meses hasta el primer mantenimiento.

```
básico       nodo único, sin SLA            desarrollo
estándar     réplica principal-secundaria    producción básica
premium      persistencia, clúster, red virtual, zonas
empresa      módulos de Redis y replicación geográfica activa
```

La política de expulsión por defecto no expulsa casi nada. volatile-lru solo considera candidatas las claves con vencimiento. Si parte de las claves se escribe sin TTL —muy habitual: sesiones, catálogos «que se refrescan solos», banderas—, esas claves nunca salen. La caché se llena de lo que no puede expulsar y responde a las escrituras con un error de memoria:

```
síntoma      OOM command not allowed when used memory > 'maxmemory'
señal        used_memory al máximo y evicted_keys = 0
causa        volatile-lru + claves sin TTL
```

La combinación evictedkeys = 0 con memoria al máximo es diagnóstica: una caché sana expulsa. Que no expulse nada y esté llena significa que no puede.

Dos correcciones, y conviene aplicar las dos:

```
1. allkeys-lru  → cualquier clave es candidata; la caché nunca se bloquea
2. TTL siempre  → una clave sin vencimiento en una caché es una fuga de memoria
```

La memoria reservada no es opcional. Redis necesita memoria libre para replicar, para persistir y para la fragmentación. Sin maxmemory-reserved, la operación de guardado intenta duplicar estructuras en una instancia ya llena y el nodo se queda sin memoria durante la copia. La reserva se configura como porcentaje y su ausencia produce caídas justo durante el mantenimiento, que es cuando peor sientan.

Y un detalle de cliente que aparece en casi toda migración desde .NET: el multiplexor de conexión debe ser único y de larga vida. Crear uno por petición agota los puertos de salida de la máquina —el mismo agotamiento de puertos de traducción de la clase 040— y produce tiempos de espera que parecen de la caché y son del cliente:

```
un multiplexor por proceso, compartido y reutilizado          correcto
un multiplexor por petición                                   agota puertos
```

Por último, escalar una caché no es una operación transparente: cambiar de tamaño o de número de particiones reconfigura los nodos, cierra conexiones y, según el camino, puede vaciar el contenido. Toda aplicación que use caché debe funcionar —más lenta— con la caché vacía o inaccesible. Si no puede, no es una caché: es una base de datos sin copia de seguridad.

4. Las dos puertas de un solo sentido de esta clase

La clase 026 introdujo la distinción entre decisiones reversibles e irreversibles. Esta clase contiene dos de las irreversibles más caras del programa, y conviene tratarlas de forma explícita:

```
1. la clave de partición de un contenedor de Cosmos DB
2. la eliminación de un servidor lógico de Azure SQL
```

La clave de partición no se edita. Corregirla implica crear un contenedor nuevo y mover los documentos, y ese movimiento consume RU en los dos contenedores a la vez, así que la migración compite con el tráfico de producción por la misma capacidad. El procedimiento sensato:

```
1. crear el contenedor destino con la nueva clave
2. aprovisionar capacidad extra TEMPORAL en ambos
3. copiar con la fuente de cambios, no con una consulta de barrido
4. doble escritura durante la ventana de corte
5. verificar recuentos por partición antes de conmutar lecturas
6. devolver la capacidad extra
```

El paso 3 importa: leer el origen con una consulta que abanica multiplica el costo y castiga a las particiones ya saturadas. La fuente de cambios lee en orden de partición y es la vía prevista para esto.

El servidor lógico concentra las copias de seguridad de todas sus bases de datos. Eliminarlo las elimina. La protección es doble y ambas mitades hacen falta:

```
# 1. bloqueo de recurso: impide el borrado incluso a un propietario
$ az lock create --name no-borrar-sql --lock-type CanNotDelete \
    --resource-group rg-datos --resource-name sql-cloudshop-prod \
    --resource-type Microsoft.Sql/servers

# 2. retención a largo plazo: copias que NO cuelgan del ciclo de vida del servidor
$ az sql db ltr-policy set -g rg-datos -s sql-cloudshop-prod -n pedidos \
    --weekly-retention P4W --monthly-retention P12M --yearly-retention P7Y --week-of-year 1
```

El bloqueo de recurso es el mecanismo de gobierno de la clase 037 aplicado al caso concreto: no depende de que nadie se equivoque. Y conviene saber su límite: un bloqueo impide borrar, no impide vaciar. Un DROP TABLE pasa igual; para eso está la restauración a un punto anterior, que hay que haber probado.

La prueba, que es la parte que se omite:

```
$ az sql db restore -g rg-datos -s sql-cloudshop-prod -n pedidos \
    --dest-name pedidos-prueba --time "2026-07-31T22:00:00"
$ sqlcmd -S sql-cloudshop-prod.database.windows.net -d pedidos-prueba \
    -Q "SELECT COUNT(*) FROM dbo.pedidos WHERE creado < '2026-07-31 22:00'"
1284391                                                          ✓
$ az sql db delete -g rg-datos -s sql-cloudshop-prod -n pedidos-prueba --yes
```

Una restauración deja una base de datos nueva: no sobrescribe la original. Eso la hace segura de ensayar en producción, y es la razón por la que no hay excusa para no haberla ensayado nunca. El dato que hay que registrar de esa prueba no es que funcionó, sino cuánto tardó: ese número es el RTO real, y suele ser bastante mayor que el que figura en el plan.

5. Elegir entre los tres sin repetir la discusión cada trimestre

Los tres motores resuelven cosas distintas y la conversación se repite porque nadie escribe el criterio. Escrito, cabe en una tabla:

Pregunta	Azure SQL	Cosmos DB	Redis
¿Transacciones entre varias entidades?	Sí	Dentro de una partición lógica	No
¿Consultas no previstas?	Sí	Caras: abanican	No
¿Latencia de un dígito de milisegundo?	Solo crítico para la empresa	Sí	Sí
¿Escala de escritura horizontal?	Limitada	Sí, si la clave reparte	—
¿Fuente de verdad?	Sí	Sí	Nunca

La última fila es la que hay que defender cuando alguien propone «guardar el carrito solo en la caché porque es más rápido»: un dato que solo existe en una caché es un dato que se pierde en el próximo mantenimiento.

Y sobre el costo, la clase 031 avisaba de que a veces decide al revés de lo esperado. Aquí ocurre por una razón concreta: el precio de Cosmos DB depende del diseño, no del volumen. La misma carga puede costar 584 o 4.900 USD al mes según cómo reparta la clave de partición y cuántas consultas abaniquen. Ningún ajuste de infraestructura compensa un modelo de datos que obliga a leer todas las particiones para responder una pregunta frecuente.

La forma de mantener esa disciplina sin auditorías periódicas es hacer visible el número:

```
respuesta = contenedor.query_items(consulta, partition_key=cliente_id)
log.info("consulta=%s ru=%.2f", nombre, contenedor.client_connection.last_response_headers[
    "x-ms-request-charge"])
```

Con ese registro, una consulta cara aparece en el panel el día que se despliega, no en la factura del mes siguiente. Es el mismo principio que la etiqueta de costo de la clase 025 aplicado a la operación individual: lo que no tiene número no se discute, se opina.

Ejemplo trabajado

CloudShop lleva a Azure su capa de datos. El catálogo va a Cosmos DB, los pedidos a Azure SQL y las sesiones a Redis. Los tres funcionan durante seis semanas y luego fallan por motivos distintos — y ninguno es el que el panel sugiere.

Caso 1 — «la base de datos está lenta» con la CPU al 18 %.

La importación nocturna de catálogo pasa de 40 minutos a más de tres horas. El panel de la base de datos S3 —modelo DTU— muestra 100 % de DTU.

```
DTU al 100 %  →  ¿de qué?
```

No se puede saber. Se migra a vCore, propósito general, 4 vCore, y la vista aparece:

```
end_time   cpu   data_io   log_write   worker   session
02:41:00   18,2    24,7      100,0        9,1      2,4
```

El techo es el registro de transacciones. La importación insertaba fila a fila dentro de una transacción por fila:

```
tamaño de lote                      1 fila     5.000 filas
reconstrucción de índices          durante     después
registro generado por 1 M de filas   4,1 GB      0,9 GB
durante la ventana
duración de la importación         3 h 12 min    26 min
```

No hizo falta subir de nivel. La conclusión que se documenta: el modelo DTU no era caro, era ciego.

Caso 2 — el percentil 99 de los pedidos no baja de 40 ms.

Con el registro resuelto, la latencia de lectura sigue alta. La E/S de datos está en el 24 % y la latencia por operación es de 7 ms: es el almacenamiento remoto del nivel de propósito general.

```
latencia de E/S                 ~7 ms               ~1,5 ms
p99 de la operación de pedido    41 ms               9 ms
réplica de lectura               no                  incluida
costo mensual de cómputo        ~365 USD            ~985 USD
```

La decisión no se aplica a todo: solo la base de datos de pedidos sube de nivel. Catálogo e informes se quedan en propósito general, y los informes pasan a leer de la réplica que el nivel crítico incluye, lo que además quita carga de la principal.

```
pedidos      propósito general 365    crítico 985
catálogo     propósito general 365    propósito general 365
informes     propósito general 365    réplica de lectura de pedidos: 0
                          ■■■■■■■■■          ■■■■■■■■■
                           1.095 USD          1.350 USD
```

Cuesta 255 USD más al mes y elimina una base de datos entera. El p99 pasa de 41 a 9 ms.

Caso 3 — las escrituras del catálogo empiezan a fallar con 403.

No es limitación por throughput: es un rechazo.

```
403 · Partition key reaching storage quota
```

La clave de partición era /pais, y el 82 % de los documentos son de un solo país:

```
$ az cosmosdb sql container show -a cosmos-cloudshop -d tienda -n catalogo \
    -g rg-datos --query "resource.partitionKey.paths" -o tsv
/pais
```

```
partición "CL"    19,8 GB  ← contra el tope de 20 GB
partición "AR"     1,4 GB
partición "PE"     0,9 GB
resto             < 0,3 GB
```

La clave es inmutable. Se migra a /categoriaId —cardinalidad alta, reparto uniforme y presente en el 94 % de las consultas— con fuente de cambios y doble escritura:

```
capacidad temporal extra durante la migración   +6.000 RU/s durante 9 h
documentos migrados                             41,2 millones
mayor partición resultante                       1,1 GB
corte de servicio                                ninguno
```

Y con la cabecera instrumentada aparece lo que llevaba seis semanas escondido:

```
lectura de ficha por id (con clave)             2,87 RU     2,87 RU
listado por categoría                       1.240,15 RU     8,44 RU
RU/s medias del contenedor                     8.200        2.100
```

Caso 4 — el modelo de rendimiento que costaba de más.

Con 8.200 RU/s estables antes de la migración, el contenedor estaba en escalado automático con máximo 10.000. La utilización media era del 82 %, muy por encima del punto de equilibrio del 66 %:

```
escalado automático a 8.200 RU/s medias    718
aprovisionado manual a 10.000 RU/s         584
```

Después de la migración, con 2.100 RU/s medias y picos de 6.000, la aritmética se invierte y el escalado automático vuelve a ser lo correcto:

```
escalado automático, máx. 6.000, media 2.100   ~184 USD/mes
aprovisionado manual a 6.000                   ~350 USD/mes
```

El criterio queda escrito para no repetir la discusión: por encima del 66 % de utilización media, manual; por debajo, automático, y se revisa cuando el patrón de carga cambie.

Caso 5 — la caché deja de aceptar escrituras un domingo.

```
OOM command not allowed when used memory > 'maxmemory'
used_memory       6,0 GB / 6,0 GB
evicted_keys      0
```

Memoria llena y ninguna expulsión: la política era volatile-lru y el 61 % de las claves —sesiones y banderas de configuración— se escribía sin vencimiento. Además la instancia era de nivel básico, así que el reinicio de mantenimiento de esa madrugada había vaciado la caché sin réplica que la sostuviera.

```
nivel                            básico            estándar con réplica
política de expulsión          volatile-lru        allkeys-lru
claves sin TTL                      61 %                 0 %
memoria reservada                   0 %                 25 %
multiplexor de conexión        uno por petición     uno por proceso
prueba con caché vacía            nunca             en cada despliegue
```

La última fila es la que convierte esto en un diseño: la aplicación se despliega con una prueba que arranca con la caché vacía y comprueba que responde —más lenta— sin errores.

Resumen de la capa de datos:

```
modelo de compra de Azure SQL                DTU           vCore
cuello de botella identificable              no            sí, cinco señales
p99 de la operación de pedido               41 ms           9 ms
importación nocturna                      3 h 12 min        26 min
escrituras rechazadas en Cosmos DB         sí (403)          0
RU/s medias del catálogo                    8.200          2.100
bases de datos SQL                             3              2
costo mensual de la capa de datos           2.190 USD      1.640 USD
```

La lección que esta clase traslada al resto de la parte: los tres motores tienen un límite que no es la CPU y que el panel por defecto no muestra —el registro en SQL, la partición lógica en Cosmos, la política de expulsión en Redis—. Un servicio de datos que solo se vigila por CPU está vigilando el recurso que casi nunca se agota primero.

Laboratorio guiado

Ejecuta desde la raíz:

```
python classes/part-03-azure-core-platform/042-azure-sql-cosmos-db-y-azure-cache-for-redis/lab.py
```

El laboratorio selecciona el motor de práctica data y produce labresult.json. El escenario, sus comprobaciones y el artefacto esperado corresponden a esta clase; no requiere credenciales y deja explícito qué debe revalidarse en un sandbox real.

1. Lee exercise.steps y formula una predicción antes de ejecutar.
2. Ejecuta la práctica y verifica todos los elementos de checks.
3. Provoca el caso de negativetest y explica la señal observada.
4. Materializa matriz-datos-azure en evidence/ usando la plantilla indicada.
5. Para proveedor real, sigue sandbox.requires, registra costo y ejecuta destroy.

Evidencia esperada

El artefacto principal es un modelo de datos ligado a patrones de acceso. Además, la entrega debe incluir el comando ejecutado, la salida estructurada y una conclusión que no exceda lo observado.

Reto verificable

Construye matriz-datos-azure para el caso CloudShop. Incluye una alternativa descartada, un supuesto que pueda falsarse, una prueba de fallo y una decisión de rollback.

■ Criterio de aceptación

- [ ] lab.py termina con código 0 y genera JSON válido.
- [ ] La entrega conecta al menos tres requisitos con mecanismos verificables.
- [ ] Existe una prueba positiva y una prueba negativa con evidencia.
- [ ] Seguridad, costo y operación aparecen como decisiones, no como anexos.
- [ ] Se declara una limitación y una condición que obligaría a revisar el diseño.
- [ ] Otra persona puede repetir el recorrido sin conocimiento tácito.

Errores frecuentes

Síntoma	Causa probable	Corrección
La base de datos está al 100 % y no se sabe de qué recurso	El modelo DTU mezcla CPU, memoria y E/S en una sola unidad	Migra a vCore y consulta sys.dmdbresourcestats: cinco columnas, cinco agotamientos distintos.
Un proceso por lotes va lentísimo con la CPU al 18 %	El gobernador de velocidad de registro está al 100 %	Reduce el registro generado —lotes mayores, menos transacciones, índices fuera de la ventana— antes de subir de nivel.
Las escrituras en Cosmos DB fallan con 403 y no con 429	Una partición lógica alcanzó el tope duro de 20 GB	La clave es inmutable: crea otro contenedor con una clave que reparta y migra con la fuente de cambios y doble escritura.
Una consulta frecuente cuesta cientos de RU	Abanica sobre todas las particiones porque la clave no está en el filtro	Registra x-ms-request-charge en cada consulta y rediseña la clave o el modelo para que la consulta sea dirigida.
La caché responde con error de memoria y no ha expulsado ninguna clave	La política volatile-lru solo expulsa claves con vencimiento y muchas se escriben sin TTL	Cambia a allkeys-lru, exige TTL en todas las escrituras y reserva memoria para replicación y fragmentación.
Al eliminar un servidor lógico desaparecen las copias de seguridad de todas sus bases	La restauración a un punto anterior cuelga del servidor, no de la base de datos	Bloqueo de recurso contra el borrado y retención a largo plazo, con una prueba de restauración cuya duración se registra como RTO real.

Seguridad, ética y costo

Trabaja con cuentas propias o sandboxes autorizados. No publiques secretos, identificadores reales ni datos personales. Antes de crear recursos pagos define presupuesto, etiquetas y comando de destrucción; después verifica que no queden recursos huérfanos. Los ejemplos locales enseñan contratos, pero no certifican cumplimiento ni disponibilidad de producción.

Preguntas de comprobación

1. El panel muestra 100 % de DTU. ¿Qué información te falta y cómo la obtienes?
2. ¿Qué distingue un 403 de un 429 en Cosmos DB, y por qué solo uno de los dos lo resuelve el SDK?
3. ¿A partir de qué utilización media deja de compensar el escalado automático, y por qué ese número?
4. Una caché tiene la memoria llena y evictedkeys en cero. ¿Qué está ocurriendo?
5. ¿Qué dos decisiones de esta clase no se pueden deshacer editando una plantilla, y cómo se protege cada una?

Referencias

- Microsoft (2025). vCore purchasing model for Azure SQL Database — recursos separados y niveles de servicio. <https://learn.microsoft.com/en-us/azure/azure-sql/database/service-tiers-sql-database-vcore>
- Microsoft (2025). Resource limits and sys.dmdbresourcestats — CPU, E/S de datos, velocidad de registro, trabajadores y sesiones. <https://learn.microsoft.com/en-us/azure/azure-sql/database/resource-limits-logical-server>
- Microsoft (2025). Partitioning and horizontal scaling in Azure Cosmos DB — partición lógica, tope de 20 GB y claves sintéticas. <https://learn.microsoft.com/en-us/azure/cosmos-db/partitioning-overview>
- Microsoft (2025). Consistency levels in Azure Cosmos DB — los cinco niveles, testigo de sesión y costo en RU. <https://learn.microsoft.com/en-us/azure/cosmos-db/consistency-levels>
- Microsoft (2025). Azure Cache for Redis best practices — niveles, políticas de expulsión, memoria reservada y conexiones. <https://learn.microsoft.com/en-us/azure/azure-cache-for-redis/cache-best-practices-development>
- Documentación oficial vigente del servicio implementado; registra URL y fecha de consulta.

Anterior	Índice	Siguiente
← 041 · Blob Storage, Files, redundancia y lifecycle	Parte 03 · Programa	043 · App Service, Functions y Container Apps →

Evaluación — 042 Azure SQL, Cosmos DB y Azure Cache for Redis

Preguntas

1. Explica el problema que resuelve esta capacidad sin mencionar un proveedor.
2. Identifica una frontera de responsabilidad y su propietario.
3. Compara dos alternativas mediante confiabilidad, seguridad, costo y operación.
4. ¿Qué demuestra el laboratorio y qué no puede demostrar?
5. Propón un caso límite o una prueba de fallo.

Reto verificable

Entrega matriz-datos-azure con diagrama o configuración, evidencia de ejecución, alternativa descartada, riesgo residual y criterio de rollback.

Criterio de aceptación

- [ ] Resultado reproducible por una segunda persona.
- [ ] Evidencia vinculada a cada afirmación técnica.
- [ ] Prueba negativa o de fallo incluida.
- [ ] Costos y permisos expresados con unidades y alcance.
- [ ] Limitaciones declaradas sin presentar la simulación como producción.

Escala

Nivel	Evidencia
A — competente	Cumple todo y defiende trade-offs con datos.
B — en desarrollo	Cumple lo esencial; falta profundidad en una dimensión.
C — inicial	Artefacto parcial o conclusión sin evidencia suficiente.
0	Sin entrega reproducible.

043 — App Service, Functions y Container Apps

Parte: 03 — Azure: plataforma esencial Nivel: intermedio · Horas estimadas: 4 Laboratorio: serverless · Estado: EXECUTABLECORE

Propósito

Elegir dónde corre la aplicación en Azure sabiendo cuál es la unidad que escala y cuál es la que factura, porque en las tres opciones son distintas y ninguna coincide con la de AWS. El plan de App Service escala para todas sus aplicaciones a la vez, la aplicación de funciones escala para todas sus funciones a la vez, y una aplicación de contenedor escala por la regla que le pongas —y si esa regla es la equivocada, se apaga y deja de trabajar sin dar ningún error.

Resultados de aprendizaje

Al finalizar podrás:

1. Identificar la unidad de escalado y la unidad de facturación de App Service, Functions y Container Apps, y el radio de impacto de cada una.
2. Elegir plan de hospedaje de funciones a partir de duración máxima, acceso a red privada y tolerancia al arranque en frío.
3. Diagnosticar el agotamiento de puertos de traducción de salida, que se manifiesta como fallo del destino y no del origen.
4. Configurar un intercambio de espacios con calentamiento y ajustes fijados al espacio, y explicar qué se lleva el intercambio.
5. Definir reglas de escalado que no apaguen a un consumidor que sigue teniendo trabajo pendiente.

Conceptos centrales

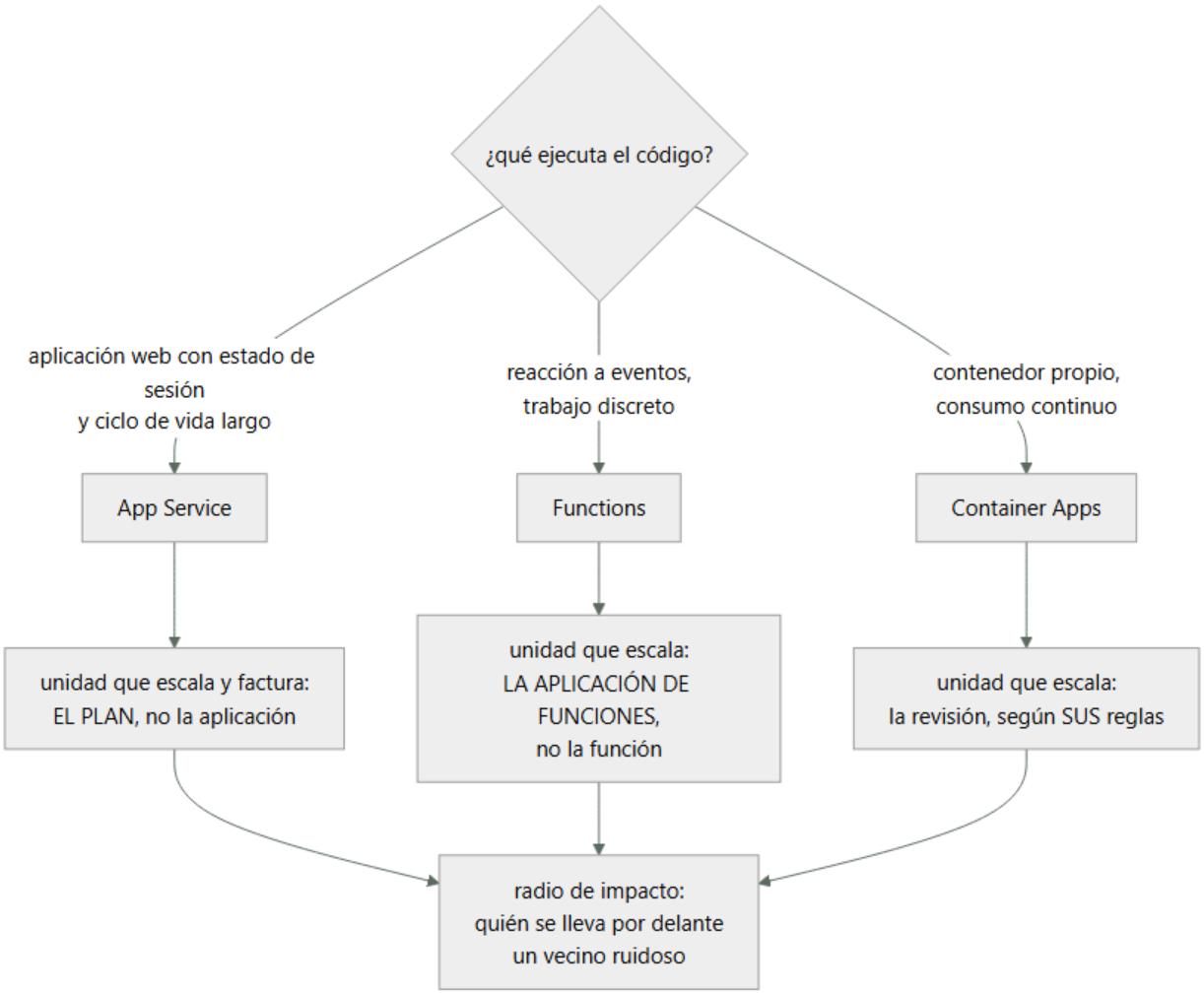
Concepto	Comprensión verificable
plan de App Service	Conjunto de máquinas que ejecuta todas las aplicaciones asignadas a él. Se factura el plan, no la aplicación, y el escalado es del plan: diez aplicaciones compiten por la misma CPU.
Always On	Ajuste que impide que la aplicación se descargue tras unos veinte minutos sin tráfico. Sin él, la primera petición tras el silencio paga un arranque completo.
ajuste fijado al espacio	Configuración marcada para quedarse en el espacio durante un intercambio. Lo que no está marcado viaja con la aplicación: una cadena de conexión de pruebas puede acabar en producción.
puertos de traducción de salida	Cupo de conexiones salientes por instancia —128 por defecto en App Service—. Al agotarse, las llamadas a servicios externos fallan de forma intermitente y el error parece del destino.
plan de hospedaje de funciones	Decide tres cosas a la vez: duración máxima de una ejecución, si hay acceso a red virtual y si existe arranque en frío. No es una decisión de precio.
regla de escalado	Señal que determina cuántas réplicas hay. Una aplicación de contenedor con una única regla HTTP se reduce a cero aunque tenga una cola llena esperando.

Modelo mental

En Azure, identidad y jerarquía organizacional preceden al recurso: tenant, management group, suscripción y resource group determinan el alcance de control.

Aplicado a esta clase, separa siempre cuatro planos: intención (qué necesita el usuario), configuración (qué declaramos), estado observado (qué existe de verdad) y evidencia (cómo sabemos que cumple). Confundirlos produce diseños que se ven correctos en un diagrama pero fallan al operar.

■ Flujo de razonamiento



Desarrollo

1. El plan es la unidad, no la aplicación

Es la primera sorpresa al llegar de AWS, donde cada servicio escala por su cuenta. En App Service se compra un plan —un conjunto de máquinas— y las aplicaciones se asignan a él:

```
plan asp-cloudshop-prod  (2 × P1v3)
  ■■ tienda            tráfico de clientes
  ■■ admin             uso interno
  ■■ informes          trabajo nocturno pesado
  ■■ webhooks          picos impredecibles
```

Las cuatro comparten CPU, memoria y conexiones salientes. El trabajo nocturno de informes degrada la tienda, y el panel de la tienda no muestra nada raro: su código no cambió y su CPU no subió, subió la del vecino.

La regla de diseño se deduce sola: el plan es la frontera del radio de impacto. Se reparten aplicaciones por plan según qué debe seguir en pie cuando otra cosa se rompa, no según qué es más barato de agrupar. Y como el escalado también es del plan, escalar por culpa de informes multiplica el costo de las cuatro.

Dos ajustes del plan que producen incidentes por sí solos:

Always On. Sin él, la aplicación se descarga tras unos veinte minutos sin tráfico y la siguiente petición paga el arranque completo del proceso. En un servicio interno que se usa dos veces al día, todo el mundo lo percibe como «esto va lentísimo» y las métricas medias no lo muestran, porque afecta a una petición de cada cien. Tampoco los trabajos programados dentro del proceso sobreviven a la descarga: se planifican y no se ejecutan.

Escalado del plan. El escalado automático es de Azure Monitor, con las mismas reglas de la clase 029, y con una diferencia de tiempos: añadir una instancia a un plan tarda entre uno y tres minutos, no segundos. Una regla que reaccione a un pico de 60 segundos llega siempre tarde. Para carga predecible —los picos de una tienda lo son— un perfil por horario evita escalar por reacción.

Y el agotamiento de conexiones salientes merece su propio párrafo porque el error apunta al sitio equivocado. Cada instancia dispone de unos 128 puertos de traducción hacia un mismo destino. Un cliente HTTP que se crea y se descarta en cada petición no reutiliza conexiones, y en cuanto el tráfico sube:

```
síntoma    tiempos de espera intermitentes hacia la pasarela de pago
           el resto de la aplicación funciona
lectura    parece que la pasarela de pago falla
señal real métrica "SNAT Connection Count" al máximo
           y "SNAT Port Usage" en ascenso
```

Las tres correcciones, en orden de eficacia:

```
1. reutilizar el cliente HTTP (una instancia por proceso, no por petición)
2. integración con red virtual + NAT Gateway  → 64.512 puertos por IP (clase 039)
3. punto de conexión privado para los destinos que sean de Azure
   → no consumen puertos de traducción en absoluto
```

La tercera es la que más rinde en una plataforma que ya hizo el trabajo de la clase 039: el tráfico a almacenamiento, bases de datos y colas deja de salir, deja de contar y deja de fallar.

Un apunte sobre la red que se confunde en casi todos los equipos: la integración con red virtual es de salida. Permite que la aplicación alcance recursos privados; no hace que la aplicación sea privada. Para eso hace falta un punto de conexión privado hacia la propia aplicación, que es la otra dirección y otro recurso.

2. Espacios de implementación: qué viaja y qué se queda

Un espacio es una instancia paralela de la aplicación con su propio nombre de host. El intercambio no vuelve a desplegar nada: cambia de sitio la configuración y redirige el tráfico, lo que hace que la vuelta atrás sea igual de rápida que la ida. Esa es toda su virtud, y es grande: un despliegue que se puede revertir en quince segundos cambia el apetito de riesgo de un equipo.

La parte que hay que dominar es qué se lleva el intercambio:

```
viaja con la aplicación   código, ajustes NO marcados, cadenas de conexión NO marcadas,
                          configuración de tiempo de ejecución
se queda en el espacio    ajustes marcados como fijados al espacio,
                          nombre de host, certificados, escalado,
                          configuración de red privada
```

La primera línea contiene el incidente clásico: si la cadena de conexión a la base de datos de pruebas no está marcada como fijada al espacio, el intercambio la lleva a producción. La aplicación arranca sin errores y empieza a escribir pedidos reales en la base de datos de pruebas. No hay excepción, no hay alerta; solo datos en el sitio equivocado.

```
$ az webapp config appsettings set -g rg-app -n app-tienda --slot staging \
    --slot-settings DB_CONNECTION=... APPINSIGHTS_KEY=...
$ az webapp config appsettings list -g rg-app -n app-tienda --slot staging \
    --query "[?slotSetting].name" -o tsv
DB_CONNECTION
APPINSIGHTS_KEY                                                    ✓
```

La comprobación de esa lista debería formar parte de la revisión de cada ajuste nuevo, porque el valor por defecto es el peligroso: lo que no se marca, viaja.

La segunda pieza es el calentamiento. El intercambio reinicia la aplicación en el espacio de destino, así que sin calentamiento las primeras peticiones tras el intercambio pagan el arranque completo. Azure espera a que la aplicación responda antes de completar el intercambio si se le indica qué pedir:

```
<applicationInitialization>
  <add initializationPage="/readyz" />
  <add initializationPage="/api/catalogo?limit=1" />
</applicationInitialization>
```

Y vale aquí, entera, la lección de la clase 029: la ruta de calentamiento debe ejercitar las dependencias, no devolver 200 desde memoria. Un /readyz que comprueba base de datos y caché convierte el intercambio en una prueba de humo automática; un / estático lo convierte en un ritual.

El patrón completo que conviene dejar escrito:

```
1. desplegar en el espacio de preparación
2. ejecutar pruebas contra el nombre de host del espacio
3. intercambiar con calentamiento
4. observar cinco minutos las señales de error y latencia
5. si algo se degrada, intercambiar de vuelta — no desplegar la versión anterior
```

El paso 5 es la razón de todo lo demás: la reversión es un intercambio, no un despliegue. Un equipo que responde a un despliegue malo desplegando otra vez tarda entre diez y cuarenta minutos; uno que intercambia tarda menos de un minuto.

3. Functions: el plan decide tres cosas y ninguna es el precio

La clase 032 estableció que en serverless la concurrencia es el recurso que se agota y que el arranque en frío se compara con el percentil del SLO, no con la media. Ese método se traslada tal cual. Lo que cambia en Azure es que el plan de hospedaje decide simultáneamente tres cosas que en Lambda son ajustes independientes:

	Duración máxima	Red virtual	Arranque en frío
Consumo	5 min por defecto, 10 como tope	No	Sí
Consumo flexible	30 min por defecto	Sí	Se mitiga con instancias siempre listas
Premium	Sin tope práctico (60 min por defecto)	Sí	No: instancias precalentadas
Dedicado (plan de App Service)	Sin tope	Sí	No, con Always On

El tope de diez minutos del plan de consumo es duro y no se negocia con una solicitud de soporte. Un proceso de facturación mensual que tarda catorce minutos no se arregla subiendo la memoria: se arregla partiéndolo o llevándolo a Durable Functions, que es el mecanismo de orquestación de la clase 032 con estado gestionado por la plataforma.

Y aquí está la diferencia estructural con AWS, la que produce el incidente que nadie anticipa:

```
AWS Lambda    escala POR FUNCIÓN
Azure         escala POR APLICACIÓN DE FUNCIONES
```

Todas las funciones de una misma aplicación comparten el plan, el proceso anfitrión y la decisión de escalado. Consecuencia directa: una avalancha de mensajes en una cola hace que el controlador escale la aplicación entera, y la función HTTP que atiende un webhook de pago compite por el mismo anfitrión con el procesador de la cola. Su percentil 99 se dispara sin que su tráfico haya cambiado.

La regla que evita la clase entera de problemas: una aplicación de funciones por perfil de carga, no por dominio funcional. Lo que responde a un usuario y lo que procesa en segundo plano no viven juntos, aunque compartan repositorio y despliegue.

Sobre el arranque en frío, los tres factores que más pesan y se pueden controlar:

```
tamaño del paquete       menos dependencias, arranque más corto
lenguaje                 los tiempos de ejecución interpretados arrancan antes
                         que los que compilan al vuelo
instancias siempre listas convierten el arranque en frío en un costo fijo
```

La tercera es un intercambio explícito: se paga por tener instancias vivas para no pagar latencia. Cuánto vale eso lo decide el SLO, con el método de la clase 032: si el percentil 99 del objetivo es de 500 ms y un arranque en frío cuesta 2,3 s, basta con que ocurra en el 1,2 % de las peticiones para incumplirlo. Ese cálculo es el que justifica el gasto, no la incomodidad de verlo en una traza.

4. Container Apps: la regla que apaga al que tenía trabajo

Container Apps ejecuta contenedores sobre Kubernetes con KEDA y Dapr debajo, sin exponer ninguno de los tres. La parte 06 entra en Kubernetes; aquí importa lo que se decide sin verlo.

La regla de escalado es la pieza central, y su valor por defecto contiene una trampa cara. Una aplicación con regla HTTP se reduce a cero cuando no hay peticiones. Eso es correcto para un servicio web y es destructivo para un consumidor de cola:

```
consumidor con regla HTTP únicamente
  no recibe peticiones HTTP nunca  →  0 réplicas
```

```
la cola crece                    →  sigue en 0 réplicas
nadie recibe un error            →  nadie se entera
```

El síntoma no es un fallo: es trabajo que no se hace. Se descubre cuando alguien pregunta por qué no llegan los correos de confirmación, con cuarenta mil mensajes acumulados. La corrección es una regla que mire la señal correcta:

```
scale:
  minReplicas: 1          # un consumidor no baja de uno
  maxReplicas: 20
  rules:
    - name: cola-pedidos
      custom:
        type: azure-servicebus
        metadata:
          queueName: pedidos
          messageCount: "20"     # una réplica por cada 20 mensajes pendientes
```

minReplicas: 1 en un consumidor no es desperdicio: es la diferencia entre un retraso de segundos y un retraso que dura hasta que alguien lo note. Reducir a cero está bien donde el disparador es la propia llegada de la petición.

Las revisiones son la otra pieza: cada despliegue crea una revisión inmutable y el tráfico se reparte entre ellas por porcentaje.

```
$ az containerapp ingress traffic set -g rg-app -n ca-pedidos \
    --revision-weight pedidos--v7=90 pedidos--v8=10
```

Es un despliegue canario sin infraestructura adicional, y su valor depende por completo de tener una señal por revisión. Repartir 90/10 sin poder comparar la tasa de error de cada una es repartir a ciegas; la clase 045 monta esa parte.

Los perfiles de carga de trabajo deciden dónde corre cada contenedor:

```
consumo    se paga por vCPU-segundo y GiB-segundo, escala a cero
dedicado   máquinas reservadas, precio por hora, aislamiento y tamaños grandes
```

El de consumo es el correcto por defecto; el dedicado aparece cuando hace falta más memoria de la que el de consumo ofrece, aislamiento de cómputo o costo predecible con carga estable — el mismo razonamiento de los modelos de compra de la clase 028.

Y la tabla que cierra la elección de la clase, que es el entregable real:

Pregunta	App Service	Functions	Container Apps
¿Contenedor propio con dependencias del sistema?	Limitado	No	Sí
¿Reducir a cero?	No	Sí	Sí
¿Ejecución de más de 30 min?	Sí	Solo en Premium o dedicado	Sí
¿Despliegue canario integrado?	Espacios	Espacios	Revisiones con peso
¿Escala por longitud de cola?	Con métrica personalizada	Sí, nativo	Sí, nativo
Unidad de escalado	El plan	La aplicación de funciones	La revisión

La última fila es la que hay que llevarse. Las tres opciones escalan; lo que las distingue es qué se arrastra cuando escalan.

5. Salir a la red privada desde una plataforma administrada

Las tres opciones corren fuera de tu red virtual por defecto. Como la clase 039 terminó cerrando el acceso público de la base de datos y del almacenamiento, este es el punto donde ambas decisiones se encuentran — y donde una plataforma bien diseñada deja de funcionar por hacerlo bien.

El mapa de lo que hace falta en cada caso:

```
App Service        integración regional con red virtual (salida)
                   + punto de conexión privado hacia la app (entrada privada)
Functions          plan de consumo: NO hay acceso a red virtual
                   consumo flexible, premium o dedicado: sí
Container Apps     el entorno se despliega EN una subred delegada
                   → la decisión se toma al crear el entorno, no después
```

La segunda línea es la que rompe migraciones: un conjunto de funciones en plan de consumo no puede alcanzar una base de datos con acceso público deshabilitado. No es un problema de reglas de red que se arregle abriendo un puerto —no hay ruta. La corrección es cambiar de plan, y cambiar de plan cambia el precio, así que el requisito de red debe entrar en la elección desde el principio y no aparecer al final.

La tercera es una decisión de un solo sentido más: la subred del entorno de Container Apps se fija al crearlo y no se cambia. Junto con el tamaño —el entorno exige un bloque de direcciones considerable— es exactamente el tipo de reserva que la planificación de la clase 039 debía haber previsto.

Y hay un detalle de DNS que reproduce el fallo silencioso de aquella clase. Con integración de red virtual, la aplicación resuelve nombres usando el DNS de la red virtual solo si se le indica:

```
$ az webapp config appsettings set -g rg-app -n app-tienda --settings \
    WEBSITE_VNET_ROUTE_ALL=1 WEBSITE_DNS_SERVER=168.63.129.16
```

Sin el primero, únicamente el tráfico hacia rangos privados entra por la red virtual y el resto sale por internet. Sin el segundo, la zona DNS privada no se consulta y el nombre del almacenamiento vuelve a resolver a su IP pública. Es el mismo síntoma de la clase 039 con otro origen: todo funciona y nada es privado, hasta que se cierra el acceso público y entonces deja de funcionar de golpe.

La comprobación honesta es siempre la misma y cuesta un minuto:

```
$ az webapp ssh -g rg-app -n app-tienda --command \
    "nslookup stcloudshopprod.blob.core.windows.net"
Address: 10.20.6.5                                                  ✓
```

Ejemplo trabajado

CloudShop lleva a Azure su capa de aplicación: tienda y administración en App Service, procesamiento de pedidos en Functions y el consumidor de notificaciones en Container Apps. Cinco semanas después hay cinco incidentes, y cuatro de ellos apuntan al componente equivocado.

Punto de partida:

```
plan asp-cloudshop-prod (2 × P1v3, ~248 USD/mes)
  tienda · admin · informes · webhooks
func-cloudshop (plan de consumo)
  procesar-pedido (cola) · webhook-pago (HTTP) · facturar-mes (temporizador)
ca-notificaciones (Container Apps, regla HTTP, mín. 0)
```

Incidente 1 — la tienda se degrada cada noche a las 02:00.

El percentil 99 de la tienda pasa de 180 ms a 1,9 s durante veinte minutos. Su CPU no sube.

```
$ az monitor metrics list --resource $PLAN_ID --metric CpuPercentage \
    --aggregation Maximum --interval PT5M --query "value[0].timeseries[0].data[-6:]"
[31.0, 94.2, 97.8, 96.1, 92.4, 28.7]
```

La CPU del plan al 97 %: es informes, que genera el cierre diario. Las cuatro aplicaciones comparten las dos instancias. Se separa por radio de impacto:

```
asp-prod   tienda, admin,     tienda, webhooks   2 × P1v3   248 USD
           informes, webhooks
asp-interno    —              admin, informes    1 × P1v3   124 USD
                              ■■■■■■■■■■■          ■■■■■■■■■■■■■■■■■■■■
                              248 USD                 372 USD
```

Cuesta 124 USD más al mes. El p99 nocturno de la tienda vuelve a 180 ms y el cierre diario deja de tener que competir por CPU: baja de 20 a 11 minutos.

Incidente 2 — la pasarela de pago «falla» en los picos.

Tiempos de espera intermitentes hacia el proveedor de pago, siempre por encima de 400 peticiones por minuto. El proveedor confirma que no ve las solicitudes.

```
$ az monitor metrics list --resource $APP_ID --metric SnatConnectionCount \
    --aggregation Total --interval PT1M --query "value[0].timeseries[0].data[-3:]"
[128, 128, 128]
```

El cupo de puertos, clavado en el techo. El código creaba un cliente HTTP por petición.

```
cliente HTTP                      uno por petición  uno por proceso
```

```
puertos en uso en el pico              128 (tope)         19
integración con red virtual              no           sí + NAT Gateway
fallos hacia la pasarela               3,1 %            0,0 %
```

Y se aprovecha para que el tráfico hacia almacenamiento y base de datos deje de contar: con puntos de conexión privados, esas llamadas no consumen ningún puerto de traducción.

Incidente 3 — un intercambio de espacios escribe pedidos en la base de datos de pruebas.

Durante cincuenta minutos, 214 pedidos acaban en el entorno equivocado. La aplicación no dio un solo error.

```
$ az webapp config appsettings list -g rg-app -n app-tienda --slot staging \
    --query "[?slotSetting].name" -o tsv
APPINSIGHTS_KEY
```

DBCONNECTION no estaba marcada, así que viajó con la aplicación. Se corrige y se añade el calentamiento que faltaba:

```
ajustes fijados al espacio                1                6
ruta de calentamiento                  ninguna          /readyz con dependencias
primeras peticiones tras intercambiar  2,4 s p99          190 ms p99
reversión ensayada                       no           sí, 14 s medidos
```

Los 214 pedidos se recuperan de la base de pruebas y se reinsertan. La medida que evita la repetición no es la corrección: es la comprobación de la lista de ajustes fijados en cada revisión de configuración.

Incidente 4 — el webhook de pago se degrada cuando llega una avalancha de pedidos.

Y el cierre mensual, además, se corta a los diez minutos:

```
FunctionTimeoutException: Timeout value of 00:10:00 exceeded
```

Dos problemas con la misma raíz: tres funciones muy distintas en una sola aplicación con plan de consumo. Se separan por perfil de carga y por requisito:

```
webhook-pago     consumo, compartido        consumo flexible, app propia,
                                            2 instancias siempre listas
procesar-pedido  consumo, compartido        consumo flexible, app propia
facturar-mes     consumo, corta a 10 min    Durable Functions, en la app de proceso
```

```
p99 del webhook durante avalancha    4,8 s      210 ms
cierre mensual                     no termina   38 min, en 6 etapas
acceso a la base de datos privada   imposible   sí (consumo flexible)
costo mensual de funciones         ~28 USD     ~96 USD
```

Los 68 USD adicionales compran tres cosas que el plan de consumo no podía dar: aislamiento entre cargas, acceso a la red privada y un proceso largo que termina.

Incidente 5 — 41.000 notificaciones sin enviar.

Nadie recibe los correos de confirmación desde hace dos días. La aplicación de contenedor está sana y en cero réplicas.

```
$ az containerapp show -g rg-app -n ca-notificaciones \
    --query "properties.template.scale" -o json
{"minReplicas": 0, "maxReplicas": 10, "rules": [{"name": "http", "http": {"metadata": {
"concurrentRequests": "50"}}}]}
```

La única regla miraba peticiones HTTP, y el consumidor no recibe ninguna: lee de una cola. Con cero peticiones, cero réplicas, y nadie vacía la cola. No hubo ningún error que alertar.

```
regla de escalado              HTTP              longitud de cola
réplicas mínimas                 0                    1
mensajes pendientes            41.000               < 30
alerta sobre profundidad de cola  no        sí, > 500 durante 5 min
```

La alerta sobre la profundidad de la cola es la corrección de fondo: el consumidor puede volver a fallar por otro motivo, y lo que hay que detectar es trabajo que se acumula, no procesos que se caen.

Resumen de la capa de aplicación:

```
planes de App Service                       1            2
aplicaciones de funciones                   1            2
fallos hacia la pasarela de pago          3,1 %         0,0 %
p99 nocturno de la tienda                 1,9 s         180 ms
p99 del webhook en avalancha              4,8 s         210 ms
```

```
cierre mensual                            no termina      38 min
trabajo en segundo plano detenido        2 días sin señal  alerta a 5 min
costo mensual de cómputo de aplicación     276 USD        468 USD
```

La lección que esta clase traslada al resto de la parte: en las tres plataformas administradas, lo que hay que preguntar antes de desplegar no es cuánto escala, sino qué más se lleva consigo cuando escala — y qué señal existe cuando decide no escalar en absoluto. Un componente en cero réplicas con trabajo pendiente es el fallo más silencioso de esta parte, porque desde fuera es indistinguible de un sistema en reposo.

Laboratorio guiado

Ejecuta desde la raíz:

```
python classes/part-03-azure-core-platform/043-app-service-functions-y-container-apps/lab.py
```

El laboratorio selecciona el motor de práctica serverless y produce labresult.json. El escenario, sus comprobaciones y el artefacto esperado corresponden a esta clase; no requiere credenciales y deja explícito qué debe revalidarse en un sandbox real.

1. Lee exercise.steps y formula una predicción antes de ejecutar.
2. Ejecuta la práctica y verifica todos los elementos de checks.
3. Provoca el caso de negativetest y explica la señal observada.
4. Materializa api-plataforma-azure en evidence/ usando la plantilla indicada.
5. Para proveedor real, sigue sandbox.requires, registra costo y ejecuta destroy.

Evidencia esperada

El artefacto principal es una función con límites, reintentos e idempotencia. Además, la entrega debe incluir el comando ejecutado, la salida estructurada y una conclusión que no exceda lo observado.

Reto verificable

Construye api-plataforma-azure para el caso CloudShop. Incluye una alternativa descartada, un supuesto que pueda falsarse, una prueba de fallo y una decisión de rollback.

■ Criterio de aceptación

- [ ] lab.py termina con código 0 y genera JSON válido.
- [ ] La entrega conecta al menos tres requisitos con mecanismos verificables.
- [ ] Existe una prueba positiva y una prueba negativa con evidencia.
- [ ] Seguridad, costo y operación aparecen como decisiones, no como anexos.
- [ ] Se declara una limitación y una condición que obligaría a revisar el diseño.
- [ ] Otra persona puede repetir el recorrido sin conocimiento tácito.

Errores frecuentes

Síntoma	Causa probable	Corrección
Una aplicación se degrada sin que su tráfico ni su CPU hayan cambiado	Comparte plan de App Service con otra que consume la CPU del plan	Reparte las aplicaciones por radio de impacto: el plan es la frontera, no una agrupación de conveniencia.
Fallos intermitentes hacia un servicio externo que confirma no recibir las solicitudes	Agotamiento de puertos de traducción de salida por crear un cliente HTTP en cada petición	Reutiliza el cliente, integra con red virtual y NAT Gateway, y usa puntos de conexión privados para los destinos de Azure.
Tras un intercambio de espacios, la aplicación escribe en la base de datos equivocada	La cadena de conexión no estaba marcada como fijada al espacio, así que viajó con la aplicación	Marca como fijado al espacio todo lo que identifique al entorno y revisa la lista en cada cambio de configuración.
Una función HTTP se degrada cuando llega una avalancha a una cola distinta	En Azure el escalado es por aplicación de funciones, no por función	Una aplicación de funciones por perfil de carga: lo que responde a un usuario no convive con lo que procesa en segundo plano.
Un proceso largo se corta a los diez minutos y subir la memoria no lo arregla	El plan de consumo tiene un tope duro de diez minutos por ejecución	Parte el trabajo con Durable Functions o usa consumo flexible, premium o dedicado.
Una cola crece durante días sin que nadie reciba un error	El consumidor tiene una regla de escalado HTTP y se redujo a cero réplicas	Escala por longitud de cola, fija un mínimo de una réplica y alerta sobre la profundidad de la cola, no sobre la salud del proceso.

Seguridad, ética y costo

Trabaja con cuentas propias o sandboxes autorizados. No publiques secretos, identificadores reales ni datos personales. Antes de crear recursos pagos define presupuesto, etiquetas y comando de destrucción; después verifica que no queden recursos huérfanos. Los ejemplos locales enseñan contratos, pero no certifican cumplimiento ni disponibilidad de producción.

Preguntas de comprobación

1. ¿Cuál es la unidad de escalado en App Service, Functions y Container Apps, y qué se arrastra en cada caso?
2. Un servicio externo falla de forma intermitente solo en los picos. ¿Qué métrica confirma que el problema es tuyo?
3. ¿Qué viaja y qué se queda en un intercambio de espacios, y cuál es el valor por defecto peligroso?
4. ¿Por qué un proceso de catorce minutos no se arregla con más memoria en el plan de consumo?
5. Un consumidor de cola está en cero réplicas y no hay ningún error. ¿Qué señal debería haber existido?

Referencias

- Microsoft (2025). App Service plan overview — la unidad de escalado y facturación, y el efecto de compartirla. <https://learn.microsoft.com/en-us/azure/app-service/overview-hosting-plans>
- Microsoft (2025). Set up staging environments in App Service — intercambio, ajustes fijados al espacio y calentamiento. <https://learn.microsoft.com/en-us/azure/app-service/deploy-staging-slots>
- Microsoft (2025). Azure Functions hosting options — duración máxima, red virtual y arranque en frío por plan. <https://learn.microsoft.com/en-us/azure/azure-functions/functions-scale>
- Microsoft (2025). Set scaling rules in Azure Container Apps — reglas KEDA, réplicas mínimas y escaladores de cola. <https://learn.microsoft.com/en-us/azure/container-apps/scale-app>
- Microsoft (2025). Troubleshoot SNAT port exhaustion — cupo por instancia, métricas y mitigaciones. <https://learn.microsoft.com/en-us/azure/app-service/troubleshoot-intermittent-outbound-connection-errors>
- Documentación oficial vigente del servicio implementado; registra URL y fecha de consulta.

Descargar: Parte 03 en PDF · Recorrido de Azure en PDF · Manual integral

Evaluación — 043 App Service, Functions y Container Apps

Preguntas

1. Explica el problema que resuelve esta capacidad sin mencionar un proveedor.
2. Identifica una frontera de responsabilidad y su propietario.
3. Compara dos alternativas mediante confiabilidad, seguridad, costo y operación.
4. ¿Qué demuestra el laboratorio y qué no puede demostrar?
5. Propón un caso límite o una prueba de fallo.

Reto verificable

Entrega api-plataforma-azure con diagrama o configuración, evidencia de ejecución, alternativa descartada, riesgo residual y criterio de rollback.

Criterio de aceptación

- [ ] Resultado reproducible por una segunda persona.
- [ ] Evidencia vinculada a cada afirmación técnica.
- [ ] Prueba negativa o de fallo incluida.
- [ ] Costos y permisos expresados con unidades y alcance.
- [ ] Limitaciones declaradas sin presentar la simulación como producción.

Escala

Nivel	Evidencia
A — competente	Cumple todo y defiende trade-offs con datos.
B — en desarrollo	Cumple lo esencial; falta profundidad en una dimensión.
C — inicial	Artefacto parcial o conclusión sin evidencia suficiente.
0	Sin entrega reproducible.

044 — Service Bus, Event Grid y Event Hubs

Parte: 03 — Azure: plataforma esencial Nivel: intermedio · Horas estimadas: 4 Laboratorio: messaging · Estado: EXECUTABLECORE

Propósito

Repartir el trabajo asíncrono entre los tres servicios de mensajería de Azure a partir de una sola pregunta que casi nunca se hace: ¿el mensaje se consume o se observa? Confundirla lleva a poner órdenes de pedido en un registro de eventos que nunca las borra, sin cola de mensajes fallidos y con un evento envenenado bloqueando una partición entera. La clase 033 dejó el criterio de la entrega y la idempotencia; aquí cambian las piezas y aparecen tres relojes —el bloqueo, la ventana de duplicados y el reintento— que hay que dimensionar juntos.

Resultados de aprendizaje

Al finalizar podrás:

1. Elegir entre Service Bus, Event Grid y Event Hubs según si el mensaje se consume, se enruta o se observa.
2. Dimensionar la duración del bloqueo frente al tiempo real de proceso y explicar qué ocurre cuando el primero es menor.
3. Configurar el traslado automático a mensajes fallidos y una alerta sobre él, en los tres servicios.
4. Distinguir la detección de duplicados —con ventana acotada— de la idempotencia del manejador.
5. Decidir el número de particiones sabiendo qué se rompe al cambiarlo después.

Conceptos centrales

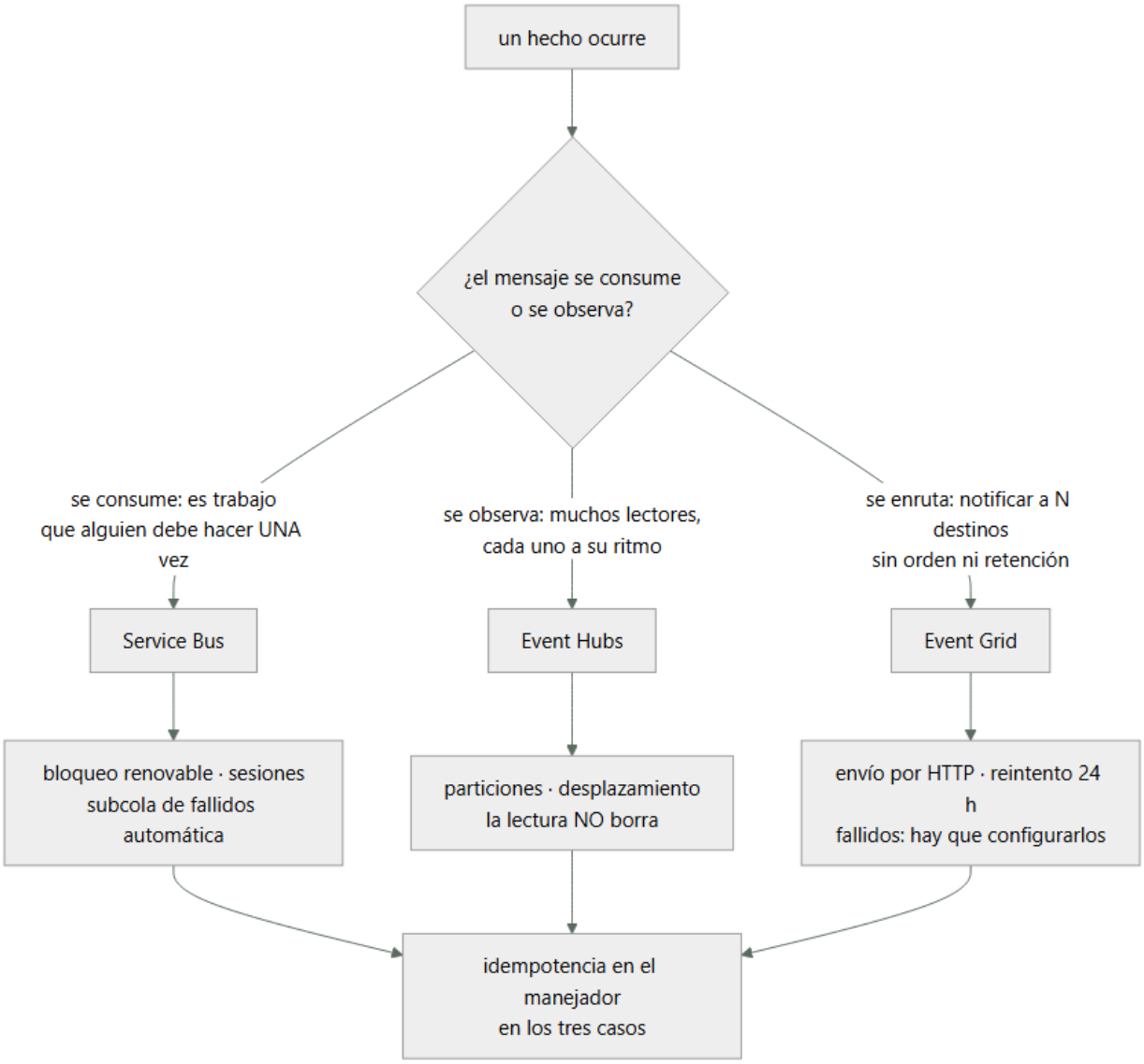
Concepto	Comprensión verificable
bloqueo por inspección	Modo de recepción en el que el mensaje queda reservado un tiempo en vez de borrarse. Equivale al tiempo de invisibilidad de la clase 033 y añade algo que aquel no tiene: se puede renovar.
subcola de mensajes fallidos	Destino automático de un mensaje que agota su cuenta de entregas. En Service Bus existe siempre, cuelga de la propia entidad y registra el motivo en DeadLetterReason.
sesión	Agrupación que garantiza orden dentro de un identificador. Una sesión la bloquea un único consumidor, así que una sesión con mucho tráfico serializa todo su trabajo.
detección de duplicados	Descarte de mensajes con el mismo identificador dentro de una ventana temporal acotada. Fuera de la ventana no descarta nada: no sustituye a la idempotencia.
grupo de consumidores y punto de control	En Event Hubs, la lectura no borra: cada grupo avanza su propio marcador. Si el punto de control se guarda poco, un reinicio reprocesa desde el último guardado.
unidad de rendimiento	Unidad de capacidad de Event Hubs: 1 MB/s de entrada y 2 MB/s de salida. El inflado automático sube y no baja: lo que escala una prueba de carga se paga todo el mes.

Modelo mental

En Azure, identidad y jerarquía organizacional preceden al recurso: tenant, management group, suscripción y resource group determinan el alcance de control.

Aplicado a esta clase, separa siempre cuatro planos: intención (qué necesita el usuario), configuración (qué declaramos), estado observado (qué existe de verdad) y evidencia (cómo sabemos que cumple). Confundirlos produce diseños que se ven correctos en un diagrama pero fallan al operar.

■ Flujo de razonamiento



Desarrollo

1. Una pregunta decide el servicio: ¿se consume o se observa?

Los tres nombres contienen la palabra «evento» o «bus» y eso lleva a elegir por eufonía. La pregunta que separa de verdad es otra:

```
¿el mensaje representa TRABAJO que alguien debe hacer exactamente una vez?
   → Service Bus.  Se consume: al completarlo, desaparece.

¿el mensaje representa un HECHO que varios interesados leen a su ritmo?
   → Event Hubs.  Se observa: leerlo no lo borra, y otro puede releerlo.

¿el mensaje es una NOTIFICACIÓN que hay que repartir a destinos que no conozco?
   → Event Grid.  Se enruta: se entrega por HTTP y no se guarda.
```

La tabla que hace operativa esa elección:

	Service Bus	Event Hubs	Event Grid
Modelo	Cola y tema con suscripciones	Registro con particiones	Enrutador de sucesos
¿Leer borra?	Sí, al completar	No	No aplica: se envía

	Service Bus	Event Hubs	Event Grid
Retención	Hasta completar o caducar	1-7 días (90 en premium)	Solo reintentos
Orden	Por sesión	Por partición	Sin garantía
Mensajes fallidos	Automático	No existe	Configurable, apagado por defecto
Tamaño	256 KB (100 MB en premium)	1 MB	1 MB
Se paga por	Operación o unidad de mensajería	Unidad de rendimiento	Operación

La fila de mensajes fallidos es la que más consecuencias tiene. En Service Bus la subcola existe siempre y recoge lo que falla; en Event Hubs no hay tal cosa, porque el modelo no es de consumo: un evento que el consumidor no sabe procesar se queda en su sitio y, si el consumidor reintenta desde el mismo desplazamiento, bloquea la partición entera — nada posterior se procesa hasta que alguien intervenga. Poner órdenes de pedido en Event Hubs por creer que es «el servicio de eventos» es el error de diseño más caro de esta clase.

El reparto habitual en una plataforma como CloudShop, escrito para no rediscutirlo:

```
Service Bus   crear pedido · cobrar · emitir factura · enviar correo
              trabajo con dueño, con reintentos y con fallo visible

Event Hubs    clics, telemetría de la aplicación, eventos de stock
              volumen alto, varios consumidores, se puede releer

Event Grid    "se subió un blob" · "se creó un recurso" · "se desplegó"
              reacción a hechos de la plataforma y notificación a terceros
```

Y un uso de Event Grid que rentabiliza el trabajo de las clases anteriores: todos los recursos de Azure emiten eventos. Un blob nuevo en el contenedor de facturas dispara la función que lo indexa; la creación de un recurso sin etiquetas dispara la corrección de gobierno de la clase 037. Es la vía por la que la plataforma reacciona a sí misma sin sondeos periódicos.

2. El bloqueo, el reloj que casi nunca se dimensiona

Service Bus recibe de dos formas, y solo una es defendible en producción:

```
recibir y borrar     el mensaje desaparece al entregarse
                     si el proceso falla después, el trabajo se pierde
bloqueo por inspección  el mensaje queda reservado; hay que completarlo,
                     abandonarlo o trasladarlo a fallidos explícitamente
```

El bloqueo cumple la función del tiempo de invisibilidad de la clase 033 y añade una capacidad que aquel no tiene: se puede renovar. Esa diferencia cambia el diseño.

La duración máxima del bloqueo es de cinco minutos. Si el manejador tarda más y nadie renueva:

```
t+0:00  se entrega el mensaje, bloqueo de 5 min
t+5:00  el bloqueo expira; el mensaje vuelve a estar disponible
t+5:01  OTRO consumidor lo recibe y empieza a procesarlo
t+6:20  el primero termina y llama a completar
        → MessageLockLostException
t+7:40  el segundo también termina
        → el pedido se ha cobrado dos veces
```

La secuencia importa porque el error aparece al final, cuando el daño ya está hecho. Y el MessageLockLostException en el registro se suele leer como un fallo transitorio del SDK.

Tres respuestas, de mejor a peor:

```
1. sacar el trabajo largo del manejador
   el mensaje dispara el proceso, no lo ejecuta
2. renovar el bloqueo mientras se trabaja
   los SDK lo hacen automáticamente SI el hilo no está bloqueado
3. subir la duración del bloqueo
   tope de 5 minutos: solo mueve el problema un poco más allá
```

La segunda tiene una letra pequeña que produce incidentes reproducibles: la renovación automática ocurre en segundo plano y necesita que el hilo pueda ejecutarse. Un manejador que bloquea el hilo con una llamada síncrona impide su propia renovación, así que el bloqueo caduca justo en los casos lentos, que son los que más lo necesitan.

El traslado automático a fallidos es la otra mitad, y en Azure viene puesto:

```
MaxDeliveryCount = 10 por defecto
al superarlo → subcola de fallidos de la propia entidad
```

No hay que crear ninguna cola ni configurar ninguna política de reenvío: existe. Y como existe sin que nadie la pida, también se llena sin que nadie la mire. El motivo queda registrado y es la primera cosa que hay que consultar:

```
$ az servicebus queue show -g rg-msg --namespace-name sb-cloudshop -n pedidos \
    --query "countDetails.deadLetterMessageCount" -o tsv
8412
```

```
DeadLetterReason
  MaxDeliveryCountExceeded   el manejador falló 10 veces: mira su excepción
  TTLExpiredException        caducó sin que nadie lo consumiera
  HeaderSizeExceeded         propiedades demasiado grandes
  <motivo propio>            el manejador lo trasladó a propósito
```

La distinción entre las dos primeras es diagnóstica: la primera es un consumidor que falla, la segunda es un consumidor que no está. Y la alerta que hay que tener en los tres servicios es la misma y casi nunca está: cuenta de mensajes fallidos mayor que cero durante N minutos. Un sistema asíncrono sano no acumula fallidos; que los acumule en silencio es la forma en que se pierde trabajo sin que nadie reciba un error.

3. Detección de duplicados, sesiones y filtros: lo que garantizan y lo que no

La clase 033 separó el duplicado de entrega del fallo de idempotencia. Azure ofrece un mecanismo que parece resolverlo y solo cubre una parte:

```
detección de duplicados
  descarta mensajes con el mismo MessageId
  DENTRO de una ventana temporal
  por defecto 30 s; hasta 7 días en premium
```

La ventana es el detalle que decide. Un cliente que reintenta a los 45 segundos con la ventana en 30 crea un duplicado real, y el sistema lo entrega dos veces. Además la ventana consume almacenamiento y capacidad: fijarla en siete días para «asegurarse» tiene un precio y sigue sin cubrir el reintento manual de un operador tres semanas después.

La conclusión es la de la clase 033, reforzada: la detección de duplicados reduce el ruido; la idempotencia del manejador es la que garantiza corrección. Un manejador idempotente hace innecesaria la primera; la primera nunca hace innecesario al segundo.

```
def manejar(mensaje):
    clave = mensaje.application_properties[b"idempotency-key"].decode()
    if repositorio.ya_procesado(clave):        # la garantía real
        return                                  # completar sin reejecutar
    with repositorio.transaccion():
        aplicar_efecto(mensaje)
        repositorio.marcar_procesado(clave)     # en la MISMA transacción
```

La última línea es la que hace que funcione: marcar y aplicar el efecto en la misma transacción. Marcadas por separado, un fallo entre ambas reproduce exactamente el problema que se quería evitar.

Las sesiones dan orden dentro de un identificador —todos los mensajes de un pedido, en secuencia— y tienen un costo estructural que hay que anticipar: una sesión la bloquea un único consumidor, así que dentro de una sesión no hay paralelismo. Si el identificador de sesión es demasiado grueso, se ha construido una cola de un solo hilo:

```
sesión = idPedido      miles de sesiones, buen reparto            ✓
sesión = idTienda      una tienda grande serializa todo su trabajo ✗
sesión = "pedidos"     una sola sesión: concurrencia = 1          ✗✗
```

Es el mismo razonamiento de la clave de partición de la clase 042: la unidad de orden es también la unidad de serialización. Solo se pide orden donde el negocio lo exige, y con el identificador más fino que lo satisfaga.

Los temas con suscripciones y filtros sustituyen al abanico de la clase 033, con dos matices:

```
$ az servicebus topic subscription rule create -g rg-msg \
    --namespace-name sb-cloudshop --topic-name pedidos --subscription-name facturacion \
    --name solo-pagados --correlation-filter properties.estado=pagado
```

Primero: un filtro de correlación —igualdad sobre propiedades— se evalúa mucho más barato que uno con sintaxis SQL. Con volumen alto, la diferencia es de factura.

Segundo: una suscripción sin ninguna regla recibe todo, y una suscripción creada hoy no recibe lo publicado ayer. La primera produce consumidores que procesan mensajes que no les tocaban; la segunda produce el hueco de datos al añadir un consumidor nuevo, que hay que rellenar a mano si importaba.

Y una decisión de nivel que se toma tarde y duele: el nivel estándar es multiinquilino, factura por operación y puede limitar; el premium reserva unidades de mensajería, da latencia predecible, admite mensajes de 100 MB y es el único que acepta punto de conexión privado. Ese último punto conecta con la clase 039: si la plataforma cerró los accesos públicos, el nivel estándar no puede participar. El salto de precio es considerable, así que conviene decidirlo con el diseño de red delante y no cuando ya no hay alternativa.

4. Event Hubs: particiones, puntos de control y el inflado que no desinfla

Event Hubs es un registro particionado, no una cola. Las tres consecuencias que definen su operación:

```
el orden es POR PARTICIÓN, nunca global
leer NO borra: el consumidor avanza un marcador propio
cada grupo de consumidores tiene su propio marcador
```

La clave de partición decide a qué partición va cada evento, y por tanto qué queda ordenado respecto a qué. Es la misma decisión de la clase 042 con otro nombre. Y el número de particiones es, en la práctica, una decisión de un solo sentido: cambiarlo altera a qué partición corresponde cada clave, así que el orden por clave se rompe en el punto del cambio y los consumidores que dependían de él procesan eventos desordenados durante la transición.

```
regla útil    particiones ≥ consumidores concurrentes máximos previstos
              porque una partición la lee UN consumidor por grupo
              particiones de más: coste de gestión, no de dinero
              particiones de menos: techo de paralelismo que no se puede subir
                                    sin romper el orden
```

El punto de control es la otra fuente de incidentes. El consumidor guarda su posición en una cuenta de almacenamiento, y la frecuencia con la que lo hace define cuánto se reprocesa tras un reinicio:

```
punto de control cada 10.000 eventos, a 30.000 eventos/min
→ hasta 20 s de eventos reprocesados en cada reinicio

punto de control cada 10.000 eventos, a 500 eventos/min
→ hasta 20 MINUTOS reprocesados en cada reinicio
```

La misma configuración produce dos comportamientos incomparables según el caudal. Por eso el punto de control se fija por tiempo además de por cantidad, y por eso el consumidor tiene que ser idempotente igual que el de Service Bus: el reprocesamiento no es una anomalía, es el funcionamiento normal tras cualquier reinicio.

Y un fallo de operación con firma propia: dos instancias del mismo grupo de consumidores compitiendo por la misma partición. La segunda toma el arrendamiento y expulsa a la primera; la primera reintenta y expulsa a la segunda. El resultado es un vaivén de arrendamientos, ningún avance del marcador y un consumo de CPU alto sin trabajo hecho. Se reconoce en el registro por el ciclo continuo de adquisición y pérdida de particiones.

El rendimiento se compra en unidades y ahí está la trampa de costo de esta clase:

```
1 unidad de rendimiento ≈ 1 MB/s de entrada, 2 MB/s de salida, 1.000 eventos/s
al superarlo → ServerBusyException, que el SDK reintenta
inflado automático → SUBE solo hasta el máximo configurado
                  → y NO BAJA nunca
```

Una prueba de carga que empuje de 2 a 20 unidades deja el espacio de nombres en 20 hasta que una persona lo baje:

```
2 unidades de rendimiento       ~44
20 tras la prueba de carga     ~440
```

Cuatrocientos dólares al mes por un experimento de veinte minutos, sin ninguna alerta que lo señale. La medida preventiva es fijar el máximo del inflado en un valor que se pueda pagar y revisar el valor actual como parte del cierre de cualquier prueba de carga.

Dos capacidades que conviene conocer porque ahorran trabajo:

Captura. Archiva automáticamente en almacenamiento en formato Avro, por ventana de tiempo o de tamaño. Es la forma más barata de tener el flujo en bruto para reprocesar, y encaja con las reglas de ciclo de vida de la clase 041 sin escribir un solo consumidor.

Compatibilidad con Kafka. Event Hubs habla el protocolo de Kafka, así que un productor o consumidor existente funciona cambiando la cadena de conexión. Para un programa multinube esto es un dato de arquitectura, no una curiosidad: el contrato de la aplicación deja de ser específico del proveedor aunque el servicio lo sea.

5. Event Grid: entrega por HTTP, reintentos y el saludo que falla

Event Grid empuja, no se consulta. Eso simplifica el consumidor —una función HTTP y nada más— y traslada al emisor tres responsabilidades que en una cola no existían.

El saludo de validación. Antes de entregar nada a un extremo HTTP, Event Grid envía un evento de validación y espera que el extremo devuelva el código recibido. Es la protección contra usar el servicio para inundar a un tercero, y es el fallo de puesta en marcha número uno: la suscripción se crea, no entrega nada y el extremo no registra ningún error porque nunca llegó a llamarse para lo demás.

```
if tipo == "Microsoft.EventGrid.SubscriptionValidationEvent":
    return {"validationResponse": datos["validationCode"]}
```

Con funciones de Azure o colas como destino, el saludo lo resuelve la plataforma; con un extremo propio, hay que escribirlo.

El reintento y su final. Event Grid reintenta con retroceso exponencial hasta 24 horas por defecto. Pasado ese plazo, descarta el evento, y el destino de mensajes fallidos está apagado:

```
$ az eventgrid event-subscription update --name indexar-facturas \
    --source-resource-id $ST_ID \
    --deadletter-endpoint "$ST_ID/blobServices/default/containers/eventos-fallidos"
```

Sin esa línea, un extremo caído durante un día produce pérdida de eventos sin traza. Con ella, los eventos aterrizan en un contenedor con su motivo, y la lección de la clase 041 vuelve a aplicar: ese contenedor necesita su propia regla de ciclo de vida o crece para siempre.

Ni orden ni unicidad. Event Grid no garantiza el orden y puede entregar el mismo evento más de una vez. Un manejador que asume orden —«el evento de creación llega antes que el de actualización»— funciona en desarrollo y falla en producción de forma intermitente. La defensa es la misma de todo el resto de la clase: idempotencia, y usar la marca de tiempo o la versión del recurso para descartar lo viejo en vez de confiar en el orden de llegada.

Un cierre que une las tres piezas. Los tres servicios entregan al menos una vez, ninguno entrega exactamente una vez, y los tres exigen lo mismo del manejador:

```
Service Bus   duplica al expirar un bloqueo o al reintentar el cliente
Event Hubs    duplica al reprocesar desde el último punto de control
Event Grid    duplica al reintentar la entrega
```

La entrega exactamente una vez no se compra: se construye en el manejador. Es la misma conclusión de la clase 033, y después de tres servicios con tres mecanismos distintos ya no es una recomendación teórica: es la única propiedad que se conserva al cambiar de servicio, y por tanto la única que merece la pena escribir una sola vez y reutilizar.

Ejemplo trabajado

CloudShop monta su columna asíncrona en Azure. El equipo elige Event Hubs para todo «porque es el servicio de eventos y escala más». Cinco incidentes en un mes reparten cada flujo donde le corresponde.

Punto de partida:

```
eh-cloudshop (Event Hubs, 4 particiones, 2 unidades de rendimiento)
  pedidos · pagos · telemetria · notificaciones
todos los consumidores en el mismo grupo, punto de control cada 10.000 eventos
```

Incidente 1 — una partición deja de avanzar y 12.000 pedidos se detienen.

Un pedido con un campo nuevo hace fallar al deserializador. El consumidor reintenta desde el mismo desplazamiento, indefinidamente.

```
partición 2   desplazamiento 4.481.209   sin avanzar desde hace 3 h 40 min
pedidos detenidos detrás del envenenado   12.184
```

No hay subcola de fallidos: Event Hubs no la tiene, porque leer no consume. La solución de urgencia es saltar el desplazamiento a mano; la de fondo es mover el flujo:

```
pedidos          Event Hubs        Service Bus (cola con sesiones por idPedido)
```

```
pagos            Event Hubs        Service Bus (tema con suscripciones filtradas)
telemetria       Event Hubs        Event Hubs                      ← se queda
notificaciones   Event Hubs        Event Grid (reacción a blob y a estado)
```

El criterio queda escrito en una línea: lo que alguien debe hacer una vez va a Service Bus; lo que muchos observan a su ritmo se queda en Event Hubs.

Incidente 2 — dos pedidos cobrados dos veces.

Con la cola ya en Service Bus, el manejador de cobro tarda entre 4 y 7 minutos porque llama a la pasarela y espera la confirmación.

```
MessageLockLostException al completar    41 veces en 24 h
cobros duplicados detectados por conciliación   2
```

El bloqueo era de cinco minutos y el manejador bloqueaba el hilo con una llamada síncrona, impidiendo su propia renovación. Se corrige por el lado del diseño, no del reloj:

```
trabajo dentro del manejador    llamada de 4-7 min   registrar intención y salir
cobro efectivo                  en el manejador      proceso propio con reintento
duración del bloqueo                 5 min              30 s (sobra)
clave de idempotencia               ninguna          idPedido + intento,
                                                     marcada en la MISMA transacción
cobros duplicados                       2                  0
```

La clave de idempotencia es la que cierra el caso: aunque el mensaje vuelva a entregarse, el efecto no se repite.

Incidente 3 — 8.412 mensajes fallidos que nadie miró en tres semanas.

```
$ az servicebus queue show -g rg-msg --namespace-name sb-cloudshop -n facturacion \
    --query "countDetails.deadLetterMessageCount" -o tsv
8412
$ # motivo del primero
MaxDeliveryCountExceeded · System.NullReferenceException en importe_neto
```

Un cambio de esquema tres semanas atrás. Cada mensaje se reintentó diez veces y acabó en la subcola, que existía desde el principio y no tenía ninguna alerta.

```
alerta sobre mensajes fallidos       ninguna     > 0 durante 15 min → aviso
procedimiento de reproceso           ninguno     documentado y ensayado
validación de esquema en el emisor   no          sí, contrato versionado
mensajes recuperados                  —          8.412 reprocesados
```

Incidente 4 — eventos de facturas perdidos durante un día.

La función que indexa facturas estuvo caída 26 horas por un despliegue fallido. Al recuperarla, faltan 1.900 documentos.

```
$ az eventgrid event-subscription show --name indexar-facturas \
    --source-resource-id $ST_ID --query "deadLetterDestination"
null
```

Event Grid reintentó 24 horas y descartó el resto. Sin destino de fallidos, no queda rastro. Se configura, y se reconstruye el hueco releyendo el contenedor —posible porque los blobs siguen ahí, no porque el sistema de eventos lo permitiera.

```
destino de mensajes fallidos       ninguno    contenedor eventos-fallidos
regla de ciclo de vida sobre él        —      borrado a 30 días (clase 041)
alerta de entregas fallidas           no      > 50 en 10 min → aviso
```

Incidente 5 — la factura de Event Hubs se multiplica por diez sin más tráfico.

```
$ az eventhubs namespace show -g rg-msg -n eh-cloudshop \
    --query "{tu:sku.capacity,inflar:isAutoInflateEnabled,max:maximumThroughputUnits}" -o tsv
20   True   20
```

Veinte unidades de rendimiento con un tráfico que necesita dos. El inflado automático había subido durante una prueba de carga de veinte minutos, tres semanas atrás, y nunca bajó — porque no baja.

```
unidades de rendimiento               20            2
máximo del inflado automático         20            6
costo mensual de Event Hubs         ~440 USD      ~44 USD
revisión tras prueba de carga       ninguna    paso obligatorio del cierre
```

Y de paso se corrige el punto de control, que con el caudal real de telemetría reprocesaba seis minutos en cada reinicio:

```
punto de control   cada 10.000 eventos   →   cada 10.000 eventos o 30 s
reproceso tras reinicio   ~6 min          →   < 30 s
```

Resumen de la columna asíncrona:

```
servicios usados                        1 (para todo)    3, por criterio escrito
trabajo detenido sin señal              12.184 pedidos          0
cobros duplicados                             2                 0
mensajes fallidos sin vigilar             8.412           alertados a 15 min
eventos perdidos sin rastro               1.900     con destino de fallidos
unidades de rendimiento pagadas              20                 2
manejadores idempotentes                   0 de 4             4 de 4
costo mensual de mensajería               ~465 USD          ~118 USD
```

La lección que esta clase traslada al resto de la parte: los tres servicios entregan al menos una vez y ninguno entrega exactamente una vez. Lo que hace correcto a un sistema asíncrono no es el servicio elegido sino la propiedad que se escribe en el manejador, y esa propiedad es la única que sobrevive intacta cuando el servicio cambia — que es exactamente lo que este programa entiende por portabilidad.

Laboratorio guiado

Ejecuta desde la raíz:

```
python classes/part-03-azure-core-platform/044-service-bus-event-grid-y-event-hubs/lab.py
```

El laboratorio selecciona el motor de práctica messaging y produce labresult.json. El escenario, sus comprobaciones y el artefacto esperado corresponden a esta clase; no requiere credenciales y deja explícito qué debe revalidarse en un sandbox real.

1. Lee exercise.steps y formula una predicción antes de ejecutar.
2. Ejecuta la práctica y verifica todos los elementos de checks.
3. Provoca el caso de negativetest y explica la señal observada.
4. Materializa flujo-eventos-azure en evidence/ usando la plantilla indicada.
5. Para proveedor real, sigue sandbox.requires, registra costo y ejecuta destroy.

Evidencia esperada

El artefacto principal es un flujo que declara orden, entrega y manejo de errores. Además, la entrega debe incluir el comando ejecutado, la salida estructurada y una conclusión que no exceda lo observado.

Reto verificable

Construye flujo-eventos-azure para el caso CloudShop. Incluye una alternativa descartada, un supuesto que pueda falsarse, una prueba de fallo y una decisión de rollback.

■ Criterio de aceptación

- [ ] lab.py termina con código 0 y genera JSON válido.
- [ ] La entrega conecta al menos tres requisitos con mecanismos verificables.
- [ ] Existe una prueba positiva y una prueba negativa con evidencia.
- [ ] Seguridad, costo y operación aparecen como decisiones, no como anexos.
- [ ] Se declara una limitación y una condición que obligaría a revisar el diseño.
- [ ] Otra persona puede repetir el recorrido sin conocimiento tácito.

Errores frecuentes

Síntoma	Causa probable	Corrección
Una partición deja de avanzar y el trabajo posterior se detiene sin errores visibles	Un evento envenenado en Event Hubs, que no tiene subcola de mensajes fallidos porque leer no consume	Usa Service Bus para lo que debe procesarse una vez; en Event Hubs, aparta el evento y avanza el desplazamiento con registro del descarte.
MessageLockLostException al completar y efectos duplicados	El manejador tarda más que el bloqueo y bloquea el hilo, impidiendo su propia renovación	Saca el trabajo largo del manejador, no bloquees el hilo, y aplica una clave de idempotencia en la misma transacción que el efecto.
Miles de mensajes en la subcola de fallidos sin que nadie se entere	El traslado automático existe desde el principio y no tiene alerta asociada	Alerta sobre la cuenta de mensajes fallidos mayor que cero y documenta un procedimiento de reproceso ensayado.
Se pierden eventos de Event Grid tras una caída prolongada del destino	El reintento dura 24 h y el destino de mensajes fallidos está apagado por defecto	Configura el destino de fallidos, ponle su regla de ciclo de vida y alerta sobre las entregas fallidas.
La factura de Event Hubs se multiplica sin que el tráfico haya cambiado	El inflado automático sube y nunca baja, y una prueba de carga dejó el valor alto	Fija un máximo asumible y revisa las unidades activas como paso obligatorio del cierre de toda prueba de carga.
Se activa la detección de duplicados y siguen llegando repetidos	La ventana es acotada —30 s por defecto— y el reintento cae fuera de ella	Trátala como reducción de ruido: la corrección la garantiza la idempotencia del manejador.

Seguridad, ética y costo

Trabaja con cuentas propias o sandboxes autorizados. No publiques secretos, identificadores reales ni datos personales. Antes de crear recursos pagos define presupuesto, etiquetas y comando de destrucción; después verifica que no queden recursos huérfanos. Los ejemplos locales enseñan contratos, pero no certifican cumplimiento ni disponibilidad de producción.

Preguntas de comprobación

1. ¿Qué pregunta separa Service Bus de Event Hubs, y qué ocurre exactamente si se elige mal para un flujo de pedidos?
2. Un manejador tarda siete minutos y el bloqueo dura cinco. Describe la secuencia completa hasta el efecto duplicado.
3. ¿Por qué la detección de duplicados no sustituye a la idempotencia, y qué hace idempotente a un manejador?
4. ¿Qué se rompe al cambiar el número de particiones de un centro de eventos, y qué regla usarías para elegirlo?
5. ¿Qué tres cosas hay que configurar en Event Grid que no vienen puestas, y qué se pierde sin cada una?

Referencias

- Microsoft (2025). Compare Azure messaging services — cuándo Service Bus, cuándo Event Grid y cuándo Event Hubs. <https://learn.microsoft.com/en-us/azure/service-bus-messaging/compare-messaging-services>
- Microsoft (2025). Service Bus dead-letter queues — traslado automático, motivos y reproceso. <https://learn.microsoft.com/en-us/azure/service-bus-messaging/service-bus-dead-letter-queues>
- Microsoft (2025). Message sessions and duplicate detection — orden por sesión y ventana de duplicados. <https://learn.microsoft.com/en-us/azure/service-bus-messaging/message-sessions>
- Microsoft (2025). Event Hubs features — particiones, grupos de consumidores, puntos de control y captura. <https://learn.microsoft.com/en-us/azure/event-hubs/event-hubs-features>
- Microsoft (2025). Event Grid delivery and retry — saludo de validación, reintentos y mensajes fallidos. <https://learn.microsoft.com/en-us/azure/event-grid/delivery-and-retry>

- Documentación oficial vigente del servicio implementado; registra URL y fecha de consulta.

Descargar: Parte 03 en PDF · Recorrido de Azure en PDF · Manual integral

Anterior	Índice	Siguiente
← 043 · App Service, Functions y Container Apps	Parte 03 · Programa	045 · Azure Monitor, Log Analytics y Application Insights →

Evaluación — 044 Service Bus, Event Grid y Event Hubs

Preguntas

1. Explica el problema que resuelve esta capacidad sin mencionar un proveedor.
2. Identifica una frontera de responsabilidad y su propietario.
3. Compara dos alternativas mediante confiabilidad, seguridad, costo y operación.
4. ¿Qué demuestra el laboratorio y qué no puede demostrar?
5. Propón un caso límite o una prueba de fallo.

Reto verificable

Entrega flujo-eventos-azure con diagrama o configuración, evidencia de ejecución, alternativa descartada, riesgo residual y criterio de rollback.

Criterio de aceptación

- [ ] Resultado reproducible por una segunda persona.
- [ ] Evidencia vinculada a cada afirmación técnica.
- [ ] Prueba negativa o de fallo incluida.
- [ ] Costos y permisos expresados con unidades y alcance.
- [ ] Limitaciones declaradas sin presentar la simulación como producción.

Escala

Nivel	Evidencia
A — competente	Cumple todo y defiende trade-offs con datos.
B — en desarrollo	Cumple lo esencial; falta profundidad en una dimensión.
C — inicial	Artefacto parcial o conclusión sin evidencia suficiente.
0	Sin entrega reproducible.

045 — Azure Monitor, Log Analytics y Application Insights

Parte: 03 — Azure: plataforma esencial Nivel: intermedio · Horas estimadas: 4 Laboratorio: observability · Estado: EXECUTABLECORE

Propósito

Construir la evidencia operativa de una plataforma Azure sabiendo dos cosas que cambian todo lo demás: los registros de recurso están apagados por defecto, así que un incidente ocurrido antes de activarlos no tiene datos que analizar; y la misma señal puede almacenarse como métrica o como registro, con una diferencia de precio de dos órdenes de magnitud. La clase 034 dejó las cuatro preguntas de un incidente; aquí se responden con KQL y se paga la factura correcta por hacerlo.

Resultados de aprendizaje

Al finalizar podrás:

1. Distinguir métricas de registros por costo, latencia y capacidad de consulta, y elegir dónde va cada señal.
2. Activar registros de recurso mediante directiva en vez de recurso por recurso, y comprobar la cobertura.
3. Escribir consultas KQL que respondan tasa de error, percentil de latencia y reconstrucción de una traza.
4. Reducir la factura de ingesta con filtrado en la regla de recopilación y planes por tabla, sin perder la señal.
5. Elegir entre alerta de métrica y alerta de registro sabiendo qué latencia añade cada una.

Conceptos centrales

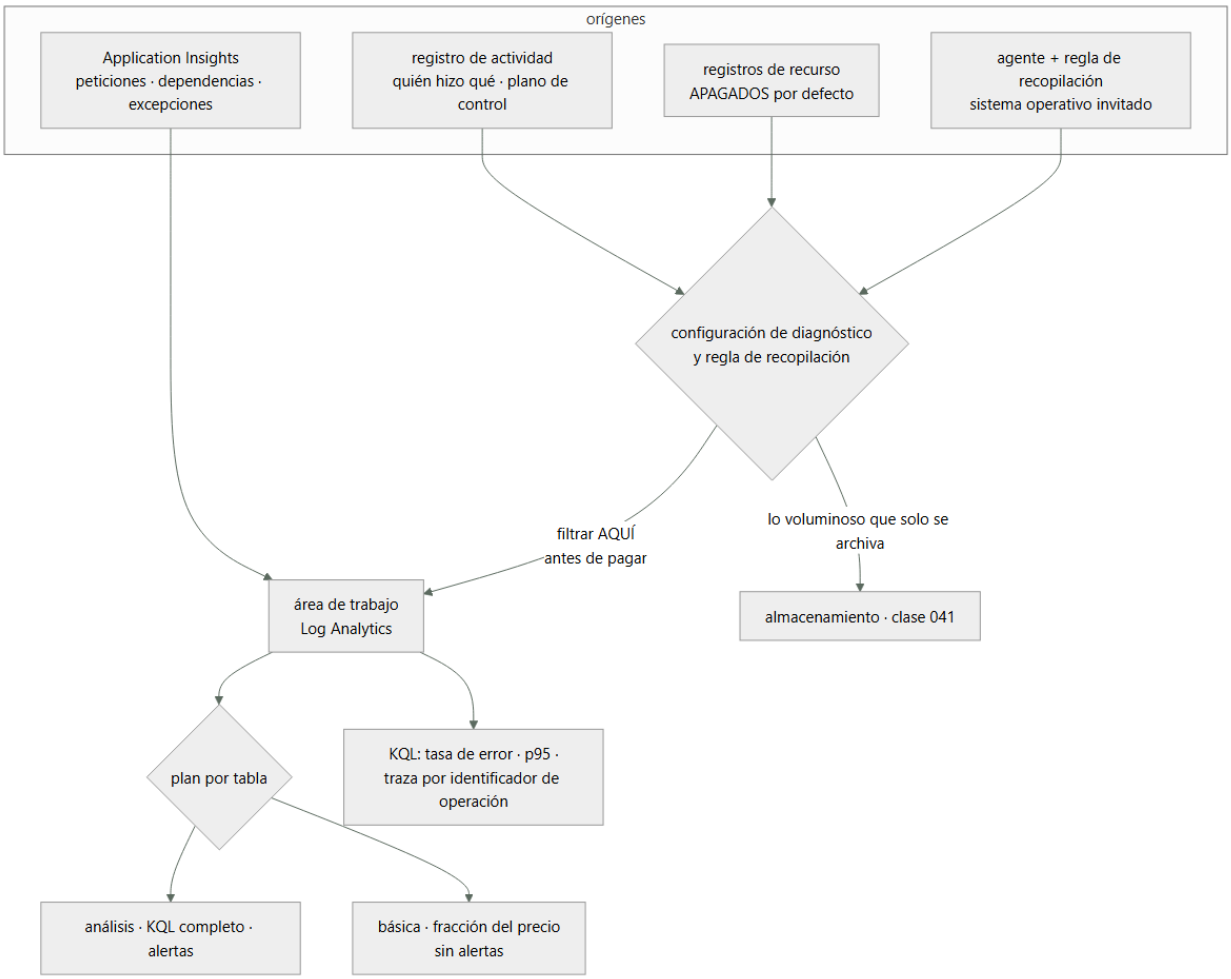
Concepto	Comprensión verificable
configuración de diagnóstico	Interruptor que envía los registros de un recurso a un destino. Está apagado por defecto en todos los recursos: sin él, el plano de datos no deja rastro.
regla de recopilación de datos	Define qué se recoge, cómo se transforma y adónde va. Es el punto donde se filtra antes de pagar la ingesta, no después.
plan por tabla	Cada tabla del área de trabajo puede ser de análisis, básica o auxiliar. La básica cuesta una fracción y renuncia a alertas y a parte de KQL.
muestreo adaptativo	Reducción automática del volumen que envía el SDK de Application Insights. Hace que count() mienta: hay que sumar itemCount.
identificador de operación	Correlador que atraviesa servicios siguiendo el estándar de traza del W3C. Es lo que convierte cuatro registros dispersos en una sola historia.
alerta de registro frente a alerta de métrica	La de métrica se evalúa en la canalización de métricas y avisa en cerca de un minuto; la de registro espera a la ingesta y a su ventana, y rara vez baja de cinco.

Modelo mental

En Azure, identidad y jerarquía organizacional preceden al recurso: tenant, management group, suscripción y resource group determinan el alcance de control.

Aplicado a esta clase, separa siempre cuatro planos: intención (qué necesita el usuario), configuración (qué declaramos), estado observado (qué existe de verdad) y evidencia (cómo sabemos que cumple). Confundirlos produce diseños que se ven correctos en un diagrama pero fallan al operar.

■ Flujo de razonamiento



Desarrollo

1. Nada se registra hasta que alguien lo enciende

Esta es la frase que hay que aprender antes que cualquier consulta. En Azure hay cuatro orígenes de telemetría y solo uno funciona sin configurar nada:

```
registro de actividad     plano de control: quién creó, modificó o borró qué
                          ACTIVO siempre · 90 días · gratis
registros de recurso      plano de datos: consultas SQL, accesos a blobs,
                          peticiones de la pasarela
                          APAGADOS por defecto en TODOS los recursos
sistema operativo invitado agente + regla de recopilación
                          hay que instalarlo
aplicación                Application Insights
                          hay que instrumentarla
```

La segunda línea es la que produce la conversación más incómoda de un postmortem: hubo una caída, se abre el portal a buscar qué pasó y no hay nada que mirar, porque la configuración de diagnóstico de ese recurso nunca se creó. Y no se puede recuperar: los registros no existían.

Activarlo recurso por recurso no funciona a escala, por el mismo motivo que cualquier control manual: alguien despliega uno nuevo y nadie se acuerda. La solución es la de la clase 037 aplicada a la telemetría — una directiva que lo despliegue por sí sola:

```
$ az policy assignment create --name diagnostico-obligatorio \
    --scope "/subscriptions/$SUB" \
    --policy-set-definition "Deploy Diagnostic Settings to Azure Services" \
    --location westeurope --identity-scope "/subscriptions/$SUB" \
    --role Contributor -p "{\"logAnalytics\":{\"value\":\"$WORKSPACE_ID\"}}"
```

El efecto DeployIfNotExists corrige lo que existe y cubre lo que venga. Y la comprobación —que es el entregable, no la asignación— consiste en preguntar cuántos recursos siguen sin ella:

```
resources
| where type in~ ("microsoft.sql/servers/databases",
                  "microsoft.network/applicationgateways",
                  "microsoft.storage/storageaccounts")
| join kind=leftanti (
    insightsresources
    | where type =~ "microsoft.insights/diagnosticsettings"
    | project id = tolower(tostring(split(id, "/providers/microsoft.insights")[0]))
  ) on $left.id == $right.id
| summarize sin_diagnostico = count() by type
```

Sobre el registro de actividad hay un límite que sorprende tarde: se conserva 90 días y no más. Una investigación sobre algo ocurrido hace cuatro meses no encuentra nada. Si la organización necesita responder «quién borró esto» más allá de ese plazo —y las auditorías lo necesitan—, hay que exportarlo con una configuración de diagnóstico, y el destino natural es una cuenta de almacenamiento con inmutabilidad de la clase 041: un registro de auditoría que el propio administrador puede borrar no es un registro de auditoría.

2. Métricas y registros: la misma señal con dos precios

Azure Monitor guarda dos tipos de datos, y la elección entre ellos es una decisión de arquitectura con consecuencias de factura:

	Métricas	Registros
Forma	Serie temporal numérica	Registros estructurados
Latencia hasta ser consultable	1 min	1-3 min, más si hay limitación
Retención	93 días, incluida	Configurable, se paga
Dimensiones	Pocas y acotadas	Las que quieras
Costo de las de plataforma	Prácticamente nulo	2,30 USD por GB ingerido
Sirven para	Alertar rápido, ver tendencia	Investigar, correlacionar, auditar

La clase 034 avisaba de que la cardinalidad multiplica la factura. En Azure ese principio se concreta en una pregunta por cada señal: ¿necesito filtrar esto por un valor que no conozco de antemano? Si la respuesta es no —CPU, latencia media, profundidad de cola—, es una métrica y sale casi gratis. Si es sí —el identificador del cliente, la ruta exacta, el mensaje de error—, es un registro y se paga por GB.

El error caro consiste en emitir como registro lo que podía ser métrica: una línea de registro por petición para poder contar peticiones. Contar peticiones es exactamente lo que hace una métrica, por una fracción del precio y con menos latencia.

Y hay un tercer nivel entre ambos que casi nadie usa y resuelve el caso voluminoso: los planes por tabla.

```
análisis    ~2,30 USD/GB   KQL completo, alertas, 31 días incluidos
básica      ~0,65 USD/GB   consultas limitadas, SIN alertas, 30 días
auxiliar    ~0,15 USD/GB   volcado barato para búsquedas ocasionales
archivo     ~0,03 USD/GB/mes  hay que lanzar un trabajo de búsqueda para leerlo
```

La decisión se toma por tabla, no por área de trabajo, y ahí está la palanca: los registros de acceso de una pasarela o la salida estándar de decenas de contenedores suelen ser el 70-80 % del volumen y casi nunca son el origen de una alerta. Ponerlos en plan básico no cambia lo que se puede investigar durante un incidente; cambia el precio de guardarlos.

El otro punto de control está antes: la regla de recopilación puede transformar y descartar antes de la ingesta, que es cuando se paga.

```
source
| where TimeGenerated > ago(1h)
| where httpStatus_d >= 400 or timeTaken_d > 1.0
| project TimeGenerated, clientIP_s, requestUri_s, httpStatus_d, timeTaken_d
```

Dos efectos a la vez: se descarta el 95 % de las líneas —las peticiones correctas y rápidas, que ya están contadas en una métrica— y se recorta el ancho de las que quedan. Filtrar después, en la consulta, no ahorra un céntimo: el GB ya se pagó al entrar.

Y una regla de higiene que evita la sorpresa: poner un presupuesto y un límite diario al área de trabajo desde el primer día. El límite corta la ingesta al alcanzarlo, lo cual es malo; despertar un lunes con una factura de cuatro cifras por un componente que se volvió locuaz el viernes es peor, y sin límite no hay nada que lo impida.

3. KQL suficiente para responder las cuatro preguntas

La clase 034 planteaba cuatro preguntas durante un incidente: qué está roto, desde cuándo, qué cambió y quién lo hizo. Con un área de trabajo, las cuatro se responden con el mismo lenguaje.

Qué está roto y cuánto.

```
requests
| where timestamp > ago(1h)
| summarize total = sum(itemCount),
            fallidas = sumif(itemCount, success == false) by operation_Name
| extend tasa_error = round(100.0 * fallidas / total, 2)
| where total > 100
| order by tasa_error desc
```

Desde cuándo, y con qué latencia.

```
requests
| where timestamp > ago(6h) and operation_Name == "POST /api/pedidos"
| summarize p50 = percentile(duration, 50),
            p95 = percentile(duration, 95),
            p99 = percentile(duration, 99) by bin(timestamp, 5m)
| render timechart
```

Qué dependencia lo causa.

```
dependencies
| where timestamp > ago(1h) and success == false
| summarize fallos = sum(itemCount) by target, type, resultCode
| order by fallos desc
```

La historia completa de una petición concreta, que es lo que convierte cuatro paneles en un diagnóstico:

```
let op = "8f2a1c9e4b7d3a05";
union requests, dependencies, exceptions, traces
| where operation_Id == op
| project timestamp, itemType, name, target, resultCode, duration, message
| order by timestamp asc
```

Ese operationId viaja entre servicios en la cabecera traceparent del estándar del W3C. Es la pieza que hace que la traza cruce la pasarela, la aplicación web, la función y la cola. Y tiene una condición: cada salto debe propagar la cabecera. Un cliente HTTP escrito a mano que no la reenvíe corta la traza justo donde empieza a ser interesante.

Quién lo hizo, en el registro de actividad:

```
AzureActivity
| where TimeGenerated > ago(7d)
| where OperationNameValue endswith "/DELETE"
| project TimeGenerated, Caller, OperationNameValue, _ResourceId, ActivityStatusValue
| order by TimeGenerated desc
```

Y ahora la advertencia que hace inútiles todas las consultas anteriores si se ignora. El SDK de Application Insights aplica muestreo adaptativo por defecto: cuando el volumen sube, envía una fracción y anota en itemCount a cuántos representa cada registro.

```
count()                 cuenta REGISTROS ALMACENADOS      41.312
sum(itemCount)          cuenta peticiones REALES         413.120
```

Un factor de diez, silencioso, en cualquier panel escrito con count(). Y peor: una alerta con umbral sobre count() deja de dispararse justo cuando el tráfico sube, porque el muestreo se vuelve más agresivo precisamente entonces. La regla es sencilla y no admite excepción: en tablas de Application Insights se suma itemCount, nunca se cuenta.

Si para un flujo concreto —pagos, por ejemplo— hace falta el detalle completo, el muestreo se desactiva para ese tipo de telemetría en lugar de para toda la aplicación, y se acepta el costo de esa decisión sabiendo cuál es.

4. Alertas: la que llega en un minuto y la que llega en nueve

Hay tres clases de alerta y elegir mal añade minutos al tiempo de detección sin que nadie se dé cuenta:

	Se evalúa sobre	Latencia típica	Costo
Métrica	La canalización de métricas	1 min	Muy bajo por regla
Registro	Una consulta KQL programada	5-15 min	Por regla y frecuencia
Registro de actividad	Sucesos del plano de control	Minutos	Muy bajo

La latencia de una alerta de registro es acumulativa y hay que sumarla entera:

```
ingesta            1-3 min
frecuencia mínima  5 min
ventana            5-15 min según se defina
■■■■■■■■■■■■■■■■■■■■■■■■■■■■
detección real     de 7 a 20 minutos
```

Eso descalifica a la alerta de registro como señal rápida. La distribución correcta:

```
métrica    disponibilidad, tasa de error HTTP, latencia, profundidad de cola,
           CPU, mensajes fallidos  →  todo lo que despierta a alguien
registro   lo que solo se puede expresar consultando: una excepción concreta,
           una combinación de campos, una correlación entre tablas
actividad  cambios sensibles: borrado de recursos, cambios de rol,
           elevación de acceso (clase 038)
```

Dos ajustes que separan una alerta útil de un generador de ruido:

Umbrales dinámicos. Aprenden la estacionalidad de una métrica y avisan de la desviación. Son excelentes para tráfico con patrón diario y semanal —una tienda lo tiene— y son inútiles para una señal que nunca ha estado sana: el modelo aprende que la anomalía es lo normal.

Grupos de acción con la severidad bien puesta. Una alerta que no corresponde a una acción no debería despertar a nadie. El criterio de la clase 034 se mantiene: si la respuesta a una alerta es «ya, eso pasa a veces», la alerta está mal definida o el sistema está mal.

Y la alerta que casi nunca existe y que esta parte ha justificado tres veces —clase 043 con el consumidor en cero réplicas y clase 044 con la subcola de fallidos—: la que detecta trabajo que se acumula. Un proceso caído se nota; un proceso que se apagó ordenadamente y dejó de trabajar no.

```
profundidad de cola creciente durante N minutos
edad del mensaje más antiguo por encima de un umbral
cuenta de mensajes fallidos mayor que cero
retraso del consumidor de Event Hubs respecto al último desplazamiento
```

Las cuatro son métricas, así que son baratas y rápidas. No tenerlas no es una cuestión de presupuesto: es que nadie se acordó de que un sistema puede fallar sin caerse.

Sobre el diseño del área de trabajo, una regla que evita una discusión recurrente: se separa por control de acceso y residencia de datos, no por orden estético. Una sola área de trabajo con acceso en contexto de recurso permite que cada equipo vea los registros de sus propios recursos sin ver los de los demás, y hace posibles las consultas que cruzan servicios — que son justamente las que resuelven los incidentes interesantes. Varias áreas por «tenerlo ordenado» obligan a consultas entre áreas de trabajo para todo y multiplican el costo fijo.

5. Operar la flota sin abrir puertos, y saber qué cambió

La clase 034 cerraba con dos capacidades que aquí tienen otras piezas y el mismo objetivo: entrar sin exponer y saber qué se movió.

Entrar sin exponer. Tres mecanismos, en orden de preferencia:

```
no entrar             la mayoría de diagnósticos se resuelven con telemetría
                      si hace falta entrar a menudo, falta instrumentación
Bastion               sesión por el portal, sin IP pública en la máquina
                      subred AzureBastionSubnet, de la clase 039
comandos remotos      `az vm run-command` ejecuta sin sesión interactiva,
                      queda registrado en el registro de actividad
```

La primera línea es la importante y suele saltarse. Una plataforma en la que hay que abrir una sesión para saber qué ocurre es una plataforma con un problema de observabilidad, no de acceso.

Y una precisión sobre el segundo: los comandos remotos se ejecutan con los permisos del agente, es decir como administrador local, y aparecen en el registro de actividad como una única operación sin el contenido del comando. Conviene saberlo por lo que implica: es un camino de ejecución privilegiada que hay que restringir con RBAC igual que cualquier otro.

Saber qué cambió. Es la pregunta que más veces resuelve un incidente, y tiene dos fuentes:

```
// cambios de configuración en las últimas 6 horas sobre los recursos afectados
AzureActivity
| where TimeGenerated > ago(6h)
| where OperationNameValue has_any ("WRITE", "DELETE", "ACTION")
| where ActivityStatusValue == "Success"
| project TimeGenerated, Caller, OperationNameValue, _ResourceId
| order by TimeGenerated asc
```

El historial de cambios de Azure Resource Graph completa la foto con el antes y el después de cada propiedad, que es lo que distingue «alguien tocó la pasarela» de «alguien cambió el tiempo de espera de 30 a 3 segundos».

Juntar ambas con la ventana del incidente es un hábito que ahorra horas:

```
1. ¿cuándo empezó?              percentil de latencia o tasa de error
2. ¿qué cambió en esa ventana?  registro de actividad e historial de cambios
3. ¿qué traza lo demuestra?     una petición fallida por identificador de operación
4. ¿quién y con qué permiso?    autor del cambio y su rol (clase 038)
```

Y la conclusión que enlaza con el proyecto de la clase 048: estas cuatro consultas se escriben una vez, se guardan en el área de trabajo y se enlazan desde el manual de operación. Un incidente no es el momento de aprender KQL. La diferencia entre una plataforma observada y una instrumentada es que la primera tiene las preguntas escritas antes de necesitarlas.

Ejemplo trabajado

CloudShop lleva un mes en Azure con todo desplegado y nada instrumentado más allá de lo que viene puesto. Una caída de 48 minutos abre cinco frentes, y el primero es que no hay datos de la caída.

Frente 1 — el incidente sin evidencia.

La tienda devuelve 502 durante 48 minutos. Al abrir el portal a investigar:

```
$ az monitor diagnostic-settings list --resource $APPGW_ID --query "value[].name" -o tsv
(vacío)
$ az monitor diagnostic-settings list --resource $SQLDB_ID --query "value[].name" -o tsv
(vacío)
```

Ni la pasarela ni la base de datos habían registrado nada del plano de datos, porque nadie lo activó. La causa se termina deduciendo por eliminación, sin poder demostrarla. Se activa por directiva, no a mano:

```
recursos con diagnóstico configurado    3 de 61   61 de 61
mecanismo                             manual    DeployIfNotExists
recursos nuevos                    sin cobertura  cubiertos al crearse
```

Y el registro de actividad se exporta a almacenamiento con inmutabilidad, porque los 90 días por defecto no cubren una auditoría anual.

Frente 2 — la factura de la telemetría, a los once días.

```
Usage
| where TimeGenerated > ago(7d) and IsBillable
| summarize GB = sum(Quantity)/1000 by DataType
| order by GB desc
```

```
AzureDiagnostics (pasarela, registros de acceso)   154 GB   48 %
ContainerLogStdout                                  98 GB   30 %
AppRequests + AppDependencies                       43 GB   13 %
resto                                               27 GB    9 %
                                                   ■■■■■■
total                                              322 GB en 7 días → ~46 GB/día
```

```
46 GB/día × 30 × 2,30 USD = 3.174 USD/mes solo de ingesta
```

Las dos primeras filas son el 78 % del gasto y nunca habían originado una alerta. Se actúa en los dos puntos, no en uno:

```
registros de acceso de la pasarela        22 GB/día     3 GB/día
  filtrado en la regla de recopilación:
  solo estado >= 400 o duración > 1 s
  el volumen completo se archiva en almacenamiento (clase 041)
salida estándar de contenedores           14 GB/día     5 GB/día en plan básico
  filtrado de líneas de depuración
telemetría de aplicación                   6 GB/día     6 GB/día  ← intacta
resto                                      4 GB/día     2 GB/día

ingesta en plan de análisis (11 GB/día)      759
ingesta en plan básico (5 GB/día)             97
archivo en almacenamiento                     14
                                             ■■■■■■
                                             870      frente a 3.174   (−73 %)
```

Lo que se conserva íntegro es lo que se usa para investigar. Lo que se abarata es lo que solo se lee cuando alguien va a buscarlo, y sigue estando.

Frente 3 — el panel que llevaba un mes mintiendo.

El panel de tráfico marcaba 41.000 peticiones diarias. La facturación del proveedor de pago no cuadraba con ese volumen.

```
requests | where timestamp > ago(1d)
| summarize registros = count(), reales = sum(itemCount)

registros  41.312
reales    413.120
```

Muestreo adaptativo al 10 %. Todos los paneles y las dos alertas de volumen usaban count().

```
agregación en paneles y alertas       count()         sum(itemCount)
muestreo para el flujo de pago       adaptativo       desactivado
muestreo para el resto               adaptativo       adaptativo (correcto)
alerta por caída de tráfico       nunca disparó    verificada con simulacro
```

La alerta por caída de tráfico era la peor de las tres: con count() y muestreo adaptativo, una subida real de tráfico podía parecer una bajada.

Frente 4 — la alerta que llegó nueve minutos tarde.

En la caída, la primera notificación llegó a las 09:31 para un incidente que empezó a las 09:22.

```
inicio real                         09:22
ingesta del registro                 09:24
frecuencia de evaluación (5 min)    09:29
ventana de 5 min ya cumplida        09:29
notificación                         09:31
```

Nueve minutos, todos estructurales. Se redistribuyen las alertas:

```
disponibilidad y 5xx de la pasarela      alerta de registro  alerta de métrica
latencia p95                             alerta de registro  alerta de métrica
excepción concreta con correlación         —                 alerta de registro
borrado de recursos                        —                 alerta de actividad
tiempo hasta la notificación             9 min               70 s
```

Frente 5 — las alertas que faltaban y ya se habían pagado dos veces.

Los incidentes de las clases 043 y 044 —el consumidor en cero réplicas y los 8.412 mensajes fallidos— tienen la misma forma: el sistema no se cayó, dejó de trabajar. Se añaden las cuatro señales de acumulación:

```
profundidad de la cola de pedidos creciendo 15 min
edad del mensaje más antiguo > 10 min
mensajes fallidos > 0 durante 15 min
retraso del consumidor de Event Hubs > 100.000 eventos
```

Las cuatro son métricas: cuestan céntimos y avisan en un minuto.

Y las cuatro consultas del manual de operación, guardadas antes de necesitarlas:

```
1. tasa de error por operación en la última hora
2. percentiles de latencia por operación, en serie temporal
3. traza completa por identificador de operación
```

```
4. cambios de configuración en la ventana del incidente, con autor
```

Resumen de la observabilidad:

```
recursos con registros activos                3 de 61         61 de 61
mecanismo de activación                       manual        por directiva
ingesta diaria                              46 GB/día       16 GB/día
costo mensual de telemetría                3.174 USD        870 USD
exactitud del recuento de peticiones          ×0,1            exacta
tiempo hasta la notificación                  9 min            70 s
alertas sobre trabajo acumulado                 0               4
consultas de incidente escritas de antemano     0               4
```

La lección que esta clase traslada al resto de la parte: la observabilidad de Azure no falla por falta de herramientas sino por dos valores por defecto: los registros de recurso vienen apagados y el muestreo viene encendido. El primero deja un incidente sin datos; el segundo deja los datos con un factor de diez. Ninguno de los dos avisa, y los dos se comprueban en un minuto — el día que se despliega, no el día del incidente.

Laboratorio guiado

Ejecuta desde la raíz:

```
python classes/part-03-azure-core-platform/045-azure-monitor-log-analytics-y-application-insights/lab.
py
```

El laboratorio selecciona el motor de práctica observability y produce labresult.json. El escenario, sus comprobaciones y el artefacto esperado corresponden a esta clase; no requiere credenciales y deja explícito qué debe revalidarse en un sandbox real.

1. Lee exercise.steps y formula una predicción antes de ejecutar.
2. Ejecuta la práctica y verifica todos los elementos de checks.
3. Provoca el caso de negativetest y explica la señal observada.
4. Materializa evidencia-operativa-azure en evidence/ usando la plantilla indicada.
5. Para proveedor real, sigue sandbox.requires, registra costo y ejecuta destroy.

Evidencia esperada

El artefacto principal es telemetría correlacionada con una pregunta operativa. Además, la entrega debe incluir el comando ejecutado, la salida estructurada y una conclusión que no exceda lo observado.

Reto verificable

Construye evidencia-operativa-azure para el caso CloudShop. Incluye una alternativa descartada, un supuesto que pueda falsarse, una prueba de fallo y una decisión de rollback.

■ Criterio de aceptación

- [ ] lab.py termina con código 0 y genera JSON válido.
- [ ] La entrega conecta al menos tres requisitos con mecanismos verificables.
- [ ] Existe una prueba positiva y una prueba negativa con evidencia.
- [ ] Seguridad, costo y operación aparecen como decisiones, no como anexos.
- [ ] Se declara una limitación y una condición que obligaría a revisar el diseño.
- [ ] Otra persona puede repetir el recorrido sin conocimiento tácito.

Errores frecuentes

Síntoma	Causa probable	Corrección
Tras una caída no hay registros que analizar	Las configuraciones de diagnóstico están apagadas por defecto y nunca se crearon	Actívalas por directiva con DeployIfNotExists y verifica la cobertura con una consulta sobre los recursos sin ella.
Los paneles muestran una fracción del tráfico real	El muestreo adaptativo de Application Insights está activo y las consultas usan count()	Suma itemCount en todas las agregaciones y desactiva el muestreo solo para los flujos que exijan detalle completo.
La factura de telemetría crece más rápido que la plataforma	Todo entra en plan de análisis, incluidos los registros de acceso y la salida estándar de los contenedores	Filtra en la regla de recopilación antes de la ingesta y asigna plan básico a las tablas voluminosas que no originan alertas.
Las alertas llegan varios minutos después del inicio del incidente	Son alertas de registro: suman ingesta, frecuencia y ventana	Pasa a alertas de métrica todo lo que despierte a alguien y reserva las de registro para lo que solo se puede expresar consultando.
Una traza se corta al llegar a un servicio interno	Ese salto no propaga la cabecera traceparent	Propaga la cabecera en todos los clientes HTTP y verifica la continuidad con una consulta por operationId.
No se puede saber quién borró un recurso hace cuatro meses	El registro de actividad se conserva 90 días	Expórtalo con una configuración de diagnóstico a almacenamiento inmutable desde el primer día.

Seguridad, ética y costo

Trabaja con cuentas propias o sandboxes autorizados. No publiques secretos, identificadores reales ni datos personales. Antes de crear recursos pagos define presupuesto, etiquetas y comando de destrucción; después verifica que no queden recursos huérfanos. Los ejemplos locales enseñan contratos, pero no certifican cumplimiento ni disponibilidad de producción.

Preguntas de comprobación

1. ¿Qué origen de telemetría funciona sin configurar nada y cuál está apagado en todos los recursos?
2. ¿Con qué criterio decides si una señal debe ser métrica o registro, y cuánto cambia el costo?
3. ¿Por qué count() sobre la tabla requests puede mentir, y qué se usa en su lugar?
4. Descompón los nueve minutos de latencia de una alerta de registro. ¿Cuáles de ellos se pueden reducir?
5. ¿Qué cuatro alertas detectan un sistema que dejó de trabajar sin caerse, y por qué son baratas?

Referencias

- Microsoft (2025). Azure Monitor overview — métricas, registros, orígenes y destinos. <https://learn.microsoft.com/en-us/azure/azure-monitor/overview>
- Microsoft (2025). Diagnostic settings in Azure Monitor — activación por recurso, destinos y despliegue por directiva. <https://learn.microsoft.com/en-us/azure/azure-monitor/essentials/diagnostic-settings>
- Microsoft (2025). Log Analytics workspace table plans and cost optimization — planes por tabla, retención y archivo. <https://learn.microsoft.com/en-us/azure/azure-monitor/logs/data-platform-logs>
- Microsoft (2025). Sampling in Application Insights — muestreo adaptativo, itemCount y sus efectos en las consultas. <https://learn.microsoft.com/en-us/azure/azure-monitor/app/sampling>
- Microsoft (2025). Types of Azure Monitor alerts — métrica, registro y registro de actividad, con sus latencias. <https://learn.microsoft.com/en-us/azure/azure-monitor/alerts/alerts-types>
- Documentación oficial vigente del servicio implementado; registra URL y fecha de consulta.

Evaluación — 045 Azure Monitor, Log Analytics y Application Insights

Preguntas

1. Explica el problema que resuelve esta capacidad sin mencionar un proveedor.
2. Identifica una frontera de responsabilidad y su propietario.
3. Compara dos alternativas mediante confiabilidad, seguridad, costo y operación.
4. ¿Qué demuestra el laboratorio y qué no puede demostrar?
5. Propón un caso límite o una prueba de fallo.

Reto verificable

Entrega evidencia-operativa-azure con diagrama o configuración, evidencia de ejecución, alternativa descartada, riesgo residual y criterio de rollback.

Criterio de aceptación

- [ ] Resultado reproducible por una segunda persona.
- [ ] Evidencia vinculada a cada afirmación técnica.
- [ ] Prueba negativa o de fallo incluida.
- [ ] Costos y permisos expresados con unidades y alcance.
- [ ] Limitaciones declaradas sin presentar la simulación como producción.

Escala

Nivel	Evidencia
A — competente	Cumple todo y defiende trade-offs con datos.
B — en desarrollo	Cumple lo esencial; falta profundidad en una dimensión.
C — inicial	Artefacto parcial o conclusión sin evidencia suficiente.
0	Sin entrega reproducible.

046 — Key Vault, Defender for Cloud y Azure Policy

Parte: 03 — Azure: plataforma esencial Nivel: intermedio · Horas estimadas: 4 Laboratorio: security · Estado: EXECUTABLECORE

Propósito

Aplicar defensa en profundidad a la plataforma Azure con las tres piezas que la sostienen, y con la precisión que más cuesta aprender: una directiva con efecto Deny no arregla lo que ya existe, porque actúa sobre la petición y no sobre el inventario. Asignarla y ver el panel en verde es el error de gobierno más repetido de esta parte. La clase 035 dejó el criterio —control de claves, rotación sin cortar conexiones, prueba negativa por control—; aquí cambian los mecanismos y aparece un motor de gobierno que no tiene equivalente directo en AWS.

Resultados de aprendizaje

Al finalizar podrás:

1. Elegir el nivel de Key Vault y el modelo de permisos, y explicar qué escalada cierra cada decisión.
2. Dimensionar el acceso a secretos para no agotar el límite de transacciones del almacén.
3. Distinguir los efectos de Azure Policy y saber cuál corrige lo existente y cuál solo guarda la puerta.
4. Rotar claves y secretos sin cortar conexiones, con el patrón de dos credenciales y una reacción a evento.
5. Verificar cada control con su prueba negativa, incluida la del antimalware y la de la directiva.

Conceptos centrales

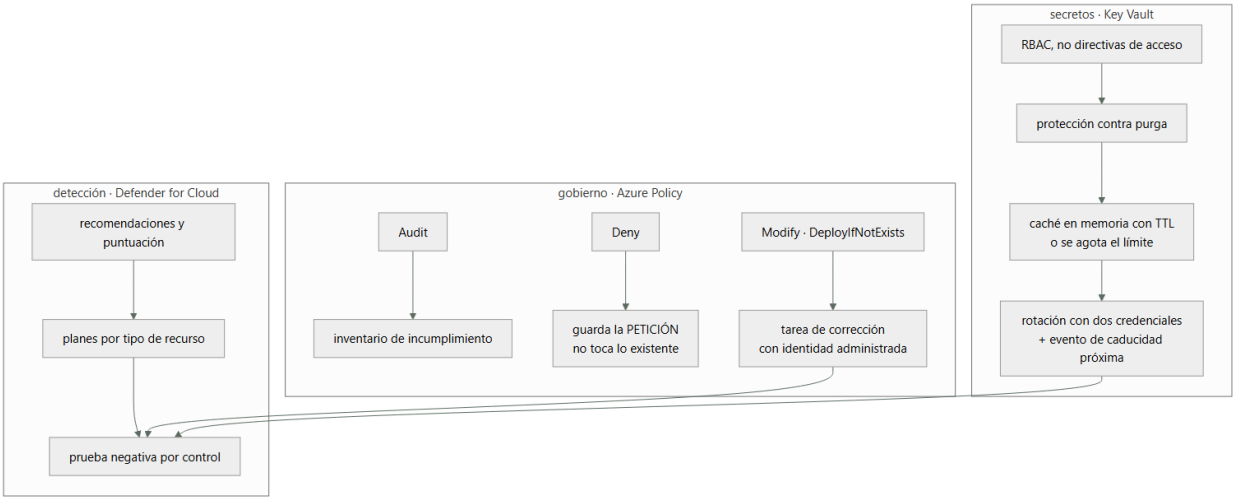
Concepto	Comprensión verificable
protección contra purga	Impide eliminar definitivamente un almacén o sus objetos durante el periodo de retención. Es irreversible al activarla, y sin ella quien puede borrar una clave puede destruir los datos que esa clave cifra.
modelo de permisos del almacén	Directivas de acceso —heredadas, del propio almacén— o Azure RBAC. Con las primeras, quien tiene Contributor sobre el almacén puede añadirse a sí mismo y leerlo todo.
clave gestionada por el cliente	Clave propia en Key Vault que cifra un servicio. Da control real y traslada una responsabilidad real: perder el acceso a la clave deja el recurso inaccesible.
efecto de directiva	Qué hace una directiva al evaluar: Audit informa, Deny rechaza la petición, Modify y DeployIfNotExists corrigen. Solo los dos últimos actúan sobre lo que ya existe, y mediante una tarea de corrección.
exención	Excepción explícita, con motivo, responsable y fecha de caducidad. Se distingue de excluir un ámbito en la asignación, que es invisible y no caduca.
puntuación de seguridad	Priorización relativa de las recomendaciones de Defender for Cloud. Es una herramienta para ordenar el trabajo, no un objetivo: llegar al 100 % no equivale a estar seguro.

Modelo mental

En Azure, identidad y jerarquía organizacional preceden al recurso: tenant, management group, suscripción y resource group determinan el alcance de control.

Aplicado a esta clase, separa siempre cuatro planos: intención (qué necesita el usuario), configuración (qué declaramos), estado observado (qué existe de verdad) y evidencia (cómo sabemos que cumple). Confundirlos produce diseños que se ven correctos en un diagrama pero fallan al operar.

■ Flujo de razonamiento



Desarrollo

1. Key Vault: el modelo de permisos decide qué escalada existe

Antes de guardar el primer secreto hay dos decisiones que se toman una vez y se sufren siempre.

El nivel, que responde a un requisito, no a una preferencia:

```
estándar        claves protegidas por software
premium         claves respaldadas por HSM, módulo compartido
HSM administrado  módulo dedicado de un solo inquilino
                  ~3,20 USD/h ≈ 2.300 USD/mes
```

El último no se elige «por si acaso»: se elige cuando una obligación regulatoria exige aislamiento de inquilino y un nivel de certificación concreto. Su precio es un dato de arquitectura.

El modelo de permisos, que es donde está la escalada. Las directivas de acceso son el modelo original: una lista en el propio almacén que dice quién puede hacer qué. Su defecto es estructural:

```
quien tiene Contributor sobre el almacén
  → puede MODIFICAR la lista de directivas de acceso
  → puede añadirse a sí mismo con permisos de lectura de secretos
  → lee todos los secretos, y la operación es legítima en el registro
```

No hace falta ninguna vulnerabilidad: es cómo funciona. Con Azure RBAC eso se cierra, porque conceder acceso a los datos exige el rol correspondiente y conceder roles exige a su vez un permiso distinto —el de administrador de acceso de usuario—, que la clase 038 ya identificó como uno de los que deben ser elegibles y no permanentes.

```
$ az keyvault update -n kv-cloudshop -g rg-seg --enable-rbac-authorization true
$ az role assignment create --assignee $APP_ID --role "Key Vault Secrets User" \
    --scope "$KV_ID/secrets/db-password"
```

El ámbito de esa segunda línea importa: RBAC llega hasta el secreto concreto, mientras que una directiva de acceso solo llegaba al almacén entero. Es la diferencia entre «esta aplicación puede leer su contraseña» y «esta aplicación puede leer todas las contraseñas de la organización».

Y la tercera decisión, que solo se puede tomar en un sentido:

```
$ az keyvault update -n kv-cloudshop -g rg-seg --enable-purge-protection true
```

La eliminación temporal ya viene activa; la protección contra purga impide eliminar definitivamente durante el periodo de retención, y no se puede desactivar. El argumento que la justifica es directo: si el almacén guarda la clave que cifra una cuenta de almacenamiento, quien puede purgar esa clave puede destruir los datos sin tocarlos. Con protección contra purga, ese camino no existe.

Y el fallo operativo más frecuente de Key Vault no es de seguridad, es de límite de transacciones:

```
tope orientativo   ~2.000 transacciones de secreto por cada 10 s y almacén
aplicación que lee el secreto en CADA petición, a 900 peticiones/s
  → 9.000 transacciones cada 10 s → 429 Too Many Requests
  → que la aplicación devuelve como 500 al usuario
```

El síntoma engaña porque parece un fallo de la base de datos: la aplicación no consigue su contraseña y lo reporta como error de conexión. La corrección es la evidente y casi nunca está puesta:

```
leer el secreto al arrancar y guardarlo en memoria
refrescar con un TTL de minutos, no por petición
y refrescar además al recibir un error de autenticación
```

La última línea es la que permite rotar sin reiniciar: si la credencial falla, se relee del almacén antes de darse por vencido.

2. Claves propias: control real y responsabilidad real

La clase 035 estableció tres niveles de control sobre las claves. En Azure son estos:

	Quién gestiona	Costo	Qué obtienes
Clave de plataforma	Microsoft	Incluido	Cifrado en reposo, sin trabajo
Clave del cliente	Tú, en Key Vault	Almacén + operaciones	Control de rotación y de revocación
Clave en HSM administrado	Tú, módulo dedicado	2.300 USD/mes	Aislamiento de inquilino

La segunda fila es la que hay que entender bien, porque su ventaja y su riesgo son la misma propiedad: puedes revocar el acceso a la clave. Eso significa que puedes dejar un dato ilegible a voluntad, que es exactamente lo que se pide en una baja de servicio o en una respuesta a incidente. Y significa también que:

```
si borras la clave                → la cuenta de almacenamiento queda inaccesible
si retiras el permiso a la identidad → el servicio deja de arrancar
si el almacén queda inalcanzable por red → lo mismo, sin haber tocado la clave
```

No es un fallo del diseño: es el precio del control, y hay que aceptarlo por escrito antes de activarlo. Las tres protecciones que lo hacen sostenible son la protección contra purga, un bloqueo de recurso sobre el almacén y una identidad asignada por el usuario —de la clase 038— para que la relación con la clave sobreviva a recrear el recurso.

Sobre la rotación, la precisión de la clase 035 se mantiene entera: rota la clave, no lo ya cifrado. Y en Azure hay un detalle comprobable que decide si la rotación sirve de algo:

```
# referencia SIN versión: el servicio toma automáticamente la versión nueva
https://kv-cloudshop.vault.azure.net/keys/clave-almacenamiento

# referencia CON versión: la rotación no cambia nada, sigue usando la vieja
https://kv-cloudshop.vault.azure.net/keys/clave-almacenamiento/7f3a…
```

Una rotación automática configurada sobre una referencia con versión fija se ejecuta puntualmente cada 90 días, genera una versión nueva, y el recurso sigue cifrando con la de siempre. El panel dice que la rotación está activa; la comprobación honesta es preguntar qué versión está usando el recurso.

La rotación de secretos sin cortar conexiones es el patrón de dos credenciales, y aquí encaja con lo construido en la clase 044:

```
1. existen dos credenciales activas: primaria y secundaria
2. las aplicaciones usan la primaria
3. se rota la SECUNDARIA (nadie la está usando)
4. se promueve: las aplicaciones pasan a la secundaria ya rotada
5. se rota la antigua primaria, que ahora está libre
```

En ningún momento hay una ventana sin credencial válida. Y el disparador no tiene por qué ser un calendario: Key Vault emite un evento cuando un secreto se acerca a su caducidad.

```
SecretNearExpiry → Event Grid → función que ejecuta la rotación
```

Es la reacción a hechos de la plataforma que la clase 044 describía, aplicada al caso donde más rinde: la rotación deja de depender de que alguien se acuerde.

3. Azure Policy: Deny guarda la puerta y no limpia la casa

Este es el mecanismo de gobierno que no tiene equivalente directo en AWS, y su malentendido más caro cabe en una línea: Deny evalúa la petición, no el inventario.

```
efecto              qué hace                        ¿toca lo que ya existe?
Audit               marca incumplimiento             informa, no cambia
Deny                rechaza la creación o el cambio  NO
```

```
Modify              añade o cambia propiedades       sí, con tarea de corrección
DeployIfNotExists   despliega un recurso asociado    sí, con tarea de corrección
AuditIfNotExists    marca la falta de algo asociado  informa
Disabled            desactiva la regla               —
```

Asignar una directiva Deny de «las cuentas de almacenamiento no deben permitir acceso público» impide crear una nueva mal configurada y deja intactas las catorce que ya lo están. El panel puede mostrar que las creaciones recientes cumplen mientras el problema real sigue ahí. Corregir lo existente exige otro efecto y un paso explícito:

```
$ az policy remediation create --name corregir-red-almacenamiento \
    --policy-assignment $ASIGNACION_ID --resource-discovery-mode ReEvaluateCompliance
```

Y esa tarea necesita una identidad administrada con permiso suficiente para hacer el cambio, lo que a su vez es una decisión de seguridad: se le concede el rol mínimo, no Contributor sobre la suscripción.

Dos comportamientos temporales que evitan diagnósticos erróneos:

```
al crear o modificar un recurso   la directiva se evalúa al instante
recursos existentes               se reevalúan cada ~24 h
```

Un panel de cumplimiento puede estar un día desfasado. Después de corregir algo, se fuerza la evaluación en vez de concluir que la corrección no funcionó.

Las iniciativas agrupan directivas con parámetros comunes, y las integradas cubren marcos completos —ISO 27001, PCI DSS, CIS—. Empezar por una integrada en modo Audit produce en un día el inventario que de otro modo cuesta semanas, y sin bloquear nada. La secuencia sensata es siempre la misma:

```
1. Audit          descubre el tamaño real del problema
2. corregir       con tareas de corrección donde el efecto lo permita
3. Deny           solo cuando el inventario ya cumple
```

Hacerlo al revés —Deny primero— bloquea despliegues legítimos, produce una avalancha de excepciones y termina con la directiva desasignada «temporalmente».

Y sobre las excepciones, una distinción de gobierno que separa una plataforma auditable de una que aparenta serlo:

```
exclusión de ámbito en la asignación
  invisible en los paneles, sin motivo, sin fecha, sin responsable
exención
  objeto propio, con categoría, justificación, responsable y CADUCIDAD
```

Las dos consiguen que un recurso deje de incumplir. Solo una de ellas se puede revisar. Una plataforma con cero incumplimientos y treinta exclusiones no documentadas está peor que una con treinta incumplimientos conocidos.

Un último enlace con la clase 038: cuando una directiva bloquea algo, el error no es de autorización, y leerlo ahorra buscar en el sitio equivocado:

```
AuthorizationFailed          falta el rol
RequestDisallowedByPolicy    hay rol y la directiva lo impide
```

4. Defender for Cloud: dos mitades y una puntuación que no es la meta

Defender for Cloud hace dos cosas distintas que se facturan distinto:

```
gestión de la postura (CSPM)   recomendaciones, puntuación de seguridad,
                               cumplimiento normativo
                               nivel gratuito incluido
protección de cargas (CWP)     detección de amenazas por tipo de recurso
                               planes de pago, uno por tipo
```

La segunda se activa plan a plan, y ahí está la decisión de costo. Encender todo en toda la suscripción es cómodo y caro; encender lo que corresponde al inventario real es trabajo de una hora:

```
Defender for Servers P2      ~15 USD por servidor y mes
Defender for Storage         ~10 USD por cuenta y mes (+ análisis por GB)
Defender for SQL             ~15 USD por instancia y mes
Defender for Containers      ~7 USD por vCPU y mes
Defender for Key Vault       por transacciones
```

La puntuación de seguridad ordena el trabajo por impacto relativo, y conviene decir con claridad qué no es: no es un objetivo. Subirla del 68 % al 74 % cerrando veinte recomendaciones de bajo impacto consume el mismo tiempo que cerrar la única que permitía movimiento lateral, y no reduce el mismo riesgo. La puntuación sirve para priorizar, y la priorización se corrige con criterio propio: qué explota un atacante primero en esta plataforma concreta.

El panel de cumplimiento normativo merece una nota, porque explica la arquitectura del producto: está construido sobre iniciativas de Azure Policy. Defender y Policy no son dos herramientas, son la misma evaluación presentada de dos formas. Lo que se corrige en una aparece en la otra.

Dos capacidades que enlazan con clases anteriores y rinden más de lo que cuestan:

Acceso justo a tiempo a máquinas virtuales. Los puertos de administración permanecen cerrados en el NSG y se abren, para una IP concreta y durante un plazo, previa solicitud registrada. Es exactamente el modelo de privilegio mínimo en el tiempo de la clase 038, aplicado a la red en lugar de a los roles. Y la aritmética es la misma: un puerto abierto 3 horas al mes en vez de 720 reduce la superficie un 99,6 %.

Análisis antimalware en almacenamiento. Analiza cada blob al subirlo. Tiene un costo por GB analizado, así que se activa en los contenedores que reciben ficheros de usuarios y no en los que guardan telemetría.

Y aquí vuelve, con toda su fuerza, la regla que cerraba la clase 035: cada control necesita su prueba negativa. Activar un plan de Defender no demuestra nada; lo que lo demuestra es provocar la detección:

```
# fichero de prueba estándar EICAR — inofensivo, reconocido por todo antimalware
$ printf 'X5O!P%%@AP[4\\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*' \
    > eicar.txt
$ az storage blob upload -c cargas -f eicar.txt -n eicar.txt \
    --account-name stcloudshopprod --auth-mode login
# la alerta debe aparecer en Defender en minutos
```

Si no aparece, el control no está donde se creía —muy a menudo porque el plan se activó en la suscripción y esa cuenta de almacenamiento estaba excluida—. Y esa comprobación hay que repetirla, porque un control verificado una vez es un control verificado en el pasado.

5. La prueba negativa de cada control de esta parte

La clase 035 dejó el principio y la clase 036 lo aplicó a AWS. Cerrar la parte 03 exige la lista equivalente, porque un control sin prueba negativa es una afirmación, no un control.

control	prueba negativa que lo demuestra
acceso público del almacenamiento deshabilitado	petición desde fuera → 403 (clase 041)
punto de conexión privado activo	nslookup desde la subred devuelve 10.x y desde fuera falla (clase 039)
salida de la subred cerrada	curl con tiempo límite → agota (clase 039)
clave compartida deshabilitada	autenticación con la clave → KeyBasedAuthenticationNotPermitted (041)
federación acotada por sujeto	petición desde otro repositorio → AADSTS70021 (clase 038)
directiva Deny asignada	crear el recurso prohibido → RequestDisallowedByPolicy
protección contra purga	intentar purgar → operación rechazada
antimalware en almacenamiento	subir el fichero EICAR → alerta
recuperación de contenedor	borrar y restaurar uno de prueba (041)
restauración de base de datos	restaurar a una copia y contar filas (042)
rotación de secreto	rotar con tráfico en curso y medir errores: deben ser cero (035)

La columna derecha es el entregable. Un informe de seguridad que dice «se ha habilitado X» no es comparable a uno que dice «se ha habilitado X y aquí está el error que devuelve al intentar lo que impide».

Y hay una prueba negativa de segundo orden que casi nunca se hace y descubre lo que las demás no: comprobar que el control sigue activo pasado un tiempo. Los controles se apagan solos por caminos perfectamente normales —alguien recrea el recurso desde una plantilla vieja, un despliegue sobrescribe la configuración, una excepción temporal se queda—. La defensa contra eso no es la memoria de nadie:

```
directiva en modo Audit sobre cada control crítico
  → el panel muestra cualquier recurso que se salga
  → y la desviación aparece en horas, no en la siguiente auditoría
```

Es el mismo principio que cierra toda la parte: lo que no tiene una señal automática, se degrada en silencio. Los registros apagados de la clase 045, el consumidor en cero réplicas de la 043, la subcola de fallidos de la 044 y el control de seguridad revertido son el mismo problema con cuatro caras: sistemas que dejan de hacer su trabajo sin producir ningún error.

Ejemplo trabajado

CloudShop cierra la seguridad de su plataforma Azure. El equipo activa Key Vault, asigna una iniciativa de cumplimiento y enciende Defender en toda la suscripción. A las seis semanas hay un panel en verde, una factura alta y cinco cosas que no eran ciertas.

Hecho 1 — la aplicación devuelve 500 en los picos y la culpa parecía de la base de datos.

```
09:40  errores 500 al 6 % · el registro dice "no se pudo conectar a la base de datos"
09:41  la base de datos está sana: CPU 22 %, ninguna conexión rechazada
```

```
AzureDiagnostics
| where ResourceProvider == "MICROSOFT.KEYVAULT" and httpStatusCode_d == 429
| summarize limitadas = count() by bin(TimeGenerated, 1m)
```

```
09:38   4.117
09:39  11.902
09:40  12.455
```

La aplicación leía la contraseña de Key Vault en cada petición. A 900 peticiones por segundo, el almacén limitaba, y el fallo se reportaba como error de base de datos porque es donde se manifestaba.

```
lecturas del almacén            1 por petición   1 cada 30 min
transacciones en el pico        ~9.000/10 s        ~2/10 s
relectura ante fallo de auth        no               sí
errores 500 en el pico             6,0 %            0,0 %
```

Hecho 2 — un ingeniero leyó todos los secretos y todo estaba permitido.

Una revisión rutinaria encuentra una modificación de la lista de directivas de acceso seguida de cuarenta lecturas de secretos, todas correctas.

```
AzureActivity
| where _ResourceId has "kv-cloudshop"
| where OperationNameValue endswith "VAULTS/WRITE"
| project TimeGenerated, Caller, ActivityStatusValue
```

No hubo intrusión: con Contributor sobre el almacén, añadirse a la lista es una operación legítima. Se cierra el camino:

```
modelo de permisos          directivas de acceso     Azure RBAC
ámbito de la aplicación       almacén entero        un secreto concreto
quién puede conceder acceso  cualquier Contributor  administrador de acceso
                                                    de usuario, y elegible (038)
protección contra purga           no                    sí
alerta sobre cambios de rol       no                    sí, alerta de actividad
```

Hecho 3 — el certificado se renovó y el sitio siguió sirviendo el viejo.

A los cuatro días de la renovación automática, los clientes empiezan a ver avisos de certificado caducado.

```
$ az keyvault certificate show --vault-name kv-cloudshop -n cloudshop-tls \
    --query "attributes.expires" -o tsv
2027-07-30T00:00:00Z                      # renovado, correcto
$ echo | openssl s_client -connect cloudshop.example:443 2>/dev/null \
    | openssl x509 -noout -enddate
notAfter=Jul 31 09:00:00 2026 GMT         # lo que ve el cliente: el viejo
```

El enlace de la aplicación apuntaba a una versión concreta del certificado. La renovación creó otra versión y nadie la usaba.

```
referencia al certificado      versión fija       sin versión
vigilancia del vencimiento     desde el almacén   desde FUERA, con una sonda TLS
aviso                          ninguno            30 días antes, sobre lo servido
```

La segunda fila es la lección: el vencimiento se vigila donde lo ve el cliente, no donde se guarda. El almacén decía la verdad y era irrelevante.

Hecho 4 — cumplimiento en verde con catorce cuentas públicas.

```
$ az policy state summarize --query "value[0].results.{cumple:nonCompliantResources}" -o tsv
0
$ az storage account list --query "[?allowBlobPublicAccess].name" -o tsv | wc -l
14
```

La contradicción se explica leyendo la asignación: efecto Deny, que solo evalúa peticiones nuevas, y un ámbito con tres exclusiones heredadas de la puesta en marcha, sin motivo ni fecha.

```
efecto de la directiva                    Deny        Audit → corregir → Deny
cuentas con acceso público                 14                 0
mecanismo de corrección                 ninguno      tarea de corrección
exclusiones de ámbito                       3                 0
exenciones documentadas                     0        2, con motivo y caducidad
```

Las dos exenciones que quedan son reales —un contenedor de recursos estáticos públicos por diseño— y ahora caducan en noventa días, así que alguien tendrá que volver a justificarlas.

Hecho 5 — 3.100 USD al mes de Defender y un control que no estaba donde se creía.

```
Defender for Servers P2                todo          22 servidores    330
Defender for Storage                   todo           3 cuentas         30
Defender for SQL                       todo           2 instancias      30
Defender for Containers                todo          16 vCPU           112
Defender for App Service, DNS, Resource Manager, APIs…                 2.598
                                                                      ■■■■■■
                                                                       3.100
```

Se conservan los planes que corresponden al inventario y a la superficie real de ataque, y se retiran los que protegían servicios que CloudShop no usa:

```
planes conservados (servidores, almacenamiento, SQL,
contenedores, Key Vault, Resource Manager)                ~640
```

Y la prueba negativa descubre lo importante:

```
$ az storage blob upload -c cargas -f eicar.txt -n eicar.txt \
    --account-name stcloudshopcargas --auth-mode login
# 20 minutos después: ninguna alerta
```

La cuenta que recibe los ficheros de los clientes —la única que de verdad importaba— se había creado después de activar el plan y estaba en otra suscripción sin cobertura. Se corrige y se repite la prueba:

```
subida del fichero EICAR → alerta "Malicious file uploaded" en 4 min      ✓
```

Resumen del cierre de seguridad:

```
modelo de permisos del almacén        directivas de acceso     RBAC
protección contra purga                       no                sí
errores 500 por limitación del almacén     6,0 %               0,0 %
cuentas de almacenamiento públicas           14                  0
exclusiones de ámbito sin justificar          3                  0
exenciones con motivo y caducidad             0                  2
planes de Defender activos                todos (3.100 USD)   los del inventario (640)
controles con prueba negativa ejecutada     0 de 11             11 de 11
```

La lección que esta clase traslada al proyecto de la clase 048: el panel en verde y el control efectivo son dos cosas distintas, y solo una de ellas se puede demostrar. Deny guardaba la puerta mientras catorce puertas ya estaban abiertas; el plan de antimalware protegía todo menos la cuenta que recibía ficheros de desconocidos. La prueba negativa no es la parte opcional del trabajo: es la única parte que produce evidencia.

Laboratorio guiado

Ejecuta desde la raíz:

```
python classes/part-03-azure-core-platform/046-key-vault-defender-for-cloud-y-azure-policy/lab.py
```

El laboratorio selecciona el motor de práctica security y produce labresult.json. El escenario, sus comprobaciones y el artefacto esperado corresponden a esta clase; no requiere credenciales y deja explícito qué debe revalidarse en un sandbox real.

1. Lee exercise.steps y formula una predicción antes de ejecutar.
2. Ejecuta la práctica y verifica todos los elementos de checks.
3. Provoca el caso de negativetest y explica la señal observada.
4. Materializa controles-azure en evidence/ usando la plantilla indicada.
5. Para proveedor real, sigue sandbox.requires, registra costo y ejecuta destroy.

Evidencia esperada

El artefacto principal es un control con amenaza, mitigación y evidencia. Además, la entrega debe incluir el comando ejecutado, la salida estructurada y una conclusión que no exceda lo observado.

Reto verificable

Construye controles-azure para el caso CloudShop. Incluye una alternativa descartada, un supuesto que pueda falsarse, una prueba de fallo y una decisión de rollback.

■ Criterio de aceptación

- [ ] lab.py termina con código 0 y genera JSON válido.
- [ ] La entrega conecta al menos tres requisitos con mecanismos verificables.
- [ ] Existe una prueba positiva y una prueba negativa con evidencia.
- [ ] Seguridad, costo y operación aparecen como decisiones, no como anexos.
- [ ] Se declara una limitación y una condición que obligaría a revisar el diseño.
- [ ] Otra persona puede repetir el recorrido sin conocimiento tácito.

■■ Errores frecuentes

Síntoma	Causa probable	Corrección
La aplicación falla al conectar con la base de datos y la base de datos está sana	Se lee el secreto de Key Vault en cada petición y el almacén limita con 429	Cachea el secreto en memoria con TTL y reléelo solo al recibir un error de autenticación.
Alguien con permisos administrativos sobre el almacén lee todos los secretos y todo aparece como legítimo	El modelo de directivas de acceso permite modificar la propia lista	Cambia a Azure RBAC, acota el ámbito al secreto y haz elegible el rol que concede acceso.
El certificado se renueva en el almacén y el cliente sigue viendo el caducado	El enlace apunta a una versión concreta, no al certificado	Referencia sin versión y vigila el vencimiento con una sonda TLS desde fuera, no desde el almacén.
El panel de cumplimiento está en verde y hay recursos incumpliendo	El efecto Deny solo evalúa peticiones nuevas, y había exclusiones de ámbito sin documentar	Empieza en Audit, corrige lo existente con tareas de corrección, y sustituye exclusiones por exenciones con motivo y caducidad.
La rotación automática de la clave se ejecuta y el recurso sigue usando la versión antigua	El recurso referencia una versión concreta en vez de la clave sin versión	Usa la referencia sin versión y comprueba qué versión está usando realmente el recurso.
Un plan de Defender está activo y no detecta nada al provocarlo	El recurso que importaba estaba fuera del ámbito donde se activó el plan	Ejecuta la prueba negativa —el fichero EICAR— contra el recurso real y repítela periódicamente.

■ Seguridad, ética y costo

Trabaja con cuentas propias o sandboxes autorizados. No publiques secretos, identificadores reales ni datos personales. Antes de crear recursos pagos define presupuesto, etiquetas y comando de destrucción; después verifica que no queden recursos huérfanos. Los ejemplos locales enseñan contratos, pero no certifican cumplimiento ni disponibilidad de producción.

■ Preguntas de comprobación

1. ¿Qué escalada permite el modelo de directivas de acceso y cómo la cierra Azure RBAC?
2. ¿Por qué una directiva con efecto Deny puede convivir con catorce recursos incumpliendo?
3. ¿Qué diferencia hay entre excluir un ámbito y crear una exención, y por qué importa para una auditoría?
4. ¿Qué ganas y qué asumes al cifrar con una clave propia en vez de con la de la plataforma?
5. Enumera tres controles de esta parte y la prueba negativa concreta que demuestra cada uno.

Referencias

• Microsoft (2025). Azure Key Vault security overview — modelos de permisos, protección contra purga y límites de servicio. <https://learn.microsoft.com/en-us/azure/key-vault/general/security-features>

• Microsoft (2025). Customer-managed keys — control, revocación y consecuencias de perder el acceso a la clave. <https://learn.microsoft.com/en-us/azure/storage/common/customer-managed-keys-overview>

• Microsoft (2025). Understand Azure Policy effects — Audit, Deny, Modify, DeployIfNotExists y su alcance. <https://learn.microsoft.com/en-us/azure/governance/policy/concepts/effects>

• Microsoft (2025). Azure Policy exemption structure — exenciones con motivo, responsable y caducidad. <https://learn.microsoft.com/en-us/azure/governance/policy/concepts/exemption-structure>

• Microsoft (2025). Microsoft Defender for Cloud plans — postura frente a protección de cargas y precios por tipo de recurso. <https://learn.microsoft.com/en-us/azure/defender-for-cloud/defender-for-cloud-introduction>

• Documentación oficial vigente del servicio implementado; registra URL y fecha de consulta.

Descargar: Parte 03 en PDF · Recorrido de Azure en PDF · Manual integral

Anterior	Índice	Siguiente
← 045 · Azure Monitor, Log Analytics y Application Insights	Parte 03 · Programa	047 · Bicep, plantillas y despliegues por alcance →

Evaluación — 046 Key Vault, Defender for Cloud y Azure Policy

Preguntas

1. Explica el problema que resuelve esta capacidad sin mencionar un proveedor.
2. Identifica una frontera de responsabilidad y su propietario.
3. Compara dos alternativas mediante confiabilidad, seguridad, costo y operación.
4. ¿Qué demuestra el laboratorio y qué no puede demostrar?
5. Propón un caso límite o una prueba de fallo.

Reto verificable

Entrega controles-azure con diagrama o configuración, evidencia de ejecución, alternativa descartada, riesgo residual y criterio de rollback.

Criterio de aceptación

• [ ] Resultado reproducible por una segunda persona.

• [ ] Evidencia vinculada a cada afirmación técnica.

• [ ] Prueba negativa o de fallo incluida.

- [ ] Costos y permisos expresados con unidades y alcance.
- [ ] Limitaciones declaradas sin presentar la simulación como producción.

Escala

Nivel	Evidencia
A — competente	Cumple todo y defiende trade-offs con datos.
B — en desarrollo	Cumple lo esencial; falta profundidad en una dimensión.
C — inicial	Artefacto parcial o conclusión sin evidencia suficiente.
0	Sin entrega reproducible.

047 — Bicep, plantillas y despliegues por alcance

Parte: 03 — Azure: plataforma esencial Nivel: intermedio · Horas estimadas: 4 Laboratorio: iac · Estado: EXECUTABLECORE

Propósito

Declarar la infraestructura de Azure con Bicep entendiendo lo que lo hace distinto de todo lo que viene después en la parte 07: no hay archivo de estado, porque el estado deseado se compara contra el propio Azure. Eso elimina una clase entera de problemas y crea otra, y en medio está el alcance del despliegue —grupo de recursos, suscripción, grupo de administración o inquilino—, que es la decisión que más plantillas rompe al empezar.

Resultados de aprendizaje

Al finalizar podrás:

1. Elegir el alcance de un despliegue y saber qué recursos solo pueden crearse en cada uno.
2. Distinguir el modo incremental del completo y explicar exactamente qué borra el segundo.
3. Leer la salida de what-if sabiendo qué diferencias son reales y cuáles son ruido del proveedor.
4. Evitar la desaparición de recursos hijo al mezclar declaración en línea y declaración independiente.
5. Manejar secretos en plantillas sin dejarlos en el historial de despliegues.

Conceptos centrales

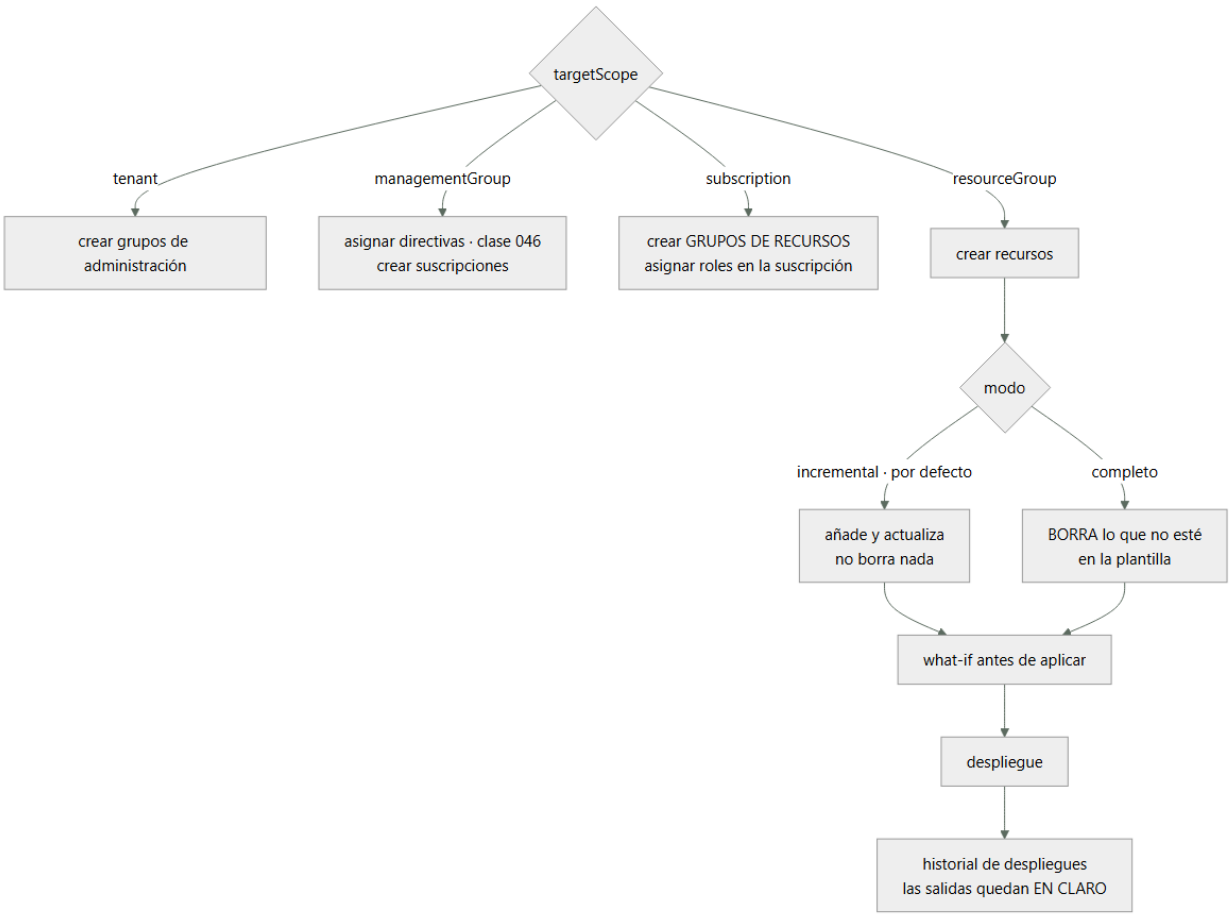
Concepto	Comprensión verificable
alcance de despliegue	Nivel en el que se ejecuta la plantilla: resourceGroup, subscription, managementGroup o tenant. Determina qué se puede crear: un grupo de recursos solo se crea desde el alcance de suscripción.
modo completo	Modo de despliegue que elimina del grupo de recursos todo lo que no aparece en la plantilla. No avisa por separado: es parte de la operación.
what-if	Simulación del cambio antes de aplicarlo. Es imprescindible y no es infalible: algunos proveedores informan de modificaciones que no ocurren, y ese ruido enseña a ignorarla.
recurso hijo en línea o independiente	Las reglas de un NSG, los ajustes de una aplicación o las entradas de DNS pueden declararse dentro del padre o como recurso propio. Mezclar ambas formas hace desaparecer las que no están en la plantilla que se despliega.
salida de despliegue	Valor devuelto por la plantilla. Se guarda en el historial de despliegues en claro y lo lee cualquiera con permiso de lectura: nunca debe contener un secreto.
pila de despliegue	Agrupación de recursos gestionada como una unidad, con control de ciclo de vida y ajustes de denegación. Es la respuesta moderna a la brusquedad del modo completo.

Modelo mental

En Azure, identidad y jerarquía organizacional preceden al recurso: tenant, management group, suscripción y resource group determinan el alcance de control.

Aplicado a esta clase, separa siempre cuatro planos: intención (qué necesita el usuario), configuración (qué declaramos), estado observado (qué existe de verdad) y evidencia (cómo sabemos que cumple). Confundirlos produce diseños que se ven correctos en un diagrama pero fallan al operar.

■ Flujo de razonamiento



Desarrollo

1. Sin archivo de estado: qué se gana y qué se pierde

Bicep es una sintaxis que compila a la plantilla JSON del Azure Resource Manager. La correspondencia es uno a uno: no hay nada que Bicep pueda expresar que ARM no acepte, ni al revés. Eso lo convierte en una abstracción transparente, y explica por qué se puede adoptar sin apuesta: se puede volver al JSON en cualquier momento, y el resultado de compilarlo es exactamente lo que se despliega.

La diferencia estructural con la herramienta de la parte 07 está en otro sitio:

```
Terraform   mantiene un archivo de ESTADO con lo que cree que existe
Bicep/ARM   no mantiene estado: pregunta a Azure qué hay ahora mismo
```

Lo que se gana con eso es una lista concreta de problemas que dejan de existir:

```
no hay estado que bloquear entre dos despliegues simultáneos
no hay estado que se corrompa ni que haya que migrar
no hay secretos guardados en el estado
no hay que importar un recurso creado a mano: la próxima ejecución lo ve
```

Ese último punto es más importante de lo que parece. Un recurso creado desde el portal no rompe el siguiente despliegue: la plantilla declara lo que quiere y Azure concilia. Un recurso creado a mano fuera de una herramienta con estado, en cambio, exige un paso de importación explícito.

Y lo que se pierde también es concreto:

```
no hay una lista de "lo que este código gestiona"
  → nadie sabe qué recursos pertenecen a esta plantilla y cuáles no
no se detecta que alguien BORRÓ algo
  → la plantilla lo vuelve a crear en el siguiente despliegue,
    sin decir que había desaparecido
no hay destrucción ordenada del conjunto
  → borrar lo desplegado es borrar el grupo de recursos, o usar modo completo
```

La primera pérdida es la que causa incidentes. Sin una lista de pertenencia, la única forma de preguntar «¿qué gestiona esta plantilla?» es leerla, y la única forma de limpiar lo que sobra es el modo completo, que es un instrumento romo. Las pilas de despliegue existen precisamente para cubrir ese hueco: agrupan los recursos de un despliegue como una unidad con ciclo de vida propio y ajustes que impiden modificarlos fuera de la pila. Es lo más parecido a una lista de pertenencia que ARM ofrece, y conviene usarlas donde el conjunto tiene identidad —un entorno, un servicio completo—.

Una consecuencia práctica que conviene tener presente desde el principio: la unidad natural de aislamiento en Azure es el grupo de recursos (clase 037), y también lo es aquí. Un grupo de recursos por unidad desplegable evita casi todos los problemas que siguen, porque hace que la pregunta «qué pertenece a esta plantilla» tenga una respuesta obvia.

2. El alcance decide qué se puede crear

Es lo primero que rompe una plantilla al empezar, y el mensaje de error rara vez lo dice con claridad. Cada plantilla declara en qué nivel se ejecuta:

```
targetScope = 'subscription'
```

Y el nivel determina qué recursos son válidos:

Alcance	Se puede crear
tenant	Grupos de administración, asignaciones en la raíz
managementGroup	Asignaciones de directiva e iniciativas, suscripciones, asignaciones de rol
subscription	Grupos de recursos, directivas y roles de la suscripción
resourceGroup	Todo lo demás: la infraestructura propiamente dicha

El caso que aparece siempre: un grupo de recursos no se puede crear desde una plantilla de alcance resourceGroup, porque el grupo tendría que existir ya para poder ejecutarla. La estructura correcta es una plantilla de suscripción que cree los grupos y llame a módulos dentro de cada uno:

```
targetScope = 'subscription'

param entorno string
param ubicacion string = 'westeurope'

resource rgRed 'Microsoft.Resources/resourceGroups@2024-03-01' = {
  name: 'rg-cloudshop-red-${entorno}'
  location: ubicacion
}

module red 'modulos/red.bicep' = {
  name: 'despliegue-red'
  scope: rgRed                      // el módulo se ejecuta DENTRO del grupo
  params: { entorno: entorno, ubicacion: ubicacion }
}
```

La palabra clave es scope en el módulo: permite que una plantilla de un nivel despliegue en otro, que es como se construye una plataforma completa desde una sola ejecución. Y para referirse a algo que ya existe en otro sitio, sin crearlo:

```
resource almacen 'Microsoft.KeyVault/vaults@2023-07-01' existing = {
  name: 'kv-cloudshop'
  scope: resourceGroup('rg-cloudshop-seg-prod')
}
```

existing no crea ni modifica nada: solo obtiene una referencia. Es la vía correcta para leer un identificador o un secreto de un recurso gestionado por otro equipo, y evita el antipatrón de pasar identificadores completos como parámetros de texto, que se rompen en cuanto algo cambia de nombre.

Dos precisiones que ahorran depuración:

Los nombres de despliegue deben ser únicos por alcance y se conservan. Reutilizar el mismo nombre sobrescribe la entrada del historial, que es justo lo que no se quiere cuando hay que investigar qué se desplegó cuando. Un nombre con marca de tiempo o con el identificador de la ejecución de la canalización resuelve el problema.

Las funciones dependen del alcance. resourceGroup() no existe en una plantilla de suscripción, y subscription() sí. Un error de «función no reconocida» casi siempre significa que la plantilla está en el nivel equivocado, no que la función esté mal escrita.

3. Modo completo y recursos hijo: las dos formas de borrar sin querer

El modo de despliegue tiene dos valores y una asimetría de consecuencias:

```
incremental (por defecto)  añade lo que falta, actualiza lo que cambió,
                           NO toca lo que no aparece en la plantilla
completo                   además, ELIMINA del grupo de recursos
                           todo lo que no esté en la plantilla
```

El modo completo tiene un uso legítimo: garantizar que un entorno contiene exactamente lo declarado, sin residuos. Y tiene un riesgo que se materializa cuando el grupo de recursos contiene algo que otra persona creó por una razón válida — un disco de diagnóstico, una instantánea de una migración, un recurso de otro equipo. El despliegue no pregunta.

La protección no es la prudencia sino el diseño:

```
1. un grupo de recursos por unidad desplegable
   → nada ajeno puede estar dentro
2. `what-if` obligatorio antes de aplicar en modo completo
   → las eliminaciones aparecen marcadas y hay que leerlas
3. bloqueo de recurso sobre lo que jamás debe borrarse (clase 042)
   → un bloqueo hace fallar el despliegue en vez de perder el recurso
```

La segunda forma de borrar sin querer es más sutil y no depende del modo. Muchos recursos de Azure admiten declarar sus hijos de dos maneras:

```
// (a) en línea, dentro del padre
resource nsg 'Microsoft.Network/networkSecurityGroups@2024-01-01' = {
  name: 'nsg-app'
  location: ubicacion
  properties: {
    securityRules: [ { name: 'allow-lb-probe', properties: { /* … */ } } ]
  }
}

// (b) como recurso independiente
resource regla 'Microsoft.Network/networkSecurityGroups/securityRules@2024-01-01' = {
  parent: nsg
  name: 'allow-appgw-8080'
  properties: { /* … */ }
}
```

Cada forma es correcta por separado. Mezclarlas destruye datos: la declaración en línea es la lista completa de reglas, así que al desplegar el padre, Azure sustituye el conjunto entero por lo que hay en línea y las reglas declaradas aparte desaparecen. En el siguiente despliegue de la plantilla que las declaraba, vuelven. El resultado es un juego de reglas que aparecen y desaparecen según qué canalización se ejecutó la última, con incidentes de conectividad intermitentes de causa muy difícil de ver.

La regla de equipo es de una línea: para cada tipo de recurso hijo, se elige una de las dos formas y se documenta. Ocurre igual con los ajustes de aplicación de App Service, las reglas de firewall de una base de datos y los registros de una zona DNS.

Y el mismo mecanismo explica otra sorpresa habitual: un despliegue que «borra» la configuración que alguien puso a mano en el portal. No la borra por malicia — la plantilla declara el estado completo de esa propiedad, y lo que no está declarado vuelve a su valor por omisión. Es el comportamiento correcto de una herramienta declarativa, y es la razón por la que la configuración manual y la declarada no pueden convivir sobre el mismo recurso.

4. what-if, validación y el ruido que enseña a no mirar

what-if compara el estado deseado con el actual y muestra el cambio antes de aplicarlo:

```
$ az deployment group what-if -g rg-cloudshop-app-prod \
    --template-file main.bicep --parameters @prod.bicepparam

Resource and property changes are indicated with these symbols:
  + Create   ~ Modify   - Delete   = NoChange
```

```
~ Microsoft.Web/sites/app-tienda
  ~ properties.siteConfig.alwaysOn:  false → true
+ Microsoft.Insights/components/appi-tienda
```

Es obligatorio antes de cualquier aplicación y especialmente antes del modo completo, donde las líneas con - son las que hay que leer con atención.

Y tiene una limitación que hay que conocer para que no la desactive de hecho: algunos proveedores informan de modificaciones que no van a ocurrir. Devuelven propiedades calculadas o normalizan valores, y what-if las presenta como diferencias. El resultado predecible es que una salida con treinta líneas de ruido en cada ejecución enseña al equipo a hojearla, y el día que aparece un cambio real pasa desapercibido.

Las medidas que lo mantienen legible:

```
módulos pequeños       una salida corta se lee; una de 300 líneas se hojea
lista de exclusiones   `--exclude-change-types NoChange Ignore`
tipos ruidosos aparte  desplegar por separado lo que siempre informa cambios
revisar el ruido       si una propiedad aparece siempre, o falta en la plantilla
                       o es calculada: en el primer caso, se añade y desaparece
```

La última es la más rentable: buena parte del ruido de what-if es real y señala propiedades que el proveedor fija y la plantilla no declara. Declararlas explícitamente elimina la línea y, de paso, documenta el valor.

La validación previa completa la red:

```
$ bicep lint main.bicep                    # reglas del linter, en el editor y en CI
$ az deployment group validate -g rg-cloudshop-app-prod \
    --template-file main.bicep --parameters @prod.bicepparam
```

validate comprueba sintaxis, referencias y permisos sin crear nada. Es rápida y detecta el error de alcance, el nombre inválido y el parámetro que falta — los tres fallos que si no aparecen en CI aparecen a mitad de un despliegue, con parte de los recursos ya creados.

Y eso lleva al punto que más tranquiliza saber de antemano: un despliegue fallido no se revierte solo. Si diez de quince recursos se crearon y el undécimo falla, los diez se quedan. ARM no tiene transacción. La consecuencia de diseño es que las plantillas deben ser idempotentes y reejecutables: la respuesta a un fallo a medias es corregir y volver a desplegar, no limpiar a mano. Una plantilla que solo funciona sobre un grupo de recursos vacío es una plantilla que no sirve para operar.

5. Secretos, módulos y el historial que todo el mundo puede leer

Tres prácticas cierran la clase, y la primera es la que más veces está mal.

Las salidas quedan en claro en el historial de despliegues. Cualquiera con permiso de lectura sobre el grupo de recursos puede consultarlas:

```
$ az deployment group show -g rg-cloudshop-app-prod -n despliegue-app \
    --query "properties.outputs" -o json
```

Devolver una cadena de conexión, una clave o una contraseña como salida es publicarla. Y no basta con dejar de hacerlo: el historial conserva las anteriores, así que un secreto que estuvo en una salida hay que considerarlo comprometido, rotarlo y borrar esa entrada del historial.

La forma correcta de que una plantilla use un secreto sin verlo:

```
resource almacen 'Microsoft.KeyVault/vaults@2023-07-01' existing = {
  name: 'kv-cloudshop'
  scope: resourceGroup('rg-cloudshop-seg-prod')
}

module baseDatos 'modulos/sql.bicep' = {
  name: 'despliegue-sql'
  params: {
    contrasena: almacen.getSecret('sql-admin-password')   // no pasa por la plantilla
  }
}
```

getSecret() solo funciona pasando el valor a un parámetro marcado como seguro dentro de un módulo, y esa restricción es deliberada: impide usarlo en una expresión que acabe en un registro o en una salida.

```
@secure()
@description('Contraseña del administrador. No se registra ni se devuelve.')
```

```
param contrasena string
```

Un parámetro @secure() se omite en el historial y en los registros. Sin la anotación, el valor del archivo de parámetros queda registrado igual que cualquier otro.

Y la mejor opción sigue siendo no tener el secreto: si el recurso admite identidad administrada, la clase 038 ya dio la respuesta — no hay contraseña que pasar.

Los módulos se versionan como código. Un módulo publicado en un registro de contenedores se referencia por versión, y eso permite que un equipo actualice cuando le convenga en vez de cuando otro despliegue:

```
module red 'br:acrcloudshop.azurecr.io/bicep/modules/red:1.4.0' = {
  name: 'despliegue-red'
  params: { entorno: 'prod' }
}
```

Sin versión —referencias a una ruta local compartida o a latest— un cambio en el módulo se aplica a todos los que lo usan en su siguiente despliegue, que puede ser el de producción de otro equipo un viernes. Es el mismo argumento de fijar versiones de dependencias, aplicado a la infraestructura.

Y la plantilla es el sitio donde vive el gobierno de la parte. Las decisiones de las clases anteriores se declaran, no se recuerdan:

```
resource cuenta 'Microsoft.Storage/storageAccounts@2023-05-01' = {
  name: nombreCuenta
  location: ubicacion
  sku: { name: 'Standard_ZRS' }                // redundancia decidida (041)
  kind: 'StorageV2'
  properties: {
    allowBlobPublicAccess: false               // acceso público cerrado (041)
    allowSharedKeyAccess: false                // sin clave compartida (041)
    minimumTlsVersion: 'TLS1_2'
    publicNetworkAccess: 'Disabled'            // solo punto privado (039)
  }
  tags: etiquetas                              // atribución de costo (037)
}
```

Seis decisiones de cuatro clases distintas, en un recurso. Esa es la función real de la infraestructura como código en este programa: convertir lo que se aprendió en algo que no se puede olvidar al crear el siguiente recurso. Y lo que la plantilla no puede garantizar por sí sola —que nadie lo cambie después— lo cubre la directiva de la clase 046. Las dos capas juntas son el mecanismo; ninguna de las dos basta sola.

Ejemplo trabajado

CloudShop pasa a declarar su plataforma Azure en Bicep. La primera versión funciona y produce cuatro incidentes en dos meses, todos por comportamientos de ARM que la plantilla no controlaba.

Punto de partida:

```
un único main.bicep de 1.100 líneas, alcance resourceGroup
un grupo de recursos compartido: rg-cloudshop-prod
despliegue en modo completo "para que quede limpio"
los parámetros incluyen la contraseña de SQL en texto
```

Incidente 1 — el despliegue borra un disco de otro equipo.

El equipo de datos había creado una instantánea en el mismo grupo de recursos para una migración. El siguiente despliegue en modo completo la eliminó, junto con una cuenta de almacenamiento temporal con 900 GB.

```
What-if habría mostrado:
  - Microsoft.Compute/snapshots/migracion-2026-06
  - Microsoft.Storage/storageAccounts/stmigraciontmp
pero no se ejecutó: el modo completo estaba en la canalización sin paso previo
```

Se corrige por diseño y no por prudencia:

```
grupos de recursos              1 compartido      4, uno por unidad desplegable
modo de despliegue              completo          incremental + pila de despliegue
`what-if` en la canalización    no                sí, obligatorio y revisado
bloqueos de recurso             ninguno           sobre almacén y servidor SQL
```

La pila de despliegue da lo que faltaba: una lista de qué pertenece a este despliegue, y un borrado ordenado cuando se retire el entorno.

Incidente 2 — reglas de NSG que aparecen y desaparecen.

Durante tres semanas hay fallos de conectividad intermitentes hacia la base de datos. La regla que permite el puerto 5432 desde snet-app está unas veces y otras no.

```
$ az network nsg rule list -g rg-cloudshop-red --nsg-name nsg-datos \
    --query "[].name" -o tsv
allow-lb-probe
deny-lateral
# falta allow-app-5432
```

La causa estaba repartida entre dos plantillas: la de red declaraba securityRules en línea dentro del NSG, y la de la aplicación declaraba allow-app-5432 como recurso hijo independiente. Cada despliegue de red sustituía la lista completa y se llevaba la regla de la otra.

```
forma de declarar reglas         mezclada (dos plantillas)  solo recurso hijo
convención documentada                    no                      sí
fallos de conectividad en 3 semanas       11                       0
```

La regla escrita para el resto de la plataforma es la que evita la repetición: para cada tipo de recurso hijo, una sola forma, y se decide una vez — reglas de NSG, ajustes de aplicación, reglas de firewall de base de datos y registros de DNS.

Incidente 3 — una cadena de conexión legible por veinte personas.

Una auditoría de accesos revisa el historial de despliegues:

```
$ az deployment group show -g rg-cloudshop-app-prod -n despliegue-app \
    --query "properties.outputs.cadenaConexion.value" -o tsv
Server=tcp:sql-cloudshop-prod.database.windows.net;…;Password=…
```

La plantilla la devolvía como salida para que la usara el paso siguiente de la canalización. Cualquiera con permiso de lectura sobre el grupo de recursos —veintiuna personas— podía consultarla, y estaba en las 47 entradas del historial.

```
secreto en salidas                   sí                    no
parámetro de contraseña          texto plano       @secure() + getSecret()
credencial de la aplicación        contraseña      identidad administrada (038)
historial con el secreto          47 entradas       purgado y contraseña rotada
```

La tercera fila es la corrección de fondo: el paso siguiente de la canalización no necesitaba la cadena, necesitaba poder conectarse. Con identidad administrada no hay ninguna cadena que pasar.

Incidente 4 — el cambio real que nadie vio entre el ruido.

Un despliegue rutinario deja el Always On de la aplicación de administración en false, y el servicio empieza a tardar ocho segundos en la primera petición de la mañana (clase 043). El what-if de esa ejecución lo mostraba:

```
~ Microsoft.Web/sites/app-admin
  ~ properties.siteConfig.alwaysOn: true → false
```

Entre otras 214 líneas, casi todas propiedades calculadas que aparecían en todas las ejecuciones. Nadie las leía desde hacía semanas.

```
líneas de what-if por ejecución        214              9
módulos                            1 de 1.100 líneas   7 módulos
propiedades calculadas declaradas       no          sí, 31 añadidas
revisión del what-if en el pull request  informal    obligatoria, con aprobación
```

La reducción de 214 a 9 no se consiguió filtrando: se consiguió declarando explícitamente las propiedades que el proveedor fijaba y la plantilla no mencionaba. El ruido era información sobre lo que faltaba en la plantilla.

Resumen del paso a infraestructura declarada:

```
tamaño de la plantilla principal           1.100 líneas    7 módulos versionados
grupos de recursos                         1 compartido    4 por unidad
modo de despliegue                         completo       incremental + pila
recursos borrados por accidente                 2               0
fallos de conectividad por reglas volátiles    11               0
secretos en el historial de despliegues        47               0
líneas de what-if por ejecución               214               9
tiempo de despliegue completo               22 min          14 min
```

```
reversión ensayada                          no        sí, versión anterior
                                                        del módulo, 6 min
```

La lección que esta clase traslada al proyecto de la clase 048 y a la parte 07: una plantilla no es un guion de creación, es la declaración del estado completo de lo que toca. Todo lo que no declara vuelve a su valor por omisión, todo lo que declara dos veces se lo disputan dos canalizaciones, y todo lo que devuelve queda escrito donde muchos pueden leerlo. Entender esas tres consecuencias es la diferencia entre automatizar la infraestructura y automatizar los incidentes.

Laboratorio guiado

Ejecuta desde la raíz:

```
python classes/part-03-azure-core-platform/047-bicep-plantillas-y-despliegues-por-alcance/lab.py
```

El laboratorio selecciona el motor de práctica iac y produce labresult.json. El escenario, sus comprobaciones y el artefacto esperado corresponden a esta clase; no requiere credenciales y deja explícito qué debe revalidarse en un sandbox real.

1. Lee exercise.steps y formula una predicción antes de ejecutar.
2. Ejecuta la práctica y verifica todos los elementos de checks.
3. Provoca el caso de negativetest y explica la señal observada.
4. Materializa infraestructura-bicep en evidence/ usando la plantilla indicada.
5. Para proveedor real, sigue sandbox.requires, registra costo y ejecuta destroy.

Evidencia esperada

El artefacto principal es un plan reproducible sin secretos ni cambios inesperados. Además, la entrega debe incluir el comando ejecutado, la salida estructurada y una conclusión que no exceda lo observado.

Reto verificable

Construye infraestructura-bicep para el caso CloudShop. Incluye una alternativa descartada, un supuesto que pueda falsarse, una prueba de fallo y una decisión de rollback.

■ Criterio de aceptación

- [ ] lab.py termina con código 0 y genera JSON válido.
- [ ] La entrega conecta al menos tres requisitos con mecanismos verificables.
- [ ] Existe una prueba positiva y una prueba negativa con evidencia.
- [ ] Seguridad, costo y operación aparecen como decisiones, no como anexos.
- [ ] Se declara una limitación y una condición que obligaría a revisar el diseño.
- [ ] Otra persona puede repetir el recorrido sin conocimiento tácito.

Errores frecuentes

Síntoma	Causa probable	Corrección
Un despliegue elimina recursos que nadie tocó en la plantilla	El modo completo borra todo lo que no aparece en ella, y el grupo de recursos era compartido	Un grupo de recursos por unidad desplegable, modo incremental con pila de despliegue y what-if obligatorio antes de aplicar.
Unas reglas de red aparecen y desaparecen según qué canalización se ejecutó la última	El mismo tipo de recurso hijo está declarado en línea en una plantilla y como recurso independiente en otra	Elige una sola forma por tipo de recurso hijo, documéntala y aplícala en todas las plantillas.
Una contraseña aparece en el historial de despliegues	Se devolvió como salida o se pasó a un parámetro sin @secure()	Marca el parámetro como seguro, obtén el valor con getSecret() y, mejor aún, sustituye la credencial por identidad administrada; rota lo que ya se expuso.
Nadie revisa la salida de what-if porque siempre muestra decenas de cambios	La plantilla no declara propiedades que el proveedor fija, y aparecen como diferencias en cada ejecución	Declara esas propiedades explícitamente y divide en módulos: el objetivo es una salida corta que se lea de verdad.
Un despliegue falla a medias y deja recursos creados	ARM no tiene transacción: no revierte lo ya aplicado	Escribe plantillas idempotentes y reejecutables; la respuesta a un fallo parcial es corregir y volver a desplegar.
Error de función no reconocida o de recurso no válido al desplegar	La plantilla está en un alcance que no permite ese recurso o esa función	Ajusta targetScope y despliega dentro del nivel correcto con módulos y el parámetro scope.

Seguridad, ética y costo

Trabaja con cuentas propias o sandboxes autorizados. No publiques secretos, identificadores reales ni datos personales. Antes de crear recursos pagos define presupuesto, etiquetas y comando de destrucción; después verifica que no queden recursos huérfanos. Los ejemplos locales enseñan contratos, pero no certifican cumplimiento ni disponibilidad de producción.

Preguntas de comprobación

1. ¿Qué problemas desaparecen al no haber archivo de estado, y qué capacidad se pierde a cambio?
2. ¿Desde qué alcance se crea un grupo de recursos, y por qué no se puede hacer desde el de grupo de recursos?
3. Describe exactamente cómo desaparece una regla de NSG al mezclar declaración en línea y recurso independiente.
4. ¿Por qué una salida de despliegue no puede contener un secreto, y qué hay que hacer si ya lo contuvo?
5. Tu what-if muestra 200 líneas en cada ejecución. ¿Qué te dice eso sobre la plantilla y cómo lo reduces?

Referencias

- Microsoft (2025). What is Bicep? — relación con ARM, módulos y compilación transparente. <https://learn.microsoft.com/en-us/azure/azure-resource-manager/bicep/overview>
- Microsoft (2025). Deployment scopes in Bicep — targetScope, módulos con scope y qué se crea en cada nivel. <https://learn.microsoft.com/en-us/azure/azure-resource-manager/bicep/deploy-to-subscription>
- Microsoft (2025). Azure Resource Manager deployment modes — incremental frente a completo y qué elimina cada uno. <https://learn.microsoft.com/en-us/azure/azure-resource-manager/templates/deployment-modes>
- Microsoft (2025). Deployment what-if operation — símbolos, limitaciones y ruido de proveedores. <https://learn.microsoft.com/en-us/azure/azure-resource-manager/bicep/deploy-what-if>
- Microsoft (2025). Use Azure Key Vault to pass secure parameter values — @secure(), getSecret() e historial de despliegues. <https://learn.microsoft.com/en-us/azure/azure-resource-manager/bicep/key-vault-parameter>
- Documentación oficial vigente del servicio implementado; registra URL y fecha de consulta.

Evaluación — 047 Bicep, plantillas y despliegues por alcance

Preguntas

1. Explica el problema que resuelve esta capacidad sin mencionar un proveedor.
2. Identifica una frontera de responsabilidad y su propietario.
3. Compara dos alternativas mediante confiabilidad, seguridad, costo y operación.
4. ¿Qué demuestra el laboratorio y qué no puede demostrar?
5. Propón un caso límite o una prueba de fallo.

Reto verificable

Entrega infraestructura-bicep con diagrama o configuración, evidencia de ejecución, alternativa descartada, riesgo residual y criterio de rollback.

Criterio de aceptación

- [ ] Resultado reproducible por una segunda persona.
- [ ] Evidencia vinculada a cada afirmación técnica.
- [ ] Prueba negativa o de fallo incluida.
- [ ] Costos y permisos expresados con unidades y alcance.
- [ ] Limitaciones declaradas sin presentar la simulación como producción.

Escala

Nivel	Evidencia
A — competente	Cumple todo y defiende trade-offs con datos.
B — en desarrollo	Cumple lo esencial; falta profundidad en una dimensión.
C — inicial	Artefacto parcial o conclusión sin evidencia suficiente.
0	Sin entrega reproducible.

048 — Proyecto: aplicación de tres capas en Azure

Parte: 03 — Azure: plataforma esencial Nivel: intermedio · Horas estimadas: 8 Laboratorio: capstone · Estado: EXECUTABLECORE

Propósito

Integrar las once clases anteriores en una plataforma Azure que funcione, se observe, falle de forma controlada y cueste algo explicable — y responder con evidencia la pregunta que abrió la parte: de todo lo aprendido en AWS, qué era arquitectura y ha reaparecido, y qué era mecanismo del proveedor. La respuesta a esa pregunta, escrita y con las excepciones señaladas, es lo que se lleva a la parte 04 y lo que hace que la tercera plataforma cueste una fracción de esfuerzo que la segunda.

Resultados de aprendizaje

Al finalizar podrás:

1. Trazar cada componente de la arquitectura hasta la clase que lo decidió y la alternativa descartada.
2. Separar el contrato portable del mecanismo específico del proveedor, señalando dónde no hubo equivalencia.
3. Declarar la línea base en dos capas: lo que la plantilla garantiza y lo que la directiva vigila.
4. Provocar tres fallos y medir detección, impacto y recuperación en cada uno.
5. Comparar dos plataformas con una unidad común, sabiendo por qué comparar facturas no sirve.

Conceptos centrales

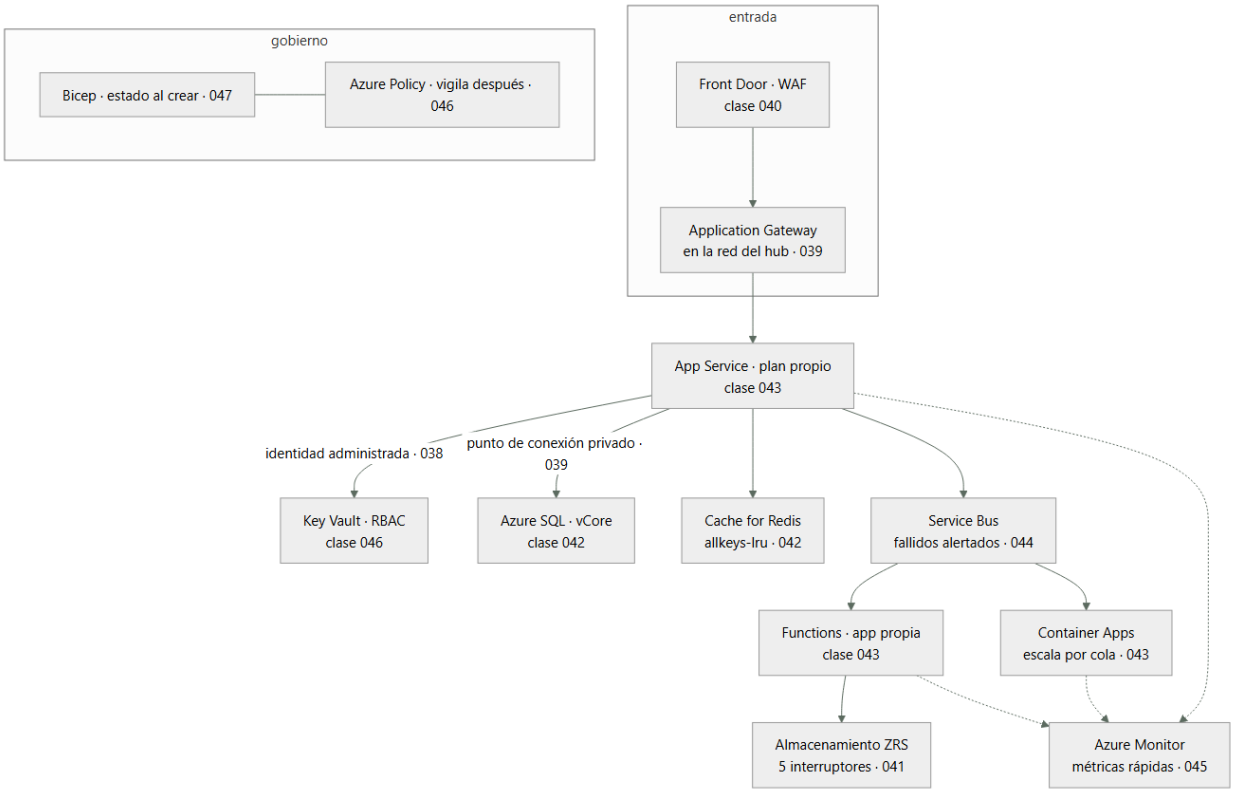
Concepto	Comprensión verificable
trazabilidad de decisión	Cada componente debe poder responder tres preguntas: qué requisito lo exige, qué alternativa se descartó y qué evidencia demuestra que cumple. Un componente sin las tres es una elección, no una decisión.
contrato portable	Lo que sobrevive al cambio de proveedor: el requisito, el patrón y la propiedad exigida al manejador. Cambia el nombre del servicio, no la obligación.
línea base en dos capas	La plantilla garantiza el estado al crear; la directiva vigila que siga así. Ninguna de las dos basta sola: la primera no impide un cambio posterior y la segunda no crea nada.
hallazgo de prueba de fallo	Lo que aparece al romper algo y no aparecía en ninguna configuración ni en ningún panel. Es el entregable real de un simulacro; que «todo aguantó» significa que el simulacro fue demasiado suave.
riesgo residual declarado	Lo que se decide no cubrir, con su motivo, su responsable y la condición que obligaría a revisarlo. Un riesgo no escrito no está aceptado: está ignorado.
costo por unidad de negocio	Única forma útil de comparar dos plataformas. Comparar facturas totales compara arquitecturas y tamaños distintos, no proveedores.

Modelo mental

En Azure, identidad y jerarquía organizacional preceden al recurso: tenant, management group, suscripción y resource group determinan el alcance de control.

Aplicado a esta clase, separa siempre cuatro planos: intención (qué necesita el usuario), configuración (qué declaramos), estado observado (qué existe de verdad) y evidencia (cómo sabemos que cumple). Confundirlos produce diseños que se ven correctos en un diagrama pero fallan al operar.

■ Flujo de razonamiento



Desarrollo

1. De dónde sale cada decisión, y qué trampa evitó

La arquitectura de la entrega no es una elección de servicios: es el resultado de once decisiones con su alternativa descartada. La tabla que hay que poder defender:

Componente	Requisito que lo exige	Alternativa descartada	Trampa de Azure que evita
Grupos de recursos por unidad	Aislamiento de despliegue y borrado (037)	Un grupo compartido	El modo completo borrando lo ajeno (047)
Identidad administrada asignada por el usuario	Sin secretos y flota compartida (038)	Secreto de aplicación con dos años	La rotación que nadie hace
Roles de datos además de gestión	Leer blobs y secretos (038)	Owner sobre la suscripción	AuthorizationPermissionMismatch con Owner
NAT Gateway + subred sin salida	Salida declarada y auditable (039)	Confiar en la salida implícita	La «subred privada» que no lo era
Puntos de conexión privados + zona DNS	Datos fuera de internet (039)	Endpoints de servicio	Funcionar por el camino público sin saberlo
App Service en planes separados	Radio de impacto (043)	Un plan para todo	El vecino ruidoso que degrada la tienda
Aplicación de funciones por perfil de carga	Aislar lo interactivo (043)	Una sola aplicación	El escalado por aplicación, no por función
Service Bus para el trabajo	Entrega con fallo visible (044)	Event Hubs «porque escala»	La partición bloqueada sin cola de fallidos
Azure SQL vCore	Diagnóstico por recurso (042)	Modelo DTU	El 100 % que no dice de qué
Alertas de métrica para lo urgente	Detección en un minuto (045)	Alertas de registro	Nueve minutos estructurales
Bicep + directiva	Estado al crear y vigilancia después (046, 047)	Solo plantillas	El control que alguien revierte

Cada fila responde a las tres preguntas de una decisión trazable: qué requisito, qué alternativa y qué evidencia. La cuarta columna es la que solo se puede escribir después de haber recorrido la parte: es la lista de errores que esta plataforma ya no puede cometer, porque su diseño los excluye en vez de confiar en que nadie los repita.

Y hay cuatro decisiones que se tomaron en contra de lo que sugería la traducción desde AWS, que conviene destacar porque son las que un revisor cuestionará:

1. No se usó el grupo de recursos como equivalente de la cuenta de AWS.
   Las cuotas son de la suscripción y no hay aislamiento de red entre
   grupos (037): la frontera de aislamiento es la suscripción.

2. No se buscó un equivalente del `Deny` de IAM.
   Azure RBAC es aditivo; lo que cumple esa función es Azure Policy,
   y opera en otro plano y con otro código de error (038, 046).

3. No se replicó el patrón de subred pública y privada por tabla de rutas.
   No existe: hay que declarar la salida y desactivarla explícitamente (039).

4. No se eligió redundancia geográfica para los datos de facturación.
   La región emparejada la asigna Microsoft y quedaba fuera de la
   jurisdicción aprobada (041): se usó ZRS más replicación explícita.

Las cuatro tienen la misma forma: el objetivo se conserva y el mecanismo no tiene equivalente. Reconocer esa forma es lo que hace útil la comparación entre proveedores; buscar la traducción literal es lo que produce los incidentes de esta parte.

2. Lo portable y lo del proveedor, con las excepciones señaladas

Este es el entregable intelectual de la parte, y sirve exactamente para una cosa: hacer que la parte 04 cueste una fracción del esfuerzo.

Lo que se repitió sin cambios —el contrato— y solo cambió de nombre:

Contrato que se conserva	En AWS (parte 02)	En Azure (parte 03)
Identidad de carga sin secretos	Rol de instancia y OIDC	Identidad administrada y federación
Sujeto de federación acotado	sub del rol federado	subject de la credencial federada
Frontera de aislamiento del entorno	Cuenta	Suscripción
Salida declarada y con costo por GB	NAT gateway	NAT Gateway
Acceso privado a servicios gestionados	Endpoint de interfaz	Punto de conexión privado
Entrega al menos una vez	SQS y sus reintentos	Service Bus y su bloqueo
Cola de mensajes fallidos con alerta a cero	Explícita	Automática, y sin alerta
Idempotencia en el manejador	Obligatoria	Obligatoria
Clave de partición como decisión irreversible	DynamoDB	Cosmos DB, con tope duro de 20 GB
Prueba negativa por control	Obligatoria	Obligatoria
Costo por unidad de negocio	Etiquetas y atribución	Etiquetas y atribución

Once filas. Ninguna de las once tuvo que volver a aprenderse, y todas se implementaron más rápido la segunda vez.

Lo que no tuvo equivalencia y hubo que aprender de cero:

```
dos planos de identidad          roles de directorio y roles de recurso
                                 son sistemas distintos (038)
actions frente a dataActions     gestionar un recurso no es leer sus datos (038)
sin subred privada por defecto   la salida es el estado inicial (039)
dos NSG en cadena                subred y NIC, ambos deben permitir (039)
dos límites de disco             el de la máquina suele mandar (040)
la cuenta como unidad            redundancia, red y claves son de la cuenta (041)
región emparejada asignada       no se elige el destino (041)
la unidad que escala             plan, aplicación de funciones, revisión (043)
Deny no toca lo existente        el gobierno actúa sobre la petición (046)
alcance de despliegue            qué se puede crear depende del nivel (047)
```

Diez conceptos. Ese es el tamaño real de la diferencia entre dos plataformas grandes: once contratos que se reutilizan y diez mecanismos que se aprenden. No es una proporción intuitiva, y explica por qué la segunda nube cuesta mucho menos de lo que la gente teme y la primera mucho más de lo que espera.

Y la conclusión operativa que se lleva a la parte 04: al abrir un proveedor nuevo, las once preguntas de la primera tabla ya están escritas y solo hay que averiguar cómo se llama la respuesta. El trabajo real está en detectar las excepciones —los sitios donde la respuesta es «aquí no existe»—, porque son las que producen incidentes cuando se asume equivalencia.

3. La línea base en dos capas, que es el entregable que más dura

La configuración de una plataforma se degrada. No por descuido excepcional, sino por caminos normales: alguien recrea un recurso desde una plantilla vieja, una excepción temporal se queda, un despliegue sobrescribe un ajuste. Un control verificado una vez es un control verificado en el pasado.

Por eso la línea base se declara en dos capas que hacen cosas distintas:

```
capa 1 · Bicep      fija el estado en el momento de crear
                    lo que no declara vuelve a su valor por omisión (047)
capa 2 · Policy     vigila que siga así después
                    y en modo Audit lo hace sin bloquear nada (046)
```

Ninguna basta sola. La plantilla no impide que alguien cambie algo mañana desde el portal; la directiva no crea nada y solo constata. Juntas, un cambio fuera de la plantilla aparece como incumplimiento en horas.

La línea base concreta de esta entrega, con la clase que la justifica:

```
red
  ninguna subred con salida no declarada                          039
  todo acceso a datos por punto de conexión privado con zona DNS  039
  segmentación explícita entre subredes                           039
datos
  acceso público deshabilitado en almacenamiento y base de datos  041, 042
  acceso con clave compartida deshabilitado                       041
  cinco interruptores de protección activos y probados            041
  retención a largo plazo y bloqueo sobre el servidor lógico      042
identidad
  cero secretos de aplicación; identidad administrada en todo     038
  Key Vault con RBAC y protección contra purga                    046
  roles privilegiados elegibles, no permanentes                   038
cómputo
  un plan por radio de impacto; una app de funciones por perfil   043
  sonda de estado que ejercita dependencias                       040
asíncrono
  manejadores idempotentes                                        044
  alerta de mensajes fallidos con umbral cero                     044
  alerta de acumulación en cada consumidor                        045
observabilidad
  diagnóstico activado por directiva en el 100 % de los recursos  045
  agregaciones con `sum(itemCount)`, nunca `count()`              045
  alertas de métrica para lo urgente                              045
gobierno
  cero exclusiones de ámbito; excepciones como exenciones con caducidad 046
  etiquetas de atribución de costo obligatorias                   037
```

Veintitrés afirmaciones. Cada una tiene su prueba negativa en el guion de verificación, y cada una tiene además una directiva en modo Audit que la vigila. Esa duplicidad es deliberada: el guion demuestra el estado hoy, la directiva detecta la desviación mañana.

Y el criterio para decidir qué entra en la línea base, que evita que crezca hasta ser inmanejable: entra lo que, si se revierte, produce un incidente o una brecha. Lo demás es una preferencia y se documenta en otro sitio.

4. Tres fallos provocados y lo que enseñó cada uno

Un simulacro en el que todo aguanta no aporta información: significa que fue demasiado suave. Los tres de esta entrega se eligieron porque atacan tres supuestos distintos.

Fallo 1 — caída de una zona. Se detienen las instancias de la zona 1 en el plan de App Service y se fuerza la conmutación de la base de datos.

```
detección por alerta de métrica             52 s
p95 durante el episodio                    212 ms  (SLO 500 ms)
errores HTTP observados por el cliente     1.104
recuperación completa                      3 min 40 s
```

Y el hallazgo, que ninguna configuración mostraba. Los 1.104 errores no vinieron del frontal ni del balanceador: vinieron de la base de datos durante los 38 segundos de su conmutación entre réplicas. Azure SQL devuelve errores transitorios identificables durante ese intervalo —40613, 40501, 49918— y la aplicación no los reintentaba: los propagaba como error 500.

```
síntoma observable      ninguno en los paneles de infraestructura
consecuencia real       1.104 peticiones perdidas, 47 pedidos sin registrar
causa                   sin manejo de fallos transitorios en el acceso a datos
```

La corrección no es de infraestructura: es de código, y es obligatoria en Azure SQL porque la conmutación es parte del funcionamiento normal, no un incidente.

```
1. reintento con retroceso exponencial para los códigos transitorios conocidos
2. tiempo de espera de conexión menor que el plazo de la petición
3. la prueba de fallo pasa a incluir una conmutación forzada, no solo
   la caída de las instancias de cómputo
```

Fallo 2 — revocación de la clave gestionada por el cliente. Se retira el permiso de la identidad sobre la clave que cifra el almacenamiento, para comprobar que el control funciona y medir su radio.

```
tiempo hasta que el almacenamiento queda inaccesible   11 min (caché de la clave)
servicios afectados esperados                          1 (subida de facturas)
servicios afectados reales                             3
recuperación tras restaurar el permiso                 6 min
```

El hallazgo: la misma cuenta de almacenamiento recibía la exportación del registro de actividad y el destino de mensajes fallidos de Event Grid. Revocar la clave no solo detuvo la subida de facturas: detuvo la evidencia de auditoría y el destino de los eventos que fallaban, justo cuando más falta hacían. El control funcionaba; el radio de impacto estaba mal estimado porque una cuenta servía a tres propósitos.

```
corrección   separar por propósito: facturas, auditoría y eventos fallidos
             en cuentas distintas, con claves distintas
registro     el orden de las operaciones de revocación y restauración
             queda documentado como procedimiento, no improvisado
```

Fallo 3 — consumidor detenido en silencio. Se detiene el consumidor de notificaciones sin apagar nada más, que es la forma en que este sistema falló dos veces durante la parte.

```
detección                          ninguna (2 días)         4 min 20 s
mensajes acumulados al detectar    41.000                     680
señal que lo detectó                 —              profundidad de cola
```

El hallazgo: la alerta se disparó correctamente y apuntaba a un procedimiento que describía cómo reiniciar el consumidor, sin decir cómo comprobar si los mensajes acumulados se habían procesado o duplicado tras el reinicio. La detección funcionaba y la respuesta estaba incompleta.

```
corrección   el procedimiento incluye la verificación posterior:
             recuento de notificaciones enviadas frente a pedidos del intervalo,
             apoyándose en la idempotencia del manejador (044)
```

Los tres hallazgos tienen la misma forma, y merece la pena decirlo explícitamente porque es la lección de la parte: la infraestructura aguantó los tres fallos y en los tres se perdió algo. Lo que falló fue el manejo de errores transitorios, la estimación del radio de impacto y la completitud del procedimiento. Ninguna de las tres cosas aparece en una revisión de configuración.

5. La entrega, la comparación honesta y la pregunta que abre la parte 04

La entrega, sin conocimiento tácito. El criterio es el de la clase 036 y no ha cambiado: otra persona debe poder repetir el recorrido sin preguntar nada.

```
infraestructura     Bicep en 7 módulos versionados, con `what-if` legible
línea base          23 afirmaciones, cada una con su prueba negativa
verificar.sh        ejecuta las 23 y devuelve código de salida
directivas          la iniciativa que vigila las 23 después
ADR                 11 decisiones con su alternativa descartada
riesgos residuales  5, con responsable y condición de revisión
consultas KQL       4, guardadas y enlazadas desde el manual
procedimientos      3 fallos ensayados, con su verificación posterior
```

```
línea base medida   rendimiento, costo y costo por pedido
```

Los cinco riesgos residuales merecen figurar porque declararlos es parte del trabajo:

```
1. una sola región: el RTO ante caída regional es de horas, aceptado
2. la conmutación de la cuenta de almacenamiento la deja en LRS (041)
3. el paso por el concentrador cuesta 32 USD/mes en inspección (039)
4. dos cuentas de emergencia con acceso permanente, documentadas (038)
5. el nivel estándar de Service Bus no admite punto de conexión privado;
   se acepta hasta que el volumen justifique el nivel premium (044)
```

La comparación con la parte 02, hecha de forma que signifique algo.

La comparación ingenua es inútil y conviene enseñarla antes de descartarla:

```
plataforma AWS (clase 036)     688,40 USD/mes
plataforma Azure, primera medida  1.389 USD/mes
```

Eso no compara proveedores: compara dos arquitecturas distintas y dos conjuntos de capacidades distintos. Al desglosarlo, casi todo el hueco tenía nombre y apellidos:

```
                                          primera      las palancas
base de datos aprovisionada frente a elástica   390            390
pasarela de aplicación con WAF                  185            185
telemetría en plan de análisis para todo        276            110   (045)
planes de Defender en todo                      230             96   (046)
cómputo, red, mensajería y almacenamiento       308            159
escalado automático de Cosmos mal dimensionado   —              —    (042)
                                              ■■■■■■■        ■■■■■■■
                                              1.389            940
```

Y la única cifra que compara de verdad, porque normaliza por trabajo hecho:

```
AWS    0,000702 USD por pedido
Azure  0,000959 USD por pedido    (+37 %)
```

Ese 37 % es real y tiene dos partidas identificadas: la pasarela con WAF y la ingesta de telemetría, ambas más caras que sus equivalentes de la parte 02 al mismo volumen. No es una penalización del proveedor: es el precio de dos capacidades concretas, y con ese desglose se puede decidir si compensan. Sin él, la conversación se convierte en «Azure es más caro», que no es una afirmación accionable.

El rendimiento, medido con la misma carga y el mismo método:

```
rps sostenidos          987,4         984,1
p50                    38,4 ms       41,2 ms
p95                    91,2 ms       96,8 ms
p99                   184,7 ms      203,5 ms
errores                    0             0
p99/p50                  4,8×          4,9×
```

La conclusión honesta es que no hay diferencia significativa, y esa es una información valiosa: descarta el rendimiento como criterio de elección a esta escala y devuelve la decisión a donde corresponde —capacidades, costo por unidad, competencias del equipo y requisitos de residencia—.

Y la pregunta que abre la parte 04. Con dos plataformas construidas y comparadas, la pregunta ya no es cuál es mejor. Es esta:

> De las once filas del contrato portable, ¿cuántas seguirán siendo válidas en Google Cloud, y cuáles de los diez conceptos sin equivalencia volverán a aparecer con otra forma?

La hipótesis que se lleva escrita —para poder equivocarse de forma comprobable— es que el contrato se conserva entero y que las excepciones serán distintas: donde Azure separó identidad de directorio e identidad de recurso, Google Cloud tiene una jerarquía de proyectos y carpetas con herencia propia; donde Azure no tiene subred privada, otro proveedor tendrá otra asimetría. Lo que se comprueba en la parte 04 no es una lista de servicios: es si el método de esta parte —contrato primero, excepciones señaladas, prueba negativa siempre— funciona a la tercera.

Ejemplo trabajado

Entrega del capstone de la parte 03, con las cifras que se llevan a la parte 04.

Verificación completa. Las 23 afirmaciones de la línea base, cada una con su prueba negativa:

```
$ ./verificar.sh
```

```
✓ federación deniega desde otro repositorio        AADSTS70021
✓ almacenamiento sin acceso público                403 desde fuera
✓ clave compartida deshabilitada                   KeyBasedAuthenticationNotPermitted
✓ nombre de blob resuelve a IP privada             10.20.6.5
✓ subred de datos sin salida a internet            tiempo agotado a los 5 s
✓ base de datos sin acceso público                 conexión rechazada
✓ tráfico lateral bloqueado salvo 5432             Deny · deny-lateral
✓ sonda del balanceador permitida                  Allow · allow-lb-probe
✓ recuperación de contenedor                       50 blobs restaurados
✓ restauración de base de datos                    1.284.391 filas · 11 min
✓ protección contra purga del almacén              operación rechazada
✓ directiva de red de almacenamiento               RequestDisallowedByPolicy
✓ antimalware en cuenta de cargas                  alerta en 4 min
✓ rotación de secreto con tráfico en curso         0 errores
✓ mensajes fallidos alertan a cero                 aviso en 15 min
✓ acumulación en cola alerta                       aviso en 4 min
✓ diagnóstico en el 100 % de recursos              61 de 61
✓ recuento de peticiones exacto                    sum(itemCount) verificado
✓ salud del servicio                               200
✓ pedido con comilla simple y punto y coma         201
✓ despliegue en modo incremental                   0 eliminaciones en what-if
✓ sin secretos en el historial de despliegues      0 coincidencias
✓ roles Owner permanentes                          2, ambos de emergencia
23/23 correctas
```

Línea base medida:

```
rps 984,1 · p50 41,2 ms · p95 96,8 ms · p99 203,5 ms · errores 0
L = 984 × 0,0412 = 40,5 de 50 solicitados → medida válida
p99/p50 = 4,9×
costo 940 USD/mes · 0,000959 USD/pedido
```

La comprobación de la ley de Little es la misma disciplina de la clase 036: si la concurrencia observada no cuadra con la solicitada, el generador de carga no estaba aplicando lo que decía y la medida no sirve.

Los tres fallos, con lo aprendido en cada uno:

```
caída de zona                  52 s        212 ms      1.104 peticiones
                                                       47 pedidos     los errores transitorios
                                                                      de SQL son normales:
                                                                      hay que reintentarlos

revocación de clave propia     11 min         —        3 servicios,    el radio de impacto
                                                       no 1            se estima mal cuando
                                                                       una cuenta sirve a tres
                                                                       propósitos

consumidor detenido          4 min 20 s    sin cambio   680 mensajes   detectar no es responder:
                                                       retrasados     el procedimiento no
                                                                      verificaba el resultado
```

El hallazgo que justificó el capstone. Todo estaba configurado correctamente, las 23 pruebas negativas pasaban y los paneles estuvieron en verde durante la caída de zona:

```
requests
| where timestamp between (datetime(2026-07-28 10:14) .. datetime(2026-07-28 10:18))
| summarize peticiones = sum(itemCount),
            fallidas = sumif(itemCount, success == false)

peticiones  58.402
fallidas     1.104        ← 1,9 %, por debajo del umbral de alerta del 3 %

exceptions
| where timestamp between (datetime(2026-07-28 10:14) .. datetime(2026-07-28 10:18))
| summarize por = count() by outerMessage

1.104 × "Database 'pedidos' on server '…' is not currently available" (40613)
```

Mil ciento cuatro errores con todas las alarmas en verde, porque el umbral estaba por encima y porque la infraestructura hizo exactamente lo que debía: conmutar en 38 segundos. La pérdida no la causó el fallo, la causó el código que no esperaba el fallo.

```
síntoma observable   ninguno: alarmas en verde, conmutación correcta,
                     p95 dentro del SLO
consecuencia real    47 pedidos no registrados
```

```
causa                sin reintento de errores transitorios documentados
qué lo destapó       provocar el fallo, no revisar la configuración
```

Se corrige y se amplía el alcance del simulacro:

```
1. reintento con retroceso para los códigos transitorios de Azure SQL
2. umbral de alerta por tasa de error bajado del 3 % al 1 %, con ventana corta
3. la prueba de fallo incluye conmutación forzada de base de datos
4. la comprobación posterior cuenta pedidos registrados frente a peticiones
   aceptadas: la diferencia debe ser cero
```

Repetido el simulacro con las cuatro correcciones:

```
errores durante la conmutación    1.104            0
pedidos no registrados               47            0
detección                       no hubo         38 s
```

Se entrega a la parte 04 con:

```
contrato portable    11 filas verificadas en dos proveedores
excepciones          10 conceptos sin equivalencia, con su síntoma
línea base           23 afirmaciones y su guion de verificación
ADR                  11 decisiones con alternativa descartada
riesgos residuales   5, con responsable y condición de revisión
comparación          costo por pedido y percentiles, con el desglose
                     que explica el 37 % de diferencia
```

Y la hipótesis escrita, para poder equivocarse de forma comprobable en la parte 04:

> El contrato de once filas se conservará entero en Google Cloud. Las excepciones serán otras diez, distintas de estas, y la que más incidentes producirá volverá a ser una en la que el sistema funcione por el camino equivocado en lugar de dar un error.

La lección que esta parte deja al programa: las tres pruebas de fallo mostraron infraestructura que aguantó y sistemas que perdieron datos. Configurar bien es necesario y no es suficiente; lo que separa una plataforma operable de una bien configurada es haber roto cada supuesto a propósito y haber medido qué se perdió.

Laboratorio guiado

Ejecuta desde la raíz:

```
python classes/part-03-azure-core-platform/048-proyecto-aplicacion-de-tres-capas-en-azure/lab.py
```

El laboratorio selecciona el motor de práctica capstone y produce labresult.json. El escenario, sus comprobaciones y el artefacto esperado corresponden a esta clase; no requiere credenciales y deja explícito qué debe revalidarse en un sandbox real.

1. Lee exercise.steps y formula una predicción antes de ejecutar.
2. Ejecuta la práctica y verifica todos los elementos de checks.
3. Provoca el caso de negativetest y explica la señal observada.
4. Materializa plataforma-azure en evidence/ usando la plantilla indicada.
5. Para proveedor real, sigue sandbox.requires, registra costo y ejecuta destroy.

Evidencia esperada

El artefacto principal es un incremento integrado, demostrable y documentado. Además, la entrega debe incluir el comando ejecutado, la salida estructurada y una conclusión que no exceda lo observado.

Reto verificable

Construye plataforma-azure para el caso CloudShop. Incluye una alternativa descartada, un supuesto que pueda falsarse, una prueba de fallo y una decisión de rollback.

■ Criterio de aceptación

- [ ] lab.py termina con código 0 y genera JSON válido.
- [ ] La entrega conecta al menos tres requisitos con mecanismos verificables.

- [ ] Existe una prueba positiva y una prueba negativa con evidencia.
- [ ] Seguridad, costo y operación aparecen como decisiones, no como anexos.
- [ ] Se declara una limitación y una condición que obligaría a revisar el diseño.
- [ ] Otra persona puede repetir el recorrido sin conocimiento tácito.

■■ Errores frecuentes

Síntoma	Causa probable	Corrección
Un simulacro en el que todo aguanta y no se aprende nada	El fallo elegido no atacaba ningún supuesto real del diseño	Elige fallos que rompan un supuesto distinto cada uno y mide pérdida real, no solo disponibilidad.
Se pierden peticiones durante una conmutación de base de datos con todas las alarmas en verde	Los errores transitorios de Azure SQL son parte del funcionamiento normal y la aplicación no los reintenta	Reintento con retroceso para los códigos transitorios, umbral de alerta acorde y conmutación forzada dentro del simulacro.
Revocar una clave afecta a más servicios de los previstos	Una misma cuenta de almacenamiento servía a tres propósitos distintos	Separa por propósito y clave, y documenta el orden de las operaciones de revocación y restauración.
La alerta se dispara, el procedimiento se ejecuta y nadie sabe si se perdió trabajo	El procedimiento describe cómo restablecer y no cómo verificar el resultado	Todo procedimiento termina con una comprobación cuantitativa apoyada en la idempotencia del manejador.
Se comparan dos plataformas por su factura total y la conclusión no es accionable	Las facturas comparan arquitecturas y capacidades distintas, no proveedores	Normaliza por unidad de negocio y desglosa la diferencia hasta que cada partida tenga nombre.
Un control desaparece semanas después de haberse verificado	La plantilla fija el estado al crear y nada vigila lo que ocurre después	Línea base en dos capas: la plantilla garantiza y una directiva en modo Audit detecta la desviación en horas.

■ Seguridad, ética y costo

Trabaja con cuentas propias o sandboxes autorizados. No publiques secretos, identificadores reales ni datos personales. Antes de crear recursos pagos define presupuesto, etiquetas y comando de destrucción; después verifica que no queden recursos huérfanos. Los ejemplos locales enseñan contratos, pero no certifican cumplimiento ni disponibilidad de producción.

■ Preguntas de comprobación

1. Elige tres componentes de la arquitectura y di qué requisito los exige, qué alternativa se descartó y qué trampa de Azure evitan.
2. ¿Cuántos contratos se reutilizaron de la parte 02 y cuántos conceptos hubo que aprender de cero? ¿Qué implica esa proporción para la parte 04?
3. ¿Por qué la línea base necesita dos capas, y qué falla si solo existe una de ellas?
4. Durante la caída de zona hubo 1.104 errores con las alarmas en verde. ¿Qué falló exactamente y por qué no lo mostraba ninguna revisión de configuración?
5. ¿Qué hace inútil comparar las facturas de dos plataformas, y qué comparación sí es accionable?

Referencias

- Microsoft (2025). Azure Well-Architected Framework — los cinco pilares como rejilla de revisión de la entrega. <https://learn.microsoft.com/en-us/azure/well-architected/>
- Microsoft (2025). Troubleshoot transient connection errors in Azure SQL — códigos transitorios y lógica de reintento obligatoria. <https://learn.microsoft.com/en-us/azure/azure-sql/database/troubleshoot-common-connectivity-issues>

- Microsoft (2025). Reliability in Azure: availability zones — comportamiento de la conmutación y qué mide un simulacro de zona. <https://learn.microsoft.com/en-us/azure/reliability/availability-zones-overview>
- Microsoft (2025). Cloud Adoption Framework: landing zone design areas — línea base declarada y vigilada, y sus áreas de decisión. <https://learn.microsoft.com/en-us/azure/cloud-adoption-framework/ready/landing-zone/design-areas>
- Microsoft (2025). Chaos Studio experiment design — provocar fallos con alcance acotado y medir el resultado. <https://learn.microsoft.com/en-us/azure/chaos-studio/chaos-studio-overview>
- Documentación oficial vigente del servicio implementado; registra URL y fecha de consulta.

Descargar: Parte 03 en PDF · Recorrido de Azure en PDF · Manual integral

Anterior	Índice	Siguiente
← 047 · Bicep, plantillas y despliegues por alcance	Parte 03 · Programa	049 · Organización, folders, proyectos, billing y cuotas →

Evaluación — 048 Proyecto: aplicación de tres capas en Azure

Preguntas

1. Explica el problema que resuelve esta capacidad sin mencionar un proveedor.
2. Identifica una frontera de responsabilidad y su propietario.
3. Compara dos alternativas mediante confiabilidad, seguridad, costo y operación.
4. ¿Qué demuestra el laboratorio y qué no puede demostrar?
5. Propón un caso límite o una prueba de fallo.

Reto verificable

Entrega plataforma-azure con diagrama o configuración, evidencia de ejecución, alternativa descartada, riesgo residual y criterio de rollback.

Criterio de aceptación

- [ ] Resultado reproducible por una segunda persona.
- [ ] Evidencia vinculada a cada afirmación técnica.
- [ ] Prueba negativa o de fallo incluida.
- [ ] Costos y permisos expresados con unidades y alcance.
- [ ] Limitaciones declaradas sin presentar la simulación como producción.

Escala

Nivel	Evidencia
A — competente	Cumple todo y defiende trade-offs con datos.
B — en desarrollo	Cumple lo esencial; falta profundidad en una dimensión.
C — inicial	Artefacto parcial o conclusión sin evidencia suficiente.
0	Sin entrega reproducible.

Parte 06 — Kubernetes y plataformas administradas

083 — EKS, AKS y GKE: similitudes y diferencias

Parte: 06 — Kubernetes y plataformas administradas Nivel: intermedio-avanzado · Horas estimadas: 4 Laboratorio: decision · Estado: EXECUTABLECORE

Propósito

Comparar las tres plataformas gestionadas de Kubernetes por lo que de verdad diferencia la operación, y no por una tabla de funciones. Tres ejes deciden casi todo: quién decide cuándo actualizas, cuántos pods caben en tu plan de direcciones y cómo obtiene una carga la identidad de la nube. Y la conclusión que interesa al programa es cuantificable: qué parte del trabajo de las clases 073 a 082 se conserva al cambiar de proveedor y qué parte hay que rehacer.

Resultados de aprendizaje

Al finalizar podrás:

1. Delimitar qué gestiona el proveedor y qué sigue siendo trabajo propio.
2. Anticipar una actualización forzada por fin de soporte y lo que puede bloquearla.
3. Calcular cuántos pods caben según el modelo de red de cada plataforma.
4. Federar la identidad de una carga con la nube, acotando la confianza correctamente.
5. Estimar qué se conserva y qué se rehace al cambiar de plataforma gestionada.

Conceptos centrales

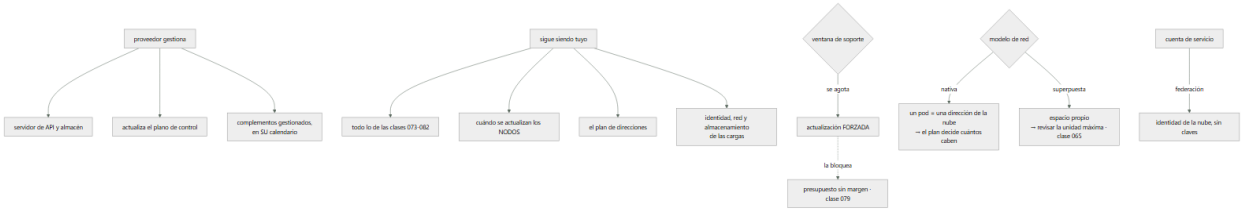
Concepto	Comprensión verificable
plano de control gestionado	El proveedor opera el servidor de API y el almacén de estado. No opera tus cargas ni tus complementos, que es donde está casi todo el trabajo de esta parte.
ventana de soporte	Periodo durante el cual una versión recibe correcciones. Al agotarse, el proveedor actualiza el clúster aunque nadie lo pida.
dirección de pod nativa de la red	Cada pod recibe una dirección del espacio de la nube. Simplifica el enrutamiento y consume el plan de direcciones de las clases 039 y 051.
red superpuesta	Los pods usan un espacio propio y su tráfico se encapsula. Ahorra direcciones y añade una capa —con la trampa de la unidad máxima de transmisión de la clase 065.
identidad federada de carga	La cuenta de servicio del clúster obtiene credenciales de la nube sin ninguna clave. Séptima aparición del mismo contrato del programa.
complemento gestionado	Pieza que el proveedor instala y actualiza en su calendario. Quita trabajo y quita control: su versión puede cambiar sin que nadie lo decida.
desviación de versiones	Diferencia admitida entre el plano de control y los nodos. Permite actualizar por fases y tiene un límite que, si se cruza, deja nodos sin soporte.

Modelo mental

Kubernetes es un sistema de reconciliación: declaras estado deseado y controladores reducen continuamente la diferencia con el estado observado.

Aplicado a esta clase, separa siempre cuatro planos: intención (qué necesita el usuario), configuración (qué declaramos), estado observado (qué existe de verdad) y evidencia (cómo sabemos que cumple). Confundirlos produce diseños que se ven correctos en un diagrama pero fallan al operar.

■ Flujo de razonamiento



Desarrollo

1. Qué gestiona el proveedor, y qué no

«Gestionado» se entiende como más de lo que es, y conviene delimitarlo antes de comparar nada:

```
el proveedor gestiona
  el servidor de API y el almacén de estado: disponibilidad, copias, parches
  la actualización del plano de control
  algunos complementos, en su calendario
  la integración con su red, su almacenamiento y su identidad

sigue siendo trabajo propio
  ABSOLUTAMENTE todo lo de las clases 073 a 082
  cuándo y cómo se actualizan los NODOS
  el plan de direcciones y sus consecuencias
  las políticas de red, de admisión y de permisos
  la observabilidad, las copias de las cargas con estado y su restauración
```

Es decir: el proveedor quita el trabajo que menos se nota y deja el que produce los incidentes de esta parte. Y merece decirse con claridad porque la expectativa contraria lleva a no dedicar equipo:

> Un clúster gestionado no reduce el trabajo de operar aplicaciones en Kubernetes. Reduce el de operar Kubernetes.

Sobre el coste del plano de control, las tres cobran de forma distinta y la cifra rara vez decide:

```
una tarifa por hora y por clúster, con independencia del tamaño
o una tarifa de gestión con algún nivel gratuito acotado
o un nivel gratuito sin acuerdo de servicio y uno de pago con él
```

Lo que sí decide es la consecuencia estructural: un clúster tiene un coste fijo, así que muchos clústeres pequeños cuestan más que uno grande. Y esa aritmética empuja a compartir clúster entre equipos, lo que devuelve al aislamiento de la clase 080 y a su límite: los espacios de nombres no bastan para inquilinos no confiables.

Y la decisión que aparece pronto, con la misma forma en las tres plataformas:

```
un clúster por entorno            frontera clara, coste fijo × entornos
un clúster con espacios por equipo  más barato, aislamiento más débil
un clúster por equipo y entorno     lo más aislado y lo más caro de operar
```

La respuesta razonable en la mayoría de las organizaciones es un clúster por entorno con espacios de nombres por equipo, y producción separada de todo lo demás — que es la misma conclusión a la que llegaron las clases 025, 037 y 049 con cuentas, suscripciones y proyectos. Cuarta aparición del mismo criterio.

2. Quién decide cuándo actualizas

Este es el eje que más cambia la operación y el que menos aparece en las comparativas.

Cada versión tiene una ventana de soporte de aproximadamente un año, y al agotarse el proveedor actualiza el clúster:

```
se publica una versión              cada ~4 meses
soporte estándar                    ~12-14 meses desde su publicación
soporte extendido, si existe         más meses, con sobrecoste
al agotarse                         ACTUALIZACIÓN AUTOMÁTICA por el proveedor
```

De ahí salen tres consecuencias operativas:

Actualizar es una tarea recurrente, no un proyecto. Con una ventana de un año y una versión cada cuatro meses, un clúster se actualiza dos o tres veces al año, siempre. Un equipo que trate cada actualización como un proyecto excepcional vive permanentemente en uno.

Lo que bloquea la actualización lo has puesto tú. El proveedor respeta los presupuestos de interrupción de la clase 079, así que un presupuesto sin margen bloquea también su actualización automática. Y ahí ocurre lo peor:

```
el proveedor intenta actualizar
un presupuesto sin margen bloquea el vaciado del nodo
la actualización se detiene o se agota su plazo
el clúster se queda con nodos en dos versiones, o sin actualizar
→ y el aviso llega por correo a una dirección que nadie lee
```

Es el incidente de la clase 079 con un origen distinto y una consecuencia mayor, porque termina con nodos ejecutando una versión sin soporte.

La desviación de versiones limita el orden. El plano de control se actualiza primero y los nodos después, y hay un margen máximo entre ambos:

```
los nodos pueden ir por detrás del plano de control, hasta un límite
nunca por delante
→ y saltarse versiones intermedias no está soportado:
  hay que pasar por cada una
```

La última línea es la que convierte un retraso en un problema: un clúster tres versiones por detrás necesita tres actualizaciones encadenadas, cada una con su ventana y su riesgo.

Y las funciones que se retiran son la otra mitad del trabajo. Cada versión elimina versiones de API antiguas, y un manifiesto que las use deja de aplicarse:

```
$ kubectl get --raw /metrics | grep apiserver_requested_deprecated_apis
```

Esa métrica dice qué APIs obsoletas se están usando ahora mismo y quién. Debería revisarse antes de cada actualización, y es la comprobación que evita descubrir el problema con el clúster ya actualizado.

Y una recomendación que se paga sola: un clúster de pruebas una versión por delante del de producción. Cuesta el plano de control y unos nodos pequeños, y convierte cada actualización en algo ya ensayado.

3. Cuántos pods caben: el modelo de red decide

Aquí está la divergencia real entre plataformas, y tiene consecuencias que se descubren tarde.

Hay dos modelos:

```
DIRECCIONES NATIVAS
  cada pod recibe una dirección del espacio de la nube
  ventaja: el enrutamiento es directo; los cortafuegos y el registro de flujo
           de las clases 039 y 051 ven al pod, no al nodo
  coste:   consume el plan de direcciones, y mucho

RED SUPERPUESTA
  los pods usan un espacio propio y el tráfico se encapsula
  ventaja: no consume direcciones de la nube
  coste:   una capa más, y la unidad máxima de transmisión de la clase 065
```

Y el cálculo que hay que hacer antes de crear el clúster, no después:

```
con direcciones nativas y una subred /22 (1.024 direcciones)
  reservadas por la nube                    ~5
  una por nodo                              × nodos
  una por pod                               × pods
  y direcciones en tránsito durante los despliegues, que no se liberan al instante

40 nodos × 30 pods = 1.200 direcciones necesarias
→ una /22 NO llega; hace falta al menos una /21, con margen
```

La última línea del cálculo es la que se olvida: durante un despliegue coexisten pods viejos y nuevos, así que el pico de direcciones es mayor que el estado estable. Y el síntoma del agotamiento es característico y despista:

```
pods en ContainerCreating de forma intermitente, sobre todo al desplegar
  failed to allocate for range 0: no IP addresses available in range
```

Parece un problema del complemento de red y es un problema de aritmética de la clase 051.

Y hay un segundo límite propio de algunos modelos: el número de pods por nodo puede depender del tipo de máquina, porque las direcciones se asignan al nodo en bloques ligados a sus interfaces. Un nodo pequeño puede admitir muy pocos pods aunque le sobren CPU y memoria:

```
síntoma   nodos con recursos libres y pods en Pending
          el suceso menciona pods insuficientes, no CPU ni memoria
```

Ese caso se corrige con máquinas mayores o con modos que reparten direcciones por prefijo en vez de una a una, y hay que conocerlo porque contradice la intuición de la clase 078: no siempre falta capacidad; a veces faltan direcciones.

Y los rangos secundarios que la clase 051 pedía reservar aparecen aquí con su factura: en las plataformas que los usan, pods y servicios consumen bloques propios, grandes y fijados al crear el clúster. Ampliarlos después no siempre es posible, lo que convierte el plan de direcciones en otra decisión de un solo sentido.

Y la conclusión que enlaza las dos cosas:

```
el tamaño máximo del clúster lo decide, en la práctica,
el plan de direcciones que alguien hizo antes de crearlo
```

4. La identidad, por séptima vez

Las tres plataformas resuelven lo mismo con el mismo mecanismo y distinta sintaxis: una carga del clúster obtiene credenciales de la nube sin ninguna clave.

```
la cuenta de servicio del pod recibe un testigo firmado por el clúster
la nube confía en el emisor del clúster
y cambia ese testigo por credenciales suyas
```

Es el mismo contrato de las clases 026, 038, 050, 069 y 080. Séptima aparición, con la misma pieza crítica:

```
la confianza del lado de la NUBE debe acotarse al espacio de nombres
y a la cuenta de servicio concretos
```

Sin acotar, cualquier pod del clúster puede pedir esa identidad. Y es exactamente el mismo fallo que el sujeto sin acotar de las federaciones anteriores, con un cuarto vocabulario:

```
condición de confianza correcta
  emisor = el del clúster
  sujeto = system:serviceaccount:<espacio>:<cuenta>
```

La comprobación es la misma prueba negativa que este programa ha ejecutado cuatro veces:

```
# desde un pod con la cuenta correcta
$ kubectl exec -n tienda api-x -- /probar-acceso-nube
ok
# desde un pod de otro espacio con otra cuenta
$ kubectl exec -n desarrollo prueba-y -- /probar-acceso-nube
acceso denegado                                                    ✓
```

Y la parte de la sintaxis, que es lo único que cambia entre plataformas:

```
# el mecanismo es el mismo; la anotación y el nombre del recurso, no
apiVersion: v1
kind: ServiceAccount
metadata:
  name: api
  namespace: tienda
  annotations:
    # <clave específica del proveedor>: <identidad de la nube>
```

Y los complementos gestionados merecen su nota porque cambian el reparto de responsabilidades:

```
complemento gestionado
  el proveedor lo instala y lo actualiza en su calendario
  menos trabajo, y su versión puede cambiar sin que nadie lo decida

complemento propio
  se elige la versión y el momento
  más trabajo, y es la única forma de fijar el comportamiento
```

La decisión importa sobre todo en el complemento de red, por lo que la clase 080 mostró: si no implementa políticas, los objetos no filtran nada. Y algunas plataformas ofrecen varios, con soporte distinto de políticas, así que la elección

del complemento al crear el clúster decide si la segmentación de red es posible. Es una decisión de instalación con consecuencias de seguridad, y hay que tomarla con la clase 080 delante.

5. Qué se conserva al cambiar de plataforma

La pregunta que este programa lleva haciendo desde la clase 048 tiene aquí una respuesta cuantificable, y es la mejor noticia de la parte.

```
se conserva SIN CAMBIOS
  todos los manifiestos de carga: despliegues, servicios, configuración,
  presupuestos, políticas de red, perfiles de seguridad, escalado
  las imágenes y su cadena de suministro (clases 061-067)
  el contrato de la aplicación: señales, comprobaciones, apagado (068)
  la instrumentación: el formato de métricas es el mismo (082)
  las herramientas de gestión de manifiestos (081)

hay que REHACER
  la creación del clúster y su plan de direcciones
  la sintaxis de la federación de identidad
  los grupos de nodos y su escalado
  la elección e instalación de complementos
  la integración con el almacenamiento: nombres de clases y parámetros
  la integración con el balanceador de entrada: anotaciones
  la recogida de registros y métricas hacia el sistema del proveedor
```

El primer bloque es grande y el segundo es la capa de plataforma, no las aplicaciones. Expresado como proporción, en una migración real entre dos proveedores:

```
ficheros de manifiesto de aplicaciones      ~95 % sin cambios
capa de plataforma                          ~100 % reescrita
esfuerzo total                              concentrado en 6-8 componentes
```

Y eso confirma con datos lo que la clase 072 predijo: el contrato del contenedor es portable y las fugas están en los bordes. Aquí los bordes tienen nombre: red, almacenamiento, identidad y entrada — las mismas cuatro, por tercera vez.

Y hay una conclusión práctica que conviene sacar y que casi nadie saca: si la capa de plataforma se va a reescribir de todos modos, conviene aislarla desde el principio.

```
repositorio de aplicaciones     manifiestos portables, sin nada del proveedor
repositorio de plataforma       creación del clúster, complementos,
                                clases de almacenamiento, anotaciones de entrada,
                                federación de identidad
```

Con esa separación, cambiar de proveedor toca un repositorio y no doscientos ficheros. Y tiene un beneficio inmediato aunque nunca se cambie: hace visible qué parte del sistema está atada al proveedor, que es una cifra que casi ninguna organización conoce.

Y un aviso honesto sobre la portabilidad, para no exagerarla: que los manifiestos sean portables no significa que el sistema lo sea. Los servicios gestionados que las cargas usan —bases de datos, colas, almacenamiento de objetos— siguen siendo del proveedor, y suelen ser la parte más difícil de mover. Kubernetes hace portable el cómputo, que es la parte que ya era más fácil.

La lista de comprobación para elegir y montar una plataforma gestionada:

- plan de direcciones calculado para el tamaño máximo previsto, con margen de despliegue, y comprobado el límite de pods por nodo
- complemento de red que implementa políticas, verificado con prueba negativa
- federación de identidad acotada a espacio y cuenta, con prueba negativa
- ventana de soporte anotada, con fecha de la próxima actualización obligatoria
- clúster de pruebas una versión por delante
- métrica de APIs obsoletas revisada antes de cada actualización
- presupuestos con margen, para que la actualización del proveedor no se bloquee
- separación entre repositorio de aplicaciones y de plataforma
- complementos: decidido cuáles son gestionados y cuáles propios, y por qué

Ejemplo trabajado

CloudShop opera un clúster en una nube y evalúa mover una parte a otra. El ejercicio se hace en serio —con una migración real de un entorno— y produce cinco datos que ninguna comparativa daba.

Dato 1 — el 94 % de los manifiestos no cambió.

```
manifiestos de aplicaciones                214          13
  de esos 13:
    clase de almacenamiento                  4
    anotaciones del objeto de entrada        5
    anotaciones de cuenta de servicio        3
    tolerancias de nodos especiales          1
capa de plataforma                          31          31
```

Los trece cambios son exactamente las cuatro fugas de la clase 072, y ninguno estaba en la lógica de la aplicación.

```
reescribir la capa de plataforma          9 días-persona
adaptar los 13 manifiestos                1 día-persona
validar y comparar                        4 días-persona
```

Y la conclusión que se documentó: el coste de cambiar de plataforma gestionada está en seis componentes de infraestructura, no en las aplicaciones.

Dato 2 — el plan de direcciones limitaba el clúster a 27 nodos.

Al dimensionar el clúster nuevo:

```
subred asignada                    /22 · 1.024 direcciones
modelo                             direcciones nativas
pods por nodo previstos            30
nodos que caben                    1.024 / 31 ≈ 33
menos margen de despliegue (~20 %)  ≈ 27
```

El clúster original tenía el mismo problema y no lo sabían: llevaban meses sin poder pasar de 24 nodos, y lo atribuían a cuotas del proveedor.

```
$ kubectl describe node nodo-14 | grep -i 'pods:'
  pods:  29        ← límite por tipo de máquina, no por CPU ni memoria
```

```
subred del clúster                          /22              /20
nodos posibles                               27              110
límite de pods por nodo                      29               110 (modo por prefijo)
sucesos "no IP addresses available"     41 en un mes           0
pods en Pending por límite de pods       sí, en los picos       no
```

Dos correcciones distintas: la subred resolvía el techo del clúster y el modo de asignación por prefijo resolvía el techo por nodo. Ninguna se puede aplicar sobre un clúster existente sin recrearlo, lo que confirma que es una decisión de un solo sentido.

Dato 3 — una actualización forzada que llevaba tres meses bloqueada.

```
versión del clúster            fin de soporte hace 6 semanas
intentos del proveedor         4, todos detenidos
nodos actualizados             11 de 24
motivo                         presupuesto sin margen en 2 servicios (clase 079)
aviso                          correo a una lista que nadie leía
```

El clúster llevaba seis semanas con la mitad de los nodos en una versión sin soporte, y con el plano de control ya actualizado por el proveedor.

```
presupuestos que bloquean                    2                0
notificación del proveedor             correo a lista    alerta en el canal
                                                          de guardia
clúster de pruebas por delante           no había          sí, una versión
métrica de APIs obsoletas revisada        nunca      antes de cada actualización
duración de la actualización completa   no terminaba      4 h 10 min
```

Y la revisión de APIs obsoletas, hecha por primera vez, encontró dos manifiestos que la versión siguiente habría rechazado.

Dato 4 — el complemento de red decidía si la segmentación era posible.

La plataforma de destino ofrecía dos complementos, y solo uno implementaba políticas. La elección por defecto del asistente de creación era el otro.

```
políticas de red efectivas                   0               14
direcciones consumidas por pod               1                1
unidad máxima de transmisión            sin cambios      revisada (clase 065)
complemento gestionado por el proveedor      sí               sí
```

Si la migración se hubiera hecho aceptando el valor por defecto, las catorce políticas de la clase 080 habrían vuelto a no filtrar nada — el mismo incidente, en una plataforma nueva, por una casilla del asistente.

Dato 5 — la federación de identidad, sin acotar por defecto.

La primera configuración funcionaba desde el primer intento, lo que hizo sospechar:

```
$ kubectl run prueba -n desarrollo --rm -it --image=… -- /probar-acceso-nube
ok        ← desde OTRO espacio, con OTRA cuenta
```

La condición de confianza del lado de la nube aceptaba cualquier testigo del clúster.

```
condición de confianza              emisor del clúster   emisor + espacio + cuenta
pods que podían obtener la identidad    todos            solo el previsto
prueba negativa                          ninguna       en el guion de verificación
```

Cuarta vez en el programa que esta misma prueba negativa se ejecuta y tercera vez que falla la primera vez.

Resumen de la evaluación:

```
manifiestos de aplicación reescritos         —              13 de 214
capa de plataforma reescrita                 —              31 de 31
nodos que caben en el plan de direcciones    27              110
políticas de red efectivas                   14              14
federación acotada                           sí            sí (tras corregir)
semanas con versión sin soporte              6               0
esfuerzo total de la migración               —           14 días-persona
```

La lección que esta clase traslada al proyecto de la clase 084: la comparación entre plataformas gestionadas no se decide por funciones sino por tres cosas que se fijan al crear el clúster y no se pueden cambiar después — el plan de direcciones, el complemento de red y el modelo de identidad. Las tres son decisiones de un solo sentido, las tres se toman en un asistente de creación en cinco minutos, y las tres deciden el techo de crecimiento, la posibilidad de segmentar y el alcance de una credencial. El resto —el 94 % de los manifiestos— es portable, y esa es la confirmación cuantificada de la hipótesis que abrió la parte 05.

Laboratorio guiado

Ejecuta desde la raíz:

```
python classes/part-06-kubernetes-managed-platforms/083-eks-aks-y-gke-similitudes-y-diferencias/lab.py
```

El laboratorio selecciona el motor de práctica decision y produce labresult.json. El escenario, sus comprobaciones y el artefacto esperado corresponden a esta clase; no requiere credenciales y deja explícito qué debe revalidarse en un sandbox real.

1. Lee exercise.steps y formula una predicción antes de ejecutar.
2. Ejecuta la práctica y verifica todos los elementos de checks.
3. Provoca el caso de negativetest y explica la señal observada.
4. Materializa matriz-kubernetes-administrado en evidence/ usando la plantilla indicada.
5. Para proveedor real, sigue sandbox.requires, registra costo y ejecuta destroy.

Evidencia esperada

El artefacto principal es una matriz de decisión ponderada y un ADR. Además, la entrega debe incluir el comando ejecutado, la salida estructurada y una conclusión que no exceda lo observado.

Reto verificable

Construye matriz-kubernetes-administrado para el caso CloudShop. Incluye una alternativa descartada, un supuesto que pueda falsarse, una prueba de fallo y una decisión de rollback.

■ Criterio de aceptación

- [ ] lab.py termina con código 0 y genera JSON válido.
- [ ] La entrega conecta al menos tres requisitos con mecanismos verificables.

- [ ] Existe una prueba positiva y una prueba negativa con evidencia.
- [ ] Seguridad, costo y operación aparecen como decisiones, no como anexos.
- [ ] Se declara una limitación y una condición que obligaría a revisar el diseño.
- [ ] Otra persona puede repetir el recorrido sin conocimiento tácito.

■■ Errores frecuentes

Síntoma	Causa probable	Corrección
El clúster no puede crecer más allá de cierto número de nodos	El plan de direcciones no da para más pods, y el pico de despliegue consume más que el estado estable	Calcula el tamaño máximo previsto con margen antes de crear el clúster; ampliarlo después suele exigir recrearlo.
Hay nodos con CPU y memoria libres y pods en Pending	El límite de pods por nodo depende del tipo de máquina y del modo de asignación de direcciones	Usa máquinas mayores o el modo de asignación por prefijo; el suceso menciona pods insuficientes, no recursos.
El proveedor intenta actualizar y el clúster queda a medias durante semanas	Un presupuesto de interrupción sin margen bloquea también la actualización automática	Mantén margen en los presupuestos y dirige los avisos del proveedor al canal de guardia, no a una lista.
Tras una actualización, algunos manifiestos dejan de aplicarse	La versión nueva retiró versiones de API que esos manifiestos usaban	Revisa la métrica de APIs obsoletas antes de cada actualización y mantén un clúster de pruebas por delante.
Las políticas de red no filtran en la plataforma nueva	El complemento elegido en el asistente de creación no las implementa	Elige el complemento con soporte de políticas y verifica con una prueba negativa antes de dar por buena la migración.
Cualquier pod del clúster obtiene la identidad de la nube	La condición de confianza acepta cualquier testigo del clúster, sin acotar espacio ni cuenta	Acota la confianza al espacio de nombres y a la cuenta concretos, y comprueba desde otro espacio que se deniega.
Se espera que la plataforma gestionada reduzca el trabajo de operación	Gestiona el plano de control, no las aplicaciones ni los complementos	Dimensiona el equipo para el trabajo de las clases 073 a 082, que es el que produce los incidentes.

■ Seguridad, ética y costo

Trabaja con cuentas propias o sandboxes autorizados. No publiques secretos, identificadores reales ni datos personales. Antes de crear recursos pagos define presupuesto, etiquetas y comando de destrucción; después verifica que no queden recursos huérfanos. Los ejemplos locales enseñan contratos, pero no certifican cumplimiento ni disponibilidad de producción.

■ Preguntas de comprobación

1. ¿Qué gestiona exactamente el proveedor y qué sigue siendo trabajo propio?
2. ¿Qué ocurre cuando se agota la ventana de soporte y qué puede impedir esa actualización?
3. Calcula cuántos nodos caben en una subred /22 con 30 pods por nodo y margen de despliegue.
4. ¿Por qué la elección del complemento de red al crear el clúster es una decisión de seguridad?
5. ¿Qué proporción de manifiestos se conserva al cambiar de plataforma, y dónde se concentra el esfuerzo?

Referencias

- Kubernetes (2025). Version skew policy — desviación admitida entre plano de control y nodos. <https://kubernetes.io/releases/version-skew-policy/>
- Kubernetes (2025). Deprecated API migration guide — versiones retiradas y cómo detectarlas. <https://kubernetes.io/docs/reference/using-api/deprecation-guide/>

- AWS (2025). Amazon EKS networking and IP address planning — direcciones por nodo y modos de asignación. <https://docs.aws.amazon.com/eks/latest/userguide/eks-networking.html>
- Microsoft (2025). AKS networking concepts — modelos de red y sus consecuencias de direccionamiento. <https://learn.microsoft.com/en-us/azure/aks/concepts-network>
- Google Cloud (2025). GKE cluster networking and IP allocation — rangos secundarios y límites por nodo. <https://cloud.google.com/kubernetes-engine/docs/concepts/alias-ips>
- Documentación oficial vigente del servicio implementado; registra URL y fecha de consulta.

Descargar: Parte 06 en PDF · Manual integral

Anterior	Índice	Siguiente
← 082 · Logs, métricas, eventos y depuración	Parte 06 · Programa	084 · Proyecto: plataforma Kubernetes portable →

Evaluación — 083 EKS, AKS y GKE: similitudes y diferencias

Preguntas

1. Explica el problema que resuelve esta capacidad sin mencionar un proveedor.
2. Identifica una frontera de responsabilidad y su propietario.
3. Compara dos alternativas mediante confiabilidad, seguridad, costo y operación.
4. ¿Qué demuestra el laboratorio y qué no puede demostrar?
5. Propón un caso límite o una prueba de fallo.

Reto verificable

Entrega matriz-kubernetes-administrado con diagrama o configuración, evidencia de ejecución, alternativa descartada, riesgo residual y criterio de rollback.

Criterio de aceptación

- [ ] Resultado reproducible por una segunda persona.
- [ ] Evidencia vinculada a cada afirmación técnica.
- [ ] Prueba negativa o de fallo incluida.
- [ ] Costos y permisos expresados con unidades y alcance.
- [ ] Limitaciones declaradas sin presentar la simulación como producción.

Escala

Nivel	Evidencia
A — competente	Cumple todo y defiende trade-offs con datos.
B — en desarrollo	Cumple lo esencial; falta profundidad en una dimensión.
C — inicial	Artefacto parcial o conclusión sin evidencia suficiente.
0	Sin entrega reproducible.

Parte 07 — Infraestructura como código y configuración

093 — CloudFormation, Bicep, Pulumi y Terraform

Parte: 07 — Infraestructura como código y configuración Nivel: intermedio-avanzado · Horas estimadas: 4 Laboratorio: decision · Estado: EXECUTABLECORE

Propósito

Comparar las cuatro familias de herramientas por lo que cambia la operación, que son cuatro ejes y ninguno es la sintaxis: quién guarda el estado, quién ejecuta el motor, qué ocurre cuando una aplicación falla a medias y si el lenguaje es un lenguaje de programación. Las clases 047 y 059 ya midieron el primero; aquí se completan los otros tres y se llega a la conclusión que este programa lleva sugiriendo desde la parte 03: la disciplina es la misma en las cuatro, y es lo que decide si funciona.

Resultados de aprendizaje

Al finalizar podrás:

1. Comparar las cuatro familias por los ejes que cambian la operación, no por funciones.
2. Anticipar qué ocurre cuando una aplicación falla a medias en cada una.
3. Valorar el uso de un lenguaje de programación general con sus dos caras.
4. Elegir herramienta con un criterio defendible y sabiendo qué se hereda con ella.
5. Reconocer la parte de la práctica que no depende de la herramienta.

Conceptos centrales

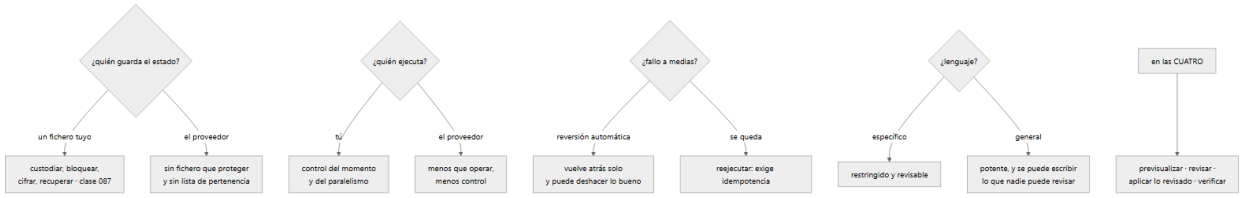
Concepto	Comprensión verificable
estado externo frente a estado del proveedor	O hay un fichero que hay que custodiar, o el propio servicio recuerda lo que gestiona. Decide la mitad de las obligaciones operativas.
motor gestionado	El proveedor ejecuta el despliegue. No hay estado que proteger ni ejecución que orquestar, y se pierde control sobre el momento y sobre el detalle.
comportamiento ante fallo parcial	Qué queda cuando la mitad se aplicó y algo falló: reversión automática, o lo aplicado se queda. Cambia por completo el procedimiento de recuperación.
lenguaje general	Escribir infraestructura en un lenguaje de programación. Da bucles, abstracciones y pruebas conocidas, y quita la restricción que hacía revisable el resultado.
retraso del proveedor	Tiempo entre que una nube publica una función y la herramienta la soporta. Es cero en las herramientas de la propia nube y variable en las demás.
disciplina portable	Previsualizar, revisar, aplicar lo revisado, verificar convergencia, política sobre el resultado y secretos fuera. Es idéntica en las cuatro familias.

Modelo mental

IaC trata la infraestructura como producto versionado: el plan explica intención, el estado conecta intención y realidad, y la revisión reduce cambios accidentales.

Aplicado a esta clase, separa siempre cuatro planos: intención (qué necesita el usuario), configuración (qué declaramos), estado observado (qué existe de verdad) y evidencia (cómo sabemos que cumple). Confundirlos produce diseños que se ven correctos en un diagrama pero fallan al operar.

■ Flujo de razonamiento



Desarrollo

1. Cuatro ejes, y la sintaxis no es ninguno

Las comparativas habituales enumeran funciones. Los cuatro ejes que de verdad cambian el trabajo diario son otros:

	Estado	Motor	Fallo parcial	Lenguaje
Nativa de la nube (plantillas del proveedor)	Del proveedor	Gestionado	Revierte solo	Específico
Terraform	Fichero propio	Tuyo	Se queda	Específico
Lenguaje general con estado	Fichero propio	Tuyo	Se queda	General
Kit de desarrollo que genera plantillas	Del proveedor	Gestionado	Revierte solo	General

Y las consecuencias de cada columna, que son lo que se hereda al elegir.

El estado ya se midió en las clases 047 y 059, y el resumen se sostiene:

```
fichero propio      hay que custodiarlo, bloquearlo, cifrarlo y recuperarlo
                    → toda la clase 087
                    a cambio: se sabe QUÉ gestiona este código
                    y se detecta que alguien BORRÓ algo

del proveedor       no hay fichero que proteger ni secretos que se filtren por ahí
                    a cambio: no hay lista de pertenencia clara,
                    y detectar un borrado externo depende del servicio
```

El motor es el eje que menos se discute y más cambia la operación:

```
motor gestionado    el proveedor despliega, con su ritmo y sus reintentos
                    no hay agentes que operar, ni bloqueos, ni concurrencia
                    y hay poco control: el paralelismo, el orden fino
                    y los tiempos de espera los decide él

motor propio        control total del momento, del paralelismo y del alcance
                    y hay que operar el agente, su identidad y su bloqueo
```

La segunda fila incluye una ventaja que se nota al depurar: con motor propio se puede aplicar una parte, planificar sin refrescar o reintentar una operación concreta (clase 090). Con motor gestionado, esas herramientas no existen.

El retraso del proveedor es un argumento real y suele exagerarse:

```
herramienta de la propia nube   soporte el mismo día del anuncio
otras                           de días a meses, según el servicio
```

Y su importancia depende de un dato que cada equipo puede medir: cuántas veces al año necesita una función publicada esta semana. En la mayoría de las organizaciones la respuesta es pocas, y para esos casos existe la salida de declarar el recurso con la API genérica del proveedor hasta que llegue el soporte.

2. Qué pasa cuando falla a medias

Este es el eje que menos aparece en las comparativas y el que más cambia el procedimiento de recuperación.

Reversión automática. El motor deshace lo que ya había aplicado y devuelve el conjunto a su estado anterior.

```
ventaja   nunca queda un estado intermedio que nadie ha probado
coste     la reversión también deshace lo que SÍ funcionó
          y puede fallar ella misma, dejando un estado del que hay que salir a mano
          y en un despliegue grande, tarda
```

La segunda línea produce una situación conocida: el motor intenta revertir un recurso que no se puede eliminar —tiene protección, tiene dependencias, tiene datos— y el conjunto queda en un estado que exige intervención. Y ahí el procedimiento no es evidente y conviene tenerlo escrito antes.

Lo aplicado se queda. Es el comportamiento de Terraform, y la clase 059 ya lo enunció: no hay transacción.

```
ventaja   lo que funcionó, funcionó; se corrige y se vuelve a aplicar
coste     hay un estado intermedio real, y hay que asumir que existirá
          → EXIGE que las plantillas sean idempotentes y reejecutables
```

Y de ahí sale la propiedad que este programa lleva pidiendo desde la clase 047: la respuesta a un fallo parcial es corregir y volver a aplicar. Una plantilla que solo funciona sobre un entorno vacío no sirve para operar, con ninguna herramienta.

La comparación honesta entre ambos comportamientos:

```
la reversión automática es mejor para        despliegues acotados y frecuentes
                                             de un conjunto pequeño
lo aplicado se queda es mejor para           conjuntos grandes, donde revertir
                                             todo por un fallo al final es peor
                                             que corregir y continuar
```

Y una coincidencia entre las dos familias que conviene señalar: ninguna revierte los efectos que no son recursos. Una migración de datos ejecutada por un gancho, un mensaje publicado, un fichero escrito — nada de eso vuelve atrás. Es el mismo límite que la clase 079 estableció para las vueltas atrás de despliegue, y sigue exigiendo lo mismo: cambios compatibles en ambos sentidos y una pregunta previa —«si esto falla a mitad, ¿qué queda roto?»—.

Y una diferencia operativa que aparece con motores gestionados y sorprende: la detección de desviación es una función del propio servicio en algunas plataformas, con lo que se obtiene sin montar la ejecución programada de la clase 090. Y suele tener límites: no todos los tipos de recurso, no todos los campos. Conviene comprobar la cobertura real antes de darla por buena, con la misma desconfianza que la clase 080 aplicó a las políticas de red.

3. El lenguaje general: dos caras

Escribir infraestructura en un lenguaje de programación completo resuelve problemas reales y quita una restricción que era útil. Merece las dos columnas.

Lo que resuelve:

```
abstracciones de verdad     clases, funciones, herencia
                            un módulo con lógica compleja se expresa mejor
bucles y condicionales      sin las contorsiones de un lenguaje declarativo
pruebas con las herramientas del lenguaje
                            unitarias, sin desplegar nada
reutilización de código     bibliotecas ya existentes, tipos compartidos
                            con la aplicación
un solo lenguaje            el equipo no aprende una sintaxis más
```

La tercera fila es la más valiosa y la menos citada: se pueden escribir pruebas unitarias sobre la lógica que decide la infraestructura, sin crear nada.

Lo que quita:

```
la restricción que hacía revisable el resultado
```

Un lenguaje declarativo limitado obliga a que el código se parezca al resultado. Con un lenguaje general se puede escribir esto:

```
una función que lee una base de datos y decide cuántos recursos crear
un bucle cuyo número de iteraciones depende de la hora
una jerarquía de clases de cinco niveles para tres tipos de servicio
```

Y nada de eso es revisable leyendo el código: hay que ejecutarlo para saber qué hace.

Y aquí está la parte que reconcilia las dos columnas y que conviene tener clara antes de elegir:

> El artefacto de revisión no es el código, es el plan. Con un lenguaje general, el plan sigue existiendo y sigue siendo el sitio donde se revisa, se aplica la política y se aprueban los borrados.

Eso reduce mucho el riesgo, y no lo elimina: un plan de doscientos recursos generado por un bucle cuyo criterio nadie entiende es revisable en su resultado y no en su intención. Por eso la disciplina que hay que añadir es de estilo:

```
la lógica que decide QUÉ existe: simple, plana, y probada
la complejidad, en las bibliotecas de apoyo, no en la definición
ninguna consulta a sistemas externos en tiempo de definición
  → hace que el resultado dependa de cuándo se ejecute
```

La tercera es la más importante: una definición que consulta una base de datos o una API para decidir qué crear no es reproducible, y rompe la propiedad que hace útil todo lo demás.

Y un matiz sobre los kits de desarrollo que generan plantillas nativas: dan el lenguaje general y conservan el motor gestionado, así que heredan la reversión automática y la ausencia de estado propio. Es una combinación coherente y con un coste conocido: el resultado es una plantilla generada, a veces grande y poco legible, y depurar exige leerla.

4. Elegir, y qué se hereda con cada elección

Un criterio defendible en cinco preguntas, en orden:

```
1. ¿una nube o varias?
   una sola y sin previsión de cambiar  → la nativa es una opción seria
   varias, o Kubernetes, o servicios de terceros → una herramienta común

2. ¿quién va a operarlo?
   equipo pequeño sin capacidad de operar agentes → motor gestionado
   equipo de plataforma con canalizaciones → motor propio

3. ¿el equipo ya tiene un lenguaje?
   sí, y la infraestructura la escriben desarrolladores → lenguaje general
   la escribe un equipo de plataforma → específico, más restringido y más legible

4. ¿cuánta lógica hace falta de verdad?
   poca, casi siempre → un lenguaje declarativo basta y se lee mejor

5. ¿qué se hereda?
   → la tabla del primer apartado, entera
```

Y la respuesta más común en organizaciones medianas, dicha sin adornos: una herramienta común con motor propio y lenguaje específico, porque cubre varias nubes, Kubernetes y servicios de terceros con un solo flujo, y porque la lógica que hace falta es poca.

Y dos situaciones en las que la nativa gana con claridad:

```
una sola nube, equipo pequeño, sin canalización de plataforma
  → el motor gestionado quita la mitad del trabajo de las clases 087 y 090

ofrecer plantillas a clientes o a otras organizaciones
  → el formato nativo lo entiende cualquiera de esa nube sin instalar nada
```

Y la situación que aparece de verdad y que ninguna comparativa contempla: una organización con tres herramientas a la vez, porque cada equipo eligió la suya. Consolidar cuesta y no consolidar también, y la decisión se toma con dos cifras:

```
coste de consolidar    reescribir e importar, medible en días-persona
coste de no hacerlo    tres canalizaciones, tres políticas, tres auditorías,
                       tres formaciones y ninguna revisión cruzada posible
```

Y una salida intermedia que funciona mejor de lo que parece: consolidar la disciplina antes que la herramienta. Las mismas políticas, la misma secuencia de revisión, el mismo tratamiento de secretos y la misma detección de desviación, con tres herramientas distintas. Eso captura la mayor parte del beneficio y no exige reescribir nada.

Y para migrar de una herramienta a otra, cuando se decide, el procedimiento tiene una forma conocida:

```
1. la herramienta nueva IMPORTA lo que existe (clase 087)
2. se verifica con una previsualización sin cambios
3. la herramienta antigua deja de gestionarlo, sin destruirlo (clase 090)
4. se retira la antigua cuando ya no gestione nada
```

El paso 3 es el que evita el desastre: retirar de la gestión, no destruir. Y el 2 es la comprobación que decide si el paso 1 se hizo bien, exactamente como en la clase 087.

5. Lo que no depende de la herramienta

Con cuatro familias comparadas, lo que queda es la conclusión que este programa lleva sugiriendo desde la parte 03: la mayor parte de la práctica es idéntica.

```
previsualizar antes de aplicar             047 · 059 · 081 · 090
aplicar exactamente lo previsualizado      059 · 090
verificar la convergencia después          073 · 085 · 090
revisión legible: sin ruido                047 · 081 · 085 · 090
política sobre el RESULTADO, no el código  091
secretos fuera, y rotar lo expuesto        047 · 059 · 061 · 087 · 092
versiones fijadas de todo                  047 · 059 · 062 · 081 · 088
un estado o alcance por radio de impacto   047 · 059 · 087
proteger lo que no se puede perder         042 · 059 · 077 · 086
identidad federada, sin claves             026 · 038 · 050 · 059 · 092
refactorizar sin destruir                  087 · 090
detección de desviación con señal          085 · 090
```

Doce prácticas, todas presentes en las cuatro familias con nombres distintos, y todas responsables de más incidentes que cualquier decisión de herramienta.

Y los tres criterios que la clase 085 fijó para juzgar una solución de esta parte siguen siendo los mismos y siguen sin depender de la herramienta:

```
1. ¿se puede ver el cambio antes de aplicarlo, y es legible?
2. ¿lo que se revisó es exactamente lo que se aplica?
3. ¿alguien sabría que el sistema dejó de converger?
```

Y una observación que conviene decir con claridad, porque ahorra discusiones improductivas:

> Un equipo con una herramienta mediocre y las doce prácticas tiene una infraestructura operable. Un equipo con la mejor herramienta y sin ellas tiene los mismos incidentes que tenía antes, con más ficheros.

Y la lista de comprobación de la clase, que es de decisión y no de configuración:

- los cuatro ejes evaluados para el caso concreto, no una tabla de funciones
- el comportamiento ante fallo parcial, conocido y con procedimiento escrito
- si el lenguaje es general: reglas de estilo sobre la lógica de definición
- ninguna consulta a sistemas externos en tiempo de definición
- si hay varias herramientas: la disciplina consolidada aunque no lo esté la herramienta
- si se migra: importar, verificar sin cambios, retirar de la gestión, destruir nunca
- las doce prácticas portables, presentes con independencia de la elección

Ejemplo trabajado

CloudShop tiene tres herramientas de infraestructura como código y una discusión anual sobre cuál usar. El ejercicio de este año se hace midiendo en vez de opinando, y la conclusión no es la que nadie esperaba.

El punto de partida.

```
equipo de plataforma      Terraform, 4 estados, 88 % de la infraestructura
equipo de datos           plantillas nativas de la nube, 9 %
equipo de aplicación      un kit de desarrollo en el lenguaje de la aplicación, 3 %
```

Y los tres argumentos de la discusión anual, siempre los mismos:

```
"las plantillas nativas no necesitan estado"
"con el kit escribimos infraestructura en el mismo lenguaje"
"tener tres herramientas es insostenible"
```

La medición: incidentes por causa, últimos doce meses.

```
causa                                  incidentes   herramienta
falta de previsualización revisada           6      las tres
plantilla no idempotente                     4      las tres
secreto en un rastro no auditado             3      las tres
desviación no detectada                      5      las tres
recurso destruido sin querer                 2      Terraform, kit
estado perdido o bloqueado                   2      Terraform
```

```
reversión automática que falló a medias             1      nativa
lógica de definición imposible de revisar           1      kit
```

Veinticuatro incidentes. Dieciocho de los veinticuatro son de las doce prácticas portables y no de la herramienta: ocurrieron con las tres, y habrían ocurrido con cualquier otra.

Los seis restantes sí son atribuibles:

```
dos por destrucción         falta de protección contra el borrado (clases 059, 086)
dos por el estado           custodia y bloqueo (clase 087)
uno por reversión           el motor gestionado no pudo revertir y quedó a medias
uno por lógica              un bucle en el kit cuyo criterio nadie entendía
```

La decisión, que no fue consolidar.

```
consolidar en una herramienta        34 días-persona
consolidar la DISCIPLINA               9 días-persona
```

Y lo que cubría cada opción:

```
consolidar la herramienta   los 6 incidentes atribuibles, y de rebote
                            facilitaría las prácticas
consolidar la disciplina    los 18 incidentes portables, en las tres herramientas
```

Se hizo la segunda primero. Y las nueve jornadas se repartieron así:

```
política común sobre el resultado, para las tres            3 días
detección de desviación programada, para las tres           2 días
auditoría de los cinco rastros de secretos (clase 092)      2 días
protección contra destrucción en lo que no se puede perder  1 día
procedimiento escrito de fallo parcial, uno por herramienta 1 día
```

La política común fue lo más interesante del ejercicio: las tres herramientas producen una representación estructurada de lo que van a hacer, así que las mismas diecinueve reglas se pudieron aplicar a las tres con adaptadores pequeños.

```
reglas de política                    19, solo en Terraform   19, en las tres
detección de desviación                1 de 3 herramientas    3 de 3
audita rastros de secretos             1 de 3                 3 de 3
procedimiento de fallo parcial escrito 0 de 3                 3 de 3
recursos con protección de borrado     6 de 41               41 de 41
```

Los seis meses siguientes.

```
incidentes de causa portable          18 → 2
incidentes atribuibles a la herramienta 6 → 3
```

Los dos portables restantes fueron plantillas no idempotentes en el kit, corregidas.

Y con esos datos, la decisión sobre consolidar la herramienta se tomó por fin con criterio:

```
las plantillas nativas se quedan       el equipo de datos es pequeño,
                                       no opera canalizaciones, una sola nube
                                       y el motor gestionado le quita trabajo real
el kit se retira                       3 % de la infraestructura, un incidente
                                       por lógica irrevisable, y el equipo
                                       prefería no mantener dos flujos
Terraform sigue                        cubre varias nubes y Kubernetes
```

La migración del kit se hizo con el procedimiento de esta clase:

```
recursos importados                    31
verificados con previsualización vacía 31 de 31, a la tercera iteración
retirados de la gestión del kit        31, sin destruir
recursos destruidos en el proceso       0
esfuerzo                               4 días-persona
```

La conclusión que se escribió, y que cerró la discusión anual.

```
de 24 incidentes en un año, 18 no dependían de la herramienta
consolidar la disciplina costó 9 días y eliminó 16 de esos 18
consolidar la herramienta habría costado 34 días y eliminado 6
y la decisión final NO fue una sola herramienta:
  dos, cada una donde su modelo operativo encaja
```

La lección que esta clase traslada al resto de la parte 07: la discusión sobre herramientas consume tiempo desproporcionado respecto de lo que decide. Tres de cada cuatro incidentes eran de las doce prácticas portables, presentes o ausentes con independencia de la elección. Y cuando por fin se eligió, el criterio no fue cuál es mejor sino

qué modelo operativo encaja con cada equipo: motor gestionado donde no hay capacidad de operar canalizaciones, motor propio donde hay varias nubes que cubrir.

Laboratorio guiado

Ejecuta desde la raíz:

```
python classes/part-07-infrastructure-as-code-configuration/093-cloudformation-bicep-pulumi-y-terraform/lab.py
```

El laboratorio selecciona el motor de práctica decision y produce labresult.json. El escenario, sus comprobaciones y el artefacto esperado corresponden a esta clase; no requiere credenciales y deja explícito qué debe revalidarse en un sandbox real.

1. Lee exercise.steps y formula una predicción antes de ejecutar.
2. Ejecuta la práctica y verifica todos los elementos de checks.
3. Provoca el caso de negativetest y explica la señal observada.
4. Materializa adr-herramienta-iac en evidence/ usando la plantilla indicada.
5. Para proveedor real, sigue sandbox.requires, registra costo y ejecuta destroy.

Evidencia esperada

El artefacto principal es una matriz de decisión ponderada y un ADR. Además, la entrega debe incluir el comando ejecutado, la salida estructurada y una conclusión que no exceda lo observado.

Reto verificable

Construye adr-herramienta-iac para el caso CloudShop. Incluye una alternativa descartada, un supuesto que pueda falsarse, una prueba de fallo y una decisión de rollback.

■ Criterio de aceptación

- [ ] lab.py termina con código 0 y genera JSON válido.
- [ ] La entrega conecta al menos tres requisitos con mecanismos verificables.
- [ ] Existe una prueba positiva y una prueba negativa con evidencia.
- [ ] Seguridad, costo y operación aparecen como decisiones, no como anexos.
- [ ] Se declara una limitación y una condición que obligaría a revisar el diseño.
- [ ] Otra persona puede repetir el recorrido sin conocimiento tácito.

Errores frecuentes

Síntoma	Causa probable	Corrección
La discusión sobre qué herramienta usar se repite cada año sin resolverse	Se compara por funciones en vez de por los ejes que cambian la operación	Mide los incidentes del último año por causa: la mayoría suele ser de prácticas portables, no de la herramienta.
Una aplicación falla a medias y el motor no consigue revertir	La reversión automática intenta eliminar algo protegido o con dependencias	Ten escrito el procedimiento de salida antes de necesitarlo; el comportamiento ante fallo parcial se conoce, no se descubre.
Un cambio en el kit genera doscientos recursos y nadie sabe por qué	La lógica de definición usa bucles o consultas cuyo criterio no se puede leer	Reglas de estilo: lógica de definición simple y plana; la complejidad, en bibliotecas de apoyo probadas.
El resultado de una ejecución depende de cuándo se ejecute	La definición consulta un sistema externo para decidir qué crear	Prohíbe las consultas en tiempo de definición: rompen la reproducibilidad, que es la propiedad que sostiene todo lo demás.
Migrar entre herramientas destruye recursos	Se retiró la antigua antes de que la nueva los gestionara, o se destruyó en vez de retirar de la gestión	Importar, verificar con previsualización vacía, retirar de la gestión sin destruir, y solo entonces retirar la antigua.
Se adopta la mejor herramienta y los incidentes siguen igual	Las doce prácticas portables no estaban antes y siguen sin estar	Consolida la disciplina antes que la herramienta: cubre más incidentes por menos esfuerzo.

Seguridad, ética y costo

Trabaja con cuentas propias o sandboxes autorizados. No publiques secretos, identificadores reales ni datos personales. Antes de crear recursos pagos define presupuesto, etiquetas y comando de destrucción; después verifica que no queden recursos huérfanos. Los ejemplos locales enseñan contratos, pero no certifican cumplimiento ni disponibilidad de producción.

Preguntas de comprobación

1. ¿Cuáles son los cuatro ejes que cambian la operación, y qué se hereda con cada opción?
2. ¿Qué ventaja y qué coste tiene la reversión automática frente a que lo aplicado se quede?
3. ¿Qué resuelve y qué quita usar un lenguaje de programación general, y qué reduce el riesgo?
4. ¿Qué procedimiento sigue una migración entre herramientas sin destruir nada?
5. Enumera cinco de las doce prácticas que no dependen de la herramienta.

Referencias

- AWS (2025). CloudFormation stack failure options and rollback — comportamiento ante fallo parcial. <https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/stack-failure-options.html>
- Microsoft (2025). Bicep vs ARM templates and deployment behaviour — motor gestionado y modos de despliegue. <https://learn.microsoft.com/en-us/azure/azure-resource-manager/bicep/overview>
- HashiCorp (2025). Terraform: no partial rollback — por qué las plantillas deben ser reejecutables. <https://developer.hashicorp.com/terraform/language/resources/behavior>
- Pulumi (2025). Programming model and preview — lenguajes generales con previsualización del cambio. <https://www.pulumi.com/docs/concepts/>
- AWS (2025). CDK: synthesized templates — generar plantillas nativas desde un lenguaje general. <https://docs.aws.amazon.com/cdk/v2/guide/home.html>
- Documentación oficial vigente del servicio implementado; registra URL y fecha de consulta.

Descargar: Parte 07 en PDF · Manual integral

Evaluación — 093 CloudFormation, Bicep, Pulumi y Terraform

Preguntas

1. Explica el problema que resuelve esta capacidad sin mencionar un proveedor.
2. Identifica una frontera de responsabilidad y su propietario.
3. Compara dos alternativas mediante confiabilidad, seguridad, costo y operación.
4. ¿Qué demuestra el laboratorio y qué no puede demostrar?
5. Propón un caso límite o una prueba de fallo.

Reto verificable

Entrega adr-herramienta-iac con diagrama o configuración, evidencia de ejecución, alternativa descartada, riesgo residual y criterio de rollback.

Criterio de aceptación

- [ ] Resultado reproducible por una segunda persona.
- [ ] Evidencia vinculada a cada afirmación técnica.
- [ ] Prueba negativa o de fallo incluida.
- [ ] Costos y permisos expresados con unidades y alcance.
- [ ] Limitaciones declaradas sin presentar la simulación como producción.

Escala

Nivel	Evidencia
A — competente	Cumple todo y defiende trade-offs con datos.
B — en desarrollo	Cumple lo esencial; falta profundidad en una dimensión.
C — inicial	Artefacto parcial o conclusión sin evidencia suficiente.
0	Sin entrega reproducible.

Parte 18 — Azure: arquitectura empresarial y operación en producción

217 — Enterprise-scale landing zones y management groups

Parte: 18 — Azure: arquitectura empresarial y operación en producción Nivel: avanzado · Horas estimadas: 4
Laboratorio: governance · Estado: EXECUTABLECORE

Propósito

Diseñar la jerarquía de Azure —grupos de administración, suscripciones y grupos de recursos— que es a esta nube lo que el plan de direcciones a la red: se decide en una tarde y condiciona una década. La clase da el criterio de reparto, explica dónde se aplican los controles para que no se puedan quitar desde abajo, y desarrolla la parte que más cuesta corregir después: mover una suscripción de sitio no mueve lo que ya se creó con la política antigua.

Resultados de aprendizaje

Al finalizar podrás:

1. Repartir cargas entre grupos de administración y suscripciones con criterio.
2. Aplicar controles en el nivel correcto, sin que se puedan quitar desde abajo.
3. Distinguir política, iniciativa y bloqueo, y saber qué resuelve cada uno.
4. Desplegar políticas en modo auditoría antes de denegar.
5. Corregir una jerarquía mal repartida sin romper lo existente.

Conceptos centrales

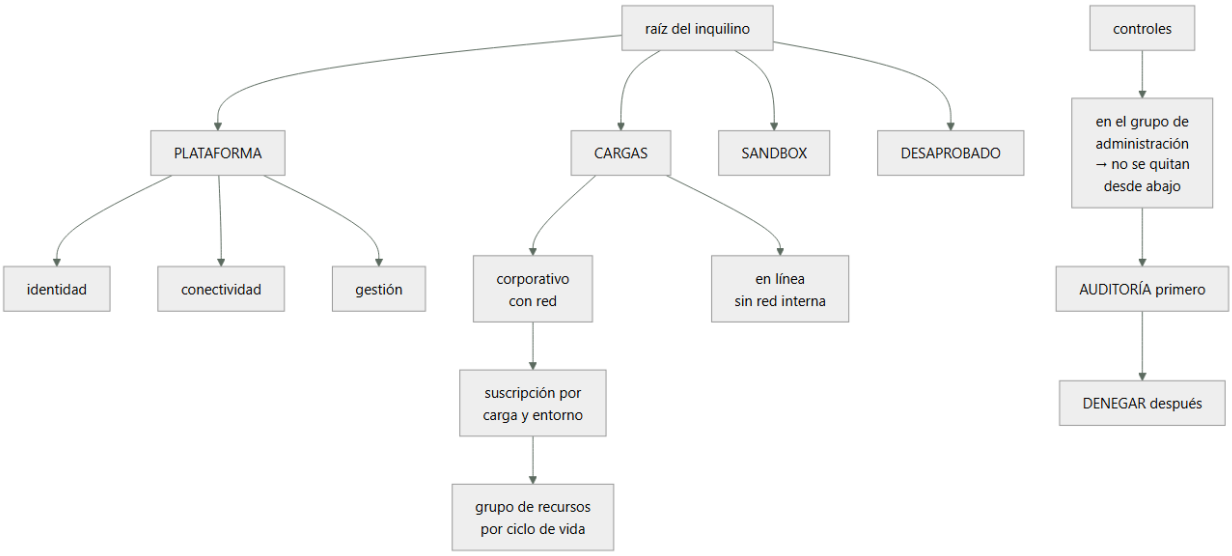
Concepto	Comprensión verificable
grupo de administración	Contenedor de suscripciones donde se aplican políticas y permisos heredados. Es el nivel que fija el gobierno.
suscripción	Unidad de facturación, de cuota y de aislamiento. Equivale funcionalmente a una cuenta.
grupo de recursos	Agrupación dentro de una suscripción, con ciclo de vida común. Se borra entero.
política	Regla que audita, deniega, modifica o despliega. Se hereda hacia abajo y no se puede quitar desde abajo.
iniciativa	Conjunto de políticas asignado como una unidad, con parámetros. Es la forma práctica de gobernar.
zona de aterrizaje	Suscripción preparada con red, identidad, controles y observabilidad, lista para recibir cargas.

Modelo mental

La arquitectura Azure nace del tenant y las políticas heredables, y llega al workload mediante identidades administradas y redes privadas.

Aplicado a esta clase, separa siempre cuatro planos: intención (qué necesita el usuario), configuración (qué declaramos), estado observado (qué existe de verdad) y evidencia (cómo sabemos que cumple). Confundirlos produce diseños que se ven correctos en un diagrama pero fallan al operar.

■ Flujo de razonamiento



Desarrollo

1. La jerarquía, y por qué se decide pronto

Azure tiene tres niveles y cada uno resuelve algo distinto. Confundirlos es el error inicial.

```
GRUPO DE ADMINISTRACIÓN
  contenedor de suscripciones
  aquí van las POLÍTICAS y las asignaciones de permisos que
  se heredan
  → es el nivel del gobierno

SUSCRIPCIÓN
  unidad de facturación, de CUOTA y de aislamiento
  → los límites de recursos son por suscripción, y ese es
    el motivo práctico más frecuente para separar

GRUPO DE RECURSOS
  agrupación con ciclo de vida común
  → se borra entero; los recursos de un grupo deberían
    nacer y morir juntos
  → no es un mecanismo de aislamiento de seguridad fuerte
```

Y el reparto que funciona, que conviene copiar salvo buena razón:

```
raíz
■■■ PLATAFORMA
■    ■■■ identidad          directorio, servidores heredados
■    ■■■ conectividad       red central, cortafuegos, DNS
■    ■■■ gestión            registros, copias, herramientas
■■■ CARGAS
■    ■■■ corporativo        necesita red interna
■    ■■■ en línea           público, sin red interna
■■■ SANDBOX                 experimentación, con caducidad
■■■ DESAPROBADO             lo que se retira, con controles
                           más duros y sin crecimiento
```

Y las tres decisiones que hay que tomar y que cuestan corregir:

```
1  ¿UNA SUSCRIPCIÓN POR ENTORNO O POR CARGA?
   por entorno y por carga: producción de pedidos, una;
   preproducción de pedidos, otra
   → aísla cuota, facturación y permisos a la vez
   → y es lo que permite dar acceso de administrador a un
     equipo sobre SU entorno sin exponer los demás

2  ¿DÓNDE VAN LOS CONTROLES?
   en el grupo de administración, nunca en la suscripción
   → una política asignada en la suscripción la puede
     quitar quien administre esa suscripción
   → una asignada arriba, no                    clase 169
```

```
3  ¿QUÉ SE HEREDA Y QUÉ SE PARAMETRIZA?
   la misma iniciativa arriba, con parámetros distintos por
   rama
   → «regiones permitidas» puede ser más estricto en
     corporativo que en sandbox
```

Y la advertencia que da nombre a la clase:

```
MOVER UNA SUSCRIPCIÓN DE SITIO NO ARREGLA LO YA CREADO
  las políticas de denegación solo actúan sobre
  CREACIONES y MODIFICACIONES
  → los recursos que ya existen siguen incumpliendo
  → aparecen como no conformes, y hay que corregirlos uno a
    uno o con tareas de remediación

→ y por eso la jerarquía se decide antes de crear nada
                                                    ley 14
```

2. Políticas: auditar antes de denegar

Las políticas son el mecanismo de gobierno de Azure y tienen cuatro efectos, con usos distintos.

```
AUDITAR      registra el incumplimiento, no impide nada
             → siempre el primer paso
DENEGAR      impide crear o modificar
             → el control real
MODIFICAR    corrige el recurso al crearlo (añade etiquetas,
             activa cifrado)
             → mejor que denegar cuando se puede arreglar
               solo                              clase 214
DESPLEGAR SI NO EXISTE
             crea lo que falta (diagnóstico, agente)
             → potente y peligroso: despliega cosas que
               nadie pidió y que facturan
```

Y el orden de implantación, que es el mismo de la clase 200:

```
1  ASIGNAR EN MODO AUDITORÍA, en el grupo de administración
2  MEDIR 2-4 semanas: cuántos recursos incumplen y cuáles
3  CLASIFICAR: incumplimientos reales frente a excepciones
   legítimas
4  CORREGIR o EXCEPTUAR, con dueño y fecha
5  CAMBIAR A DENEGAR
6  VIGILAR el cumplimiento como una señal más   clase 211

→ asignar en denegar el primer día bloquea despliegues
  legítimos y el gobierno pierde credibilidad     ley 16
```

Las iniciativas, que es como se gobierna de verdad:

```
una política suelta es difícil de mantener
una INICIATIVA agrupa decenas con parámetros
  → se asigna una vez por rama, con sus parámetros
  → y el cumplimiento se mide por iniciativa, no política a
    política

las iniciativas mínimas de una organización
  etiquetas obligatorias y sus valores          clase 214
  regiones permitidas
  tipos de recurso permitidos o prohibidos
  cifrado obligatorio y versión mínima de TLS
  sin acceso público en almacenamiento ni en bases
  diagnóstico enviado al destino central
  y las de cumplimiento normativo, si aplican
```

Y una advertencia sobre el modo de despliegue automático:

```
una política que despliega el agente de diagnóstico en
cada recurso nuevo
  → resuelve el problema de la observabilidad ausente
  → y crea coste que nadie pidió                clase 214
→ hay que estimar el coste de lo que la política despliega
  ANTES de asignarla
```

Los bloqueos, que son otra cosa y se confunden con las políticas:

```
BLOQUEO DE BORRADO      impide eliminar el recurso
```

```
BLOQUEO DE SOLO LECTURA impide modificarlo

  se aplican al recurso o al grupo, y los hereda lo de dentro
  → útiles para lo que nunca debe borrarse: bases de datos
    de producción, red central, copias
  → y son un incordio si se ponen de más: bloquean el
    despliegue declarativo            ← y entonces se quitan
                                        y nadie los repone
                                                  ley 25
```

3. Suscripciones: cuotas, facturación y aislamiento

La suscripción es el nivel donde se topan los límites, y ese es el motivo práctico que más veces obliga a separar.

```
LO QUE ES POR SUSCRIPCIÓN
  cuotas de cómputo por familia y por región
  límites de recursos por tipo
  la factura y su desglose
  y el ámbito natural de muchos permisos

CONSECUENCIAS PRÁCTICAS
  una carga que consume mucha cuota puede impedir crecer a
  otra de la misma suscripción
  → separar por carga aísla ese riesgo

  las cuotas NO son automáticas: hay que pedirlas con
  antelación
  → y en un incidente que requiera escalar mucho, la cuota
    es el límite real, no la capacidad de Azure
  → por eso se pide margen antes, y se vigila el consumo
    frente al límite                          clase 262
```

Las zonas de aterrizaje, que es la forma de que una suscripción nueva nazca lista:

```
QUÉ TRAE UNA SUSCRIPCIÓN ENTREGADA
  colocada en el grupo de administración correcto
  red conectada al centro, con sus rangos del plan
                                            clase 219
  identidad y permisos base asignados
  diagnóstico enrutado al destino central
  presupuesto y etiquetas obligatorias      clase 214
  y las políticas heredadas ya aplicando

Y EL PLAZO IMPORTA
  si entregar una suscripción tarda tres semanas, los
  equipos usarán la suya, o crearán recursos donde puedan
                                                  ley 16
  → automatizar la entrega es lo que hace que el gobierno
    se cumpla                                 clase 171
```

Y la decisión sobre quién administra qué:

```
EL EQUIPO DE PLATAFORMA administra
  grupos de administración, políticas, red central,
  identidad, destinos de diagnóstico

EL EQUIPO DE LA CARGA administra
  su suscripción: crea, despliega y opera dentro de las
  reglas heredadas

Y LO QUE NADIE DEBE PODER HACER
  quitar una política heredada
  cambiar el enrutamiento del diagnóstico
  crear emparejamientos de red por su cuenta
  → estas son las que van arriba, denegadas
```

4. Corregir una jerarquía mal repartida

Casi ninguna organización empieza con la jerarquía correcta. Corregirla se puede, y hay un orden.

```
LO QUE SE PUEDE MOVER
  una suscripción puede cambiar de grupo de administración
  un recurso puede cambiar de grupo de recursos (a veces)
  y en algunos casos, de suscripción
```

```
LO QUE NO SE MUEVE FÁCIL
  recursos con dependencias de red
  recursos con identidad administrada asignada
  claves y secretos con referencias
  y todo lo que tenga la suscripción escrita en su
  identificador en otro sitio
```

Y el procedimiento:

```
1  DEFINIR la jerarquía objetivo y las iniciativas
2  ASIGNARLAS EN AUDITORÍA en el nivel correcto
3  MEDIR el incumplimiento actual
   → esta es la fotografía que dice cuánto trabajo hay
4  MOVER las suscripciones al grupo que les corresponde
   → recordando que lo ya creado sigue incumpliendo
5  REMEDIAR lo existente, por lotes y por prioridad
   → primero lo que afecta a seguridad y a datos
6  PASAR A DENEGAR cuando el incumplimiento sea residual
7  DEJAR el cumplimiento como señal vigilada
```

Y lo que hay que medir mientras tanto:

```
recursos conformes frente a totales, por iniciativa
y su TENDENCIA
  → si no baja, la remediación no está ocurriendo
excepciones vivas y su antigüedad             clase 190
  → si crecen siempre, la política está mal calibrada
```

Y una advertencia sobre las exenciones:

```
una exención sin fecha es una política que no existe
                                                  ley 25
→ toda exención con dueño, motivo y caducidad
→ y un panel con las que vencen este mes
```

Y la lista de comprobación de la clase:

- hay grupos separados para plataforma, cargas, sandbox y desaprobado
- las suscripciones se separan por carga Y por entorno
- todas las políticas se asignan en grupos de administración, no en suscripciones
- las políticas se asignaron primero en auditoría
- se midió el incumplimiento antes de denegar
- las iniciativas están parametrizadas por rama
- está estimado el coste de lo que despliegan las políticas
- los bloqueos están solo donde hacen falta y no estorban al despliegue declarativo
- las cuotas están pedidas con margen y se vigilan
- entregar una suscripción nueva está automatizado y tarda minutos
- ningún equipo de carga puede quitar una política heredada
- toda exención tiene dueño, motivo y caducidad
- el cumplimiento por iniciativa se vigila como una señal

Y el cierre que enlaza con la clase siguiente: con la jerarquía puesta, el control que de verdad decide el alcance en Azure no es la política sino quién puede hacer qué y desde dónde. Identidad, identidades de carga, elevación temporal y acceso condicional es la materia de la clase 218.

Ejemplo trabajado

CloudShop lleva cuatro años en Azure con 61 suscripciones creadas sin criterio. Lo que sigue es la fotografía del incumplimiento, la jerarquía nueva, y el hallazgo de que mover las suscripciones no arregló casi nada.

El punto de partida:

```
suscripciones                                  61
  bajo la raíz, sin grupo de administración    47
  en un grupo llamado «Producción»              9
  en un grupo llamado «Test»                    5

políticas asignadas                            12
  en grupos de administración                   3
  EN SUSCRIPCIONES                              9  ←
  de las 9, quitadas por el equipo de la carga  4
```

```
y la consecuencia
  4 equipos habían quitado las políticas de su propia
  suscripción porque «bloqueaban el despliegue»
  → y nadie se enteró: no había alerta de cambio de
    asignación                                          ley 15
```

Y el detalle de por qué las quitaron:

```
la política denegaba crear cuentas de almacenamiento sin
cifrado con clave propia
la plantilla del equipo no lo declaraba
y el mensaje de error decía solo «denegado por política»
→ nadie sabía qué faltaba
→ y quitar la política era más rápido que averiguarlo
                                                        ley 16
```

La jerarquía nueva:

```
raíz
■■■ plataforma
■    ■■■ identidad          1 suscripción
■    ■■■ conectividad       2 (una por región)
■    ■■■ gestión            1
■■■ cargas
■    ■■■ corporativo        34 suscripciones
■    ■     pedidos-prod, pedidos-pre, pedidos-dev,
■    ■     catalogo-prod, … (una por carga y entorno)
■    ■■■ en línea            9
■■■ sandbox                12  ← con caducidad de 90 días
■■■ desaprobado             2

y el criterio escrito
  una suscripción por CARGA y por ENTORNO
  motivo   aísla cuota, factura y permisos a la vez
  coste de cambio si nos equivocamos   alto
```

Las iniciativas, con sus parámetros por rama:

```
iniciativa                    corporativo  en línea  sandbox
etiquetas obligatorias           denegar    denegar   auditar
regiones permitidas              2 (UE)     2 (UE)    2 (UE)
sin acceso público                denegar    denegar   denegar
cifrado en reposo                 denegar    denegar   auditar
TLS mínimo 1.2                    denegar    denegar   denegar
diagnóstico al destino central   desplegar  desplegar auditar
tipos de recurso permitidos      restringido  amplio   amplio
sin IP pública en máquinas        denegar    auditar   auditar
```

Y el coste que se estimó antes de asignar:

```
la política que despliega el diagnóstico crearía
  un ajuste de diagnóstico por recurso
  ingesta estimada a partir del inventario  ~1.900 €/mes
→ se ajustó qué categorías se envían y con qué caducidad
→ ingesta real                                640 €/mes
→ sin esta estimación previa, habría sido una sorpresa en
  la factura del mes siguiente                clase 214
```

La fase de auditoría, cuatro semanas.

```
recursos totales                              14.200

incumplimientos por iniciativa
  etiquetas obligatorias                       9.140  64 %
  cifrado en reposo                            1.870  13 %
  sin acceso público                             340   2,4 %
  TLS mínimo                                     620   4,4 %
  regiones permitidas                            112   0,8 %
  tipos de recurso                                88   0,6 %
  sin IP pública                                 210   1,5 %

y los que preocupaban de verdad
  340 recursos con acceso público
    de ellos, cuentas de almacenamiento          214
    de ellas, con datos de clientes               19  ←
  112 recursos fuera de las regiones permitidas
```

```
    de ellos, 8 con datos personales fuera de la UE  ←
```

Y la decisión sobre el orden de remediación:

```
no se atacó lo más numeroso (etiquetas, 9.140)
se atacó lo más grave: los 19 almacenes públicos con datos
y los 8 recursos fuera de región
→ resueltos en 9 días
→ las etiquetas, en 6 semanas con tareas de remediación
```

El hallazgo: mover las suscripciones no arregló casi nada.

```
al mover las 61 suscripciones a su grupo correcto
  políticas heredadas aplicando                    sí
  recursos existentes corregidos                    0

  porque las políticas de denegación solo actúan sobre
  creaciones y modificaciones
  → los 14.200 recursos existentes siguieron exactamente
    igual
  → aparecieron como no conformes, y nada más

lo que sí corrigió recursos existentes
  las políticas de tipo MODIFICAR con tareas de remediación
  → añadieron etiquetas a 8.100 recursos en 3 días
  → y el resto hubo que tocarlo a mano o redesplegarlo

lo que NO se pudo remediar automáticamente
  cifrado en reposo de recursos ya creados: 1.870
  → exige recrear el recurso en la mayoría de los casos
  → 1.410 se recrearon en 4 meses
  →   460 quedaron con exención, con dueño y fecha
```

El paso a denegar, y la lección de la comunicación.

```
primer intento
  se cambiaron 7 iniciativas a denegar el mismo día
  a las 4 horas, 23 despliegues bloqueados
  el mensaje decía «denegado por política»
  → 11 tiquetes de soporte en una tarde

corrección
  se volvió a auditoría
  se añadió a cada política un mensaje con
    qué falta exactamente
    el enlace a la plantilla que ya lo cumple
    y a quién pedir una exención, con el plazo
  y la plantilla de servicio nuevo se actualizó para
    cumplirlas todas                            clase 171

segundo intento, dos semanas después
  despliegues bloqueados en la primera semana         4
  de ellos, incumplimientos reales                    4
  tiquetes de soporte                                 0
  → porque el mensaje decía qué hacer
```

La entrega de suscripciones, automatizada:

```
antes   petición por correo → 3 semanas
        y 12 de las 61 suscripciones se habían creado por
        fuera del proceso                           ley 16

después una plantilla en la canalización crea
          la suscripción en el grupo correcto
          red conectada al centro, con rangos del plan
          identidad y permisos base
          diagnóstico enrutado
          presupuesto y etiquetas
        plazo                                    22 min
        suscripciones creadas por fuera en 6 meses      0
```

El resultado, seis meses después:

```
suscripciones sin grupo de administración      47           0
políticas asignadas en suscripciones            9           0
políticas quitadas por equipos                  4      imposible
```

```
recursos conformes (etiquetas)                36 %        98 %
almacenes públicos con datos                   19           0
recursos fuera de región permitida            112           0
exenciones vivas                                0          22
  con dueño, motivo y caducidad                 —          22
  vencidas y sin renovar                        —           1
plazo de entrega de una suscripción       3 semanas     22 min
suscripciones creadas por fuera del proceso    12           0
```

La lección que esta clase deja: mover sesenta y una suscripciones al grupo correcto no corrigió ni un solo recurso existente, porque las políticas de denegación solo actúan al crear o modificar; lo que corrigió catorce mil recursos fue una combinación de políticas de tipo modificar, tareas de remediación y cuatro meses de recrear cosas. Y las cuatro políticas que los equipos habían quitado no las quitaron por mala fe: las quitaron porque el mensaje de error decía «denegado por política» y no decía qué faltaba.

Laboratorio guiado

Ejecuta desde la raíz:

```
python classes/part-18-azure-production-architecture/217-enterprise-scale-landing-zones-y-management-groups/lab.py
```

El laboratorio selecciona el motor de práctica governance y produce labresult.json. El escenario, sus comprobaciones y el artefacto esperado corresponden a esta clase; no requiere credenciales y deja explícito qué debe revalidarse en un sandbox real.

1. Lee exercise.steps y formula una predicción antes de ejecutar.
2. Ejecuta la práctica y verifica todos los elementos de checks.
3. Provoca el caso de negativetest y explica la señal observada.
4. Materializa azure-landing-zone en evidence/ usando la plantilla indicada.
5. Para proveedor real, sigue sandbox.requires, registra costo y ejecuta destroy.

Evidencia esperada

El artefacto principal es una jerarquía de política, ownership y evidencia. Además, la entrega debe incluir el comando ejecutado, la salida estructurada y una conclusión que no exceda lo observado.

Reto verificable

Construye azure-landing-zone para el caso CloudShop. Incluye una alternativa descartada, un supuesto que pueda falsarse, una prueba de fallo y una decisión de rollback.

■ Criterio de aceptación

- [ ] lab.py termina con código 0 y genera JSON válido.
- [ ] La entrega conecta al menos tres requisitos con mecanismos verificables.
- [ ] Existe una prueba positiva y una prueba negativa con evidencia.
- [ ] Seguridad, costo y operación aparecen como decisiones, no como anexos.
- [ ] Se declara una limitación y una condición que obligaría a revisar el diseño.
- [ ] Otra persona puede repetir el recorrido sin conocimiento tácito.

■■ Errores frecuentes

Síntoma	Causa probable	Corrección
Un equipo quita la política que le estorba y nadie se entera	La política estaba asignada en la suscripción, que ese equipo administra	Asigna siempre en el grupo de administración y alerta sobre cambios de asignación.
Mover las suscripciones no corrige ningún recurso	Las políticas de denegación solo actúan sobre creaciones y modificaciones	Usa políticas de tipo modificar con tareas de remediación para lo existente, y planifica recrear lo que no se pueda corregir.
Al activar el modo denegar se bloquean despliegues legítimos y llueven los tiquetes	Se pasó de cero a denegar sin auditar y sin mensajes útiles	Audita semanas, corrige lo real, añade mensajes que digan qué falta y actualiza la plantilla por defecto antes de denegar.
Una carga impide crecer a otra sin relación con ella	Comparten suscripción y por tanto cuota	Separa por carga y por entorno, y pide cuota con margen antes de necesitarla.
Aparecen recursos creados fuera del proceso	Conseguir una suscripción tarda semanas	Automatiza la entrega de suscripciones preparadas; si tarda minutos, nadie busca atajos.
La factura crece tras aplicar el gobierno	Una política despliega agentes o ajustes de diagnóstico en cada recurso nuevo	Estima el coste de lo que la política despliega antes de asignarla y ajusta categorías y caducidad.

■ Seguridad, ética y costo

Trabaja con cuentas propias o sandboxes autorizados. No publiques secretos, identificadores reales ni datos personales. Antes de crear recursos pagos define presupuesto, etiquetas y comando de destrucción; después verifica que no queden recursos huérfanos. Los ejemplos locales enseñan contratos, pero no certifican cumplimiento ni disponibilidad de producción.

■ Preguntas de comprobación

1. ¿Qué resuelve cada uno de los tres niveles de la jerarquía?
2. ¿Por qué las políticas se asignan en grupos de administración y no en suscripciones?
3. ¿Qué corrige mover una suscripción de grupo y qué no?
4. ¿Cuál es el orden correcto para implantar una política y por qué?
5. ¿Qué debe llevar una exención para ser útil?

Referencias

- Microsoft (2025). Cloud Adoption Framework: Azure landing zone design areas. <https://learn.microsoft.com/en-us/azure/cloud-adoption-framework/ready/landing-zone/>
- Microsoft (2025). Management groups and organizing resources. <https://learn.microsoft.com/en-us/azure/governance/management-groups/overview>
- Microsoft (2025). Azure Policy effects and remediation. <https://learn.microsoft.com/en-us/azure/governance/policy/concepts/effects>
- Microsoft (2025). Azure Policy exemptions. <https://learn.microsoft.com/en-us/azure/governance/policy/concepts/exemption-structure>
- Microsoft (2025). Subscription vending and landing zone automation. <https://learn.microsoft.com/en-us/azure/cloud-adoption-framework/ready/landing-zone/design-area/subscription-vending>
- Documentación oficial vigente del servicio implementado; registra URL y fecha de consulta.

Descargar: Parte 18 en PDF · Recorrido de Azure en PDF · Manual integral

Evaluación — 217 Enterprise-scale landing zones y management groups

Preguntas

1. Explica el problema que resuelve esta capacidad sin mencionar un proveedor.
2. Identifica una frontera de responsabilidad y su propietario.
3. Compara dos alternativas mediante confiabilidad, seguridad, costo y operación.
4. ¿Qué demuestra el laboratorio y qué no puede demostrar?
5. Propón un caso límite o una prueba de fallo.

Reto verificable

Entrega azure-landing-zone con diagrama o configuración, evidencia de ejecución, alternativa descartada, riesgo residual y criterio de rollback.

Criterio de aceptación

- [ ] Resultado reproducible por una segunda persona.
- [ ] Evidencia vinculada a cada afirmación técnica.
- [ ] Prueba negativa o de fallo incluida.
- [ ] Costos y permisos expresados con unidades y alcance.
- [ ] Limitaciones declaradas sin presentar la simulación como producción.

Escala

Nivel	Evidencia
A — competente	Cumple todo y defiende trade-offs con datos.
B — en desarrollo	Cumple lo esencial; falta profundidad en una dimensión.
C — inicial	Artefacto parcial o conclusión sin evidencia suficiente.
0	Sin entrega reproducible.

218 — Entra ID, workload identity, PIM y Conditional Access

Parte: 18 — Azure: arquitectura empresarial y operación en producción Nivel: avanzado · Horas estimadas: 4
Laboratorio: iam · Estado: EXECUTABLECORE

Propósito

Resolver la identidad en Azure, que es donde se decide el alcance real de todo lo demás. La clase cubre las identidades de carga sin secretos, el modelo de asignación de permisos con su error característico —un ámbito demasiado amplio, que es el equivalente exacto del comodín de la clase 206—, la elevación temporal para el acceso administrativo y el acceso condicional como control de contexto que se aplica a personas y a cargas.

Resultados de aprendizaje

Al finalizar podrás:

1. Eliminar secretos de aplicación usando identidades administradas y federación.
2. Asignar permisos en el ámbito mínimo y detectar los amplios.
3. Configurar elevación temporal con aprobación para el acceso administrativo.
4. Aplicar acceso condicional sin quedarse fuera del propio inquilino.
5. Revisar accesos periódicamente y retirar lo que no se usa.

Conceptos centrales

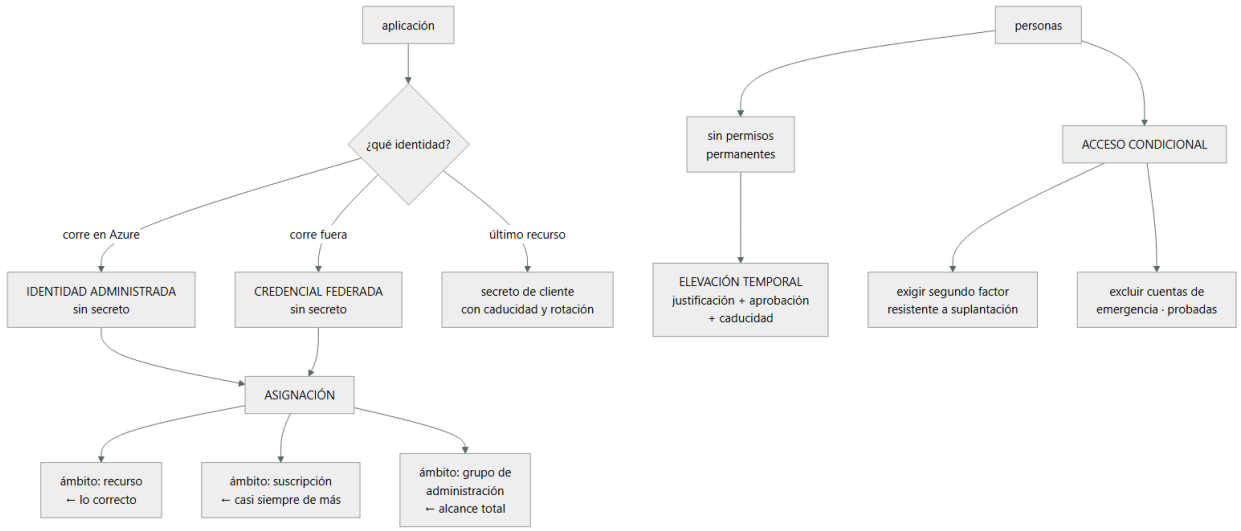
Concepto	Comprensión verificable
identidad administrada	Identidad ligada a un recurso de Azure, sin credenciales que gestionar. Asignada por el sistema o por el usuario.
credencial federada	Confianza declarada en un emisor externo para que sus testigos obtengan credenciales sin secreto.
ámbito de asignación	Nivel al que se concede un permiso: grupo de administración, suscripción, grupo de recursos o recurso.
elevación temporal	Activación de un permiso administrativo por tiempo limitado, con justificación y aprobación.
acceso condicional	Regla que decide si una autenticación se permite según usuario, aplicación, dispositivo, red y riesgo.
acceso de emergencia	Cuentas excluidas de las políticas, para no quedarse fuera. Vigiladas y probadas.

Modelo mental

La arquitectura Azure nace del tenant y las políticas heredables, y llega al workload mediante identidades administradas y redes privadas.

Aplicado a esta clase, separa siempre cuatro planos: intención (qué necesita el usuario), configuración (qué declaramos), estado observado (qué existe de verdad) y evidencia (cómo sabemos que cumple). Confundirlos produce diseños que se ven correctos en un diagrama pero fallan al operar.

■ Flujo de razonamiento



Desarrollo

1. Identidad de carga sin secretos

El objetivo es el mismo de la clase 206: que ninguna aplicación tenga una credencial guardada.

```
IDENTIDAD ADMINISTRADA — para lo que corre EN Azure
  el recurso tiene identidad propia; la plataforma le
  entrega testigos
  sin secretos, sin rotación, sin nada que robar

  ASIGNADA POR EL SISTEMA
    nace y muere con el recurso
    + no queda huérfana
    – no se puede preasignar permisos antes de crearlo

  ASIGNADA POR EL USUARIO
    recurso independiente, reutilizable por varios
    + permisos preparados antes; sirve para conjuntos
    – sobrevive al recurso                          ley 25
    → y hay que retirarla cuando ya no la use nadie

CREDENCIAL FEDERADA — para lo que corre FUERA
  canalizaciones, cargas en otro clúster, otra nube
  se declara confianza en el emisor y en el sujeto
  → mismo mecanismo y mismos errores que en la clase 206

SECRETO DE CLIENTE — último recurso
  con caducidad corta, rotación automatizada y una
  excepción registrada con fecha                    ley 25
```

Y la regla que ordena la elección:

```
¿corre en Azure?        identidad administrada
¿corre fuera?           credencial federada
¿ninguna de las dos?    secreto, con fecha de revisión

→ y una función de aptitud: cero secretos de cliente sin
  excepción registrada                          clase 190
```

Y el detalle de configuración donde se falla, igual que en AWS:

```
LA CREDENCIAL FEDERADA SE ATA A
  emisor, sujeto y audiencia

✗ sujeto con comodín, o sin comprobar audiencia
  → cualquier repositorio u organización puede obtener
    el testigo
✓ sujeto exacto: organización, repositorio y entorno

→ es el mismo fallo de la clase 206 con otro nombre, y por
  eso la prueba negativa es idéntica: intentarlo desde otro
```

```
  origen y comprobar que falla                    ley 22
```

2. El ámbito de asignación: el error característico

En Azure los permisos se conceden asignando un papel a una identidad en un ámbito, y el ámbito se hereda hacia abajo. Ahí está el error.

```
LOS CUATRO ÁMBITOS, de mayor a menor
  grupo de administración   → todas sus suscripciones
  suscripción               → todos sus grupos y recursos
  grupo de recursos         → todo lo de dentro
  recurso                   → solo ese

Y LA COSTUMBRE
  asignar «Colaborador» en la suscripción, porque es
  cómodo y funciona
  → esa identidad puede crear, modificar y borrar
    CUALQUIER cosa de la suscripción
  → incluida la base de datos de producción
```

Y el equivalente exacto de la clase 206:

```
en AWS   una condición de sujeto con comodín permite que
         cualquiera asuma el rol
en Azure una asignación en un ámbito amplio permite que la
         identidad toque cualquier cosa de ese ámbito

→ el mecanismo es distinto; el resultado, el mismo:
  alcance enorme desde un punto comprometido    clase 189
```

Cómo se hace bien, con el coste que tiene:

```
asignar en el ÁMBITO DEL RECURSO o del grupo de recursos
con el papel MÁS ESPECÍFICO que exista
  no «Colaborador» sino «Colaborador de datos de blob»
  no «Propietario» nunca, salvo casos contados

y cuando no existe un papel adecuado
  papel personalizado, con las acciones justas
  → cuesta mantenerlo, y compensa en los casos de mucho
    alcance
```

Y los papeles que hay que tratar como excepcionales:

```
Propietario                puede conceder permisos
                           → quien lo tiene, lo tiene todo
Administrador de acceso    puede conceder permisos
                           → equivalente en la práctica
Colaborador en suscripción puede borrar casi todo

→ estos tres se conceden con elevación temporal, nunca de
  forma permanente
```

Cómo detectar los ámbitos amplios, que es el trabajo continuo:

```
inventario de asignaciones, por ámbito
  ¿cuántas hay en grupo de administración?
  ¿cuántas en suscripción?
  ¿cuántas identidades tienen Propietario?

y la medida que importa
  ALCANCE desde cada identidad: a cuántos recursos llega
  → se mide, y es la cifra que ordena el trabajo
                                        clases 133, 189

y las asignaciones que no se usan
  → los registros de actividad dicen qué permisos se
    ejercen de verdad
  → lo concedido y no usado se retira            clase 134
```

Y una fuente de alcance que se olvida:

```
LOS GRUPOS ANIDADOS
  un grupo dentro de otro hereda sus asignaciones
  → alguien añadido a un grupo «de proyecto» acaba con
    permisos de producción sin que nadie lo decida
```

```
→ hay que revisar la pertenencia efectiva, no la directa
```

3. Elevación temporal y revisión de accesos

Para las personas, el objetivo es que nadie tenga permisos administrativos permanentes.

```
EL MODELO
  la persona es ELEGIBLE para un papel, no lo tiene
  cuando lo necesita, lo activa
    con justificación escrita
    con aprobación de otra persona, si el papel es
      sensible
    con segundo factor en el momento de activar
    y con CADUCIDAD: 1, 4 u 8 horas

QUÉ RESUELVE
  el permiso permanente que nadie revisa
  el administrador que se va y conserva accesos
  el compromiso de una cuenta con permisos totales
  y deja registro de POR QUÉ se usó, no solo de que se usó
                                                clase 141
```

Y los detalles que hacen que funcione o que se rodee:

```
SI ACTIVAR TARDA MUCHO, la gente pedirá permanente
  → activación en segundos cuando no requiere aprobación
  → y aprobadores suficientes, con turnos, cuando sí
                                                  ley 16

SI NO HAY APROBADOR DISPONIBLE DE MADRUGADA
  → un incidente se atasca esperando
  → por eso los papeles de emergencia se activan sin
    aprobación, con alerta inmediata y revisión posterior

Y LA PRUEBA QUE HAY QUE HACER
  activar el papel de emergencia y cronometrar
  → en la clase 179, el acceso de emergencia estaba creado,
    documentado y NO funcionaba                   ley 22
```

Las revisiones de acceso, que es lo que impide la acumulación:

```
cada trimestre, quien es dueño revisa
  quién es elegible para cada papel
  qué identidades tienen asignaciones amplias
  qué invitados externos siguen teniendo acceso

y la regla que las hace útiles
  quien no responde a la revisión → se retira el acceso
  → si la falta de respuesta lo conserva, la revisión no
    sirve de nada
```

Y la señal que dice si el modelo está funcionando:

```
número de asignaciones PERMANENTES de papeles sensibles
  objetivo: cero, salvo cuentas de emergencia
y número de activaciones al mes, con su justificación
  → si nadie activa nunca, o sobran los papeles o la gente
    tiene permisos por otra vía
```

4. Acceso condicional, sin quedarse fuera

El acceso condicional decide si una autenticación se permite según el contexto, y es el control más potente del directorio.

```
LO QUE PUEDE EVALUAR
  quién: usuario, grupo, papel
  qué: aplicación o recurso
  desde dónde: red, país, dispositivo
  con qué: dispositivo conforme o unido al directorio
  y el riesgo: de la sesión y del usuario

LO QUE PUEDE EXIGIR
  segundo factor
  dispositivo conforme
  aplicación cliente aprobada
```

```
  frecuencia de reautenticación
  o directamente bloquear
```

Las políticas mínimas de una organización:

```
1  segundo factor para TODOS los usuarios
   → y resistente a suplantación (llave física o similar)
     para los papeles administrativos
2  bloquear protocolos de autenticación antiguos
   → son los que se saltan el segundo factor
3  exigir dispositivo conforme para el acceso
   administrativo
4  bloquear países donde no se opera
   → medida sencilla que reduce mucho el ruido
5  exigir reautenticación al elevar privilegios
6  y proteger los flujos de registro de credenciales
   → registrar un segundo factor es un momento crítico
```

Y el error que deja a la organización fuera de su propio inquilino:

```
UNA POLÍTICA QUE EXIGE DISPOSITIVO CONFORME A TODOS
  incluidas las cuentas administrativas
  y el día que falla el sistema de conformidad, o el
  dispositivo del administrador se rompe
  → nadie puede entrar a arreglarlo

LA PROTECCIÓN
  cuentas de acceso de emergencia
    excluidas de todas las políticas condicionales
    con credenciales largas guardadas físicamente
    con alerta INMEDIATA en cada uso
    y probadas cada trimestre                    ley 22
  y siempre DOS, por si una falla
```

Y la disciplina de despliegue, que es la misma de siempre:

```
1  crear la política en modo SOLO INFORME
2  medir a quién afectaría, durante semanas
3  excluir lo que haya que excluir, con registro
4  activar para un grupo piloto
5  activar para todos
→ activar una política condicional para todos el primer día
  es la forma más rápida de dejar a la empresa sin acceso
                                              clase 200
```

Y la lista de comprobación de la clase:

- ninguna aplicación tiene secretos de cliente sin excepción registrada
- lo que corre en Azure usa identidad administrada
- las credenciales federadas atan emisor, sujeto y audiencia
- no hay asignaciones de Propietario permanentes
- las asignaciones se hacen en el ámbito del recurso o del grupo
- se usa el papel más específico disponible
- está medido el alcance desde cada identidad
- se revisan los permisos concedidos y no usados
- se revisa la pertenencia efectiva a grupos, no la directa
- los papeles sensibles requieren elevación temporal
- hay aprobadores suficientes y activación rápida
- el acceso de emergencia se ha probado este trimestre
- hay dos cuentas de emergencia excluidas de las políticas
- cada uso de emergencia dispara alerta inmediata
- las políticas condicionales se desplegaron en solo informe
- los protocolos de autenticación antiguos están bloqueados

Y el cierre que enlaza con la clase siguiente: con la jerarquía y la identidad resueltas, queda conectar las suscripciones entre sí y con el centro de datos, y decidir por dónde entra y sale el tráfico. Red en Azure es la materia de la clase 219.

Ejemplo trabajado

CloudShop revisa la identidad de su inquilino de Azure. Lo que sigue es el inventario de asignaciones, las tres identidades que podían tocar toda la producción, y la política condicional que dejó al equipo fuera durante cuarenta minutos.

El inventario:

```
identidades de aplicación                          184
  con identidad administrada                        61
  con credencial federada                            9
  CON SECRETO DE CLIENTE                           114  ←
    de ellos, caducados y aún referenciados         18
    de ellos, con más de 2 años                     47

asignaciones de papel                            1.940
  en grupo de administración                        31  ←
  en suscripción                                   820  ←
  en grupo de recursos                             760
  en recurso                                       329

papeles sensibles, permanentes
  Propietario                                       41
  Administrador de acceso                           12
  Colaborador en suscripción                       214
```

Y las tres asignaciones que resultaron peores:

```
1  la identidad de la canalización de despliegue tenía
   Colaborador en el grupo de administración «cargas»
   → alcance: 43 suscripciones, todo
   → y podía borrar bases de datos de producción

2  una identidad administrada asignada por el usuario,
   creada en 2022 para una prueba, tenía Propietario en la
   suscripción de producción de pedidos
   → el recurso que la usaba se borró hace 2 años
   → la identidad seguía viva, con Propietario     ley 25

3  un grupo llamado «Proyecto Migración 2023» tenía
   Colaborador en 9 suscripciones
   → 34 personas seguían en él
   → 11 ya no trabajaban en esas cargas
   → y 3 habían entrado por pertenencia ANIDADA, sin que
     nadie los añadiera a propósito
```

El alcance medido, que es lo que ordenó el trabajo:

```
identidad                     recursos alcanzables
canalización de despliegue              14.200   ← todo
identidad huérfana de 2022               1.840
grupo Proyecto Migración                 6.100
api-pedidos (correcta)                       4
procesador-eventos (correcta)                7

→ dos identidades y un grupo concentraban casi todo el
  riesgo del inquilino
```

Las correcciones:

```
CANALIZACIÓN
  antes    Colaborador en grupo de administración
  después  4 identidades federadas, una por propósito
           cada una con asignación en el GRUPO DE RECURSOS
           que le toca
           y una barrera: ninguna puede asignar papeles ni
           modificar políticas
  alcance  14.200 → 340 recursos, repartidos en 4

IDENTIDAD HUÉRFANA
  retirada
  y una comprobación nueva: identidades administradas
  asignadas por el usuario sin ningún recurso asociado
  → se encontraron 23 más                          ley 25

GRUPO DE PROYECTO
  retirado; sustituido por grupos por carga y entorno
  y revisión trimestral de pertenencia EFECTIVA

SECRETOS DE CLIENTE
  114 → 12 en cuatro meses
```

```
    61 pasaron a identidad administrada
    28 a credencial federada
    13 se apagaron: eran aplicaciones que ya no existían
    12 quedan, todos de terceros, con rotación automática
       y excepción registrada con fecha
  los 18 caducados y referenciados
    → 4 de ellos, en aplicaciones que fallaban
      silenciosamente desde hacía meses            ley 13
```

La elevación temporal:

```
se retiraron TODAS las asignaciones permanentes de
Propietario y Administrador de acceso
y pasaron a elegibles con activación

  Propietario                aprobación de 2, caducidad 4 h
  Administrador de acceso    aprobación de 2, caducidad 2 h
  Colaborador en suscripción activación directa, 8 h
  Emergencia                 sin aprobación, alerta
                             inmediata, revisión posterior

los primeros dos meses
  activaciones                                      412
  con justificación útil                            340
  con justificación tipo «trabajo»                   72
    → se pidió mejorarlas; a los 3 meses, 11
  activaciones de emergencia                          3
    · 2 incidentes reales
    · 1 porque el aprobador estaba de vacaciones y nadie
      había cubierto el turno            ← se corrigió

y la prueba de emergencia, ejecutada
  primer intento   la cuenta de emergencia no podía activar
                   porque la política condicional creada en
                   el paso siguiente la había incluido
                   sin querer
  → corregido, y añadido al calendario trimestral
```

La política condicional que dejó al equipo fuera.

```
se creó una política: «exigir dispositivo conforme para
acceder al portal de Azure»
se activó directamente para todos

  a las 09:40, el sistema de evaluación de conformidad
  tuvo una incidencia y marcó como no conformes a 190
  dispositivos
  → nadie del equipo de plataforma podía entrar al portal
  → incluidas las 2 cuentas de emergencia, que se habían
    incluido en el ámbito por error

  09:40  se detecta
  09:52  se localiza a alguien con acceso desde otro camino
  10:20  política desactivada
  duración                                        40 min

qué faltó
  modo solo informe durante semanas
  exclusión explícita y COMPROBADA de las cuentas de
    emergencia
  y despliegue por grupos                    clase 200

segundo intento
  1  solo informe, 3 semanas
     → 1.140 accesos que se habrían bloqueado
     → de ellos, 210 legítimos desde dispositivos no
       registrados: contratistas y 2 equipos con portátiles
       propios
  2  se registraron los dispositivos legítimos
  3  exclusión de cuentas de emergencia, comprobada
     entrando con ellas
  4  grupo piloto de 12 personas, 2 semanas
  5  activación general
  accesos bloqueados legítimos tras activar          2
```

El resto de políticas, con su despliegue:

```
política                              informe   activa
segundo factor para todos               3 sem     sí
segundo factor resistente para
  papeles administrativos               2 sem     sí
bloquear autenticación antigua          4 sem     sí
  → el informe encontró 6 aplicaciones que la usaban
  → 4 se corrigieron, 2 se apagaron
dispositivo conforme para portal        3 sem     sí
bloquear países no operados             1 sem     sí
  → 41.000 intentos bloqueados el primer mes
reautenticación al elevar               1 sem     sí
```

El resultado, seis meses después:

```
secretos de cliente                          114          12
  caducados y referenciados                   18           0
asignaciones en grupo de administración       31           4
asignaciones en suscripción                  820         190
Propietario permanente                        41           0
alcance de la identidad de despliegue     14.200         340
identidades huérfanas                         24           0
personas con permisos administrativos
  permanentes                                267           2
  (las 2 cuentas de emergencia)
prueba de emergencia ejecutada                 no    trimestral
incidentes por política condicional              1           0
```

La lección que esta clase deja: el error característico de Azure resultó ser exactamente el equivalente del comodín de la clase 206: no una condición mal escrita, sino un ámbito de asignación demasiado alto, con una sola identidad capaz de tocar catorce mil recursos. Y la política de seguridad que se activó para proteger el portal dejó a la empresa fuera durante cuarenta minutos, incluidas las cuentas de emergencia que existían precisamente para eso y que nadie había comprobado.

Laboratorio guiado

Ejecuta desde la raíz:

```
python classes/part-18-azure-production-architecture/218-entra-id-workload-identity-pim-y-conditional-access/lab.py
```

El laboratorio selecciona el motor de práctica iam y produce labresult.json. El escenario, sus comprobaciones y el artefacto esperado corresponden a esta clase; no requiere credenciales y deja explícito qué debe revalidarse en un sandbox real.

1. Lee exercise.steps y formula una predicción antes de ejecutar.
2. Ejecuta la práctica y verifica todos los elementos de checks.
3. Provoca el caso de negativetest y explica la señal observada.
4. Materializa azure-identity en evidence/ usando la plantilla indicada.
5. Para proveedor real, sigue sandbox.requires, registra costo y ejecuta destroy.

Evidencia esperada

El artefacto principal es una matriz de acceso mínimo con prueba de denegación. Además, la entrega debe incluir el comando ejecutado, la salida estructurada y una conclusión que no exceda lo observado.

Reto verificable

Construye azure-identity para el caso CloudShop. Incluye una alternativa descartada, un supuesto que pueda falsarse, una prueba de fallo y una decisión de rollback.

■ Criterio de aceptación

- [ ] lab.py termina con código 0 y genera JSON válido.
- [ ] La entrega conecta al menos tres requisitos con mecanismos verificables.

- [ ] Existe una prueba positiva y una prueba negativa con evidencia.
- [ ] Seguridad, costo y operación aparecen como decisiones, no como anexos.
- [ ] Se declara una limitación y una condición que obligaría a revisar el diseño.
- [ ] Otra persona puede repetir el recorrido sin conocimiento tácito.

Errores frecuentes

Síntoma	Causa probable	Corrección
Una identidad de aplicación puede tocar recursos que no tienen nada que ver con ella	La asignación se hizo en la suscripción o en el grupo de administración	Asigna en el ámbito del recurso o del grupo de recursos, con el papel más específico disponible, y mide el alcance resultante.
Alguien acaba con permisos de producción sin que nadie se los diera	Pertenencia anidada a un grupo con asignaciones amplias	Revisa la pertenencia efectiva, no la directa, y retira los grupos de proyecto cuando el proyecto termina.
Una identidad sigue teniendo permisos y su recurso ya no existe	Identidad administrada asignada por el usuario, que sobrevive al recurso	Inventaría periódicamente las identidades sin recurso asociado y retíralas.
Una aplicación falla en silencio desde hace meses	Su secreto de cliente caducó y nadie lo vigilaba	Sustituye por identidad administrada o federada; para lo que quede, rotación automática y alerta por caducidad próxima.
Nadie puede entrar al portal cuando falla un sistema auxiliar	Una política condicional se activó para todos sin exclusiones comprobadas	Despliega en solo informe, excluye dos cuentas de emergencia y comprueba entrando con ellas antes de activar.
La gente pide permisos permanentes pese a existir elevación temporal	Activar tarda demasiado o no hay aprobadores disponibles	Activación en segundos cuando no requiere aprobación, aprobadores con turnos y un papel de emergencia sin aprobación con alerta inmediata.

Seguridad, ética y costo

Trabaja con cuentas propias o sandboxes autorizados. No publiques secretos, identificadores reales ni datos personales. Antes de crear recursos pagos define presupuesto, etiquetas y comando de destrucción; después verifica que no queden recursos huérfanos. Los ejemplos locales enseñan contratos, pero no certifican cumplimiento ni disponibilidad de producción.

Preguntas de comprobación

1. ¿Qué identidad corresponde a una carga que corre en Azure y cuál a una que corre fuera?
2. ¿Cuál es el equivalente en Azure del comodín en la condición de sujeto de AWS?
3. ¿Qué hace que la elevación temporal se rodee y cómo se evita?
4. ¿Por qué hacen falta dos cuentas de acceso de emergencia y qué hay que probar de ellas?
5. ¿Cómo se despliega una política condicional sin dejar a la organización fuera?

Referencias

- Microsoft (2025). Managed identities for Azure resources. <https://learn.microsoft.com/en-us/entra/identity/managed-identities-azure-resources/overview>
- Microsoft (2025). Workload identity federation. <https://learn.microsoft.com/en-us/entra/workload-id/workload-identity-federation>
- Microsoft (2025). Azure RBAC scope and best practices. <https://learn.microsoft.com/en-us/azure/role-based-access-control/best-practices>
- Microsoft (2025). Privileged Identity Management. <https://learn.microsoft.com/en-us/entra/id-governance/privileged-identity-management/pim-configure>

- Microsoft (2025). Conditional Access and emergency access accounts. <https://learn.microsoft.com/en-us/entra/identity/role-based-access-control/security-emergency-access>
- Documentación oficial vigente del servicio implementado; registra URL y fecha de consulta.

Descargar: Parte 18 en PDF · Recorrido de Azure en PDF · Manual integral

Anterior	Índice	Siguiente
← 217 · Enterprise-scale landing zones y management groups	Parte 18 · Programa	219 · Hub-spoke, Virtual WAN, Private Link y DNS privado →

Evaluación — 218 Entra ID, workload identity, PIM y Conditional Access

Preguntas

1. Explica el problema que resuelve esta capacidad sin mencionar un proveedor.
2. Identifica una frontera de responsabilidad y su propietario.
3. Compara dos alternativas mediante confiabilidad, seguridad, costo y operación.
4. ¿Qué demuestra el laboratorio y qué no puede demostrar?
5. Propón un caso límite o una prueba de fallo.

Reto verificable

Entrega azure-identity con diagrama o configuración, evidencia de ejecución, alternativa descartada, riesgo residual y criterio de rollback.

Criterio de aceptación

- [ ] Resultado reproducible por una segunda persona.
- [ ] Evidencia vinculada a cada afirmación técnica.
- [ ] Prueba negativa o de fallo incluida.
- [ ] Costos y permisos expresados con unidades y alcance.
- [ ] Limitaciones declaradas sin presentar la simulación como producción.

Escala

Nivel	Evidencia
A — competente	Cumple todo y defiende trade-offs con datos.
B — en desarrollo	Cumple lo esencial; falta profundidad en una dimensión.
C — inicial	Artefacto parcial o conclusión sin evidencia suficiente.
0	Sin entrega reproducible.

219 — Hub-spoke, Virtual WAN, Private Link y DNS privado

Parte: 18 — Azure: arquitectura empresarial y operación en producción Nivel: avanzado · Horas estimadas: 4
Laboratorio: network · Estado: EXECUTABLECORE

Propósito

Montar la red de Azure conectando decenas de suscripciones sin abrir caminos indeseados. La clase cubre la topología de centro y radios frente a la gestionada, los puntos privados hacia los servicios de la plataforma, y la parte que produce la mayoría de los incidentes de esta capa: la resolución de nombres privados, que en Azure exige enlazar cada zona a cada red y falla en silencio cuando no se hace.

Resultados de aprendizaje

Al finalizar podrás:

1. Elegir entre centro y radios propio y topología gestionada.
2. Segmentar con grupos de seguridad y rutas, cerrando por ausencia de camino.
3. Conectar a los servicios de la plataforma con puntos privados.
4. Resolver nombres privados en todas las redes, sin huecos.
5. Diagnosticar los fallos característicos de esta topología.

Conceptos centrales

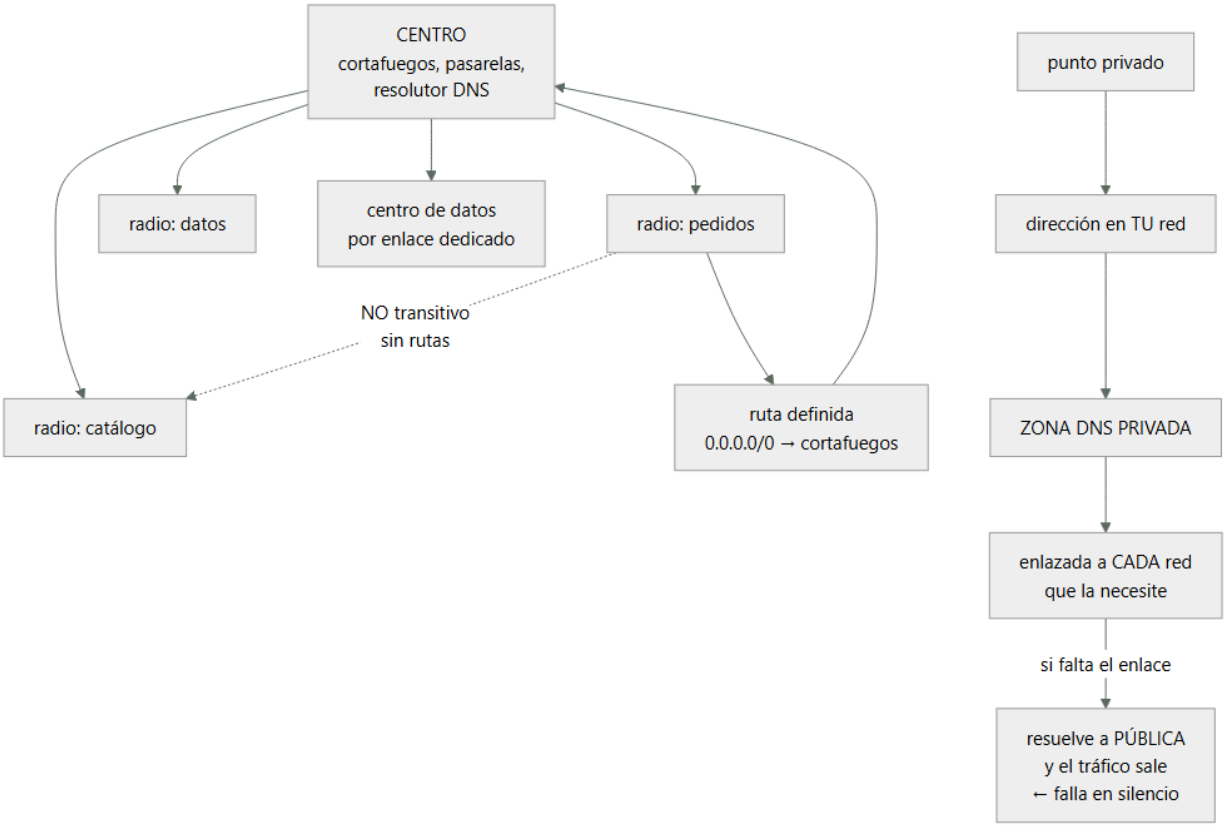
Concepto	Comprensión verificable
centro y radios	Topología con una red central que concentra conectividad e inspección, y redes de carga conectadas a ella.
emparejamiento	Conexión entre dos redes virtuales. No es transitivo por sí solo: hace falta encaminar.
ruta definida por el usuario	Ruta que obliga al tráfico a pasar por un dispositivo. Es lo que fuerza la inspección.
punto privado	Interfaz con dirección de tu red que representa un servicio de la plataforma.
zona DNS privada	Zona que resuelve el nombre del servicio a la dirección privada. Debe enlazarse a cada red que la necesite.
grupo de seguridad de red	Reglas de filtrado por subred o interfaz, con prioridades y reglas por defecto que hay que conocer.

Modelo mental

La arquitectura Azure nace del tenant y las políticas heredables, y llega al workload mediante identidades administradas y redes privadas.

Aplicado a esta clase, separa siempre cuatro planos: intención (qué necesita el usuario), configuración (qué declaramos), estado observado (qué existe de verdad) y evidencia (cómo sabemos que cumple). Confundirlos produce diseños que se ven correctos en un diagrama pero fallan al operar.

■ Flujo de razonamiento



Desarrollo

1. Topología: centro y radios

La topología estándar de Azure concentra en una red central lo que no debe repetirse.

```
EN EL CENTRO
  cortafuegos o dispositivo de inspección
  pasarelas hacia el centro de datos y hacia internet
  resolutor DNS y sus reglas de reenvío
  y, a veces, los puntos privados compartidos

EN CADA RADIO
  las cargas, en sus subredes
  sin pasarela propia ni salida directa a internet
```

Y la propiedad que hay que entender antes de nada:

```
EL EMPAREJAMIENTO NO ES TRANSITIVO
  radio A ↔ centro y radio B ↔ centro
  → A NO alcanza B automáticamente
  → hace falta que el tráfico pase por un dispositivo del
    centro y haya rutas que lo lleven allí

→ y esto es una VENTAJA: el aislamiento entre radios es el
  estado por defecto                          clase 199
→ abrirlo es una decisión, no un descuido
```

La elección entre montar el centro o usar la topología gestionada:

```
CENTRO PROPIO
  + control total: cualquier dispositivo, cualquier diseño
  + más barato en tráfico
  – hay que montar y operar el encaminamiento y la
    alta disponibilidad

TOPOLOGÍA GESTIONADA (Virtual WAN)
  + conecta radios, sedes y usuarios remotos con menos
    trabajo
  + encaminamiento y escala gestionados
```

```
  + tablas de rutas para segmentar              clase 199
  − coste por conexión y por gigabyte
  − menos control fino

CRITERIO
  pocas redes y una región    → centro propio
  muchas sedes, varias regiones, usuarios remotos
                              → gestionada
```

Y lo que hay que decidir en ambos casos:

```
¿POR DÓNDE SALE EL TRÁFICO A INTERNET?
  salida centralizada por el cortafuegos del centro
  → registro, control y direcciones fijas       clase 200
  → y una ruta definida por el usuario en cada subred que
    mande 0.0.0.0/0 al cortafuegos

¿QUÉ RADIOS SE VEN ENTRE SÍ?
  por defecto, ninguno
  → y lo que se abra, con rutas explícitas y pasando por
    inspección si cruza fronteras de confianza  clase 189
```

Y una decisión que ahorra dinero y latencia:

```
el tráfico entre dos radios que se hablan mucho puede ir
directo con un emparejamiento entre ellos
→ evita pagar el proceso del cortafuegos dos veces
→ y hay que decidirlo, no dejarlo pasar por costumbre
                                                clase 199
```

2. Filtrado y rutas: lo que hay que saber

Los grupos de seguridad de red tienen reglas por defecto que sorprenden si no se conocen.

```
REGLAS POR DEFECTO, que existen aunque no se vean
  entrante   permitido desde la propia red virtual
             permitido desde el balanceador
             DENEGADO todo lo demás
  saliente   permitido hacia la propia red virtual
             PERMITIDO HACIA INTERNET            ←
             denegado el resto

→ la salida a internet está PERMITIDA por defecto
→ y por eso el control de salida es una decisión explícita
                                        clase 200, ley 26
```

Y el funcionamiento por prioridad, que causa confusión:

```
las reglas se evalúan por número de prioridad, de menor a
mayor, y la PRIMERA que coincide decide
→ una regla permisiva con prioridad baja anula las
  restrictivas posteriores
→ y las reglas por defecto tienen prioridades muy altas:
  siempre se evalúan al final

regla práctica
  deja huecos entre prioridades (100, 200, 300…)
  y documenta el orden esperado
```

Y dónde aplicarlos:

```
A LA SUBRED     lo habitual; se aplica a todo lo de dentro
A LA INTERFAZ   para excepciones; complica el diagnóstico
→ aplicar en los dos sitios a la vez es la causa más común
  de «no entiendo por qué no llega»
```

Las rutas definidas por el usuario, que son lo que fuerza la inspección:

```
por defecto, cada subred tiene rutas del sistema hacia
  la propia red, los emparejamientos e internet

una ruta definida por el usuario las sustituye
  0.0.0.0/0 → dirección del cortafuegos
  → todo el tráfico saliente pasa por él

y el error clásico
  poner la ruta en la subred del cortafuegos también
```

```
  → el tráfico entra en bucle
  → la subred del dispositivo NO lleva esa ruta
```

Y la regla del prefijo más específico, que aquí también manda:

```
si hay una ruta más específica hacia otro sitio, gana
  → así se implementan las excepciones            clase 194
→ y así también se abren caminos sin querer, si alguien
  añade una ruta «temporal»                        ley 25
```

Y una comprobación que resuelve la mitad de los casos:

```
la herramienta de diagnóstico de siguiente salto dice, para
un origen y un destino concretos, qué ruta se aplica y hacia
dónde va
→ y hay otra que dice qué regla de seguridad permite o
  deniega ese flujo
→ usarlas antes de suponer                       clase 202
```

3. Puntos privados y la trampa del DNS

Los puntos privados en Azure resuelven lo mismo que en la clase 200, y tienen una particularidad que causa la mayoría de los incidentes de esta clase.

```
CÓMO FUNCIONA
  se crea un punto privado en una subred: obtiene una
  dirección de TU red
  el servicio queda alcanzable por esa dirección
  y se puede desactivar su acceso público

Y LA PARTE QUE FALLA
  el nombre público del servicio debe resolver a la
  dirección PRIVADA
  eso lo hace una ZONA DNS PRIVADA
  y esa zona debe estar ENLAZADA a cada red virtual que la
  necesite

  ✗ si falta el enlace
    el nombre resuelve a la dirección PÚBLICA
    → el tráfico sale a internet
    → y si el acceso público está desactivado, FALLA
    → y si no lo está, FUNCIONA pagando salida y sin
      pasar por ningún control            ← lo peor
```

Y por eso el fallo es silencioso:

```
funciona desde la red donde se creó el punto privado
y no funciona —o funciona por internet— desde las demás
→ el síntoma es «a mí me va bien»                clase 195
```

El montaje que evita el problema:

```
ZONAS PRIVADAS CENTRALIZADAS
  todas las zonas de puntos privados en la suscripción de
  conectividad
  enlazadas a TODAS las redes que las necesitan
  y creadas automáticamente por política          clase 217
    → una política de tipo «desplegar si no existe» que
      registra el punto privado en la zona correcta
    → sin ella, cada equipo crea su zona y aparecen
      duplicadas y divergentes

RESOLUTOR PRIVADO EN EL CENTRO
  con regla de reenvío desde el centro de datos hacia
  Azure y viceversa
  → y hay que comprobarlo EN LOS DOS SENTIDOS  clase 195

COMPROBACIÓN AUTOMÁTICA
  para cada servicio con punto privado, resolver su nombre
  desde CADA red y verificar que devuelve dirección privada
  → función de aptitud, ejecutada a diario     clase 190
```

Y una decisión que hay que tomar y que se olvida:

```
DESACTIVAR EL ACCESO PÚBLICO del servicio
  crear el punto privado no cierra el público
```

```
   → el servicio sigue alcanzable desde internet
   → y la política de «sin acceso público» de la clase 217
     es lo que lo impide de verdad
```

Y el consumo de direcciones, que hay que planificar:

```
un punto privado = una dirección de la subred
30 servicios × 3 entornos = 90 direcciones
→ y a eso se suman las subredes que ciertos servicios
  exigen en exclusiva y con tamaño mínimo
→ el plan de direcciones tiene que contarlo    clase 193
```

4. Diagnóstico y operación

Los fallos de esta topología tienen firmas reconocibles.

SÍNTOMA	CAUSA HABITUAL
funciona desde una red y no desde otra	falta el enlace de la zona DNS privada
el tráfico sale a internet pese a tener punto privado	el nombre resuelve a la dirección pública
dos radios no se ven	el emparejamiento no es transitivo y faltan rutas
el tráfico no pasa por el cortafuegos	falta la ruta definida por el usuario en esa subred
bucle de encaminamiento	la ruta 0.0.0.0/0 se puso también en la subred del cortafuegos
se permite algo que no debería	una regla permisiva con prioridad menor gana
no llega y las reglas parecen correctas	hay grupo de seguridad en la subred Y en la interfaz
todo deja de funcionar tras un cambio de red	alguien cambió una ruta y el prefijo más específico desvió el tráfico

Y las herramientas, por orden de coste:

```
1  comprobación de siguiente salto y de flujo permitido
   → responde en segundos qué ruta y qué regla aplican
2  registros de flujo de los grupos de seguridad
   → quién habla con quién, qué se deniega   clase 202
3  captura de paquetes
   → solo cuando lo anterior no basta
```

Lo que hay que vigilar:

```
zonas privadas sin enlazar a redes que las necesitan
puntos privados cuyo nombre resuelve a dirección pública
reglas de seguridad con origen o destino demasiado amplios
rutas definidas por el usuario creadas fuera del proceso
estado de las pasarelas y de las sesiones hacia el centro
  de datos                                    clase 198
uso de direcciones por subred, frente al total
  → una subred agotada bloquea el escalado     clase 193
```

Y una alerta que evita sorpresas:

```
«ha aparecido una ruta o una regla que no está en el
 repositorio»
→ detecta los cambios manuales, que son los que se quedan
                                                  ley 25
```

Y la lista de comprobación de la clase:

- ■ la topología está decidida y justificada
- ■ los radios no se ven entre sí salvo decisión explícita
- ■ toda subred de carga tiene ruta 0.0.0.0/0 al cortafuegos

- la subred del cortafuegos NO tiene esa ruta
- los grupos de seguridad están en la subred, no en ambos sitios
- las prioridades dejan huecos y están documentadas
- se sabe que la salida a internet está permitida por defecto
- los servicios de plataforma se alcanzan por punto privado
- el acceso público de esos servicios está desactivado
- las zonas DNS privadas están centralizadas y enlazadas a todas las redes
- hay política que registra los puntos privados en la zona correcta
- hay comprobación diaria de que cada nombre resuelve a dirección privada desde cada red
- el reenvío de nombres con el centro de datos funciona en ambos sentidos
- el plan de direcciones cuenta los puntos privados y las subredes exclusivas
- hay alerta ante rutas o reglas creadas fuera del proceso

Y el cierre que enlaza con la clase siguiente: con jerarquía, identidad y red resueltas, hace falta declarar todo esto como código de forma que se pueda desplegar, revisar y retirar. Es la materia de la clase 220.

Ejemplo trabajado

CloudShop monta la red de sus 61 suscripciones en Azure. Lo que sigue es el incidente del punto privado que llevaba meses saliendo a internet, el bucle de encaminamiento del primer día, y la comprobación diaria que quedó montada.

La topología elegida:

```
centro propio, uno por región (2)
  cortafuegos con inspección
  pasarela hacia el centro de datos              clase 198
  resolutor DNS privado con reglas de reenvío
  zonas DNS privadas de todos los servicios

43 radios: uno por carga y entorno
  sin pasarela propia
  sin salida directa a internet
  ruta 0.0.0.0/0 → cortafuegos del centro

motivo de centro propio y no gestionado
  2 regiones, sin sedes remotas ni usuarios remotos
  y el volumen entre radios y centro justificaba el ahorro
  → decisión registrada, con la señal que la reabriría:
    «si se conectan más de 10 sedes»             clase 190
```

Incidente 1 · El bucle de encaminamiento, día uno.

```
síntoma   al aplicar la ruta 0.0.0.0/0 en todas las subredes
          del centro y de los radios, nada funcionaba
          el cortafuegos mostraba tráfico que entraba y
          volvía a entrar

causa     la ruta se había aplicado también a la subred del
          propio cortafuegos
          → el tráfico salía del cortafuegos, encontraba la
            ruta que apuntaba al cortafuegos, y volvía

corrección
  la subred del cortafuegos usa las rutas del sistema
  y se añadió una función de aptitud: ninguna tabla de
  rutas con 0.0.0.0/0 puede asociarse a la subred del
  dispositivo de inspección                      clase 190

tiempo perdido                                     3 h
```

Incidente 2 · El punto privado que salía a internet.

```
síntoma reportado   ninguno

se descubrió al revisar el coste de salida     clase 214
  transferencia de salida a internet          1.940 €/mes
  esperada, con puntos privados                 ~200 €
```

```
diagnóstico
  27 servicios tenían punto privado creado
  las zonas DNS privadas estaban enlazadas a la red del
  centro y a 4 radios
  → y había 43 radios

  desde los 39 radios sin enlace
    el nombre del servicio resolvía a la dirección PÚBLICA
    el tráfico salía por el cortafuegos a internet
    y volvía a entrar por el punto público del servicio
    → funcionaba perfectamente
    → y por eso nadie lo reportó                    ley 13

  cuánto llevaba así
    los primeros radios se crearon hacía 7 meses
    los enlaces se hicieron a mano para los 4 primeros y
    luego nadie los hizo más                       ley 25

lo que implicaba, además del coste
  el tráfico hacia las bases de datos salía a internet
  cifrado, pero por internet
  y el acceso público de esos servicios seguía activado
  → cualquiera con la credencial podía llegar desde fuera
                                                clase 189

corrección
  1  política de tipo «desplegar si no existe» que registra
     cada punto privado en la zona centralizada correcta
  2  enlace de todas las zonas a las 43 redes, automatizado
  3  desactivación del acceso público de los 27 servicios
     → y aquí fallaron 3: dos bases y un almacén tenían
       clientes externos legítimos que se descubrieron al
       cortar
     → 2 se movieron a punto privado desde el centro de
       datos; 1 quedó con excepción, con dueño y fecha
  4  comprobación diaria: resolver el nombre de cada
     servicio desde cada red y verificar dirección privada

resultado
  transferencia de salida             1.940 € → 180 €/mes
  servicios con acceso público            27 → 1 (con
                                          excepción)
  fallos de la comprobación diaria en 6 meses         4
    → los 4, radios nuevos creados antes de que la
      automatización enlazara sus zonas
    → corregidos en menos de 1 hora cada uno
```

Incidente 3 · «No llega y las reglas parecen correctas».

```
síntoma   el servicio de catálogo no podía llamar al de
          precios, en otro radio

lo que se revisó primero (mal)
  las reglas del grupo de seguridad: correctas
  el emparejamiento: existía
  2 horas de revisión manual

lo que lo resolvió, en 90 segundos
  la comprobación de siguiente salto
  → dijo que el tráfico iba al cortafuegos y ahí se
    detenía
  la comprobación de flujo permitido
  → dijo que la regla del CORTAFUEGOS, no la del grupo de
    seguridad, lo denegaba

causa
  el emparejamiento entre radios no es transitivo
  el tráfico pasaba por el centro, correctamente
  y el cortafuegos no tenía regla para ese par
  → se había asumido que el emparejamiento bastaba

corrección
  regla en el cortafuegos, y documentación del camino
```

```
  esperado para los 8 flujos entre radios     clase 194
```

El plan de direcciones, ajustado tras contar:

```
cálculo inicial por radio           /24 (256)

cálculo real
  cargas                                          ~60
  puntos privados (27 servicios)                   27
  subredes exclusivas de servicios gestionados
    pasarela de aplicación (mínimo /24 propia)
    bases gestionadas con integración de red        2 × /26
    entorno de contenedores                        1 × /23
  margen de despliegue escalonado                 ×2
  ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
  → /21 por radio, no /24

→ el plan de la clase 193 tuvo que revisarse para Azure
→ y los 4 radios creados con /24 hubo que ampliarlos
  antes de que se agotaran
```

La vigilancia montada:

```
zonas privadas sin enlazar a una red existente   → alerta
nombre de servicio que resuelve a pública        → alerta
reglas de seguridad con origen «cualquiera»      → auditada
rutas creadas fuera del repositorio              → alerta
  → 6 en el primer trimestre; 5 eran temporales de
    diagnóstico que nadie quitó                    ley 25
direcciones libres por subred, alerta al 80 %
estado de la sesión hacia el centro de datos   clase 198
```

El resultado:

```
transferencia de salida                   1.940 €       180 €
servicios con acceso público                   27           1
radios con zonas DNS enlazadas               4/43       43/43
tiempo de diagnóstico de red               2 h 10       9 min
rutas manuales sin registrar                    6           0
subredes agotadas o en riesgo                   4           0
bucles de encaminamiento                        1           0
```

La lección que esta clase deja: los puntos privados estaban creados, se pagaban y el tráfico salía a internet igualmente porque las zonas DNS privadas solo estaban enlazadas a cuatro de cuarenta y tres redes. No falló nada, no saltó ninguna alerta y nadie lo reportó: se descubrió mirando la factura de salida. Y el diagnóstico que costó dos horas revisando reglas a mano se resolvía en noventa segundos con dos comprobaciones que la plataforma ya ofrecía.

Laboratorio guiado

Ejecuta desde la raíz:

```
python classes/part-18-azure-production-architecture/219-hub-spoke-virtual-wan-private-link-y-dns-privado/lab.py
```

El laboratorio selecciona el motor de práctica network y produce labresult.json. El escenario, sus comprobaciones y el artefacto esperado corresponden a esta clase; no requiere credenciales y deja explícito qué debe revalidarse en un sandbox real.

1. Lee exercise.steps y formula una predicción antes de ejecutar.
2. Ejecuta la práctica y verifica todos los elementos de checks.
3. Provoca el caso de negativetest y explica la señal observada.
4. Materializa azure-network en evidence/ usando la plantilla indicada.
5. Para proveedor real, sigue sandbox.requires, registra costo y ejecuta destroy.

Evidencia esperada

El artefacto principal es una topología con rutas, puertos y controles. Además, la entrega debe incluir el comando ejecutado, la salida estructurada y una conclusión que no exceda lo observado.

Reto verificable

Construye azure-network para el caso CloudShop. Incluye una alternativa descartada, un supuesto que pueda falsarse, una prueba de fallo y una decisión de rollback.

■ Criterio de aceptación

- [ ] lab.py termina con código 0 y genera JSON válido.
- [ ] La entrega conecta al menos tres requisitos con mecanismos verificables.
- [ ] Existe una prueba positiva y una prueba negativa con evidencia.
- [ ] Seguridad, costo y operación aparecen como decisiones, no como anexos.
- [ ] Se declara una limitación y una condición que obligaría a revisar el diseño.
- [ ] Otra persona puede repetir el recorrido sin conocimiento tácito.

■■ Errores frecuentes

Síntoma	Causa probable	Corrección
Un punto privado existe y el tráfico sigue saliendo a internet	La zona DNS privada no está enlazada a esa red virtual y el nombre resuelve a la dirección pública	Centraliza las zonas, enlázalas a todas las redes por automatización y comprueba a diario que cada nombre resuelve a dirección privada desde cada red.
El servicio sigue siendo alcanzable desde internet pese al punto privado	Crear el punto privado no desactiva el acceso público	Desactiva el acceso público explícitamente y respáldalo con una política de denegación heredada.
Nada funciona tras aplicar la ruta hacia el cortafuegos	La ruta 0.0.0.0/0 se aplicó también a la subred del propio dispositivo	Deja esa subred con las rutas del sistema y comprueba la asociación con una función de aptitud.
Dos radios no se comunican aunque estén emparejados con el centro	El emparejamiento no es transitivo y falta la regla en el cortafuegos	Documenta el camino esperado de cada flujo entre radios y añade la regla correspondiente.
Una regla de seguridad no surte efecto	Otra regla permisiva con prioridad menor decide antes, o hay grupos aplicados en subred y en interfaz	Aplica en la subred, deja huecos entre prioridades y usa la comprobación de flujo permitido antes de suponer.
Una subred se queda sin direcciones y bloquea el escalado	El dimensionado no contó puntos privados ni las subredes exclusivas que exigen ciertos servicios	Cuenta todos los consumidores y dimensiona con margen, revisando el plan para las particularidades de esta nube.

■ Seguridad, ética y costo

Trabaja con cuentas propias o sandboxes autorizados. No publiques secretos, identificadores reales ni datos personales. Antes de crear recursos pagos define presupuesto, etiquetas y comando de destrucción; después verifica que no queden recursos huérfanos. Los ejemplos locales enseñan contratos, pero no certifican cumplimiento ni disponibilidad de producción.

■ Preguntas de comprobación

1. ¿Por qué el aislamiento entre radios es el estado por defecto y qué hace falta para abrirlo?
2. ¿Qué regla por defecto de los grupos de seguridad sorprende y qué implica?
3. ¿Qué ocurre exactamente cuando falta el enlace de una zona DNS privada?
4. ¿Por qué la subred del cortafuegos no debe llevar la ruta hacia el cortafuegos?
5. ¿Qué dos comprobaciones resuelven la mayoría de los diagnósticos de red en Azure?

Referencias

- Microsoft (2025). Hub-spoke network topology in Azure. <https://learn.microsoft.com/en-us/azure/architecture/networking/architecture/hub-spoke>
- Microsoft (2025). Azure Private Link and private endpoint DNS configuration. <https://learn.microsoft.com/en-us/azure/private-link/private-endpoint-dns>
- Microsoft (2025). Network security groups: default rules and evaluation. <https://learn.microsoft.com/en-us/azure/virtual-network/network-security-groups-overview>
- Microsoft (2025). User-defined routes and forced tunneling. <https://learn.microsoft.com/en-us/azure/virtual-network/virtual-networks-udr-overview>
- Microsoft (2025). Azure Virtual WAN architecture. <https://learn.microsoft.com/en-us/azure/virtual-wan/virtual-wan-about>
- Documentación oficial vigente del servicio implementado; registra URL y fecha de consulta.

Descargar: Parte 18 en PDF · Recorrido de Azure en PDF · Manual integral

Anterior	Índice	Siguiente
← 218 · Entra ID, workload identity, PIM y Conditional Access	Parte 18 · Programa	220 · Bicep, deployment stacks y Azure Verified Modules →

Evaluación — 219 Hub-spoke, Virtual WAN, Private Link y DNS privado

Preguntas

1. Explica el problema que resuelve esta capacidad sin mencionar un proveedor.
2. Identifica una frontera de responsabilidad y su propietario.
3. Compara dos alternativas mediante confiabilidad, seguridad, costo y operación.
4. ¿Qué demuestra el laboratorio y qué no puede demostrar?
5. Propón un caso límite o una prueba de fallo.

Reto verificable

Entrega azure-network con diagrama o configuración, evidencia de ejecución, alternativa descartada, riesgo residual y criterio de rollback.

Criterio de aceptación

- [ ] Resultado reproducible por una segunda persona.
- [ ] Evidencia vinculada a cada afirmación técnica.
- [ ] Prueba negativa o de fallo incluida.
- [ ] Costos y permisos expresados con unidades y alcance.
- [ ] Limitaciones declaradas sin presentar la simulación como producción.

Escala

Nivel	Evidencia
A — competente	Cumple todo y defiende trade-offs con datos.
B — en desarrollo	Cumple lo esencial; falta profundidad en una dimensión.
C — inicial	Artefacto parcial o conclusión sin evidencia suficiente.
0	Sin entrega reproducible.

220 — Bicep, deployment stacks y Azure Verified Modules

Parte: 18 — Azure: arquitectura empresarial y operación en producción Nivel: avanzado · Horas estimadas: 4
Laboratorio: iac · Estado: EXECUTABLECORE

Propósito

Declarar la infraestructura de Azure como código de forma que se pueda desplegar, revisar y —lo que casi nunca se resuelve— retirar. La clase cubre Bicep frente a las plantillas nativas y a las herramientas de terceros, las pilas de despliegue que sí saben borrar lo que sobra, los módulos verificados y cuándo conviene usarlos, y los modos de despliegue con su trampa: el modo completo borra lo que no está en la plantilla, y eso incluye lo que creó otro equipo.

Resultados de aprendizaje

Al finalizar podrás:

1. Elegir entre Bicep y herramientas de terceros con criterio.
2. Estructurar el código por ciclo de vida y por ámbito.
3. Usar pilas de despliegue para gestionar el ciclo de vida completo.
4. Aplicar módulos verificados sin perder control.
5. Evitar los tres errores que borran recursos en producción.

Conceptos centrales

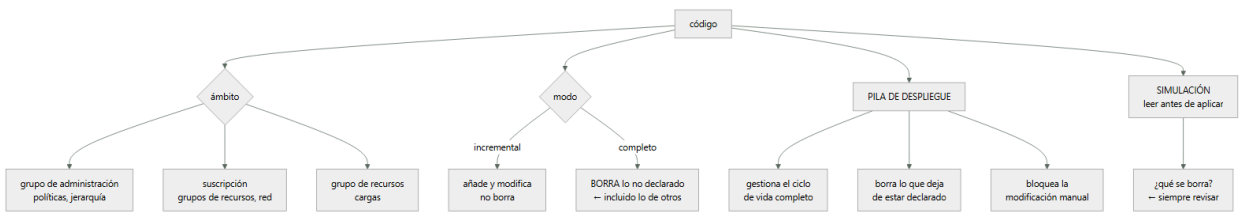
Concepto	Comprensión verificable
Bicep	Lenguaje declarativo que compila a plantillas nativas de Azure, con módulos, tipos y validación previa.
ámbito del despliegue	Nivel al que se despliega: grupo de administración, suscripción, grupo de recursos o inquilino.
modo de despliegue	Incremental añade y modifica; completo además borra lo que no está declarado.
pila de despliegue	Recurso que gestiona un conjunto como unidad, con capacidad de borrar lo que deja de estar declarado.
simulación de cambios	Previsualización de lo que el despliegue creará, modificará o borrará. Se lee antes de aplicar.
módulo verificado	Módulo mantenido por el proveedor con valores por defecto revisados y opciones de seguridad activadas.

Modelo mental

La arquitectura Azure nace del tenant y las políticas heredables, y llega al workload mediante identidades administradas y redes privadas.

Aplicado a esta clase, separa siempre cuatro planos: intención (qué necesita el usuario), configuración (qué declaramos), estado observado (qué existe de verdad) y evidencia (cómo sabemos que cumple). Confundirlos produce diseños que se ven correctos en un diagrama pero fallan al operar.

■ Flujo de razonamiento



Desarrollo

1. Bicep, y cuándo no

Azure tiene su lenguaje propio y también funciona con herramientas multinube. La elección importa menos de lo que parece y más de lo que se cree.

```
BICEP
  + no hay estado que gestionar: el estado es Azure
  + soporte inmediato de recursos nuevos
  + validación y simulación de cambios nativas
  + módulos verificados mantenidos por el proveedor
  − solo Azure
  − la gestión del ciclo de vida completo exige pilas de
    despliegue; el despliegue normal no borra nada

HERRAMIENTA MULTINUBE (Terraform o similar)
  + una sola forma de trabajar para varias nubes
  + el estado permite saber qué gestiona y borrar lo que
    sobra
  + ecosistema de módulos enorme
  − hay que gestionar y proteger el estado    clase 100
  − los recursos nuevos tardan en estar soportados
  − y la portabilidad del CÓDIGO es menor de lo que se
    espera: los recursos son específicos    clase 158
```

Y el criterio honesto:

```
¿la organización usa varias nubes de verdad y el equipo ya
domina una herramienta multinube?
  → úsala también aquí; la coherencia vale más que la
    diferencia técnica

¿es una organización solo Azure, o el equipo empieza?
  → Bicep; menos piezas que operar

¿hay que declarar políticas y jerarquía?
  → cualquiera de las dos sirve, y conviene que sea la
    misma que el resto                        clase 217
```

Y una advertencia sobre la mezcla:

```
usar las dos a la vez sobre los mismos recursos produce
conflictos: cada una cree que gestiona algo que la otra
modifica
→ si conviven, que sea con fronteras claras: quién gestiona
  qué, escrito                                  clase 183
```

Los ámbitos, que hay que entender antes de estructurar:

```
INQUILINO                  grupos de administración
GRUPO DE ADMINISTRACIÓN    políticas, asignaciones
SUSCRIPCIÓN                grupos de recursos, red, roles
GRUPO DE RECURSOS          la mayoría de los recursos

→ un despliegue tiene un ámbito, y desde él puede desplegar
  módulos en ámbitos inferiores
→ estructurar el código por ámbito es lo que permite
  desplegar la jerarquía y las cargas por separado
```

2. Estructura por ciclo de vida

El error de estructura es el mismo de la clase 207: una plantilla enorme con todo.

```
✗ UNA PLANTILLA QUE LO CONTIENE ABSOLUTAMENTE todo
  cada despliegue toca todo
  un fallo bloquea el conjunto
  y los tiempos de despliegue crecen sin control

✓ SEPARADO POR CICLO DE VIDA
  jerarquía y políticas          cambian al trimestre
  red y conectividad             cambian al mes
  datos: bases, almacenes        cambian poco, NUNCA se
                                 borran con el código
  cargas: aplicaciones           cambian a diario
```

```
→ y los datos SIEMPRE en su propio despliegue, con
  bloqueos de borrado                          clase 217
```

Y las reglas que evitan los problemas frecuentes:

```
SIN VALORES INCRUSTADOS POR ENTORNO
  parámetros por entorno, en ficheros separados

NOMBRES DERIVADOS, NO FIJOS
  y con la unicidad global resuelta donde haga falta
  → varios servicios exigen nombre único en todo Azure
  → un nombre fijo funciona en el primer entorno y falla en
    el segundo

SIN SECRETOS EN PARÁMETROS
  referencia al almacén de secretos, no el valor
  → y los valores de salida NUNCA devuelven secretos: el
    historial de despliegues los conserva

IDENTIDADES Y PERMISOS DECLARADOS
  la asignación de papel es un recurso más, y debe estar
  en el código con su ámbito exacto            clase 218
```

Los módulos verificados, que resuelven un problema real:

```
QUÉ SON
  módulos mantenidos por el proveedor, con los valores por
  defecto revisados y las opciones de seguridad activadas

QUÉ RESUELVEN
  exactamente la ley 26: los valores por defecto de los
  recursos están elegidos para que la demostración
  funcione, y estos módulos los cambian por los de
  producción
  → cifrado, sin acceso público, diagnóstico, versión
    mínima de TLS, identidad administrada

CUÁNDO NO USARLOS
  cuando traen mucho más de lo que hace falta y hay que
  entender todo lo que despliegan
  cuando su versión va por detrás de lo que se necesita

→ y en cualquier caso: FIJAR LA VERSIÓN
  un módulo sin versión fija cambia bajo los pies
                                        clase 106, ley 25
```

Y una práctica que ahorra revisiones:

```
la plantilla de servicio nuevo usa los módulos verificados
y ya cumple las políticas de la organización
→ el carril fácil cumple solo                  clase 171
→ y las funciones de aptitud casi nunca fallan
```

3. Modos, pilas y lo que borra recursos

Aquí están los tres errores que borran cosas en producción.

```
MODO INCREMENTAL (el predeterminado)
  crea lo que falta, modifica lo que difiere
  NO borra nada de lo que no está en la plantilla
  → seguro, y deja recursos huérfanos acumulándose
                                                 ley 25

MODO COMPLETO
  además, BORRA todo lo que exista en el grupo de recursos
  y no esté declarado
  → resuelve los huérfanos
  → y borra lo que creó otro equipo en ese grupo
  → y lo que se creó a mano durante un incidente

  ERROR 1   usar modo completo sobre un grupo compartido
  ERROR 2   usar modo completo con una plantilla parcial
            (por ejemplo, tras un error de compilación que
             dejó fuera medio fichero)
```

Y el tercero, el más frecuente:

```
ERROR 3   no leer la simulación de cambios
  la herramienta dice exactamente qué va a crear, modificar
  y BORRAR
  → y en una canalización, ese resultado debe publicarse y
    revisarse; si incluye borrados en producción, se para
  → una puerta automática: «si hay borrados y el entorno es
    producción, exige aprobación»              clase 106
```

Las pilas de despliegue, que resuelven el ciclo de vida sin la brutalidad del modo completo:

```
QUÉ APORTAN
  gestionan un conjunto de recursos como una unidad
  saben qué recursos les pertenecen
  al desplegar, BORRAN o DESVINCULAN lo que deja de estar
    declarado
  y pueden BLOQUEAR la modificación manual de lo que
    gestionan

→ es lo más parecido al estado de una herramienta
  multinube, sin gestionar un fichero de estado

LA DECISIÓN QUE HAY QUE TOMAR AL CREARLAS
  qué hacer con lo que se quita de la plantilla
    borrar          para cargas
    desvincular     para datos ← lo prudente
  y qué bloqueo aplicar a lo gestionado
    ninguno · solo lectura · sin borrado
```

Y la comprobación que hay que hacer antes de usarlas en producción:

```
probar en un entorno inferior
  quitar un recurso de la plantilla y desplegar
  → comprobar que hace lo que se espera
  → y que los datos NO se borran                    ley 22
```

La canalización de infraestructura, con las puertas que hacen falta:

```
1  compilar y validar la plantilla
2  comprobaciones de política: ¿el despliegue cumpliría?
   → hay herramientas que evalúan la política ANTES
   → evita el rechazo en el momento de desplegar
3  SIMULACIÓN de cambios, publicada
4  puerta: si hay borrados en producción → aprobación
5  desplegar con identidad federada, sin secretos
                                              clase 218
6  comprobar que el resultado es el esperado
7  y en caso de fallo, saber cómo se vuelve atrás
```

Y una nota sobre la vuelta atrás:

```
volver atrás en infraestructura NO es volver a desplegar la
versión anterior sin más
  un recurso borrado no se recupera desplegando otra vez
  un cambio de tamaño de base no siempre es reversible
  y algunos cambios exigen recrear
→ por eso los recursos con datos llevan bloqueo y política
  de retención, y el borrado se trata como irreversible
                                              clase 166
```

4. Revisar, probar y retirar

Qué se revisa en un cambio de infraestructura, que no es lo mismo que en un cambio de código:

```
¿qué se borra? ← lo primero
¿qué se recrea? (un cambio de nombre o de ubicación
  destruye y crea)
¿qué ámbito tiene cada asignación de permisos? clase 218
¿hay acceso público en algo?
¿está el diagnóstico enrutado?
¿tiene etiquetas obligatorias?                clase 214
¿cuánto va a costar al mes?
  → algunos recursos son muy caros y se aprueban solos en
    una plantilla
```

Y la última merece automatismo:

```
estimación de coste del cambio, publicada en la revisión
→ hay herramientas que la calculan a partir de la
  simulación
→ y evita la sorpresa del mes siguiente        clase 214
```

Las pruebas de infraestructura, que son distintas de las de código:

```
VALIDACIÓN         la plantilla es correcta
SIMULACIÓN         el cambio es el esperado
DESPLIEGUE EN
  ENTORNO EFÍMERO  se despliega de cero y funciona
  → esto es lo que detecta las dependencias no declaradas
    y los recursos que alguien creó a mano y nadie metió
    en el código                              clase 104
PRUEBAS DE
  COMPORTAMIENTO   ¿el almacén rechaza el acceso público?
                   ¿la subred no llega a internet?
  → las pruebas negativas de las clases 217 a 219
```

Y el desplegar de cero es la más valiosa:

```
si el entorno completo no se puede desplegar desde el
código en un entorno vacío, el código NO describe el
sistema
→ y en un desastre, no sirve                    clase 215
→ conviene hacerlo periódicamente, no una vez
```

La retirada, que es lo que el código como declaración permite y casi nadie usa:

```
quitar el recurso de la plantilla y desplegar
  con pila de despliegue → se borra o desvincula
  con modo incremental   → NO se borra, y queda huérfano

y la comprobación periódica que encuentra los huérfanos
  inventario real frente a lo declarado
  → lo que existe y no está en ningún código es un hallazgo
                                          ley 25, clase 253
```

Y la lista de comprobación de la clase:

- la herramienta elegida es coherente con el resto de la organización
- el código está separado por ciclo de vida y por ámbito
- los datos tienen su propio despliegue, con bloqueos
- no hay valores incrustados por entorno
- los nombres son derivados y resuelven la unicidad global
- no hay secretos en parámetros ni en salidas
- las asignaciones de permisos están en el código, con su ámbito exacto
- los módulos verificados están con versión fija
- el modo completo no se usa sobre grupos compartidos
- la simulación de cambios se publica y se revisa
- hay puerta de aprobación si hay borrados en producción
- las pilas de despliegue se probaron en un entorno inferior
- el entorno se despliega de cero periódicamente
- hay estimación de coste en la revisión
- se comparan periódicamente el inventario real y lo declarado

Y el cierre que enlaza con la clase siguiente: con la base declarada, queda desplegar las cargas. Las tres opciones de cómputo gestionado de Azure —y cuándo conviene cada una— son la materia de la clase 221.

Ejemplo trabajado

CloudShop pasa su infraestructura de Azure a código. Lo que sigue es el despliegue que borró una base de datos de preproducción, la comparación entre inventario real y código declarado, y las pilas de despliegue que resolvieron los huérfanos.

El punto de partida:

```
recursos en el inventario                    14.200
  declarados en algún código                  4.100  29 %
  creados a mano                             10.100  71 %
```

```
plantillas existentes                              31
  con valores incrustados por entorno              24
  con secretos en parámetros                        7
  con nombres fijos                                19
    → y por eso solo funcionaban en un entorno
```

El incidente: el modo completo, semana 3.

```
qué pasó
  un equipo migró su carga a Bicep
  desplegó en modo COMPLETO sobre el grupo de recursos de
  preproducción
  su plantilla declaraba la aplicación y el almacén

  el grupo contenía además
    una base de datos de preproducción, creada a mano
    dos cuentas de almacenamiento de otro equipo
    un espacio de trabajo de análisis

  el modo completo BORRÓ los cuatro

efectos
  la base de datos tenía copia de seguridad: recuperada en
  2 h 40
  las cuentas de almacenamiento del otro equipo: 1 se
  recuperó del borrado reversible, la otra había superado
  la ventana → 11 GB de resultados de pruebas perdidos
  el espacio de análisis: recreado, sin las consultas
  guardadas

tiempo total de recuperación                      6 h
```

Y el análisis:

```
la simulación de cambios SE HABÍA EJECUTADO
y decía claramente «se eliminarán 4 recursos»
nadie la leyó: la canalización la imprimía en un registro
de 400 líneas y pasaba al paso siguiente        ley 15

correcciones
  1  la simulación se publica como resumen legible, con los
     borrados destacados
  2  puerta automática: si hay borrados, exige aprobación
     explícita, en cualquier entorno
  3  el modo completo queda PROHIBIDO por política; se usan
     pilas de despliegue
  4  un grupo de recursos por equipo y por carga: nada
     compartido                                clase 183
  5  bloqueo de borrado en todos los recursos con datos
```

La estructura nueva:

```
infra/
  jerarquia/        ámbito: grupo de administración
    grupos de administración, iniciativas de política
    despliegue trimestral
  conectividad/     ámbito: suscripción
    centro, radios, cortafuegos, zonas DNS privadas
    despliegue mensual
  datos/            ámbito: grupo de recursos
    bases, almacenes, espacios de análisis
    con bloqueo de borrado y política de retención
    despliegue: raro, y siempre con aprobación
  cargas/           ámbito: grupo de recursos
    aplicaciones, planes, identidades, asignaciones
    despliegue diario

parámetros por entorno, en ficheros separados
nombres derivados: {carga}-{entorno}-{region}-{sufijo único}
secretos: referencias al almacén, nunca valores
```

Los módulos verificados, y lo que cambiaron.

```
se migraron 9 tipos de recurso a módulos verificados
con versión fija
```

```
lo que traían activado que las plantillas propias no tenían
  cuenta de almacenamiento
    acceso público                      desactivado
    versión mínima de TLS               1.2
    diagnóstico                         enrutado
    identidad administrada              activada
    clave gestionada por el cliente     opcional, activada
  base de datos
    acceso público                      desactivado
    conexión por punto privado          sí
    auditoría                           activada
    retención de copias                 35 días
  plan de aplicación
    versión mínima de TLS               1.2
    registro de aplicación              activado

→ 14 valores por defecto cambiados sin escribir una línea
→ y el cumplimiento de las iniciativas de la clase 217
  pasó del 62 % al 97 % en las cargas nuevas
                                                    ley 26
```

Y la decisión sobre versiones:

```
los módulos se fijan a una versión concreta
se actualizan una vez al trimestre, revisando el registro
de cambios
→ en la primera actualización, un módulo cambió un valor
  por defecto que recreaba el recurso
→ la simulación lo detectó y se planificó, en vez de
  ocurrir de madrugada                          clase 106
```

Las pilas de despliegue:

```
se adoptaron para cargas y para conectividad

configuración
  cargas         lo que se quita de la plantilla → BORRAR
                 bloqueo de lo gestionado → sin borrado
  conectividad   lo que se quita → BORRAR
                 bloqueo → solo lectura
  datos          NO se usan pilas; despliegue incremental
                 con bloqueos y aprobación

y la prueba, en preproducción
  se quitó un recurso de la plantilla de cargas
  → borrado, correctamente
  se quitó un recurso de la plantilla de datos por error
  → NO se borró, porque datos no usa pilas         ✓

efecto sobre los huérfanos
  recursos huérfanos en cargas         1.140 → 0 en 3 meses
  → al migrar cada carga a pila, lo no declarado se
    revisaba y se borraba o se declaraba
```

La comparación entre inventario y código:

```
se montó una comprobación semanal
  inventario real frente a lo declarado en el repositorio

primera ejecución
  recursos existentes y no declarados            10.100

seis meses después
  recursos existentes y no declarados               410
    de ellos, con excepción registrada              340
      · recursos gestionados por servicios (subredes de
        integración, identidades del sistema)
    sin excepción                                    70
      → 70 hallazgos, revisados uno a uno
      → 41 se declararon, 22 se borraron, 7 resultaron
        ser de un proyecto que nadie recordaba   ley 25
```

El despliegue de cero, ejecutado dos veces:

```
primer intento (mes 4)
```

```
  se desplegó el entorno de preproducción completo en una
  suscripción vacía
  falló en 6 puntos
    · una zona DNS privada que se creaba a mano
    · una asignación de papel al grupo del equipo, que
      existía desde 2022 y no estaba en el código
    · un certificado subido manualmente
    · dos reglas del cortafuegos
    · una cuota que había que pedir con antelación
  tiempo                            3 días con correcciones

segundo intento (mes 7)
  desplegado de cero, sin intervención
  tiempo                                              94 min
  → y ese número es el que se usa en el plan de
    continuidad                                  clase 215
```

El resultado:

```
recursos declarados en código               29 %        97 %
plantillas con secretos                        7           0
plantillas con nombres fijos                  19           0
recursos borrados por error                    4           0
cumplimiento de iniciativas (cargas nuevas) 62 %        97 %
huérfanos en grupos de cargas              1.140           0
tiempo de despliegue de cero             imposible      94 min
valores por defecto corregidos por
  módulos verificados                          —          14
```

La lección que esta clase deja: el incidente que borró cuatro recursos no fue por una herramienta peligrosa: la simulación había dicho exactamente qué iba a borrar, y estaba en un registro de cuatrocientas líneas que nadie leía. Y los módulos verificados corrigieron catorce valores por defecto sin escribir una línea de código, que es la forma más barata que este programa ha encontrado de tratar la ley 26.

Laboratorio guiado

Ejecuta desde la raíz:

```
python classes/part-18-azure-production-architecture/220-bicep-deployment-stacks-y-azure-verified-modules/lab.py
```

El laboratorio selecciona el motor de práctica iac y produce labresult.json. El escenario, sus comprobaciones y el artefacto esperado corresponden a esta clase; no requiere credenciales y deja explícito qué debe revalidarse en un sandbox real.

1. Lee exercise.steps y formula una predicción antes de ejecutar.
2. Ejecuta la práctica y verifica todos los elementos de checks.
3. Provoca el caso de negativetest y explica la señal observada.
4. Materializa azure-bicep-stack en evidence/ usando la plantilla indicada.
5. Para proveedor real, sigue sandbox.requires, registra costo y ejecuta destroy.

Evidencia esperada

El artefacto principal es un plan reproducible sin secretos ni cambios inesperados. Además, la entrega debe incluir el comando ejecutado, la salida estructurada y una conclusión que no exceda lo observado.

Reto verificable

Construye azure-bicep-stack para el caso CloudShop. Incluye una alternativa descartada, un supuesto que pueda falsarse, una prueba de fallo y una decisión de rollback.

■ Criterio de aceptación

- [ ] lab.py termina con código 0 y genera JSON válido.
- [ ] La entrega conecta al menos tres requisitos con mecanismos verificables.

- [ ] Existe una prueba positiva y una prueba negativa con evidencia.
- [ ] Seguridad, costo y operación aparecen como decisiones, no como anexos.
- [ ] Se declara una limitación y una condición que obligaría a revisar el diseño.
- [ ] Otra persona puede repetir el recorrido sin conocimiento tácito.

Errores frecuentes

Síntoma	Causa probable	Corrección
Un despliegue borra recursos que no eran suyos	Modo completo sobre un grupo de recursos compartido	Un grupo por equipo y carga, prohíbe el modo completo y usa pilas de despliegue con la política de borrado decidida.
La simulación avisaba del borrado y nadie lo vio	El resultado se imprimía en un registro larguísimo y la canalización seguía	Publica un resumen legible con los borrados destacados y pon una puerta de aprobación cuando existan.
La plantilla funciona en un entorno y falla en el siguiente	Nombres fijos y valores incrustados por entorno	Deriva los nombres del entorno y resuelve la unicidad global; parámetros en ficheros separados.
Los recursos nuevos no cumplen las políticas de la organización	Se declaran con los valores por defecto del recurso	Usa módulos verificados con versión fija; traen los valores de producción activados.
Se acumulan recursos que nadie declaró ni recuerda	El despliegue incremental nunca borra lo que deja de estar en la plantilla	Usa pilas de despliegue y compara periódicamente el inventario real con lo declarado.
En un desastre, el entorno no se puede reconstruir desde el código	Hay piezas creadas a mano que nunca entraron en el repositorio	Despliega el entorno de cero en una suscripción vacía periódicamente y corrige lo que falte.

Seguridad, ética y costo

Trabaja con cuentas propias o sandboxes autorizados. No publiques secretos, identificadores reales ni datos personales. Antes de crear recursos pagos define presupuesto, etiquetas y comando de destrucción; después verifica que no queden recursos huérfanos. Los ejemplos locales enseñan contratos, pero no certifican cumplimiento ni disponibilidad de producción.

Preguntas de comprobación

1. ¿Qué diferencia hay entre el modo incremental y el completo, y qué riesgo tiene el segundo?
2. ¿Qué aportan las pilas de despliegue frente al despliegue normal?
3. ¿Por qué los módulos verificados son una respuesta directa a la ley 26?
4. ¿Qué se revisa primero en un cambio de infraestructura?
5. ¿Qué detecta desplegar el entorno de cero que ninguna otra prueba detecta?

Referencias

- Microsoft (2025). Bicep documentation. <https://learn.microsoft.com/en-us/azure/azure-resource-manager/bicep/overview>
- Microsoft (2025). Deployment stacks. <https://learn.microsoft.com/en-us/azure/azure-resource-manager/bicep/deployment-stacks>
- Microsoft (2025). ARM template deployment modes. <https://learn.microsoft.com/en-us/azure/azure-resource-manager/templates/deployment-modes>
- Microsoft (2025). Azure Verified Modules. <https://azure.github.io/Azure-Verified-Modules/>

- Microsoft (2025). ARM template what-if operation. <https://learn.microsoft.com/en-us/azure/azure-resource-manager/templates/deploy-what-if>
- Documentación oficial vigente del servicio implementado; registra URL y fecha de consulta.

Descargar: Parte 18 en PDF · Recorrido de Azure en PDF · Manual integral

Anterior	Índice	Siguiente
← 219 · Hub-spoke, Virtual WAN, Private Link y DNS privado	Parte 18 · Programa	221 · App Service, Functions y Container Apps en producción →

Evaluación — 220 Bicep, deployment stacks y Azure Verified Modules

Preguntas

1. Explica el problema que resuelve esta capacidad sin mencionar un proveedor.
2. Identifica una frontera de responsabilidad y su propietario.
3. Compara dos alternativas mediante confiabilidad, seguridad, costo y operación.
4. ¿Qué demuestra el laboratorio y qué no puede demostrar?
5. Propón un caso límite o una prueba de fallo.

Reto verificable

Entrega azure-bicep-stack con diagrama o configuración, evidencia de ejecución, alternativa descartada, riesgo residual y criterio de rollback.

Criterio de aceptación

- [ ] Resultado reproducible por una segunda persona.
- [ ] Evidencia vinculada a cada afirmación técnica.
- [ ] Prueba negativa o de fallo incluida.
- [ ] Costos y permisos expresados con unidades y alcance.
- [ ] Limitaciones declaradas sin presentar la simulación como producción.

Escala

Nivel	Evidencia
A — competente	Cumple todo y defiende trade-offs con datos.
B — en desarrollo	Cumple lo esencial; falta profundidad en una dimensión.
C — inicial	Artefacto parcial o conclusión sin evidencia suficiente.
0	Sin entrega reproducible.

221 — App Service, Functions y Container Apps en producción

Parte: 18 — Azure: arquitectura empresarial y operación en producción Nivel: avanzado · Horas estimadas: 4 Laboratorio: serverless · Estado: EXECUTABLECORE

Propósito

Elegir entre las tres opciones de cómputo gestionado de Azure y configurarlas para producción, que es distinto de configurarlas para que funcionen. La clase compara servicio de aplicaciones, funciones y aplicaciones de contenedor por lo que de verdad las diferencia —modelo de escalado, arranque en frío, integración de red y coste—, y desarrolla los ajustes que separan un despliegue estable de uno que corta peticiones y se queda sin conexiones.

Resultados de aprendizaje

Al finalizar podrás:

1. Elegir entre las tres opciones con criterios comprobables.
2. Integrar las cargas en la red privada, de entrada y de salida.
3. Configurar escalado, arranque en frío y límites sin desperdiciar.
4. Desplegar con ranuras o revisiones, sin cortar peticiones.
5. Evitar el agotamiento de puertos y de conexiones bajo carga.

Conceptos centrales

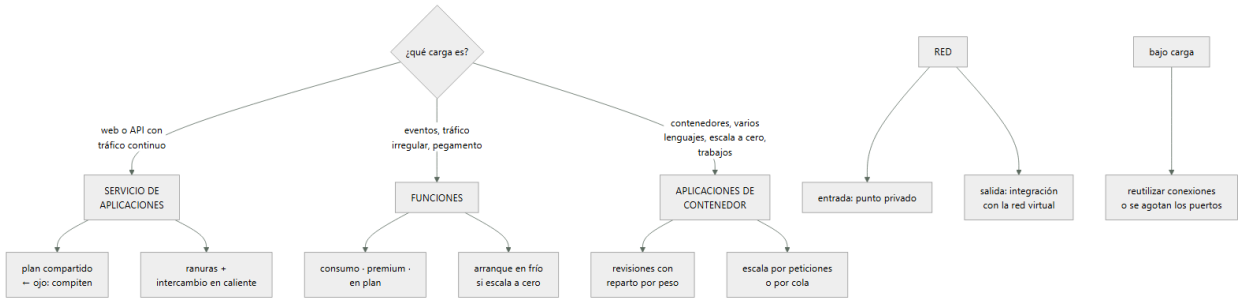
Concepto	Comprensión verificable
plan de servicio	Conjunto de recursos de cómputo compartido por varias aplicaciones. Es donde se paga y donde se compite.
ranura de despliegue	Copia de la aplicación con su propia dirección, que se intercambia con producción en caliente.
intercambio en caliente	Cambio de la ranura de preparación a producción tras calentarla, sin reiniciar.
plan de consumo	Modelo de facturación por ejecución, con escalado a cero y arranque en frío.
integración de red saliente	Mecanismo que hace que el tráfico de salida de la aplicación pase por la red virtual.
agotamiento de puertos	Falta de puertos efímeros por no reutilizar conexiones. Produce fallos intermitentes bajo carga.

Modelo mental

La arquitectura Azure nace del tenant y las políticas heredables, y llega al workload mediante identidades administradas y redes privadas.

Aplicado a esta clase, separa siempre cuatro planos: intención (qué necesita el usuario), configuración (qué declaramos), estado observado (qué existe de verdad) y evidencia (cómo sabemos que cumple). Confundirlos produce diseños que se ven correctos en un diagrama pero fallan al operar.

■ Flujo de razonamiento



Desarrollo

1. Tres opciones, y cómo elegir

Las tres ejecutan código gestionado y se diferencian en el modelo de escalado, en lo que hay que empaquetar y en el coste.

```
SERVICIO DE APLICACIONES
  aplicaciones web y APIs con proceso continuo
  + ranuras de despliegue con intercambio en caliente
  + escalado por reglas, con instancias siempre encendidas
  + integración de red madura
  – se paga el plan aunque no haya tráfico
  – varias aplicaciones en un plan COMPITEN por sus
    recursos                                  ← ver abajo

FUNCIONES
  para eventos, trabajos programados y pegamento
  + escalado por evento; en consumo, escala a cero
  + integración directa con colas, temas y almacenes
  – arranque en frío si escala a cero
  – los planes que lo evitan cuestan como una instancia

APLICACIONES DE CONTENEDOR
  contenedores sin gestionar orquestador
  + escala a cero y escalado por peticiones o por cola
  + revisiones con reparto de tráfico por peso
  + trabajos por lotes y procesos de larga duración
  – menos control que un orquestador completo
  – arranque en frío si escala a cero
```

Y el criterio, en preguntas:

```
¿es una web o API con tráfico continuo y en un lenguaje
 soportado?
  → servicio de aplicaciones

¿es código que reacciona a eventos y no está siempre
 activo?
  → funciones

¿hay contenedores, varios lenguajes, o hace falta escalar a
 cero con reparto de tráfico?
  → aplicaciones de contenedor

¿hace falta control fino del planificador, operadores o
 cargas heterogéneas?
  → Kubernetes                                clase 222
```

Y el error de elección más frecuente:

```
usar funciones para una API con tráfico continuo
  → se paga por ejecución algo que estaría más barato en un
    plan
  → y se sufre arranque en frío sin necesidad
→ el modelo de facturación por ejecución gana cuando el
  tráfico es IRREGULAR, no cuando es alto y constante
```

El plan compartido, que produce incidentes difíciles de diagnosticar:

```
varias aplicaciones en el mismo plan comparten CPU, memoria
```

```
y puertos
→ una aplicación con una fuga de memoria afecta a las
  demás
→ y el escalado del plan lo dispara cualquiera de ellas

regla   producción, en su propio plan
        y las aplicaciones de un mismo plan, del mismo
        equipo y con perfiles parecidos          clase 183
```

2. Red: entrada privada y salida integrada

Estos servicios nacen con dirección pública. Meterlos en la red privada exige dos cosas distintas que se confunden.

```
ENTRADA — punto privado
  la aplicación se alcanza por una dirección de tu red
  y el acceso público se desactiva
  → y hace falta la zona DNS privada enlazada a todas las
    redes que la llamen                       clase 219

SALIDA — integración con la red virtual
  el tráfico SALIENTE de la aplicación pasa por una subred
  → sin esto, la aplicación sale a internet aunque tenga
    punto privado de entrada
  → y por tanto no alcanza recursos privados y no pasa por
    el cortafuegos                            clase 200
```

Y la confusión que produce incidentes:

```
«ya tiene punto privado, está en la red»
  → NO: la entrada es privada y la salida sigue siendo
    pública
  → hay que configurar las dos

y el síntoma
  la aplicación no puede llegar a la base de datos privada
  o el tráfico hacia el almacén sale a internet y se paga
                                        clase 219, ley 26
```

Y los detalles de la integración de salida:

```
exige una subred DEDICADA y delegada al servicio
  con tamaño suficiente: cada instancia consume direcciones
  → dimensionar por el máximo del escalado    clase 193

y hay que decidir si CUANTO tráfico salga va por la red
  → por defecto, solo el privado; el resto sale directo
  → para forzar la salida por el cortafuegos hay que
    activarlo explícitamente
```

Y una comprobación que hay que hacer:

```
desde la aplicación, resolver el nombre de un servicio con
punto privado
  → debe devolver dirección privada
y comprobar la dirección de salida observada
  → debe ser la del cortafuegos, no una pública del
    servicio                                  ley 22
```

3. Escalado, arranque en frío y despliegue

El escalado, con la misma lección de la clase 212:

```
LA SEÑAL
  ✗ CPU, en servicios que esperan a la red
  ✓ peticiones en cola, longitud de cola, o métrica propia
  → en aplicaciones de contenedor, el escalado por cola es
    directo y es el que corresponde a un trabajador
                                              clase 210

EL RETRASO
  detección + decisión + arranque de instancia
  → minutos, igual que en la clase 212
  → y por eso el margen y el escalado programado siguen
    haciendo falta

MÍNIMO DE INSTANCIAS
```

```
  con escala a cero, la primera petición paga el arranque
  → para el camino crítico, mínimo 1 o más
  → y ahí desaparece la ventaja de coste de escalar a cero
```

El arranque en frío, con las palancas de siempre:

```
paquete pequeño y dependencias mínimas
trabajo fuera de la inicialización, salvo lo reutilizable
instancias siempre listas (se pagan)
calentamiento antes de recibir tráfico
  → y esto último es lo que hacen bien las ranuras
```

El despliegue sin cortar peticiones, que aquí tiene mecanismos propios:

```
RANURAS DE DESPLIEGUE (servicio de aplicaciones)
  se despliega en una ranura de preparación
  se CALIENTA: se piden rutas que fuercen la
    inicialización
  se intercambia con producción
  → el intercambio cambia el enrutamiento, no reinicia
  → y si algo va mal, se intercambia de vuelta en segundos

  y el detalle que decide si funciona
    la configuración con marca de «ranura» NO se
    intercambia
    → así la ranura de preparación apunta a la base de
      preparación, y producción a la suya
    → olvidarlo hace que preparación escriba en producción

REVISIONES (aplicaciones de contenedor)
  cada despliegue crea una revisión
  el tráfico se reparte por peso entre revisiones
  → despliegue escalonado nativo, con vuelta atrás
    inmediata                                clase 102

Y EN AMBOS CASOS
  parada elegante: atender la señal de terminación
  drenaje mayor que la petición más larga    clase 212
```

Y una advertencia sobre el calentamiento:

```
intercambiar sin calentar traslada el arranque en frío a
los primeros usuarios de producción
→ y el síntoma es «tras cada despliegue, unos minutos de
  latencia alta»
```

4. Lo que rompe bajo carga

El agotamiento de puertos efímeros es el fallo característico de estos servicios y se diagnostica mal.

```
QUÉ PASA
  cada conexión saliente consume un puerto efímero
  el puerto queda ocupado un tiempo tras cerrarse
  el número de puertos por instancia es limitado

  si el código abre una conexión NUEVA en cada llamada
  → bajo carga se agotan
  → y aparecen fallos intermitentes de conexión, sin
    patrón claro

LA CAUSA CASI SIEMPRE
  crear el cliente HTTP dentro de la función o del
  controlador, en vez de reutilizarlo
  → cada petición abre una conexión nueva

LA CORRECCIÓN
  cliente único y reutilizado, con agrupación de conexiones
  y para bases de datos, agrupación configurada con límite

Y EL DIAGNÓSTICO
  la métrica de conexiones salientes por instancia
  → si crece linealmente con el tráfico y no se estabiliza,
    no se están reutilizando
```

Y el otro fallo característico, ya conocido:

```
CONEXIONES A LA BASE DE DATOS
  cada instancia abre su propia agrupación
  20 instancias × 100 conexiones = 2.000
  → y la base admite 400                        clase 207
→ el límite de la agrupación se calcula por instancia,
  contando el máximo del escalado
```

Lo que hay que vigilar:

```
peticiones, errores y latencia por percentil  clase 211
instancias activas frente al máximo
conexiones salientes por instancia
tiempo de arranque y peticiones que lo pagan
uso de CPU y memoria, y REINICIOS por memoria
y en funciones: ejecuciones, retrasos de disparador y
  mensajes pendientes
```

Y una alerta específica de estos servicios:

```
«la aplicación se ha reiniciado N veces en la última hora»
  → los reinicios silenciosos por memoria o por fallo de
    la comprobación son la señal que más se pierde
                                                  ley 13
```

Y la lista de comprobación de la clase:

- la opción elegida corresponde al perfil de la carga
- producción no comparte plan con otras cargas
- la entrada es por punto privado y el acceso público está desactivado
- la salida está integrada con la red virtual
- la subred de integración está dimensionada por el máximo
- se ha comprobado que la salida pasa por el cortafuegos
- el escalado usa una señal distinta de la CPU si procede
- el mínimo de instancias cubre el camino crítico
- hay ranuras o revisiones, con calentamiento previo
- la configuración específica de ranura está marcada
- la aplicación atiende la señal de terminación
- los clientes HTTP y de base se reutilizan
- el límite de agrupación se calculó por instancia
- hay alerta de reinicios y de conexiones salientes

Y el cierre que enlaza con la clase siguiente: cuando la carga exige control fino, operadores o cargas heterogéneas, aparece Kubernetes gestionado con sus particularidades de identidad y de entrada en esta nube. Es la materia de la clase 222.

Ejemplo trabajado

CloudShop despliega su plataforma en Azure. Lo que sigue es la elección de opción por carga, el incidente de la ranura que escribió en producción, y el agotamiento de puertos que se diagnosticó como «problema de red» durante tres semanas.

La elección, carga por carga:

```
carga                perfil               elección
api de pedidos       continuo, 900 pet/s  servicio de
                                          aplicaciones
web de tienda        continuo             servicio de
                                          aplicaciones
procesador de
  eventos            por cola, irregular  aplicaciones de
                                          contenedor
trabajos programados 12/día               funciones,
                                          plan de consumo
notificaciones       picos, escala a cero funciones,
                                          plan de consumo
informes nocturnos   1/día, 40 min        trabajo de
                                          aplicaciones de
                                          contenedor
búsqueda             contenedor propio,
                     imagen específica    aplicaciones de
                                          contenedor

y lo que se descartó
```

```
  poner la api de pedidos en funciones
  → estimación: 1.840 €/mes en consumo frente a 610 € en
    un plan, con tráfico continuo
  → y arranque en frío innecesario
```

Incidente 1 · La ranura que escribió en producción.

```
síntoma   tras un despliegue de rutina, aparecieron 41
          pedidos de prueba en la base de producción
          con importes de 0,01 € y direcciones falsas

diagnóstico
  la cadena de conexión a la base estaba en la
  configuración de la aplicación
  NO estaba marcada como específica de ranura
  → al intercambiar, la configuración de preparación viajó
    a producción y la de producción a preparación
  → durante 8 minutos, antes de que alguien lo notara, las
    pruebas de humo de preparación escribieron en
    producción

corrección
  toda la configuración que apunta a recursos externos,
  marcada como específica de ranura
  y mejor: las cadenas se resuelven por referencia al
  almacén de secretos, con el nombre del secreto distinto
  por ranura
  y una comprobación previa al intercambio: la ranura de
  preparación debe apuntar a la base de preparación

y lo que se limpió
  41 pedidos borrados, con registro de la incidencia
```

Incidente 2 · «Fallos intermitentes de red», tres semanas.

```
síntoma   entre las 11:00 y las 13:00, un 2-4 % de las
          llamadas de la api de pedidos al servicio de
          precios fallaban con error de conexión
          fuera de esas horas, nunca

lo que se revisó (mal)
  reglas del grupo de seguridad          correctas
  rutas y cortafuegos                    correctos
  capturas de paquetes                   sin nada raro
  se abrió incidencia con el proveedor   3 semanas

lo que lo resolvió
  la métrica de conexiones salientes por instancia

    hora        peticiones/s   conexiones salientes
    09:00          210              240
    10:00          390              450
    11:00          720              830
    12:00        1.100            1.270   ← límite ~1.280
    14:00          380              440

  → las conexiones crecían LINEALMENTE con las peticiones
  → no se estaban reutilizando

causa
  el código creaba un cliente HTTP nuevo en cada llamada
  al servicio de precios
  cada uno abría su conexión, que quedaba ocupada tras
  cerrarse
  → a partir de cierto tráfico, no había puertos libres

corrección
  un cliente único reutilizado, con agrupación configurada
  conexiones salientes en el pico    1.270 → 46
  fallos intermitentes                 2-4 % → 0

y lo que enseñó
  la métrica que resolvió el caso estaba disponible desde
  el primer día y no estaba en ningún panel      ley 15
```

```
  → se añadió al panel estándar de servicio     clase 211
```

La red, montada en dos partes:

```
ENTRADA
  punto privado para la api y para la web
  acceso público desactivado
  zonas DNS privadas enlazadas a las 43 redes clase 219
  el tráfico público entra por la pasarela de aplicación,
  que sí es pública, con filtrado

SALIDA
  integración con la red virtual, subred dedicada /24 por
  entorno
  → dimensionada para el máximo de 30 instancias
  encaminamiento de CUANTO tráfico salga por la red
  → así pasa por el cortafuegos y se registra

y la comprobación que se añadió
  desde la aplicación, resolver el nombre de la base
  → debe dar dirección privada
  y consultar la dirección de salida observada
  → debe ser la del cortafuegos
  → en la primera ejecución, la web daba una dirección
    pública: le faltaba activar el encaminamiento completo
                                                    ley 22
```

El escalado y el despliegue:

```
api de pedidos (servicio de aplicaciones)
  escalado por peticiones en cola, no por CPU
  mínimo 3 instancias, máximo 30
  escalado programado: +8 instancias a las 10:30 en
    campañas
  ranuras: preparación y producción
    calentamiento: 6 rutas pedidas antes de intercambiar
    y tras corregir el incidente 1, comprobación previa

  peticiones con latencia alta tras despliegue
    antes del calentamiento                    ~340
    después                                       0

procesador de eventos (aplicaciones de contenedor)
  escalado por longitud de cola: 1 réplica por cada 50
    mensajes pendientes, máximo 40
  mínimo 1 réplica: la cola nunca está vacía mucho tiempo
  revisiones con reparto: 10 % durante 10 min, luego 100 %
  vuelta atrás: cambiar el peso, 15 segundos

notificaciones (funciones, consumo)
  escala a cero; arranque en frío de 400-900 ms
  aceptable: el envío es asíncrono           clase 210
  y se midió antes de aceptarlo
```

El coste, comparado:

```
api de pedidos (plan P1v3 × 3-30)             610 €    720 €
web de tienda (plan compartido con api)         —        —
  → se separó: comparten plan y compiten
  → plan propio                                         310 €
procesador de eventos                         280 €    240 €
funciones (consumo)                            90 €     71 €
trabajos de informes                           40 €     38 €
pasarela de aplicación                        320 €    320 €
puntos privados                                —       190 €  ←
inspección del cortafuegos                     —       410 €  ←
[redacted]
total                                       1.340 €  2.299 €

y las dos partidas no estimadas son las de siempre:
lo que cuesta poner las cargas en la red privada
                                          clase 216, ley 26
```

El resultado:

```
fallos intermitentes de conexión            2-4 %          0
conexiones salientes en el pico             1.270          46
pedidos de prueba en producción                41           0
peticiones lentas tras despliegue             340           0
aplicaciones compartiendo plan con
  producción                                    2           0
salida que no pasa por el cortafuegos           1           0
tiempo de vuelta atrás                    redesplegar     15 s
```

La lección que esta clase deja: tres semanas de incidencia con el proveedor y capturas de paquetes por un problema que no era de red: el código abría una conexión nueva en cada llamada y agotaba los puertos efímeros a partir de cierto tráfico. La métrica que lo resolvía estaba disponible desde el primer día. Y el incidente más embarazoso —cuarenta y un pedidos de prueba en producción— lo causó una configuración que no estaba marcada como específica de ranura, que es exactamente el tipo de detalle que funciona hasta el día del intercambio.

Laboratorio guiado

Ejecuta desde la raíz:

```
python classes/part-18-azure-production-architecture/221-app-service-functions-y-container-apps-en-produccion/lab.py
```

El laboratorio selecciona el motor de práctica serverless y produce labresult.json. El escenario, sus comprobaciones y el artefacto esperado corresponden a esta clase; no requiere credenciales y deja explícito qué debe revalidarse en un sandbox real.

1. Lee exercise.steps y formula una predicción antes de ejecutar.
2. Ejecuta la práctica y verifica todos los elementos de checks.
3. Provoca el caso de negativetest y explica la señal observada.
4. Materializa azure-app-platform en evidence/ usando la plantilla indicada.
5. Para proveedor real, sigue sandbox.requires, registra costo y ejecuta destroy.

Evidencia esperada

El artefacto principal es una función con límites, reintentos e idempotencia. Además, la entrega debe incluir el comando ejecutado, la salida estructurada y una conclusión que no exceda lo observado.

Reto verificable

Construye azure-app-platform para el caso CloudShop. Incluye una alternativa descartada, un supuesto que pueda falsarse, una prueba de fallo y una decisión de rollback.

■ Criterio de aceptación

- [ ] lab.py termina con código 0 y genera JSON válido.
- [ ] La entrega conecta al menos tres requisitos con mecanismos verificables.
- [ ] Existe una prueba positiva y una prueba negativa con evidencia.
- [ ] Seguridad, costo y operación aparecen como decisiones, no como anexos.
- [ ] Se declara una limitación y una condición que obligaría a revisar el diseño.
- [ ] Otra persona puede repetir el recorrido sin conocimiento tácito.

Errores frecuentes

Síntoma	Causa probable	Corrección
Fallos intermitentes de conexión que aparecen solo con tráfico alto	El código crea un cliente nuevo por llamada y agota los puertos efímeros	Reutiliza un cliente único con agrupación de conexiones y vigila las conexiones salientes por instancia.
La aplicación no alcanza recursos privados pese a tener punto privado	El punto privado resuelve la entrada; la salida sigue siendo pública	Configura además la integración de salida con la red virtual y comprueba la dirección de salida observada.
Tras un intercambio de ranuras, la preparación escribe en producción	La configuración con la cadena de conexión no estaba marcada como específica de ranura	Marca toda la configuración que apunte a recursos externos y comprueba antes del intercambio a qué apunta cada ranura.
Tras cada despliegue hay unos minutos de latencia alta	Se intercambia sin calentar y el arranque en frío lo pagan los usuarios	Pide rutas que fuercen la inicialización en la ranura de preparación antes de intercambiar.
Una aplicación con problemas degrada a otras sin relación	Comparten plan de servicio y por tanto recursos	Producción en su propio plan; agrupa solo aplicaciones del mismo equipo con perfiles parecidos.
Se paga mucho por una API con tráfico continuo	Se eligió el modelo de facturación por ejecución para una carga constante	El pago por ejecución gana con tráfico irregular; con tráfico alto y continuo, un plan sale más barato y evita arranques en frío.

Seguridad, ética y costo

Trabaja con cuentas propias o sandboxes autorizados. No publiques secretos, identificadores reales ni datos personales. Antes de crear recursos pagos define presupuesto, etiquetas y comando de destrucción; después verifica que no queden recursos huérfanos. Los ejemplos locales enseñan contratos, pero no certifican cumplimiento ni disponibilidad de producción.

Preguntas de comprobación

1. ¿Qué pregunta decide entre las tres opciones de cómputo gestionado?
2. ¿Qué diferencia hay entre el punto privado de entrada y la integración de salida?
3. ¿Qué configuración no debe intercambiarse al usar ranuras y por qué?
4. ¿Cuál es la causa habitual del agotamiento de puertos y cómo se detecta?
5. ¿Por qué el calentamiento previo al intercambio es parte del despliegue?

Referencias

- Microsoft (2025). Azure App Service deployment slots. <https://learn.microsoft.com/en-us/azure/app-service/deploy-staging-slots>
- Microsoft (2025). Azure Functions hosting options. <https://learn.microsoft.com/en-us/azure/azure-functions/functions-scale>
- Microsoft (2025). Azure Container Apps revisions and traffic splitting. <https://learn.microsoft.com/en-us/azure/container-apps/revisions>
- Microsoft (2025). Integrate your app with an Azure virtual network. <https://learn.microsoft.com/en-us/azure/app-service/overview-vnet-integration>
- Microsoft (2025). Troubleshooting SNAT port exhaustion. <https://learn.microsoft.com/en-us/azure/load-balancer/troubleshoot-outbound-connection>
- Documentación oficial vigente del servicio implementado; registra URL y fecha de consulta.

Descargar: Parte 18 en PDF · Recorrido de Azure en PDF · Manual integral

Evaluación — 221 App Service, Functions y Container Apps en producción

Preguntas

1. Explica el problema que resuelve esta capacidad sin mencionar un proveedor.
2. Identifica una frontera de responsabilidad y su propietario.
3. Compara dos alternativas mediante confiabilidad, seguridad, costo y operación.
4. ¿Qué demuestra el laboratorio y qué no puede demostrar?
5. Propón un caso límite o una prueba de fallo.

Reto verificable

Entrega azure-app-platform con diagrama o configuración, evidencia de ejecución, alternativa descartada, riesgo residual y criterio de rollback.

Criterio de aceptación

- [ ] Resultado reproducible por una segunda persona.
- [ ] Evidencia vinculada a cada afirmación técnica.
- [ ] Prueba negativa o de fallo incluida.
- [ ] Costos y permisos expresados con unidades y alcance.
- [ ] Limitaciones declaradas sin presentar la simulación como producción.

Escala

Nivel	Evidencia
A — competente	Cumple todo y defiende trade-offs con datos.
B — en desarrollo	Cumple lo esencial; falta profundidad en una dimensión.
C — inicial	Artefacto parcial o conclusión sin evidencia suficiente.
0	Sin entrega reproducible.

222 — AKS, workload identity, ingress y GitOps

Parte: 18 — Azure: arquitectura empresarial y operación en producción Nivel: avanzado · Horas estimadas: 4
Laboratorio: kubernetes · Estado: EXECUTABLECORE

Propósito

Operar Kubernetes gestionado en Azure con las particularidades que lo diferencian de lo visto en la clase 213: la identidad de carga federada con el directorio, el modelo de red —que en esta nube tiene una decisión irreversible sobre el consumo de direcciones—, la entrada gestionada y la reconciliación desde el repositorio. Y la pregunta de siempre antes de empezar: ¿hace falta, o basta con lo de la clase 221?

Resultados de aprendizaje

Al finalizar podrás:

1. Decidir si el clúster compensa frente a las opciones gestionadas.
2. Elegir el modelo de red conociendo su consumo de direcciones.
3. Dar identidad a las cargas con federación, sin secretos.
4. Configurar la entrada gestionada con certificados y filtrado.
5. Planificar actualizaciones de versión y de complementos.

Conceptos centrales

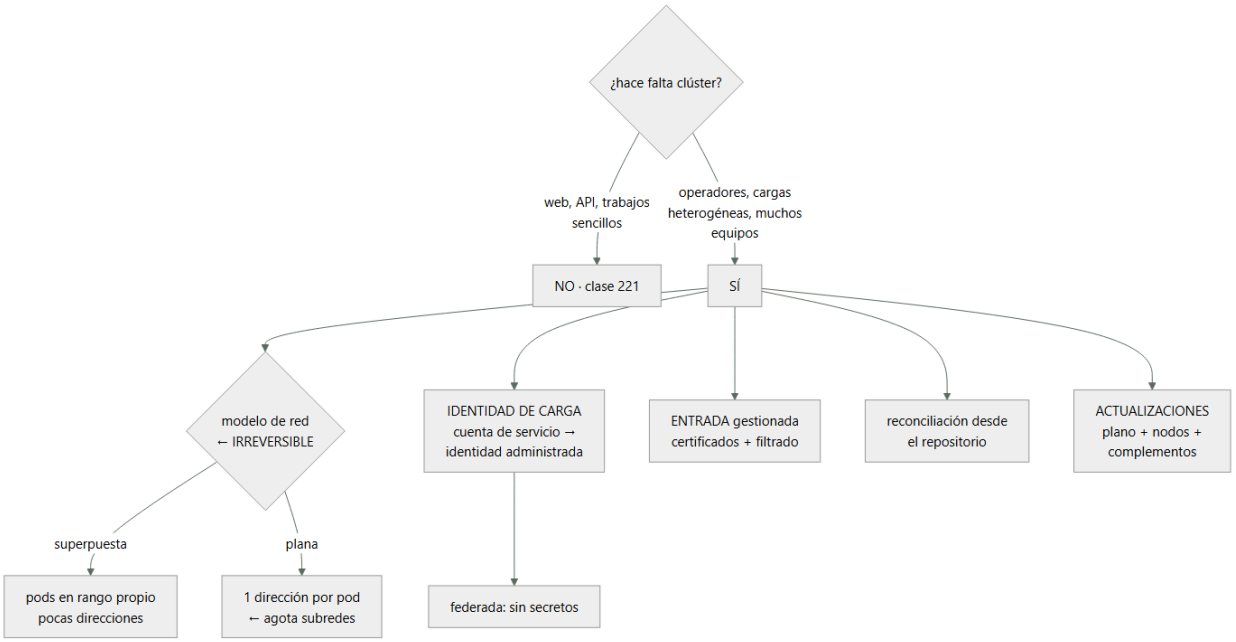
Concepto	Comprensión verificable
identidad de carga	Cuenta de servicio del clúster federada con una identidad administrada del directorio. Sin secretos.
modelo de red superpuesta	Los pods usan un rango propio no enrutado; solo los nodos consumen direcciones de la subred.
modelo de red plana	Cada pod recibe una dirección de la subred. Consume muchísimas direcciones.
controlador de entrada gestionado	Entrada operada por la plataforma, con certificados y filtrado integrados.
grupo de nodos	Conjunto de nodos con el mismo tamaño y configuración. Se actualizan y escalan por grupo.
canal de actualización	Política que decide cuándo se aplican versiones nuevas de plano de control y de nodos.

Modelo mental

La arquitectura Azure nace del tenant y las políticas heredables, y llega al workload mediante identidades administradas y redes privadas.

Aplicado a esta clase, separa siempre cuatro planos: intención (qué necesita el usuario), configuración (qué declaramos), estado observado (qué existe de verdad) y evidencia (cómo sabemos que cumple). Confundirlos produce diseños que se ven correctos en un diagrama pero fallan al operar.

■ Flujo de razonamiento



Desarrollo

1. ¿Hace falta, y con qué configuración?

La pregunta de la clase 213 sigue siendo la primera, y en Azure tiene una respuesta adicional: las aplicaciones de contenedor cubren muchos casos que antes exigían clúster.

```
NO HACE FALTA SI
  son contenedores web y trabajadores por cola
  → las aplicaciones de contenedor lo hacen con escala a
    cero y reparto por revisión                clase 221
  el equipo es pequeño
  no hay cargas con requisitos especiales

SÍ COMPENSA SI
  hay operadores o recursos propios que gestionar
  hay cargas heterogéneas: GPU, procesos largos, lotes con
    prioridades
  varios equipos necesitan autonomía sobre una base común
  o ya se opera Kubernetes en otra nube y la coherencia
    vale                                       clase 158
```

Y las decisiones de configuración que se toman al crear y cuestan corregir:

```
1  MODELO DE RED           ← irreversible sin recrear
2  MODO DE IDENTIDAD       ← se puede migrar, con trabajo
3  PLANO DE CONTROL PRIVADO O PÚBLICO
4  GRUPOS DE NODOS: uno de sistema, y los de carga aparte
5  CANAL DE ACTUALIZACIÓN
```

El modelo de red, que es la decisión irreversible:

```
RED PLANA (cada pod una dirección de la subred)
  + los pods son alcanzables directamente desde la red
  + útil si algo externo tiene que llamar a un pod concreto
  − CONSUME MUCHÍSIMAS DIRECCIONES
    dimensionado = nodos máximos × pods por nodo
    40 nodos × 110 pods = 4.400 direcciones
    → una subred /22 se queda corta
  − y el número de pods por nodo se fija al crear el grupo

RED SUPERPUESTA (pods en rango propio)
  + los nodos consumen direcciones; los pods, no
    40 nodos = 40 direcciones
  + el rango de pods puede reutilizarse entre clústeres
  − los pods no son alcanzables directamente desde fuera
    → hay que salir por servicios y por la entrada
```

```
→ la superpuesta es la elección razonable salvo requisito
  concreto                                    clase 193
→ y cambiar de una a otra exige recrear el clúster
```

Y una decisión relacionada:

```
el plano de control puede ser público o privado
  privado    solo alcanzable desde la red; más seguro
  y entonces la canalización debe alcanzarlo
    → agente autoalojado en la red, o punto privado del
      plano de control
  → decidirlo antes evita rehacer la canalización
```

2. Identidad de carga, sin secretos

El mecanismo es el mismo de la clase 213, con las piezas de Azure.

```
CÓMO FUNCIONA
  el clúster expone un emisor de testigos
  se declara una CREDENCIAL FEDERADA en una identidad
    administrada, atada a
      el emisor del clúster
      el ESPACIO DE NOMBRES
      y el NOMBRE de la cuenta de servicio
  la cuenta de servicio se anota con el identificador de
    esa identidad
  el pod recibe un testigo y lo intercambia

→ y los permisos se asignan a esa identidad, en el ámbito
  del recurso                                 clase 218
```

Y los patrones que hay que dejar atrás:

```
✗ IDENTIDAD DEL NODO
  permisos en la identidad de la máquina
  → todos los pods del nodo los tienen
  → y hay que bloquear el acceso de los pods al servicio
    de metadatos, o los piden aunque tengan la suya

✗ SECRETOS DE KUBERNETES CON CREDENCIALES
  un objeto de tipo secreto está codificado, no cifrado
  → activar el cifrado del almacén del plano de control
  → y para secretos reales, obtenerlos del almacén externo
    en ejecución con un controlador, no guardarlos
```

Y el error de configuración que repite el de las clases 206 y 218:

```
la credencial federada debe atar emisor, espacio de nombres
Y cuenta de servicio
✗ atar solo el emisor
  → cualquier cuenta de servicio de cualquier espacio de
    nombres obtiene esa identidad

prueba negativa
  desde un pod de otro espacio de nombres, intentar obtener
  el testigo y usarlo
  → debe fallar                                 ley 22
```

Y la separación por equipos, con lo que aporta cada pieza:

```
espacio de nombres por equipo o carga
cuotas de recursos por espacio            clase 213
política de admisión, con mensajes que expliquen
  → qué imágenes se admiten: del registro propio y firmadas
  → sin privilegios, sin red del anfitrión
  → etiquetas obligatorias                  clase 214
política de red con denegación por defecto
  → y hay que activarla al crear: por defecto, todo pod
    habla con todo pod                        ley 26
```

3. Entrada, salida y complementos

La entrada, con la elección que hay que hacer:

```
CONTROLADOR GESTIONADO POR LA PLATAFORMA
  + certificados, filtrado y escalado integrados
```

```
  + menos que operar
  − menos control de configuración fina

CONTROLADOR PROPIO EN EL CLÚSTER
  + control total, y portable a otras nubes  clase 158
  − hay que operarlo, escalarlo y actualizarlo
  − y gestionar certificados con un componente más

→ si el objetivo es reducir lo que se mantiene, el
  gestionado; si la portabilidad es un requisito real, el
  propio                                          ley 23
```

Y lo que hay que resolver en cualquiera de los dos:

```
certificados: emisión y renovación automáticas
  y ALERTA POR ANTIGÜEDAD, no solo por caducidad
                                                clase 196
filtrado de aplicación delante                clase 209
plazos y drenaje coherentes                   clase 212
y la entrada NO debe ser el único control de acceso:
  la autorización entre servicios va aparte     clase 201
```

La salida, que en un clúster es fácil de dejar abierta:

```
por defecto, los pods salen a internet por la dirección
del balanceador de salida
→ y eso es salida sin control                     ley 26

lo que hay que hacer
  ruta 0.0.0.0/0 hacia el cortafuegos           clase 219
  y política de red que restrinja a dónde puede salir cada
    espacio de nombres                          clase 200
  → y comprobar con una prueba negativa: sacar datos a un
    destino no declarado debe fallar
```

Los complementos, que son el trabajo continuo:

```
los habituales
  controlador de entrada
  proveedor de secretos desde el almacén externo
  autoescalado de nodos
  recolección de métricas y registros
  política de admisión
  controlador de certificados
  y los operadores de cada equipo

y cada uno
  tiene su matriz de compatibilidad con la versión del
    clúster
  se actualiza en su propio calendario
  y puede bloquear la actualización del clúster  ley 23

→ los que la plataforma ofrece como complemento gestionado
  se actualizan con el clúster, y eso vale mucho
→ los instalados a mano, no                     clase 213
```

Y la decisión práctica:

```
usa complementos gestionados donde existan
y para el resto, mantén la matriz de compatibilidad
escrita y comprobada antes de cada actualización
```

4. Actualizaciones y operación

Las actualizaciones tienen aquí tres piezas que se planifican por separado.

```
1  PLANO DE CONTROL
   se actualiza primero; la ventana de soporte es corta
   → 2 o 3 veces al año, obligatorio              ley 25

2  GRUPOS DE NODOS
   se actualizan después, por grupos
   con acordonado y drenaje, respetando el presupuesto de
   interrupción                                 clase 213
   → y hay que tener capacidad de sobra para el reemplazo
```

```
3  IMAGEN DE LOS NODOS
   parches del sistema operativo, más frecuentes
   → conviene automatizarlos con ventana de mantenimiento
```

Y las decisiones que evitan sorpresas:

```
CANAL DE ACTUALIZACIÓN
  manual        control total, y se olvida            ley 25
  parches       aplica parches automáticamente ← razonable
  estable       sube versión menor automáticamente
  → con VENTANA DE MANTENIMIENTO declarada, para que no
    ocurra en hora punta

Y LA COMPROBACIÓN PREVIA
  interfaces retiradas en los manifiestos
  compatibilidad de complementos
  → y un clúster de prueba GENERADO DEL MISMO CÓDIGO
    → en la clase 213, la prueba pasó porque el clúster de
      prueba no era idéntico
```

La reconciliación desde el repositorio, con la misma alerta imprescindible:

```
el estado deseado en el repositorio; un controlador lo
aplica y corrige la deriva                    clase 213

y la alerta
  «la sincronización lleva N minutos sin completarse»
  → un bucle parado no da error: deja de aplicar   ley 13
```

Y el modo automático de la plataforma, que conviene conocer:

```
hay una variante gestionada de reconciliación como
complemento
  + menos que operar
  − menos control sobre versiones y extensiones
→ misma decisión que con los demás complementos
```

Lo que hay que vigilar, además de lo de la clase 213:

```
versión del clúster y días hasta el fin de soporte ley 25
nodos con imagen desactualizada
antigüedad de la sincronización
direcciones libres en la subred de nodos
  → si se agota, no se puede escalar          clase 193
cuota de cómputo frente al máximo del escalado
  → la cuota es de la suscripción y se topa    clase 217
y reinicios por memoria y estrangulamiento por CPU
                                                clase 213
```

Y la lista de comprobación de la clase:

- ■ la decisión de usar clúster está justificada frente a las opciones gestionadas
- ■ el modelo de red se eligió contando direcciones
- ■ el plano de control es privado, y la canalización lo alcanza
- ■ hay grupo de nodos de sistema separado de los de carga
- ■ las cargas usan identidad federada, no la del nodo
- ■ los pods no pueden alcanzar el servicio de metadatos
- ■ la credencial federada ata emisor, espacio de nombres y cuenta de servicio
- ■ los secretos vienen del almacén externo en ejecución
- ■ la política de red está activada, con denegación por defecto
- ■ la salida pasa por el cortafuegos y está restringida
- ■ los certificados tienen alerta por antigüedad
- ■ se usan complementos gestionados donde existen
- ■ hay matriz de compatibilidad de los instalados a mano
- ■ el canal de actualización y la ventana están decididos
- ■ el clúster de prueba se genera del mismo código
- ■ hay alerta de antigüedad de la sincronización
- ■ se vigilan direcciones libres y cuota frente al máximo

Y el cierre que enlaza con la clase siguiente: con el cómputo resuelto, quedan los datos, que en Azure ofrecen tres familias con modelos de consistencia distintos y una de ellas con cinco niveles a elegir. Es la materia de la clase 223.

Ejemplo trabajado

CloudShop monta su clúster en Azure para las cargas del equipo de datos y el buscador. Lo que sigue es la decisión del modelo de red que evitó recrear el clúster a los cuatro meses, y los dos incidentes del primer año.

La decisión de adoptar, con el método de siempre:

```
lo que se pedía
  2 cargas con GPU para el equipo de datos
  un operador de base vectorial que ese equipo ya usa
  trabajos por lotes con prioridades
  el buscador, con imagen propia y ajustes de sistema

lo que ya estaba resuelto
  web, API y trabajadores por cola, en aplicaciones de
  contenedor                                  clase 221

decisión
  clúster SOLO para esas 4 cargas
  las 11 aplicaciones de contenedor se quedan donde están
  motivo   funcionan, y moverlas añade trabajo sin
           beneficio                            ley 23
```

El modelo de red: el cálculo que decidió.

```
el equipo iba a elegir red plana «porque es lo que usamos
en la otra nube»

el cálculo, hecho antes
  nodos máximos previstos                        34
  pods por nodo (valor por defecto)             110
  direcciones necesarias con red plana        3.740
  más el margen de despliegue escalonado      ×1,5
  ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
  total                                      ~5.610
  → una subred /19

  con red superpuesta
  nodos máximos                                  34
  direcciones necesarias                         34
  más margen                                    ~60
  → una subred /26 basta

y el contexto
  el plan de direcciones asignaba /21 por radio
                                        clases 193, 219
  → con red plana, el clúster se habría comido más de la
    mitad del radio y habría bloqueado el crecimiento

decisión   red superpuesta
y lo que se comprobó antes
  ¿algo externo necesita llamar a un pod concreto?
  → no; todo pasa por servicios y por la entrada
  → sin ese requisito, la superpuesta no quita nada

y el coste de equivocarse
  cambiar el modelo exige RECREAR el clúster
  → migrar 4 cargas, sus datos y sus operadores: semanas
                                                  ley 14
```

Incidente 1 · Todos los pods podían hablar con todos, mes 2.

```
se descubrió en la revisión de seguridad, no en un incidente

la política de red NO estaba activada
  → es una opción que se activa al crear el clúster
  → y por defecto está desactivada                ley 26

qué implicaba
  un pod del espacio de nombres de pruebas del equipo de
  datos podía llamar al buscador y a la base vectorial
  y el alcance desde cualquier pod comprometido era el
  clúster entero                              clase 189
```

```
y lo peor
  activarla exige recrear el clúster en algunas
  configuraciones
  → en este caso se pudo activar sin recrear, con suerte

corrección
  política de red activada
  denegación por defecto entre espacios de nombres
  y reglas explícitas: 9 pares permitidos, de 42 posibles
  alcance desde un pod comprometido       clúster → 2
                                          servicios

y una comprobación añadida a la creación de clústeres
  función de aptitud: ningún clúster sin política de red
                                                clase 190
```

Incidente 2 · La actualización que no cabía, mes 9.

```
situación   la versión entraba en fin de soporte
            se lanzó la actualización de los grupos de nodos

qué pasó
  la actualización reemplaza nodos: acordona, drena y crea
  uno nuevo
  para crear el nuevo hacía falta cuota de cómputo
  → la suscripción estaba al 96 % de su cuota para esa
    familia de máquinas                          clase 217
  → el nodo nuevo no se pudo crear
  → la actualización quedó a medias: 3 nodos acordonados y
    drenados, sin sustituto

  capacidad efectiva                          -22 %
  y las cargas con GPU no tenían dónde ir: su grupo tenía 2
  nodos y uno estaba acordonado
  duración                                    4 h 20

corrección inmediata
  se descordonó lo drenado y se pidió ampliación de cuota
  → tardó 6 h en concederse

corrección de fondo
  cuota pedida con margen del 30 % sobre el máximo del
  escalado
  alerta al 80 % de la cuota                    clase 262
  y regla: no se actualiza si el margen de cuota es menor
  que un nodo del grupo más grande
```

La identidad, montada bien desde el principio:

```
7 identidades administradas, una por carga
credencial federada atando emisor + espacio de nombres +
  cuenta de servicio
permisos asignados en el ámbito del RECURSO   clase 218
acceso de los pods al servicio de metadatos: bloqueado
secretos desde el almacén externo, montados en ejecución

pruebas negativas
  ✓  pod de «datos-pruebas» intentando obtener la identidad
     de «buscador»                             denegado
  ✓  pod leyendo credenciales del nodo          bloqueado
  ✓  cuenta de servicio sin anotación             sin
                                                identidad
  ✗  un pod del espacio «datos» podía leer un almacén de
     producción
     → la asignación se había hecho en el ámbito del grupo
       de recursos, no del almacén
     → corregido                                clase 218
```

La entrada y los complementos:

```
entrada     controlador gestionado por la plataforma
            motivo: la portabilidad no era requisito y hay
            menos que mantener                    ley 23
            certificados automáticos, con alerta por
            antigüedad                          clase 196
```

```
            filtrado delante                    clase 209

complementos
  gestionados      métricas, registros, proveedor de
                   secretos, política, autoescalado
  a mano           el operador de base vectorial del equipo
                   de datos
                   → 1 solo, con su matriz de compatibilidad

→ frente a los 12 complementos a mano de la clase 213,
  aquí quedó 1
→ y la actualización siguiente tardó 2 h en vez de 4
```

Las actualizaciones, planificadas:

```
canal                parches automáticos
ventana              domingos 02:00-06:00
versión menor        manual, 3 veces al año, planificada

antes de cada una
  revisión de interfaces retiradas en la canalización
  clúster de prueba generado del mismo código   clase 213
  comprobación de cuota disponible
  matriz de compatibilidad del operador

segunda actualización (mes 13)
  duración                                      2 h
  incidencias                                     0
  manifiestos corregidos antes                    6
```

El resultado, al año:

```
direcciones consumidas por el clúster      ~5.610          60
  (si se hubiera elegido red plana)
alcance desde un pod comprometido      todo el clúster  2 serv.
complementos instalados a mano                  —           1
duración de una actualización              4 h 20        2 h
incidentes por cuota                            1           0
cargas movidas al clúster sin necesidad         0           0
capacidad de equipo consumida                 n/d    0,3 pers.
```

La lección que esta clase deja: la decisión que más valor tuvo se tomó antes de crear el clúster: elegir el modelo de red superpuesta ahorró más de cinco mil quinientas direcciones y evitó que el clúster se comiera medio radio del plan. Cambiarla después habría exigido recrearlo. Y el hallazgo de seguridad más grave —todos los pods podían hablar con todos— no lo produjo ningún error: era el valor por defecto, y se descubrió en una revisión, no en un incidente.

Laboratorio guiado

Ejecuta desde la raíz:

```
python classes/part-18-azure-production-architecture/222-aks-workload-identity-ingress-y-gitops/lab.py
```

El laboratorio selecciona el motor de práctica kubernetes y produce labresult.json. El escenario, sus comprobaciones y el artefacto esperado corresponden a esta clase; no requiere credenciales y deja explícito qué debe revalidarse en un sandbox real.

1. Lee exercise.steps y formula una predicción antes de ejecutar.
2. Ejecuta la práctica y verifica todos los elementos de checks.
3. Provoca el caso de negativetest y explica la señal observada.
4. Materializa azure-aks-platform en evidence/ usando la plantilla indicada.
5. Para proveedor real, sigue sandbox.requires, registra costo y ejecuta destroy.

Evidencia esperada

El artefacto principal es manifiestos declarativos con estado observado. Además, la entrega debe incluir el comando ejecutado, la salida estructurada y una conclusión que no exceda lo observado.

Reto verificable

Construye azure-aks-platform para el caso CloudShop. Incluye una alternativa descartada, un supuesto que pueda falsarse, una prueba de fallo y una decisión de rollback.

■ Criterio de aceptación

- [ ] lab.py termina con código 0 y genera JSON válido.
- [ ] La entrega conecta al menos tres requisitos con mecanismos verificables.
- [ ] Existe una prueba positiva y una prueba negativa con evidencia.
- [ ] Seguridad, costo y operación aparecen como decisiones, no como anexos.
- [ ] Se declara una limitación y una condición que obligaría a revisar el diseño.
- [ ] Otra persona puede repetir el recorrido sin conocimiento tácito.

■■ Errores frecuentes

Síntoma	Causa probable	Corrección
El clúster consume miles de direcciones y bloquea el crecimiento de la red	Se eligió el modelo de red plana, en el que cada pod toma una dirección de la subred	Calcula nodos por pods antes de crear el clúster y elige el modelo superpuesto salvo requisito concreto; cambiarlo exige recrear.
Cualquier pod puede hablar con cualquier otro	La política de red no está activada; por defecto no lo está	Actívala al crear el clúster, con denegación por defecto y reglas explícitas, y compruébalo con una función de aptitud.
Un pod usa permisos que no le corresponden	Los permisos están en la identidad del nodo o la credencial federada solo ata al emisor	Identidad por carga, credencial atada a espacio de nombres y cuenta de servicio, y bloqueo del acceso al servicio de metadatos.
Una actualización deja nodos drenados sin sustituto	No hay cuota de cómputo para crear los nodos nuevos	Pide cuota con margen sobre el máximo del escalado, alerta al 80 % y no actualices si el margen es menor que un nodo.
La actualización del clúster se bloquea por los complementos	Muchos complementos instalados a mano, cada uno con su compatibilidad	Usa complementos gestionados donde existan y mantén escrita la matriz de los que queden.
Los cambios del repositorio dejan de aplicarse sin aviso	El bucle de reconciliación falló en silencio	Alerta por antigüedad de la sincronización, con destino a un canal con guardia.

■ Seguridad, ética y costo

Trabaja con cuentas propias o sandboxes autorizados. No publiques secretos, identificadores reales ni datos personales. Antes de crear recursos pagos define presupuesto, etiquetas y comando de destrucción; después verifica que no queden recursos huérfanos. Los ejemplos locales enseñan contratos, pero no certifican cumplimiento ni disponibilidad de producción.

■ Preguntas de comprobación

1. ¿Qué diferencia hay entre el modelo de red plana y el superpuesto, y por qué es irreversible?
2. ¿A qué debe atarse una credencial federada para una carga del clúster?
3. ¿Qué está permitido por defecto entre pods y cómo se cierra?
4. ¿Qué puede impedir que una actualización de nodos termine?
5. ¿Qué ventaja tienen los complementos gestionados frente a los instalados a mano?

Referencias

- Microsoft (2025). Azure Kubernetes Service: workload identity. <https://learn.microsoft.com/en-us/azure/aks/workload-identity-overview>
- Microsoft (2025). AKS networking: Azure CNI overlay. <https://learn.microsoft.com/en-us/azure/aks/azure-cni-overlay>
- Microsoft (2025). Application routing add-on and managed ingress. <https://learn.microsoft.com/en-us/azure/aks/app-routing>
- Microsoft (2025). AKS cluster upgrades and maintenance windows. <https://learn.microsoft.com/en-us/azure/aks/upgrade-cluster>
- Microsoft (2025). AKS baseline architecture. <https://learn.microsoft.com/en-us/azure/architecture/reference-architectures/containers/aks/baseline-aks>
- Documentación oficial vigente del servicio implementado; registra URL y fecha de consulta.

Descargar: Parte 18 en PDF · Recorrido de Azure en PDF · Manual integral

Anterior	Índice	Siguiente
← 221 · App Service, Functions y Container Apps en producción	Parte 18 · Programa	223 · Azure SQL, Cosmos DB y consistencia distribuida →

Evaluación — 222 AKS, workload identity, ingress y GitOps

Preguntas

1. Explica el problema que resuelve esta capacidad sin mencionar un proveedor.
2. Identifica una frontera de responsabilidad y su propietario.
3. Compara dos alternativas mediante confiabilidad, seguridad, costo y operación.
4. ¿Qué demuestra el laboratorio y qué no puede demostrar?
5. Propón un caso límite o una prueba de fallo.

Reto verificable

Entrega azure-aks-platform con diagrama o configuración, evidencia de ejecución, alternativa descartada, riesgo residual y criterio de rollback.

Criterio de aceptación

- [ ] Resultado reproducible por una segunda persona.
- [ ] Evidencia vinculada a cada afirmación técnica.
- [ ] Prueba negativa o de fallo incluida.
- [ ] Costos y permisos expresados con unidades y alcance.
- [ ] Limitaciones declaradas sin presentar la simulación como producción.

Escala

Nivel	Evidencia
A — competente	Cumple todo y defiende trade-offs con datos.
B — en desarrollo	Cumple lo esencial; falta profundidad en una dimensión.
C — inicial	Artefacto parcial o conclusión sin evidencia suficiente.
0	Sin entrega reproducible.

223 — Azure SQL, Cosmos DB y consistencia distribuida

Parte: 18 — Azure: arquitectura empresarial y operación en producción Nivel: avanzado · Horas estimadas: 4
Laboratorio: data · Estado: EXECUTABLECORE

Propósito

Elegir y configurar el almacén de datos en Azure, donde hay una decisión que no existe en otras nubes: cinco niveles de consistencia a elegir, con su coste y su latencia. La clase compara la base relacional gestionada, la distribuida multimodelo y el resto de la familia, explica qué significa cada nivel de consistencia en la práctica, y desarrolla el modelo de capacidad —unidades de petición— que es donde se rompen los presupuestos y las latencias.

Resultados de aprendizaje

Al finalizar podrás:

1. Elegir entre base relacional gestionada y distribuida con criterios.
2. Decidir el nivel de consistencia por operación y conocer su coste.
3. Dimensionar capacidad con unidades de petición y evitar el estrangulamiento.
4. Distribuir datos entre regiones sabiendo qué se gana y qué se paga.
5. Diagnosticar los fallos característicos de cada familia.

Conceptos centrales

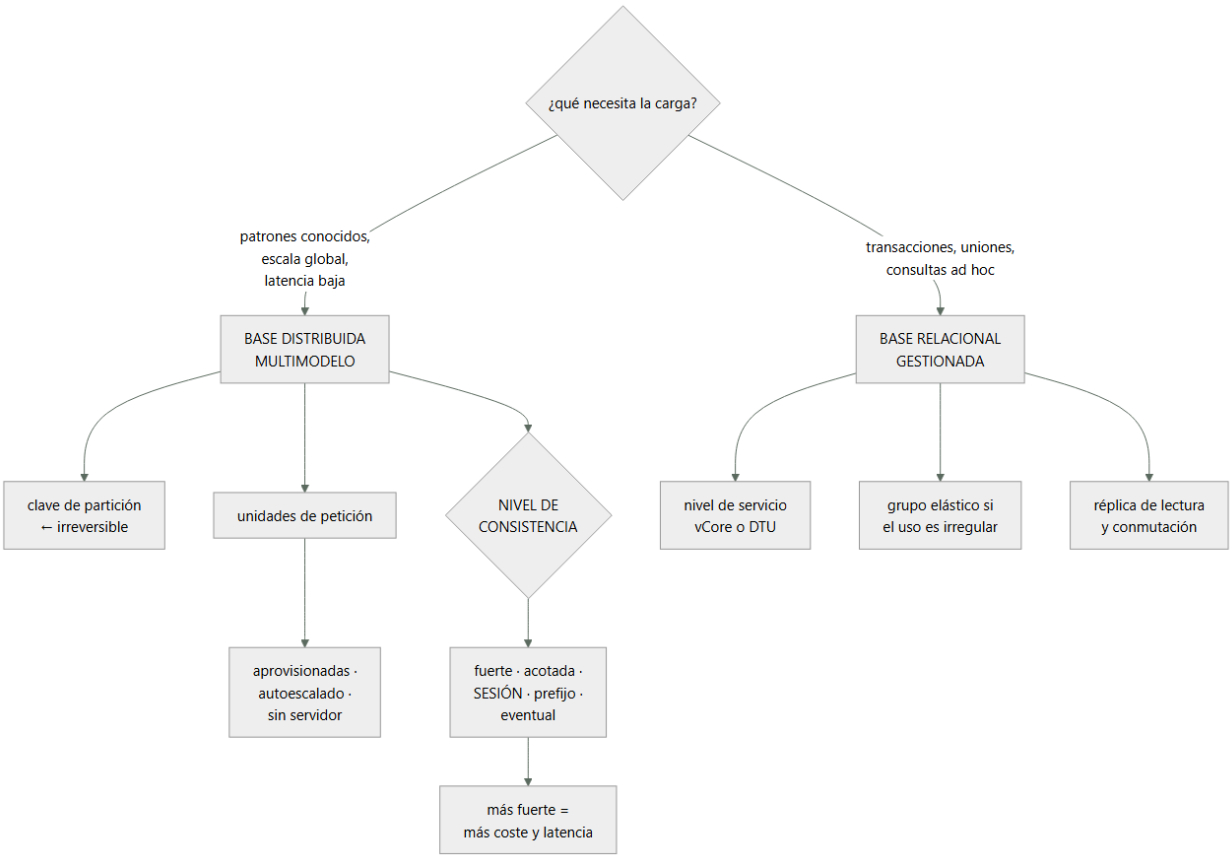
Concepto	Comprensión verificable
unidad de petición	Medida normalizada del coste de una operación. Determina capacidad, latencia y factura.
clave de partición lógica	Campo que agrupa elementos. Decide el reparto y no se puede cambiar sin migrar.
consistencia acotada	Nivel que garantiza un desfase máximo, en versiones o en tiempo.
consistencia de sesión	Nivel por defecto: un cliente ve sus propias escrituras y no retrocede.
grupo de escalado elástico	Conjunto de bases que comparten recursos, para cargas de uso irregular.
escritura multirregión	Opción que permite escribir en varias regiones. Duplica el coste y obliga a resolver conflictos.

Modelo mental

La arquitectura Azure nace del tenant y las políticas heredables, y llega al workload mediante identidades administradas y redes privadas.

Aplicado a esta clase, separa siempre cuatro planos: intención (qué necesita el usuario), configuración (qué declaramos), estado observado (qué existe de verdad) y evidencia (cómo sabemos que cumple). Confundirlos produce diseños que se ven correctos en un diagrama pero fallan al operar.

■ Flujo de razonamiento



Desarrollo

1. Elegir la familia

Azure ofrece varias familias y la elección se hace igual que siempre: por los patrones de acceso.

```
BASE RELACIONAL GESTIONADA
  transacciones, uniones, consultas que no se conocen de
  antemano, informes
  + SQL, herramientas maduras, migración desde lo heredado
  + réplicas de lectura y conmutación gestionada
  – escala vertical con límite; la horizontal exige
    particionar a mano
  – y las conexiones son un recurso limitado  clase 207

BASE DISTRIBUIDA MULTIMODELO
  patrones conocidos, escala grande, latencia baja y
  distribución global
  + escala horizontal transparente
  + distribución multirregión con una casilla
  + cinco niveles de consistencia a elegir
  – el modelo se deduce de los patrones      clase 208
  – y el coste se mide en unidades de petición, que hay que
    entender

Y EL RESTO DE LA FAMILIA
  caché en memoria       para lecturas calientes
  almacén de tablas      barato y limitado
  almacenamiento de
    objetos con capas    para lo frío
  y motores gestionados de código abierto para migraciones
```

Y el criterio, que ya conocemos:

```
¿las consultas se conocen de antemano?
  no  → relacional; o distribuida MÁS un almacén analítico
  sí  → distribuida, si el volumen o la latencia lo piden

¿hace falta transacción entre entidades no relacionadas?
```

```
  → relacional

¿hace falta latencia baja en varias regiones?
  → distribuida
```

Y el patrón que resuelve la mayoría de los casos reales:

```
distribuida para el camino operativo
+ enlace analítico o flujo de cambios hacia un almacén de
  análisis para informes y exploración
→ y así ninguna de las dos hace lo que no sabe hacer
                                        clases 208, 150
```

Y una advertencia sobre la migración desde lo heredado:

```
la instancia gestionada permite migrar bases antiguas con
casi cero cambios
  + resuelve la migración
  − y traslada el modelo antiguo, con sus problemas
→ es una decisión de plazo, no de arquitectura; conviene
  escribir que es temporal, con fecha            ley 25
```

2. Los cinco niveles de consistencia

Esta es la particularidad que no existe en otras nubes: el nivel de consistencia se elige, por cuenta y por operación.

```
FUERTE
  toda lectura ve la última escritura confirmada
  coste   el más alto; y NO se puede usar con escritura
          multirregión
  latencia añadida entre regiones

OBSOLESCENCIA ACOTADA
  el desfase máximo está garantizado: N versiones o T
  segundos
  → útil cuando se puede tolerar un retraso conocido
  → y es la única que da una COTA, que es lo que suele
    hacer falta

SESIÓN  ← el valor por defecto
  el cliente ve sus propias escrituras y no retrocede
  coste   la mitad de lectura que la fuerte
  → y es lo que la mayoría de los sistemas necesitan de
    verdad                                  clase 187

PREFIJO COHERENTE
  las escrituras se leen en orden, sin huecos
  → sin garantía de recencia

EVENTUAL
  converge, sin plazo ni orden
  coste   el más bajo
```

Y las tres cosas que hay que saber para decidir:

```
1  EL COSTE DE LECTURA CAMBIA
   fuerte y acotada cuestan el DOBLE de unidades de
   petición que sesión, prefijo o eventual
   → cambiar el nivel por defecto de fuerte a sesión puede
     reducir la factura de lectura a la mitad

2  SE PUEDE RELAJAR POR PETICIÓN
   la cuenta fija el nivel por defecto
   y cada petición puede pedir uno MÁS DÉBIL, nunca más
   fuerte
   → conviene poner la cuenta en el nivel más fuerte que
     necesite alguna operación, y relajar el resto

3  LA ESCRITURA MULTIRREGIÓN LIMITA LAS OPCIONES
   con varias regiones de escritura, fuerte no está
   disponible
   → y hay que resolver conflictos, con las trampas de los
     relojes                                clase 187
```

Y la decisión práctica, con el método de la clase 187:

```
por cada operación
  reservar plaza / cobrar   → fuerte, o escritura
                              condicionada
  ver lo propio             → sesión
  listar catálogo           → eventual
  panel                     → eventual, marcado

y el nivel de la cuenta = el más fuerte que necesite alguna
→ el resto relaja explícitamente
```

Y una advertencia sobre la consistencia fuerte:

```
con regiones distribuidas, la lectura fuerte obliga a
coordinar
→ la latencia crece con la distancia
→ y por eso «pongo todo en fuerte por si acaso» es la
  decisión más cara que se puede tomar aquí       ley 26
```

3. Unidades de petición y estrangulamiento

Todo el coste, la capacidad y la latencia de la base distribuida se expresan en unidades de petición, y no entenderlas es la causa de los sustos.

```
QUÉ CONSUME
  leer un elemento por su clave: poco, y proporcional al
    tamaño
  escribir: bastante más que leer
  índices: cada índice encarece la escritura
  consulta que recorre varias particiones: MUCHO
  → y esa es la que arruina el presupuesto

LOS MODELOS DE CAPACIDAD
  APROVISIONADA        se reservan unidades por segundo
    + barata con tráfico alto y estable
    – hay que dimensionar y vigilar el estrangulamiento
  AUTOESCALADO         entre un mínimo y diez veces ese
                       mínimo
    + absorbe picos
    – precio por unidad mayor
  SIN SERVIDOR         se paga por consumo
    + ideal para desarrollo y cargas pequeñas
    – límites de caudal y de almacenamiento
```

Y el reparto por partición, que es donde se falla:

```
LAS UNIDADES SE REPARTEN ENTRE PARTICIONES FÍSICAS
  10.000 unidades y 10 particiones → 1.000 por partición

→ si el tráfico va a una sola clave, se dispone de 1.000,
  no de 10.000
→ y aparece el estrangulamiento con la base «al 10 % de
  uso»
→ es exactamente el problema de la clase 208, con otro
  nombre
```

Y el diagnóstico y las correcciones:

```
SÍNTOMA   error de petición limitada, con reintento sugerido
          latencia alta en horas concretas
          y consumo total muy por debajo de lo
          aprovisionado

QUÉ MIRAR
  unidades consumidas POR PARTICIÓN, no en total
  y las claves de partición más consumidoras

CORRECCIONES
  clave que reparta mejor            ← lo correcto
  clave sintética con sufijo         ← si ya es tarde
  caché delante para lecturas calientes
  y revisar las consultas que cruzan particiones
```

Y dos ajustes que reducen mucho el consumo:

```
POLÍTICA DE ÍNDICES
  por defecto se indexa CUALQUIER campo
```

```
  → cada escritura paga por indexar campos que nadie
    consulta                                        ley 26
  → excluir lo que no se filtra reduce el consumo de
    escritura de forma notable

TAMAÑO DEL ELEMENTO
  el coste de lectura crece con el tamaño
  → guardar campos grandes fuera y referenciarlos
```

Y la vigilancia imprescindible:

```
peticiones limitadas: alerta si superan un umbral bajo
unidades consumidas frente a las aprovisionadas
las 10 claves de partición que más consumen
y el coste por operación de negocio           clase 214
```

4. Distribución, continuidad y la familia relacional

La distribución multirregión de la base distribuida se activa fácilmente y tiene consecuencias.

```
AÑADIR UNA REGIÓN DE LECTURA
  duplica el almacenamiento y las unidades de petición
  → el coste se multiplica por el número de regiones
  + latencia baja en esa región
  + y conmutación gestionada

AÑADIR ESCRITURA MULTIRREGIÓN
  + escritura local en cada región
  − conflictos: hay que definir la política de resolución
  − y con relojes derivando, «el último gana» tiene las
    trampas conocidas                          clase 187
  → solo tiene sentido si cada dato lo escribe una sola
    región                                        ley 21

CONMUTACIÓN
  automática o manual
  → automática es cómoda y quita el control del momento
  → y volver exige que la región original se ponga al día
                                                clase 215
```

Y el consejo:

```
empieza con una región de escritura y las de lectura que
hagan falta
y mide antes de activar escritura multirregión: casi nunca
es lo que se necesita
```

La familia relacional, con sus decisiones propias:

```
NIVEL DE SERVICIO
  el modelo por núcleos permite elegir cómputo y
  almacenamiento por separado
  → más predecible que el modelo por unidades combinadas

GRUPOS ELÁSTICOS
  varias bases comparten recursos
  → ideal para multi-inquilino con uso irregular
  → y evita pagar capacidad por cada base

RÉPLICAS DE LECTURA
  descargan informes y consultas pesadas
  → con el retraso de réplica vigilado          clase 161

CONTINUIDAD
  grupo de conmutación con nombre de escritura y de lectura
  → la aplicación usa el nombre, no la instancia
  → y el plazo real se MIDE, incluida la caché del cliente
                                          clases 195, 215

CONEXIONES
  el límite depende del nivel
  → y con cómputo que escala, hay que calcularlo por
    instancia                                   clase 221
  → o usar un intermediario de conexiones
```

Y una decisión de coste que se olvida:

```
el nivel sin servidor de la base relacional pausa la base
tras un tiempo sin uso
  + muy barato en desarrollo
  − la primera conexión tras la pausa tarda
  → excelente para entornos no productivos    clase 214
  → y una mala idea en producción con tráfico esporádico
```

Y la lista de comprobación de la clase:

- la familia elegida corresponde a los patrones de acceso
- los patrones están escritos antes del modelo
- la clave de partición reparte y está en las consultas
- el nivel de consistencia de la cuenta es el más fuerte necesario, y el resto se relaja por petición
- se conoce el coste doble de lectura de los niveles fuertes
- el modelo de capacidad corresponde al perfil de tráfico
- se vigilan las unidades por partición, no solo el total
- hay alerta de peticiones limitadas
- la política de índices excluye lo que no se consulta
- los campos grandes no viven dentro del elemento
- las regiones añadidas están justificadas por su coste
- la escritura multirregión, si existe, tiene política de conflictos escrita
- la aplicación usa nombres de conmutación, no instancias
- el límite de conexiones se calculó por instancia
- el plazo de conmutación se ha medido, no calculado

Y el cierre que enlaza con la clase siguiente: con datos y cómputo resueltos, el trabajo que no cabe en la petición se envía a procesar, y en Azure hay tres servicios de mensajería con propósitos distintos que se confunden constantemente. Es la materia de la clase 224.

Ejemplo trabajado

CloudShop elige y configura sus almacenes en Azure. Lo que sigue es la factura de la base distribuida que triplicaba lo previsto, las dos causas que la explicaban, y la decisión de consistencia que redujo el coste de lectura a la mitad.

El punto de partida, tras dos meses:

```
estimación inicial                          1.800 €/mes
factura real                                5.740 €/mes

desglose
  base distribuida (pedidos y clientes)     4.100 €
  base relacional (facturación heredada)      980 €
  caché en memoria                            340 €
  almacenamiento de objetos                   320 €
```

Causa 1 · La consistencia fuerte por defecto.

```
la cuenta se había creado con consistencia FUERTE
motivo del equipo   «es la más segura»

lo que costaba
  las lecturas fuertes consumen el doble de unidades que
  las de sesión
  lecturas al mes                             980 M
  → y el 96 % de ellas no necesitaban recencia

el análisis por operación                    clase 187
  operación                  necesita          nivel
  reservar plaza             atomicidad     escritura
                                            condicionada
  ver mi pedido              verme a mí     sesión
  listar catálogo            nada           eventual
  ver precio                 ≤ 15 min       eventual
  panel de ventas            ≤ 60 s         eventual
  cierre contable            recencia       acotada 5 s

  → ninguna operación necesitaba FUERTE
  → la más exigente era el cierre contable, con acotada
```

```
cambio
  nivel de la cuenta          fuerte → obsolescencia
                              acotada (5 s)
  y las lecturas normales piden explícitamente sesión o
  eventual

efecto
  unidades de lectura consumidas          -47 %
  coste                          4.100 € → 2.380 €
  latencia p99 de lectura           38 ms → 9 ms
```

Y la observación:

```
el nivel de consistencia se había elegido en el asistente
de creación, en 20 segundos, sin escribir ningún patrón
→ y costaba 1.720 €/mes                    ley 14, ley 26
```

Causa 2 · La política de índices por defecto.

```
por defecto se indexan TODOS los campos

el documento de pedido tenía 61 campos
  campos usados en filtros u ordenaciones           7
  campos indexados                                 61

consumo de escritura por pedido        42 unidades
tras excluir los 54 campos no consultados
                                       17 unidades

efecto
  escrituras al mes                         41 M
  coste de escritura              1.720 € → 700 €
```

Y un efecto secundario que no se buscaba:

```
dos campos indexados contenían texto largo de descripción
→ excluirlos redujo también el TAMAÑO del índice
→ almacenamiento                          410 GB → 240 GB
```

El estrangulamiento con la base al 11 %.

```
síntoma   entre las 11:00 y las 12:30, errores de petición
          limitada
          consumo total                    2.100 de 20.000
          unidades aprovisionadas

diagnóstico
  unidades por partición, no en total
  el panel de claves más consumidoras

    clave de partición          unidades consumidas
    cliente#EMPRESA-A                 1.740   ← el 83 %
    resto (41.000 claves)               360

  el cliente EMPRESA-A hacía pedidos automáticos desde su
  sistema, con ráfagas a mediodía
  su partición disponía de ~1.800 unidades de las 20.000
  → estrangulada, mientras el resto de la base estaba
    ociosa

corrección
  no se cambió la clave: habría exigido migrar
  → el sistema de EMPRESA-A pasó a agrupar sus pedidos en
    lotes con límite de ritmo
  → y se añadió una alerta: «una sola clave consume más del
    30 % del total»

y la decisión registrada
  «si aparece un segundo cliente con este perfil, se
   introduce clave sintética con sufijo y se migra»
                                       clases 208, 190
```

La distribución multirregión, evaluada y recortada.

```
se había activado escritura multirregión en 3 regiones
motivo   «para estar preparados»
```

```
lo que costaba
  almacenamiento × 3
  unidades de petición × 3
  → 2.380 € pasaban a ser ~4.900 € solo por eso

lo que aportaba
  el 98,4 % del tráfico venía de Europa occidental
  las otras dos regiones servían 1,6 %
  y la escritura multirregión obligaba a una política de
  resolución de conflictos que nadie había definido

decisión
  1 región de escritura (Europa occidental)
  1 región de lectura (Europa norte), para continuidad
                                                clase 215
  la tercera, retirada
  escritura multirregión, desactivada

coste                              4.900 € → 2.380 €
  (con la región de lectura ya incluida)
```

La base relacional, con lo suyo:

```
facturación heredada, migrada a instancia gestionada
  → decisión registrada como TEMPORAL, con revisión a 18
    meses                                        ley 25

ajustes
  nivel   por núcleos, cómputo y almacenamiento separados
  réplica de lectura para los informes
    → descargó el 61 % de las consultas del primario
  grupo de conmutación con nombres de escritura y lectura
    → la aplicación usa el nombre, no la instancia

y las conexiones
  cómputo con escalado a 30 instancias           clase 221
  agrupación por instancia                        20
  30 × 20 = 600  > 400 del nivel contratado
  → se bajó la agrupación a 12 y se añadió intermediario
  → conexiones máximas observadas                 214

y los entornos no productivos
  nivel sin servidor con pausa automática
  coste de desarrollo y preproducción     480 € → 90 €
                                                clase 214
```

El resultado:

```
coste mensual de datos                    5.740 €    2.910 €
  base distribuida                        4.100 €    1.680 €
  base relacional                           980 €      810 €
  no productivos                            480 €       90 €
unidades por escritura de pedido              42          17
latencia p99 de lectura                    38 ms        9 ms
errores de petición limitada           140/día           0
almacenamiento de la base distribuida     410 GB      240 GB
regiones activas                              3           2
escritura multirregión                       sí          no
conexiones máximas a la relacional           600         214
```

La lección que esta clase deja: de los cinco mil setecientos euros mensuales, más de la mitad venía de dos decisiones tomadas en el asistente de creación: el nivel de consistencia fuerte y la política de índices que lo indexa todo. Ninguna operación necesitaba consistencia fuerte, y siete de sesenta y un campos se consultaban. Y el estrangulamiento con la base al once por ciento de uso resultó ser un solo cliente concentrando el ochenta y tres por ciento del consumo en una partición, que es el mismo problema de la clase 208 con otro nombre.

Laboratorio guiado

Ejecuta desde la raíz:

```
python classes/part-18-azure-production-architecture/223-azure-sql-cosmos-db-y-consistencia-distribuida/lab.py
```

El laboratorio selecciona el motor de práctica data y produce labresult.json. El escenario, sus comprobaciones y el artefacto esperado corresponden a esta clase; no requiere credenciales y deja explícito qué debe revalidarse en un sandbox real.

1. Lee exercise.steps y formula una predicción antes de ejecutar.
2. Ejecuta la práctica y verifica todos los elementos de checks.
3. Provoca el caso de negativetest y explica la señal observada.
4. Materializa azure-data-platform en evidence/ usando la plantilla indicada.
5. Para proveedor real, sigue sandbox.requires, registra costo y ejecuta destroy.

Evidencia esperada

El artefacto principal es un modelo de datos ligado a patrones de acceso. Además, la entrega debe incluir el comando ejecutado, la salida estructurada y una conclusión que no exceda lo observado.

Reto verificable

Construye azure-data-platform para el caso CloudShop. Incluye una alternativa descartada, un supuesto que pueda falsarse, una prueba de fallo y una decisión de rollback.

■ Criterio de aceptación

- [ ] lab.py termina con código 0 y genera JSON válido.
- [ ] La entrega conecta al menos tres requisitos con mecanismos verificables.
- [ ] Existe una prueba positiva y una prueba negativa con evidencia.
- [ ] Seguridad, costo y operación aparecen como decisiones, no como anexos.
- [ ] Se declara una limitación y una condición que obligaría a revisar el diseño.
- [ ] Otra persona puede repetir el recorrido sin conocimiento tácito.

■■ Errores frecuentes

Síntoma	Causa probable	Corrección
El coste de lectura es el doble de lo esperado	La cuenta usa un nivel de consistencia fuerte que ninguna operación necesita	Decide el nivel por operación, fija en la cuenta el más fuerte necesario y relaja el resto por petición.
Cada escritura consume muchísimas unidades	La política de índices indexa todos los campos por defecto	Excluye del índice los campos que no se filtran ni se ordenan, y saca los textos largos del elemento.
Hay estrangulamiento con la base al diez por ciento de uso	Las unidades se reparten entre particiones y el tráfico va a una sola clave	Mira el consumo por partición y las claves más consumidoras; corrige la clave, agrupa en lotes o añade sufijo sintético.
La factura se multiplica al añadir regiones	Cada región replica almacenamiento y unidades de petición	Justifica cada región con el tráfico que sirve y activa la escritura multirregión solo si cada dato lo escribe una sola región.
La base relacional agota conexiones al escalar el cómputo	Cada instancia abre su propia agrupación y el total supera el límite del nivel	Calcula la agrupación por instancia contando el máximo del escalado, o usa un intermediario de conexiones.
La aplicación no sigue la conmutación de la base	Se conecta a la instancia y no al nombre del grupo de conmutación	Usa siempre el nombre de escritura o de lectura, y mide el plazo real de conmutación incluyendo la caché del cliente.

■ Seguridad, ética y costo

Trabaja con cuentas propias o sandboxes autorizados. No publiques secretos, identificadores reales ni datos personales. Antes de crear recursos pagos define presupuesto, etiquetas y comando de destrucción; después verifica que no queden recursos huérfanos. Los ejemplos locales enseñan contratos, pero no certifican cumplimiento ni disponibilidad de producción.

■ Preguntas de comprobación

1. ¿Cuáles son los cinco niveles de consistencia y cuál es el valor por defecto?
2. ¿Qué niveles cuestan el doble de unidades al leer?
3. ¿Por qué puede haber estrangulamiento con la base muy por debajo de su capacidad total?
4. ¿Qué efecto tiene la política de índices por defecto sobre el coste de escritura?
5. ¿Qué obliga a resolver la escritura multirregión y cuándo tiene sentido?

Referencias

- Microsoft (2025). Azure Cosmos DB consistency levels. <https://learn.microsoft.com/en-us/azure/cosmos-db/consistency-levels>
- Microsoft (2025). Request units in Azure Cosmos DB. <https://learn.microsoft.com/en-us/azure/cosmos-db/request-units>
- Microsoft (2025). Partitioning and horizontal scaling. <https://learn.microsoft.com/en-us/azure/cosmos-db/partitioning-overview>
- Microsoft (2025). Indexing policies in Azure Cosmos DB. <https://learn.microsoft.com/en-us/azure/cosmos-db/index-policy>
- Microsoft (2025). Azure SQL Database: failover groups and elastic pools. <https://learn.microsoft.com/en-us/azure/azure-sql/database/failover-group-sql-db>
- Documentación oficial vigente del servicio implementado; registra URL y fecha de consulta.

Descargar: Parte 18 en PDF · Recorrido de Azure en PDF · Manual integral

Anterior	Índice	Siguiente
← 222 · AKS, workload identity, ingress y GitOps	Parte 18 · Programa	224 · Service Bus, Event Grid y Event Hubs →

Evaluación — 223 Azure SQL, Cosmos DB y consistencia distribuida

Preguntas

1. Explica el problema que resuelve esta capacidad sin mencionar un proveedor.
2. Identifica una frontera de responsabilidad y su propietario.
3. Compara dos alternativas mediante confiabilidad, seguridad, costo y operación.
4. ¿Qué demuestra el laboratorio y qué no puede demostrar?
5. Propón un caso límite o una prueba de fallo.

Reto verificable

Entrega azure-data-platform con diagrama o configuración, evidencia de ejecución, alternativa descartada, riesgo residual y criterio de rollback.

Criterio de aceptación

- [ ] Resultado reproducible por una segunda persona.
- [ ] Evidencia vinculada a cada afirmación técnica.
- [ ] Prueba negativa o de fallo incluida.
- [ ] Costos y permisos expresados con unidades y alcance.
- [ ] Limitaciones declaradas sin presentar la simulación como producción.

Escala

Nivel	Evidencia
A — competente	Cumple todo y defiende trade-offs con datos.
B — en desarrollo	Cumple lo esencial; falta profundidad en una dimensión.
C — inicial	Artefacto parcial o conclusión sin evidencia suficiente.
0	Sin entrega reproducible.

224 — Service Bus, Event Grid y Event Hubs

Parte: 18 — Azure: arquitectura empresarial y operación en producción Nivel: avanzado · Horas estimadas: 4
Laboratorio: messaging · Estado: EXECUTABLECORE

Propósito

Elegir entre los tres servicios de mensajería de Azure, que se confunden constantemente porque los tres «llevan mensajes» y resuelven cosas distintas. La clase separa el bus de mensajes empresarial, el enrutador de eventos y el flujo de gran volumen por lo que cada uno garantiza, desarrolla los mecanismos propios del primero —sesiones, transacciones, detección de duplicados— y aplica la disciplina de siempre: idempotencia, cola de fallidos vigilada y reproceso probado.

Resultados de aprendizaje

Al finalizar podrás:

1. Elegir entre bus, enrutador y flujo según lo que se garantiza.
2. Usar sesiones, transacciones y detección de duplicados donde aportan.
3. Configurar bloqueo, reintentos y cola de fallidos coherentemente.
4. Consumir un flujo con posiciones y reparto por partición.
5. Operar los mensajes fallidos con alerta, diagnóstico y reproceso.

Conceptos centrales

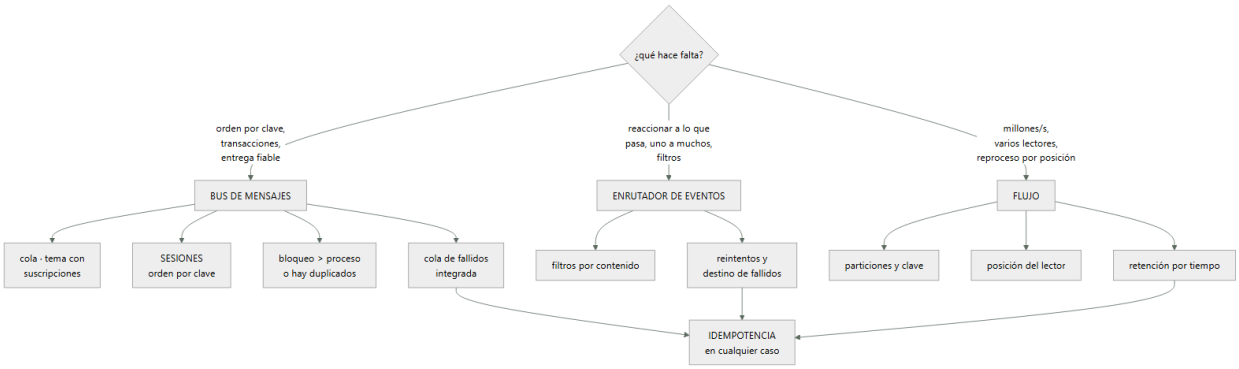
Concepto	Comprensión verificable
bus de mensajes	Colas y temas con garantías empresariales: orden por sesión, transacciones, detección de duplicados y cola de fallidos.
enrutador de eventos	Servicio que entrega eventos a destinos según filtros. Uno a muchos, con reintentos y destino de fallidos.
flujo de eventos	Registro particionado de gran volumen, con lectores independientes y reproceso por posición.
sesión	Agrupación de mensajes por clave que garantiza orden y entrega al mismo consumidor.
bloqueo del mensaje	Tiempo durante el que el mensaje tomado no se entrega a otro. Debe superar el proceso o hay duplicados.
detección de duplicados	Descarte de mensajes con el mismo identificador dentro de una ventana. No sustituye a la idempotencia.

Modelo mental

La arquitectura Azure nace del tenant y las políticas heredables, y llega al workload mediante identidades administradas y redes privadas.

Aplicado a esta clase, separa siempre cuatro planos: intención (qué necesita el usuario), configuración (qué declaramos), estado observado (qué existe de verdad) y evidencia (cómo sabemos que cumple). Confundirlos produce diseños que se ven correctos en un diagrama pero fallan al operar.

■ Flujo de razonamiento



Desarrollo

1. Tres servicios, tres propósitos

Los tres transportan mensajes y garantizan cosas distintas. Elegir mal produce sistemas que funcionan hasta que hay carga o hay que reprocesar.

```
BUS DE MENSAJES (Service Bus)
  colas y temas con suscripciones
  garantiza
    entrega al menos una vez, con bloqueo y confirmación
    ORDEN por sesión
    transacciones entre entidades del propio bus
    detección de duplicados en una ventana
    cola de fallidos integrada
    y mensajes programados y diferidos
  para   pedidos, pagos, integraciones donde perder o
         desordenar cuesta dinero
  ojo    caudal moderado; no es para millones por segundo

ENRUTADOR DE EVENTOS (Event Grid)
  entrega eventos a destinos según filtros de contenido
  garantiza
    entrega al menos una vez, con reintentos y retroceso
    destino de fallidos
    y filtros declarativos
  para   reaccionar a lo que ocurre; uno a muchos
         incluidos los eventos de la propia plataforma
  ojo    NO garantiza orden

FLUJO (Event Hubs)
  registro particionado, con lectores independientes
  garantiza
    orden DENTRO de la partición
    retención por tiempo y reproceso por posición
    varios grupos de consumidores sin duplicar el
      almacenamiento
  para   telemetría, trazas, clics, sensores: gran volumen
  ojo    el consumidor gestiona su posición; si la pierde,
         reprocesa o se salta                  clase 116
```

Y la combinación habitual, que es la que hay que reconocer:

```
algo ocurre → ENRUTADOR (uno a muchos, con filtros)
            → COLA del bus por consumidor
            → función o contenedor

y por separado
  telemetría de alto volumen → FLUJO → procesamiento y
                               almacén analítico

→ y la cola entre el enrutador y el consumidor sigue siendo
  lo que amortigua y da control de reintentos  clase 210
```

Y el error de elección más caro:

```
usar el flujo como si fuera una cola
  no hay bloqueo por mensaje ni confirmación individual
```

```
  no hay cola de fallidos integrada
  un mensaje venenoso bloquea la partición entera
  → hay que implementar todo eso a mano

y al revés: usar el bus para telemetría de alto volumen
  → caudal insuficiente y coste alto
```

2. Los mecanismos del bus

El bus tiene mecanismos que no existen en otros servicios y que resuelven problemas concretos.

```
SESIONES — orden por clave
  los mensajes con la misma clave de sesión van al mismo
  consumidor, en orden
  → resuelve «los eventos de un pedido deben procesarse en
    orden» sin serializar todo
  coste
    el caudal por sesión lo limita un solo consumidor
    y el consumidor debe mantener la sesión bloqueada
  → usar sesiones solo donde el orden importe   clase 203

TRANSACCIONES
  varias operaciones del bus en una unidad atómica
  → recibir de una cola y enviar a otra, todo o nada
  ojo   solo dentro del mismo espacio de nombres; NO
        incluye la base de datos
  → para eso sigue haciendo falta la tabla de salida
                                                clase 118

DETECCIÓN DE DUPLICADOS
  descarta mensajes con el mismo identificador dentro de
  una ventana de tiempo
  + evita duplicados del PRODUCTOR que reintenta
  − no cubre el reproceso ni la reentrega tras bloqueo
    vencido
  → NO sustituye a la idempotencia del consumidor
                                                clase 210

MENSAJES PROGRAMADOS Y DIFERIDOS
  programado   se entrega a una hora futura
  diferido     el consumidor lo aparta y lo recupera por su
               número de secuencia
  → útiles para reintentos con retroceso largo y para
    esperar a que llegue otra cosa
```

Los parámetros que producen duplicados, con la misma aritmética de la clase 210:

```
DURACIÓN DEL BLOQUEO
  el mensaje tomado queda bloqueado ese tiempo
  si el proceso tarda más, el bloqueo VENCE y el mensaje se
  reentrega
  → duplicados

  dos soluciones
    bloqueo mayor que el proceso máximo
    o RENOVAR el bloqueo desde el consumidor mientras
      trabaja
    → la renovación automática existe en las bibliotecas y
      hay que activarla

RECUENTO MÁXIMO DE ENTREGAS
  cuántas veces antes de ir a la cola de fallidos
  típico 3 a 5

TAMAÑO DE LOTE Y PREBÚSQUEDA
  la prebúsqueda trae mensajes por adelantado
  + menos latencia
  − esos mensajes están bloqueados aunque no se procesen
  → prebúsqueda alta con proceso lento = bloqueos vencidos
    y duplicados
```

Y la regla:

```
prebúsqueda × tiempo de proceso < duración del bloqueo
→ o se renueva el bloqueo activamente
```

3. El flujo y sus posiciones

El flujo funciona distinto y su operación tiene sus propias trampas.

```
PARTICIONES
  el caudal se reparte entre particiones
  la CLAVE decide en cuál cae cada evento
  → misma clave, misma partición, mismo orden  clase 116
  → y una clave caliente satura una partición mientras las
    demás están ociosas

  y el número de particiones
    condiciona el paralelismo máximo del consumo
    → más consumidores que particiones = consumidores
      ociosos

POSICIÓN DEL LECTOR
  cada grupo de consumidores guarda por dónde va
  → y esa posición hay que persistirla
  → si se pierde, se reprocesa desde el principio o se
    salta lo pendiente

  LA REGLA
    guardar la posición DESPUÉS de procesar, no antes
    → guardarla antes produce pérdida silenciosa   ley 13
    → guardarla después produce reproceso, que la
      idempotencia absorbe

RETENCIÓN
  los eventos viven un tiempo, no hasta que se consumen
  → si el consumidor está caído más que la retención, se
    pierden eventos
  → alerta por RETRASO del consumidor frente a la
    retención                                    ley 13
```

Y el mensaje venenoso, que aquí es más grave:

```
en una cola, un mensaje que falla acaba en la cola de
fallidos y los demás siguen

en un flujo, si el consumidor falla y no avanza, la
PARTICIÓN ENTERA se detiene
→ hay que decidir qué hacer con lo que no se puede
  procesar
  · apartarlo a una cola de fallidos propia y avanzar
  · o parar y alertar, si perder orden es inaceptable
→ y esa decisión hay que escribirla; no viene dada
```

Y la captura hacia almacenamiento, que ahorra trabajo:

```
el flujo puede volcar automáticamente a almacenamiento de
objetos
  → da el histórico para análisis sin escribir un
    consumidor
  → y el reproceso de meses se hace desde ahí, no desde el
    flujo                                       clase 150
```

4. Fallidos, reproceso y operación

La disciplina es la misma de la clase 210, con las piezas de Azure.

```
LA COLA DE FALLIDOS DEL BUS
  cada cola y cada suscripción tiene la suya, integrada
  los mensajes llegan con el MOTIVO y la descripción
  → y eso es lo que permite distinguir veneno de
    dependencia                                 clase 210

  y llegan también por
    caducidad del mensaje
    error de evaluación de una regla de suscripción
    y tamaño excedido
  → tres causas que no son «el consumidor falló»

LAS ALERTAS
  «hay mensajes en la cola de fallidos»          insuficiente
```

```
  «el mensaje más antiguo supera 1 hora»          ← la útil
  «la cola activa crece de forma sostenida»
  «el retraso del consumidor del flujo se acerca a la
   retención»                                     ← crítica
```

Y el reproceso, con la lección de la clase 210:

```
procedimiento escrito, con límite de ritmo y por lotes
con posibilidad de reprocesar uno solo
ejecutado por alguien que no lo escribió            ley 22
y los venenos, descartados con registro

→ reprocesar todo de golpe dispara la concurrencia y tumba
  lo que se acababa de recuperar                clase 207
```

La idempotencia, que sigue siendo obligatoria:

```
la detección de duplicados del bus cubre al productor
las sesiones cubren el orden
NINGUNA de las dos cubre
  el bloqueo vencido y la reentrega
  el reproceso desde la cola de fallidos
  el reproceso del flujo desde una posición anterior

→ clave de idempotencia del productor, registro con estados
  y escritura condicionada                      clase 210
```

Lo que hay que vigilar:

```
mensajes activos, programados y en cola de fallidos
antigüedad del más viejo, en las tres
bloqueos vencidos: señal directa de duración mal puesta
entregas repetidas del mismo mensaje
retraso del consumidor del flujo, en eventos y en tiempo
unidades de caudal del flujo frente a lo consumido
y el retraso entre que ocurre el hecho y se procesa
  → la medida que le importa al negocio         clase 211
```

Y la lista de comprobación de la clase:

- el servicio elegido corresponde a lo que hay que garantizar
- hay cola entre el enrutador y el consumidor
- las sesiones se usan solo donde el orden importa
- la duración del bloqueo supera el proceso, o se renueva
- prebúsqueda × tiempo de proceso < duración del bloqueo
- la detección de duplicados no se usa como sustituto de la idempotencia
- toda operación con efecto es idempotente
- el número de particiones del flujo permite el paralelismo necesario
- la posición del lector se guarda DESPUÉS de procesar
- está decidido qué hacer con un mensaje venenoso en el flujo
- hay alerta por antigüedad en la cola de fallidos
- hay alerta de retraso del consumidor frente a la retención
- el procedimiento de reproceso existe y se ha ejecutado
- los venenos se descartan con registro

Y el cierre que enlaza con la clase siguiente: con datos y mensajería en pie, falta saber qué ocurre cuando algo va mal. Observabilidad en Azure, con instrumentación estándar, es la materia de la clase 225.

Ejemplo trabajado

CloudShop monta la mensajería de su plataforma en Azure. Lo que sigue son los tres errores de elección del primer diseño, el incidente del flujo que perdió cuatro horas de telemetría, y los parámetros que quedaron.

El primer diseño, con las tres elecciones equivocadas:

```
confirmación de pedido    → enrutador de eventos, directo
                            a 3 funciones
telemetría de la web      → bus de mensajes
integración con el ERP    → enrutador de eventos
```

Y lo que falló en cada una:

```
1  CONFIRMACIÓN DE PEDIDO al enrutador, directo a funciones
   sin cola en medio
   → sin amortiguación: en campaña, las funciones escalaron
     a 2.400 y tumbaron la base            clase 207
   → sin control de reintentos por consumidor
   → y sin cola de fallidos por consumidor: los tres
     compartían destino de fallidos y no se distinguía
     cuál había fallado

   corrección
     enrutador → 3 colas del bus → 3 consumidores
     cada una con su bloqueo, reintentos y cola de fallidos

2  TELEMETRÍA de la web al bus de mensajes
   volumen                              41.000 eventos/s
   el espacio de nombres del bus se saturó
   coste estimado con el nivel necesario   3.900 €/mes

   corrección
     flujo de eventos, 8 particiones
     coste                                  340 €/mes
     y captura automática a almacenamiento para el
     histórico                            clase 150

3  INTEGRACIÓN CON EL ERP al enrutador
   el ERP exigía procesar las líneas de un pedido EN ORDEN
   el enrutador no garantiza orden
   → líneas procesadas fuera de orden: 61 pedidos con
     estado incorrecto en 3 semanas

   corrección
     cola del bus CON SESIONES, clave = identificador de
     pedido
     → orden garantizado dentro de cada pedido
     → y paralelismo entre pedidos distintos
```

El incidente del flujo: cuatro horas de telemetría perdidas.

```
situación   el consumidor del flujo se desplegó con un
            fallo y entró en bucle de reinicio
            retención del flujo                    1 día
            pero el consumidor guardaba su posición ANTES
            de procesar

qué pasó
  al arrancar, el consumidor leía un lote, guardaba la
  posición, y fallaba al procesar
  al reiniciar, leía DESDE LA POSICIÓN GUARDADA
  → los eventos de ese lote nunca se procesaron
  → y no quedaba rastro: la posición decía que estaban
    hechos

  duración del bucle                          4 h 10
  eventos perdidos                        ~610 millones
  cómo se detectó   el panel de negocio mostró 0 visitas
                    durante 4 horas             ley 15

qué salvó el caso
  la captura automática a almacenamiento seguía activa
  → los eventos estaban ahí
  → se reprocesaron desde los ficheros, en 6 h

correcciones
  1  la posición se guarda DESPUÉS de procesar
     → produce reproceso, que la idempotencia absorbe
  2  alerta de retraso del consumidor: «el retraso supera
     el 50 % de la retención»
  3  alerta de reinicios del consumidor       clase 221
  4  y una comprobación de negocio: «visitas por minuto
     por debajo del mínimo esperado»
     → esta es la que lo habría detectado en 5 minutos
                                              clase 211
```

Los duplicados del bus, y el bloqueo.

```
síntoma   bajo carga, algunos pedidos generaban dos
          notificaciones

diagnóstico
  duración del bloqueo                      30 s (defecto)
  prebúsqueda                               20 mensajes
  tiempo de proceso por mensaje             1,8 s

  20 × 1,8 = 36 s > 30 s
  → los últimos mensajes del lote perdían el bloqueo antes
    de procesarse
  → se reentregaban a otro consumidor

  y la métrica que lo confirmaba
    «bloqueos vencidos»: 340/hora en el pico
    → estaba disponible y no estaba en ningún panel
                                                    ley 15

correcciones
  duración del bloqueo                      30 s → 120 s
  prebúsqueda                               20 → 10
  renovación automática del bloqueo         activada
  e idempotencia en el consumidor de notificaciones

bloqueos vencidos                          340/h → 0
notificaciones duplicadas                    61 → 0
```

La configuración final:

```
ENRUTADOR DE EVENTOS
  eventos de negocio y de la plataforma
  filtros por tipo y por contenido
  destino: colas del bus, nunca funciones directas
  reintentos: 5 con retroceso; destino de fallidos
  configurado

BUS DE MENSAJES
  cola             bloqueo  prebúsq.  entregas  sesiones
  notificaciones    120 s     10         5        no
  inventario        180 s     10         5        no
  facturación       300 s      5         5        no
  erp               240 s      1         3        SÍ
  → la de sesiones con prebúsqueda 1: el orden importa más
    que la latencia

  detección de duplicados activada en la cola de pedidos,
  ventana de 10 minutos
  → y aun así, idempotencia en los cuatro consumidores

FLUJO
  8 particiones, clave = identificador de sesión de usuario
  2 grupos de consumidores: proceso en tiempo real y
    análisis
  retención 3 días (subida desde 1)
  captura automática a almacenamiento, cada 5 min
  posición guardada después de procesar
```

Las alertas montadas:

```
antigüedad del mensaje más viejo en cada cola de fallidos
  umbral 1 hora → canal de guardia
cola activa creciendo de forma sostenida
bloqueos vencidos > 0
retraso del consumidor del flujo > 50 % de la retención
eventos por minuto por debajo del mínimo esperado
  → la de negocio, que detecta lo que las técnicas no
reinicios de consumidores
```

El reproceso, probado:

```
primer ensayo
  se dejaron 8.000 mensajes en la cola de fallidos a
  propósito
```

```
  quien lo ejecutó no había escrito el procedimiento
  → 2 de 7 pasos necesitaron aclaración
  → y el límite de ritmo estaba en el paso 4, no en el 1
    → se movió al 1                              clase 216

segundo ensayo
  8.000 mensajes reprocesados en 3 min
  concurrencia máxima                            40
  incidencias                                     0
```

El resultado:

```
notificaciones duplicadas                      61          0
pedidos con estado incorrecto (orden)          61          0
eventos de telemetría perdidos            610 M            0
coste de la telemetría                    3.900 €      340 €
bloqueos vencidos                         340/hora        0
tiempo de detección de un consumidor
  parado                                     4 h 10      5 min
reprocesos ejecutados                           0          4
  que causaron incidente                        —          0
```

La lección que esta clase deja: los tres errores del primer diseño fueron de elección de servicio, no de configuración: el enrutador directo a funciones sin amortiguar, el bus para telemetría de gran volumen y el enrutador para algo que necesitaba orden. Y el incidente más grave —seiscientos diez millones de eventos perdidos— lo causó guardar la posición del lector antes de procesar, un detalle de una línea que producía pérdida silenciosa; lo salvó una captura automática a almacenamiento que se había activado por otro motivo.

Laboratorio guiado

Ejecuta desde la raíz:

```
python classes/part-18-azure-production-architecture/224-service-bus-event-grid-y-event-hubs/lab.py
```

El laboratorio selecciona el motor de práctica messaging y produce labresult.json. El escenario, sus comprobaciones y el artefacto esperado corresponden a esta clase; no requiere credenciales y deja explícito qué debe revalidarse en un sandbox real.

1. Lee exercise.steps y formula una predicción antes de ejecutar.
2. Ejecuta la práctica y verifica todos los elementos de checks.
3. Provoca el caso de negativetest y explica la señal observada.
4. Materializa azure-event-platform en evidence/ usando la plantilla indicada.
5. Para proveedor real, sigue sandbox.requires, registra costo y ejecuta destroy.

Evidencia esperada

El artefacto principal es un flujo que declara orden, entrega y manejo de errores. Además, la entrega debe incluir el comando ejecutado, la salida estructurada y una conclusión que no exceda lo observado.

Reto verificable

Construye azure-event-platform para el caso CloudShop. Incluye una alternativa descartada, un supuesto que pueda falsarse, una prueba de fallo y una decisión de rollback.

■ Criterio de aceptación

- [ ] lab.py termina con código 0 y genera JSON válido.
- [ ] La entrega conecta al menos tres requisitos con mecanismos verificables.
- [ ] Existe una prueba positiva y una prueba negativa con evidencia.
- [ ] Seguridad, costo y operación aparecen como decisiones, no como anexos.
- [ ] Se declara una limitación y una condición que obligaría a revisar el diseño.
- [ ] Otra persona puede repetir el recorrido sin conocimiento tácito.

Errores frecuentes

Síntoma	Causa probable	Corrección
Un pico de eventos tumba la base de datos	El enrutador entrega directamente a las funciones, sin cola que amortigüe	Pon una cola por consumidor entre el enrutador y el proceso, con su bloqueo, reintentos y cola de fallidos propios.
Se procesan mensajes fuera de orden y el estado queda mal	Se eligió un servicio que no garantiza orden	Usa colas con sesiones y clave por entidad; el orden se garantiza dentro de la sesión y sigue habiendo paralelismo entre entidades.
Aparecen duplicados solo con carga alta	La prebúsqueda por el tiempo de proceso supera la duración del bloqueo	Aumenta el bloqueo, reduce la prebúsqueda y activa la renovación automática; vigila la métrica de bloqueos vencidos.
Se pierden eventos sin ningún error	La posición del lector se guarda antes de procesar	Guarda la posición después de procesar y absorbe el reproceso con idempotencia; alerta si el retraso se acerca a la retención.
Un mensaje incorrecto detiene todo el consumo	En un flujo, el consumidor no avanza y bloquea la partición entera	Decide y escribe qué hacer con el veneno: apartarlo a una cola propia y avanzar, o parar y alertar.
La factura de mensajería es muy alta para telemetría	Se usa el bus de mensajes para volúmenes que corresponden a un flujo	Usa el flujo para gran volumen, con captura automática a almacenamiento para el histórico.

Seguridad, ética y costo

Trabaja con cuentas propias o sandboxes autorizados. No publiques secretos, identificadores reales ni datos personales. Antes de crear recursos pagos define presupuesto, etiquetas y comando de destrucción; después verifica que no queden recursos huérfanos. Los ejemplos locales enseñan contratos, pero no certifican cumplimiento ni disponibilidad de producción.

Preguntas de comprobación

1. ¿Qué garantiza el bus que no garantiza el enrutador de eventos?
2. ¿Qué relación deben cumplir prebúsqueda, tiempo de proceso y duración del bloqueo?
3. ¿Por qué la detección de duplicados no sustituye a la idempotencia?
4. ¿Cuándo se debe guardar la posición del lector de un flujo y por qué?
5. ¿Qué ocurre con un mensaje venenoso en un flujo y qué hay que decidir?

Referencias

- Microsoft (2025). Azure Service Bus: message sessions, transactions and dead-letter queues. <https://learn.microsoft.com/en-us/azure/service-bus-messaging/service-bus-messaging-overview>
- Microsoft (2025). Azure Event Grid: event delivery and retry. <https://learn.microsoft.com/en-us/azure/event-grid/delivery-and-retry>
- Microsoft (2025). Azure Event Hubs: partitions, consumer groups and capture. <https://learn.microsoft.com/en-us/azure/event-hubs/event-hubs-features>
- Microsoft (2025). Choose between Azure messaging services. <https://learn.microsoft.com/en-us/azure/service-bus-messaging/compare-messaging-services>
- Hohpe, G. y Woolf, B. (2003). Enterprise Integration Patterns. <https://www.enterpriseintegrationpatterns.com/>
- Documentación oficial vigente del servicio implementado; registra URL y fecha de consulta.

Descargar: Parte 18 en PDF · Recorrido de Azure en PDF · Manual integral

Anterior	Índice	Siguiente
← 223 · Azure SQL, Cosmos DB y consistencia distribuida	Parte 18 · Programa	225 · Azure Monitor, Application Insights y OpenTelemetry →

Evaluación — 224 Service Bus, Event Grid y Event Hubs

Preguntas

1. Explica el problema que resuelve esta capacidad sin mencionar un proveedor.
2. Identifica una frontera de responsabilidad y su propietario.
3. Compara dos alternativas mediante confiabilidad, seguridad, costo y operación.
4. ¿Qué demuestra el laboratorio y qué no puede demostrar?
5. Propón un caso límite o una prueba de fallo.

Reto verificable

Entrega azure-event-platform con diagrama o configuración, evidencia de ejecución, alternativa descartada, riesgo residual y criterio de rollback.

Criterio de aceptación

- [ ] Resultado reproducible por una segunda persona.
- [ ] Evidencia vinculada a cada afirmación técnica.
- [ ] Prueba negativa o de fallo incluida.
- [ ] Costos y permisos expresados con unidades y alcance.
- [ ] Limitaciones declaradas sin presentar la simulación como producción.

Escala

Nivel	Evidencia
A — competente	Cumple todo y defiende trade-offs con datos.
B — en desarrollo	Cumple lo esencial; falta profundidad en una dimensión.
C — inicial	Artefacto parcial o conclusión sin evidencia suficiente.
0	Sin entrega reproducible.

225 — Azure Monitor, Application Insights y OpenTelemetry

Parte: 18 — Azure: arquitectura empresarial y operación en producción Nivel: avanzado · Horas estimadas: 4
Laboratorio: observability · Estado: EXECUTABLECORE

Propósito

Montar la observabilidad de Azure con instrumentación estándar, de forma que no haya que rehacerla al cambiar de nube ni de proveedor de análisis. La clase cubre el destino único de registros y métricas, el muestreo que decide el coste, y los dos asuntos que este programa lleva señalando y que aquí tienen forma propia: la ingesta sin límite factura más que el cómputo, y la instrumentación específica de un proveedor se convierte en un cambio caro el día que se quiere mover.

Resultados de aprendizaje

Al finalizar podrás:

1. Enrutar registros y métricas de todos los recursos a un destino común.
2. Instrumentar con el estándar abierto para no quedar atado.
3. Controlar el coste con muestreo, niveles y planes de tabla.
4. Definir alertas y objetivos junto al servicio, como código.
5. Correlacionar desde la alerta hasta la causa sin eslabones rotos.

Conceptos centrales

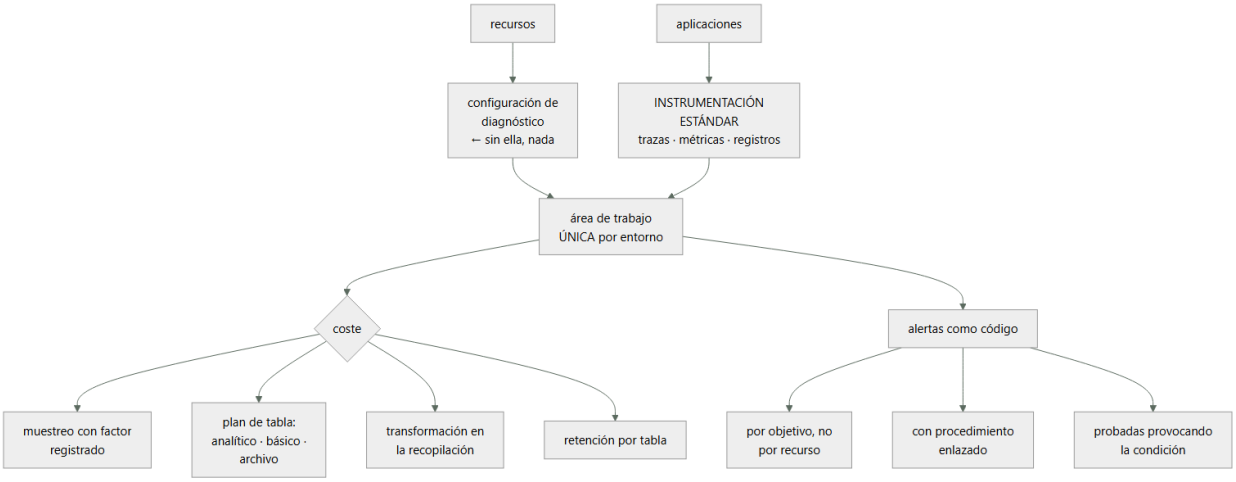
Concepto	Comprensión verificable
área de trabajo	Destino de registros y métricas, con su retención, sus permisos y su facturación.
configuración de diagnóstico	Ajuste por recurso que decide qué categorías se envían y a dónde. Sin ella, el recurso no registra nada.
muestreo adaptativo	Reducción automática del volumen enviado, con factor registrado para poder extrapolar.
instrumentación estándar	Emisión de trazas, métricas y registros con el estándar abierto, independiente del destino.
plan de tabla	Nivel de una tabla de registros: analítico, básico o de archivo. Cambia precio y capacidad de consulta.
regla de recopilación	Definición de qué datos se recogen de qué recursos y con qué transformación antes de almacenarlos.

Modelo mental

La arquitectura Azure nace del tenant y las políticas heredables, y llega al workload mediante identidades administradas y redes privadas.

Aplicado a esta clase, separa siempre cuatro planos: intención (qué necesita el usuario), configuración (qué declaramos), estado observado (qué existe de verdad) y evidencia (cómo sabemos que cumple). Confundirlos produce diseños que se ven correctos en un diagrama pero fallan al operar.

■ Flujo de razonamiento



Desarrollo

1. Un destino, y la configuración que casi nadie pone

En Azure, un recurso no envía nada por defecto. La configuración de diagnóstico es lo que enruta sus registros y métricas, y si falta, ese recurso es invisible.

```
LO QUE HAY QUE DECIDIR POR RECURSO
  qué categorías de registro se envían
  si se envían métricas
  y a qué destino: área de trabajo, almacenamiento, flujo

Y LO QUE PASA SI FALTA
  el recurso funciona
  no aparece en ninguna consulta
  y cuando hay un incidente, no hay datos de las horas
  anteriores                                          ley 13
```

Y por eso se resuelve con política, no con disciplina:

```
política de tipo «desplegar si no existe» que crea la
configuración de diagnóstico en cada recurso nuevo
                                                  clase 217
→ y con la estimación de coste hecha ANTES de asignarla
→ y con las categorías elegidas, no todas
```

El destino, con la decisión de cuántas áreas de trabajo:

```
UNA POR ENTORNO, no una por suscripción
  → correlacionar entre servicios exige que estén juntos
  → y las consultas cruzadas entre áreas son más lentas y
    más incómodas

Y LOS PERMISOS SE RESUELVEN APARTE
  quien no debe ver los registros de otro equipo se
  controla con permisos por tabla o por recurso
  → no separando áreas                          clase 218

EXCEPCIONES RAZONABLES
  registros de auditoría y de seguridad, en un área
  aparte, en una suscripción donde el comprometido no
  pueda borrarlos                          clases 141, 226
```

Y los planes de tabla, que son la palanca de coste más directa:

```
ANALÍTICO   consulta completa, alertas, retención larga
            el más caro por gigabyte
BÁSICO      consulta limitada, sin alertas, retención corta
            mucho más barato
            → para registros voluminosos que solo se miran
              al diagnosticar
ARCHIVO     almacenamiento barato, restauración bajo demanda
            → para lo que hay que conservar por norma
```

```
→ poner en básico las tablas de registros voluminosos de
  aplicación suele reducir la factura a la mitad
→ y hay que comprobar antes que no se necesitan alertas
  sobre ellas
```

2. Instrumentar con el estándar abierto

Aquí hay una decisión de arquitectura que se toma sin pensarla y se paga años después.

```
INSTRUMENTACIÓN ESPECÍFICA DEL PROVEEDOR
  la biblioteca del proveedor, con su API
  + integración inmediata y algunas funciones extra
  – el código queda atado: cambiar de destino exige tocar
    todas las aplicaciones                    clase 158

INSTRUMENTACIÓN ESTÁNDAR (OpenTelemetry)
  API y formato comunes; el destino es configuración
  + cambiar de proveedor de análisis es un cambio de
    exportador, no de código
  + la misma instrumentación vale en las tres nubes
  – algunas funciones del proveedor no están cubiertas

→ y esta es la decisión: el código emite en estándar y el
  destino se configura
→ el coste de cambiarla más adelante es alto: toca todas
  las aplicaciones                              ley 14
```

Y lo que hay que emitir, con la disciplina de la clase 211:

```
TRAZAS con contexto propagado
  incluidas las llamadas asíncronas: el contexto viaja
  DENTRO del mensaje                          clase 224
  → sin eso, la traza se corta en la primera cola

MÉTRICAS
  las cuatro señales y las de negocio
  con dimensiones de baja cardinalidad        clase 211
  → el identificador de pedido va al registro, no a una
    dimensión

REGISTROS estructurados
  con traza, servicio, versión, entorno y duración
  y sin secretos ni datos personales completos
```

Y el detalle que decide la utilidad del conjunto:

```
EL MISMO IDENTIFICADOR DE TRAZA en las tres señales
  → así se salta de una métrica a las trazas del periodo,
    y de una traza a sus registros
  → si cada capa usa su propio identificador, la cadena se
    rompe                                     clase 211
```

El muestreo, que es lo que hace viable el coste:

```
MUESTREO ADAPTATIVO
  el agente reduce el volumen según la carga y registra el
  FACTOR aplicado
  → los recuentos se pueden extrapolar
  → y por eso hay que usar las funciones que lo tienen en
    cuenta al contar

QUÉ SE MUESTREA Y QUÉ NO
  peticiones correctas        1-5 %
  errores                     100 %, siempre
  peticiones lentas           100 %
  y todo lo de un usuario concreto, activable para
    diagnosticar

EL ERROR HABITUAL
  muestrear y luego contar sin tener en cuenta el factor
  → los paneles muestran cifras que no son
```

3. Alertas y objetivos, como código

La disciplina es la de la clase 211, con las piezas de esta nube.

```
LO QUE SE DECLARA JUNTO AL SERVICIO
  configuración de diagnóstico
  reglas de recopilación
  consultas guardadas de diagnóstico
  panel del servicio
  objetivo de nivel de servicio
  reglas de alerta con su grupo de acción

→ y una función de aptitud: ningún servicio se despliega
  sin ellas                                    clase 190
```

Los tipos de alerta, con el uso de cada uno:

```
POR MÉTRICA        rápida y barata; para señales continuas
POR CONSULTA       flexible; para lo que exige agregar o
                   correlacionar
                   → con coste por ejecución: cuidado con
                     la frecuencia
POR ACTIVIDAD      cambios en recursos, políticas y
                   permisos
                   → «alguien quitó una asignación de
                     política»                 clase 217
POR SALUD DEL
  SERVICIO         incidencias del propio proveedor
                   → y esta la olvida casi todo el mundo
```

Y las que este programa exige siempre:

```
ALERTA POR AUSENCIA
  «esta función no se ha ejecutado en N minutos»
  → un trabajo programado que deja de dispararse no genera
    error                                        ley 13

ALERTA POR ANTIGÜEDAD
  cola de fallidos, certificados, sincronización
                                  clases 224, 196, 222

Y POR RITMO DE CONSUMO DEL PRESUPUESTO DE ERROR
  → la que despierta, en vez del error suelto  clase 211
```

Y el destino, con la lección repetida:

```
un grupo de acción por turno de guardia, no uno por
servicio
→ crear un grupo por equipo garantiza que alguno quede sin
  destinatarios                        ley 15, clases 210, 211

y la comprobación
  provocar la condición y ver si llega              ley 22
  → en la clase 211, 7 de 40 no llegaron a la primera
```

Y una nota sobre el coste de las alertas por consulta:

```
una alerta por consulta que se ejecuta cada minuto sobre
mucho volumen factura
→ frecuencia acorde a lo que se detecta
→ y una alerta que nunca se ha disparado en un año, o
  sobra o su umbral está mal                   clase 190
```

4. Consultar, correlacionar y controlar el gasto

Las consultas preparadas, que ahorran minutos durante un incidente:

```
«dame todo lo de esta traza, en las tres señales»
«errores de este servicio en la última hora, por tipo»
«peticiones más lentas y qué dependencia domina»
«qué cambió en los recursos en la última hora»
«compara esta hora con la misma de ayer»

→ guardadas y enlazadas desde el procedimiento de la
  alerta                                       clase 127
```

La cadena de diagnóstico, con sus eslabones:

```
alerta por objetivo
  → panel del servicio
    → trazas del periodo
```

```
      → registros de esas trazas
        → registro de actividad: ¿qué se desplegó o cambió?
                                                clase 122

→ y si un eslabón falta, el diagnóstico se hace a ojo
```

El control del gasto, que aquí tiene palancas concretas:

```
1  CATEGORÍAS ELEGIDAS, no todas
   muchas categorías de diagnóstico son voluminosas y
   nadie las consulta

2  TRANSFORMACIÓN EN LA RECOPILACIÓN
   filtrar y recortar antes de almacenar
   → quitar campos que no se usan, descartar líneas de
     nivel bajo
   → es lo que más reduce sin perder capacidad de
     diagnóstico

3  PLAN DE TABLA por volumen y uso

4  RETENCIÓN por tabla, no global
   → los registros de auditoría, largos; los de aplicación,
     cortos                                    clase 141

5  MUESTREO en la aplicación

6  Y COMPROMISO DE CAPACIDAD si el volumen es estable
   → descuento por gigabyte comprometido
```

Y la comprobación honesta:

```
¿cuánto cuesta la observabilidad frente al cómputo?
→ en la clase 211 costaba el doble
→ y en la 214 era el 11 % de la factura total
→ conviene mirarlo cada trimestre, no una vez  clase 214
```

Y la lista de comprobación de la clase:

- ■ hay política que crea la configuración de diagnóstico en cada recurso
- ■ las categorías están elegidas, no todas
- ■ hay un área de trabajo por entorno, no por suscripción
- ■ los registros de auditoría van a un área separada e inalcanzable desde producción
- ■ la instrumentación de las aplicaciones es estándar
- ■ el contexto de traza viaja dentro de los mensajes
- ■ las dimensiones no tienen cardinalidad alta
- ■ el muestreo registra el factor y las consultas lo tienen en cuenta
- ■ los errores y las peticiones lentas no se muestrean
- ■ paneles, objetivos y alertas están en la plantilla del servicio
- ■ hay alertas por ausencia, por antigüedad y por ritmo de consumo
- ■ hay alerta de salud del servicio del proveedor
- ■ el destino es un grupo de acción con guardia
- ■ cada alerta se ha probado provocando la condición
- ■ hay transformación en la recopilación y planes de tabla por volumen
- ■ el coste de observabilidad se revisa cada trimestre

Y el cierre que enlaza con la clase siguiente: con la observabilidad en pie, queda la parte que mira lo mismo con otra intención: detectar lo que no debería estar pasando. Postura de seguridad, cumplimiento y detección es la materia de la clase 226.

Ejemplo trabajado

CloudShop monta la observabilidad de su plataforma en Azure. Lo que sigue son los recursos que no registraban nada, la factura de ingesta que superaba al cómputo, y la decisión de instrumentación que evitó un cambio de meses dos años después.

El punto de partida:

```
recursos                                        14.200
  con configuración de diagnóstico                 940   6,6 %
  sin ella                                      13.260

áreas de trabajo                                    31
  una por suscripción, creadas al azar
  → correlacionar entre servicios exigía consultas
    cruzadas entre 4 o 5 áreas

instrumentación de aplicaciones
  con la biblioteca específica del proveedor        14
  con estándar abierto                               0

coste mensual de observabilidad              6.900 €
  ingesta                                    4.400 €
  retención                                  1.800 €
  alertas por consulta                         700 €
```

Y lo que reveló el primer incidente serio:

```
un fallo del cortafuegos del centro dejó sin conectividad
a 9 radios durante 40 minutos

al investigar
  el cortafuegos NO tenía configuración de diagnóstico
  → no había ni un registro de las horas anteriores
  → la causa se dedujo por descarte, en 3 horas

y los 13.260 recursos sin diagnóstico incluían
  el cortafuegos central
  4 de las 6 pasarelas
  todas las cuentas de almacenamiento
  y las 43 redes virtuales
```

Las correcciones estructurales:

```
POLÍTICA que crea la configuración de diagnóstico en cada
recurso nuevo y remedia los existentes         clase 217

  y la estimación previa del coste
    si se enviaban TODAS las categorías        ~9.400 €/mes
    con categorías elegidas                    ~1.900 €/mes

  categorías elegidas
    cortafuegos     reglas aplicadas, DNS, amenazas
                    → no: estadísticas detalladas por flujo
    almacenamiento  operaciones de escritura y borrado
                    → no: cada lectura
    redes           registros de flujo, ya existentes
    bases           auditoría, consultas lentas
                    → no: cada consulta

ÁREAS DE TRABAJO
  31 → 3 (producción, preproducción, desarrollo)
  + 1 de auditoría y seguridad, en la suscripción de
    seguridad, sin acceso desde producción     clase 141
  permisos por tabla para separar lo que cada equipo ve
```

La instrumentación: la decisión que se registró.

```
el equipo iba a seguir con la biblioteca específica
«funciona y es lo que sabemos»

el argumento que cambió la decisión
  la parte 19 iba a montar lo mismo en otra nube
  y el equipo de datos ya usaba otro proveedor de análisis
  para sus cargas
  → tres destinos distintos, y el código atado a uno

decisión
  migrar a instrumentación estándar
  el destino es un exportador configurado, no código
  coste de la migración              14 aplicaciones,
                                     6 semanas
```

```
registro de decisión                          clase 190
  premisa   habrá al menos dos destinos de análisis
  qué la reabriría   si el proveedor deja de soportar el
                     estándar, o si su coste sube por
                     encima del propio

y dos años después
  se cambió el proveedor de análisis de las cargas de datos
  cambio necesario en las aplicaciones            ninguno
  se cambió el exportador y el destino
  tiempo                                           2 días
  → sin esta decisión, habrían sido meses de tocar 14
    aplicaciones                                clase 158
```

El coste, atacado con las cinco palancas:

```
ingesta inicial tras aplicar las políticas    1.900 €/mes

1  TRANSFORMACIÓN EN LA RECOPILACIÓN
   los registros de aplicación traían 41 campos; se
   consultaban 9
   se descartan las líneas de nivel de información de 3
   servicios muy habladores
   1.900 € → 1.140 €

2  PLANES DE TABLA
   registros de flujo de red y de contenedores → básico
   → no se necesitaban alertas sobre ellos, comprobado
   1.140 € → 640 €

3  RETENCIÓN POR TABLA
   aplicación         30 días
   flujo de red       14 días
   auditoría          400 días, en archivo tras 90
   retención                       1.800 € → 410 €

4  MUESTREO EN LA APLICACIÓN
   peticiones correctas al 4 %, errores y lentas al 100 %
   → y las consultas ajustadas para tener en cuenta el
     factor
   → aquí hubo un susto: durante dos semanas, el panel de
     pedidos por minuto mostró cifras 25 veces menores
   → porque la consulta contaba filas sin aplicar el
     factor de muestreo

5  ALERTAS POR CONSULTA
   41 alertas por consulta, 12 de ellas cada minuto
   → se bajó la frecuencia de las que detectan cosas
     lentas
   700 € → 210 €

total                              6.900 € → 1.260 €/mes
```

Las alertas, revisadas:

```
antes                                            112
  disparadas al mes                              890
  accionables                                 71 (8 %)
  con destino a grupos de acción por equipo        63
    de ellos, SIN destinatarios                    14

después                                           58
  por objetivo de nivel de servicio                 9
  por ausencia                                     11
    · funciones y trabajos programados
  por antigüedad                                    8
    · colas de fallidos, certificados, sincronización
  por actividad                                     6
    · cambios de política, de permisos, de rutas
  por salud del servicio del proveedor              1
    → esta detectó 2 incidencias del proveedor antes de
      que nadie las notara
  el resto, técnicas                               23
```

```
  destino   un grupo de acción por turno
  disparadas al mes                                74
  accionables                                 68 (92 %)

y la prueba de extremo a extremo
  58 alertas, condición provocada una a una
  no llegaron                                       6
    3 por umbral mal puesto
    2 por destino equivocado
    1 porque la consulta tenía un error de sintaxis y la
      regla estaba en estado de error desde su creación
                                                  ley 22
```

La cadena de diagnóstico, comprobada:

```
02:41  alerta: el objetivo de confirmación de pedido
       consume presupuesto de error 11× más rápido
02:41  el mensaje enlaza el panel del servicio
02:42  el panel muestra el p99 disparado desde las 02:33
02:42  registro de actividad: cambio en una regla del
       cortafuegos a las 02:31               clase 122
02:43  trazas del periodo: el tramo lento es la llamada al
       servicio de precios
02:44  registros de esas trazas: tiempo de espera agotado
       al conectar
02:46  se revierte el cambio del cortafuegos
02:51  restablecido

tiempo hasta identificar la causa                3 min
```

El resultado:

```
recursos con diagnóstico                    6,6 %      99,1 %
áreas de trabajo                              31           4
coste de observabilidad                  6.900 €     1.260 €
alertas configuradas                         112          58
  accionables                            71 (8 %)   68 (92 %)
alertas sin destinatarios                     14           0
instrumentación estándar                    0/14       14/14
tiempo de cambio de proveedor de análisis   meses      2 días
tiempo medio hasta identificar causa      3 h (est.)     3 min
```

La lección que esta clase deja: el 93 % de los recursos no registraba nada, incluido el cortafuegos central, y eso se descubrió en un incidente en el que no había datos de las horas anteriores. La factura de observabilidad superaba a la de cómputo y se redujo un 82 % sin perder capacidad de diagnóstico, con cinco palancas de configuración. Y la decisión que más valió no dio ningún beneficio el primer año: instrumentar con el estándar abierto, que dos años después convirtió un cambio de proveedor de meses en dos días.

Laboratorio guiado

Ejecuta desde la raíz:

```
python classes/part-18-azure-production-architecture/225-azure-monitor-application-insights-y-opentelemetry/lab.py
```

El laboratorio selecciona el motor de práctica observability y produce labresult.json. El escenario, sus comprobaciones y el artefacto esperado corresponden a esta clase; no requiere credenciales y deja explícito qué debe revalidarse en un sandbox real.

1. Lee exercise.steps y formula una predicción antes de ejecutar.
2. Ejecuta la práctica y verifica todos los elementos de checks.
3. Provoca el caso de negativetest y explica la señal observada.
4. Materializa azure-observability en evidence/ usando la plantilla indicada.
5. Para proveedor real, sigue sandbox.requires, registra costo y ejecuta destroy.

Evidencia esperada

El artefacto principal es telemetría correlacionada con una pregunta operativa. Además, la entrega debe incluir el comando ejecutado, la salida estructurada y una conclusión que no exceda lo observado.

Reto verificable

Construye azure-observability para el caso CloudShop. Incluye una alternativa descartada, un supuesto que pueda falsarse, una prueba de fallo y una decisión de rollback.

■ Criterio de aceptación

- [ ] lab.py termina con código 0 y genera JSON válido.
- [ ] La entrega conecta al menos tres requisitos con mecanismos verificables.
- [ ] Existe una prueba positiva y una prueba negativa con evidencia.
- [ ] Seguridad, costo y operación aparecen como decisiones, no como anexos.
- [ ] Se declara una limitación y una condición que obligaría a revisar el diseño.
- [ ] Otra persona puede repetir el recorrido sin conocimiento tácito.

■■ Errores frecuentes

Síntoma	Causa probable	Corrección
Un recurso no aparece en ninguna consulta durante un incidente	No tiene configuración de diagnóstico y por defecto no envía nada	Crea la configuración con una política que la despliegue en todo recurso nuevo, con categorías elegidas y coste estimado antes.
Correlacionar entre servicios exige consultas cruzadas entre muchas áreas	Se creó un área de trabajo por suscripción	Una por entorno, con permisos por tabla para separar lo que cada equipo ve, y una aparte para auditoría.
Cambiar de proveedor de análisis obligaría a tocar todas las aplicaciones	Se instrumentó con la biblioteca específica del proveedor	Emite con el estándar abierto y deja el destino como configuración; el coste de cambiarlo después es alto.
Los paneles muestran cifras muy inferiores a la realidad	Hay muestreo y las consultas cuentan filas sin aplicar el factor	Usa las funciones que tienen en cuenta el factor de muestreo y comprueba los recuentos contra una fuente independiente.
La factura de observabilidad supera a la de cómputo	Todas las categorías, sin transformación, en plan analítico y con retención larga	Elige categorías, transforma en la recopilación, usa planes de tabla por volumen y fija retención por tabla.
Una alerta nunca ha llegado a nadie	Su grupo de acción no tiene destinatarios o la regla está en estado de error	Un grupo de acción por turno de guardia y prueba de extremo a extremo provocando la condición de cada alerta.

■ Seguridad, ética y costo

Trabaja con cuentas propias o sandboxes autorizados. No publiques secretos, identificadores reales ni datos personales. Antes de crear recursos pagos define presupuesto, etiquetas y comando de destrucción; después verifica que no queden recursos huérfanos. Los ejemplos locales enseñan contratos, pero no certifican cumplimiento ni disponibilidad de producción.

■ Preguntas de comprobación

1. ¿Qué ocurre con un recurso que no tiene configuración de diagnóstico?
2. ¿Por qué conviene un área de trabajo por entorno y no por suscripción?
3. ¿Qué se gana instrumentando con el estándar abierto y cuándo se cobra esa ganancia?
4. ¿Qué error de consulta produce el muestreo si no se tiene en cuenta?

5. ¿Cuáles son las cinco palancas para reducir el coste de ingesta?

Referencias

- Microsoft (2025). Azure Monitor: diagnostic settings. <https://learn.microsoft.com/en-us/azure/azure-monitor/essentials/diagnostic-settings>
- Microsoft (2025). Application Insights with OpenTelemetry. <https://learn.microsoft.com/en-us/azure/azure-monitor/app/opentelemetry-enable>
- Microsoft (2025). Log Analytics table plans and data transformations. <https://learn.microsoft.com/en-us/azure/azure-monitor/logs/data-platform-logs>
- Microsoft (2025). Sampling in Application Insights. <https://learn.microsoft.com/en-us/azure/azure-monitor/app/sampling>
- OpenTelemetry (2025). Specification and semantic conventions. <https://opentelemetry.io/docs/specs/otel/>
- Documentación oficial vigente del servicio implementado; registra URL y fecha de consulta.

Descargar: Parte 18 en PDF · Recorrido de Azure en PDF · Manual integral

Anterior	Índice	Siguiente
← 224 · Service Bus, Event Grid y Event Hubs	Parte 18 · Programa	226 · Defender for Cloud, Policy y Sentinel →

Evaluación — 225 Azure Monitor, Application Insights y OpenTelemetry

Preguntas

1. Explica el problema que resuelve esta capacidad sin mencionar un proveedor.
2. Identifica una frontera de responsabilidad y su propietario.
3. Compara dos alternativas mediante confiabilidad, seguridad, costo y operación.
4. ¿Qué demuestra el laboratorio y qué no puede demostrar?
5. Propón un caso límite o una prueba de fallo.

Reto verificable

Entrega azure-observability con diagrama o configuración, evidencia de ejecución, alternativa descartada, riesgo residual y criterio de rollback.

Criterio de aceptación

- [ ] Resultado reproducible por una segunda persona.
- [ ] Evidencia vinculada a cada afirmación técnica.
- [ ] Prueba negativa o de fallo incluida.
- [ ] Costos y permisos expresados con unidades y alcance.
- [ ] Limitaciones declaradas sin presentar la simulación como producción.

Escala

Nivel	Evidencia
A — competente	Cumple todo y defiende trade-offs con datos.
B — en desarrollo	Cumple lo esencial; falta profundidad en una dimensión.
C — inicial	Artefacto parcial o conclusión sin evidencia suficiente.
0	Sin entrega reproducible.

226 — Defender for Cloud, Policy y Sentinel

Parte: 18 — Azure: arquitectura empresarial y operación en producción Nivel: avanzado · Horas estimadas: 4
Laboratorio: security · Estado: EXECUTABLECORE

Propósito

Montar la seguridad operativa de Azure: la evaluación continua de la postura, el cumplimiento medido contra un marco, y la detección con su parte incómoda. La clase separa lo que cada herramienta hace de verdad, ordena las recomendaciones por alcance en vez de por su puntuación, y desarrolla el asunto que decide si un centro de operaciones funciona: una detección que nadie ha comprobado no detecta, y una que genera mil alertas al día no se mira.

Resultados de aprendizaje

Al finalizar podrás:

1. Distinguir postura, cumplimiento y detección, y qué resuelve cada uno.
2. Priorizar recomendaciones por alcance real, no por la puntuación.
3. Medir el cumplimiento contra un marco sin convertirlo en trámite.
4. Configurar detección con reglas propias y comprobarlas.
5. Controlar el coste de la ingesta de seguridad.

Conceptos centrales

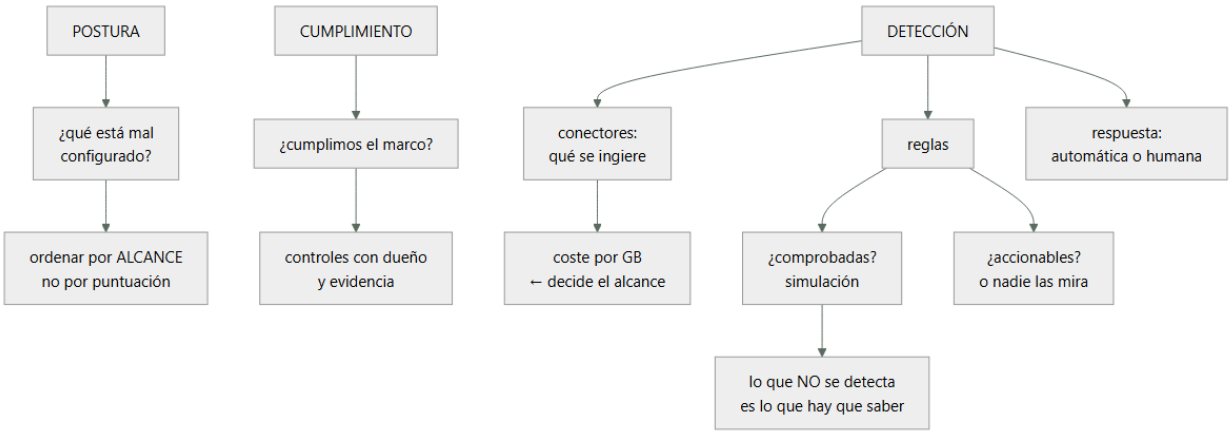
Concepto	Comprensión verificable
postura de seguridad	Evaluación continua de la configuración frente a buenas prácticas. Dice qué está mal configurado.
puntuación de seguridad	Cifra agregada de la postura. Útil como tendencia, engañosa como prioridad.
cumplimiento normativo	Medida de la configuración contra un marco concreto, con sus controles.
detección	Reglas que generan incidentes a partir de señales. Requiere ingesta, reglas y quien las atienda.
simulación de ataque	Ejecución controlada de una técnica conocida para comprobar si se detecta.
conector de datos	Origen de señales que se ingiere para detectar. Cada uno tiene su volumen y su coste.

Modelo mental

La arquitectura Azure nace del tenant y las políticas heredables, y llega al workload mediante identidades administradas y redes privadas.

Aplicado a esta clase, separa siempre cuatro planos: intención (qué necesita el usuario), configuración (qué declaramos), estado observado (qué existe de verdad) y evidencia (cómo sabemos que cumple). Confundirlos produce diseños que se ven correctos en un diagrama pero fallan al operar.

■ Flujo de razonamiento



Desarrollo

1. Postura, cumplimiento y detección

Tres cosas distintas que se mezclan y responden preguntas diferentes.

```
POSTURA
  ¿qué está mal configurado, ahora mismo?
  evaluación continua contra buenas prácticas
  produce   recomendaciones
  ejemplo   «esta base tiene acceso público»

CUMPLIMIENTO
  ¿cumplimos este marco concreto?
  la misma información, agrupada por controles de una norma
  produce   evidencia para auditoría
  ejemplo   «el control A.9.4 está cubierto por estas 12
             políticas»

DETECCIÓN
  ¿está ocurriendo algo ahora?
  reglas sobre señales que generan incidentes
  produce   incidentes que alguien atiende
  ejemplo   «esta identidad ha accedido desde dos países en
             10 minutos»
```

Y la relación entre las tres, que ordena el trabajo:

```
la POSTURA reduce lo que hay que detectar
  → un almacén sin acceso público no puede ser accedido
    desde internet
  → arreglar configuración es más barato que detectar su
    explotación

el CUMPLIMIENTO no mejora la seguridad por sí mismo
  → mide y da evidencia; no cierra nada
  → y perseguir el porcentaje produce trámite   ley 17

la DETECCIÓN cubre lo que la configuración no puede
  → uso legítimo de credenciales robadas, abuso de
    permisos concedidos, comportamiento anómalo
```

La puntuación de seguridad, con su trampa:

```
es una cifra agregada y ponderada
  + útil como TENDENCIA: si baja, algo se ha degradado
  – engañosa como prioridad: la recomendación que más
    sube la puntuación no es la que más riesgo reduce

  ejemplo típico
    «activar cifrado en 400 discos de desarrollo»
      +6 puntos, riesgo bajo
    «una identidad con Propietario en producción»
      +0,2 puntos, riesgo máximo         clase 218

→ ordena por ALCANCE desde el punto comprometido, no por
```

```
  puntos                                          clase 189
```

2. Postura: priorizar bien

La herramienta produce cientos de recomendaciones. El trabajo es ordenarlas.

```
LAS PREGUNTAS QUE ORDENAN
  1  ¿está expuesto a internet?
  2  ¿tiene datos sensibles?
  3  ¿hasta dónde se llega desde ahí?     ← la que decide
  4  ¿existe ya explotación conocida de esa vulnerabilidad?
  5  ¿cuántos recursos afecta?

→ y las herramientas modernas calculan CAMINOS DE ATAQUE:
  «desde esta máquina expuesta se llega, por esta
   identidad, a esta base con datos de clientes»
→ eso vale más que cualquier lista ordenada por puntos
```

Y las categorías de recomendación, con lo que cuesta cada una:

```
CONFIGURACIÓN (acceso público, cifrado, versión de TLS)
  → se resuelven con política, en masa           clase 217
  → y lo nuevo se impide con denegación

IDENTIDAD (permisos amplios, sin segundo factor)
  → las de más alcance y las más difíciles de arreglar
                                                 clase 218

VULNERABILIDADES (software sin parchear)
  → parcheo y actualización de imágenes          clase 254
  → y con exploración continua, no solo al construir
                                                 clase 212

DATOS (almacenes sin clasificar, sin cifrado propio)
  → hay que saber dónde están los datos sensibles primero
```

Y el orden de trabajo que funciona:

```
1  cerrar lo expuesto a internet con datos detrás
2  reducir el alcance de las identidades más amplias
3  impedir por política que lo arreglado vuelva a ocurrir
4  remediar el resto por lotes, con tareas automáticas
5  y aceptar por escrito lo que no compensa arreglar
                                                 clase 189
```

Y una advertencia sobre las exenciones:

```
una recomendación se puede marcar como exenta
→ y sin dueño ni fecha, desaparece del radar para siempre
                                                    ley 25
→ toda exención con motivo, nombre y caducidad, y un panel
  de las que vencen                              clase 217
```

El cumplimiento, con la disciplina que lo hace útil:

```
se elige el marco que aplica de verdad, no todos
cada control tiene DUEÑO y evidencia
lo que la herramienta no puede comprobar se documenta
  aparte
  → una parte importante de cualquier marco es
    organizativa, no técnica

y la medida honesta
  no «cumplimos el 94 %»
  sino «de los 114 controles, 71 automatizados y verdes,
  22 automatizados con excepciones, 21 manuales con
  evidencia, y estos 3 no los cumplimos, con este plan»
```

3. Detección: lo que cuesta y lo que hay que comprobar

Un centro de operaciones de seguridad tiene tres partes y las tres tienen que existir.

```
1  INGESTA        qué señales entran
2  REGLAS         qué se considera sospechoso
3  RESPUESTA      quién actúa, y cómo

→ si falta cualquiera, no hay detección: hay una consola
```

La ingesta, que decide el coste y el alcance:

```
CONECTORES HABITUALES, por valor y por volumen
  registros de identidad y de inicio de sesión
    valor ALTO, volumen medio      ← el primero, siempre
  registro de actividad de la nube
    valor alto, volumen bajo       ← barato y útil
  alertas de la propia herramienta de postura
    valor alto, volumen muy bajo
  registros de dispositivos y puntos finales
    valor alto, volumen alto
  registros de red (flujos, cortafuegos)
    valor medio, volumen MUY alto  ← el que dispara la
                                     factura
  registros de aplicación
    valor variable

→ y la decisión no es «todo»: es qué se ingiere para
  detectar, qué se guarda barato para investigar y qué no
  se guarda                                    clase 225
```

Y el patrón que resuelve el coste:

```
lo que alimenta REGLAS      → plan analítico
lo que solo se consulta al
  investigar                → plan básico o archivo
lo que hay que conservar
  por norma                 → archivo

→ y las reglas de detección sobre datos en plan básico no
  se pueden hacer: por eso hay que decidir antes
```

Las reglas, con las dos preguntas que deciden si valen:

```
¿ESTÁ COMPROBADA?
  simular la técnica que dice detectar
  → en la clase 179, 2 de 6 técnicas simuladas NO se
    detectaban, y las reglas existían            ley 22
  → y en la clase 189, 3 de 14 pruebas fallaron sobre
    controles documentados como implantados

¿ES ACCIONABLE?
  ¿qué hace quien la recibe?
  → si no hay respuesta posible, la regla genera ruido
  → y el ruido esconde lo real                 clase 125
```

Y la calibración, que es trabajo continuo:

```
una regla nueva se despliega en modo de observación
se mide cuántas veces se dispararía y sobre qué
se ajustan las exclusiones legítimas
  → el escáner de vulnerabilidades propio
  → la cuenta de la copia de seguridad
  → los rangos de la sede
y solo entonces genera incidentes

→ es el mismo orden de las clases 200, 209 y 217
```

Y la medida de si el centro funciona:

```
incidentes al día, y proporción que resulta ser algo
tiempo hasta la primera acción
y —la que más dice— PROPORCIÓN DE TÉCNICAS SIMULADAS QUE SE
DETECTAN
  → esa cifra es la única que mide la detección de verdad
```

4. Respuesta y coste

La respuesta, con la decisión de qué se automatiza:

```
SE AUTOMATIZA
  enriquecer el incidente: quién es esa identidad, qué
    recursos toca, de dónde viene esa dirección
  aislar un dispositivo comprometido
  revocar sesiones de una identidad
  bloquear una dirección
```

```
  → acciones reversibles y de bajo daño si el aviso es
    falso

NO SE AUTOMATIZA
  apagar producción
  borrar recursos
  bloquear cuentas de administración
  → un falso positivo con esas acciones es un incidente
    peor que el que se intentaba evitar

Y EN CUALQUIER CASO
  toda acción automática deja registro y es reversible
  y hay una forma de desactivar la automatización
    rápidamente                                clase 259
```

Y el procedimiento, con lo que este programa exige:

```
cada regla con procedimiento enlazado          clase 125
probado por alguien que no lo escribió           ley 22
y con el paso de contención al principio, no al final
                                                clase 216
```

El coste, que en seguridad se dispara con facilidad:

```
LAS DOS PARTIDAS
  planes de protección por recurso: por servidor, por
    base, por almacén
  ingesta y retención de las señales

Y LAS DECISIONES
  activar la protección donde hay datos o exposición, no
    en todo
  ingerir para detectar solo lo que alimenta reglas
  el resto, en plan barato para investigar
  y compromiso de capacidad si el volumen es estable

→ y hay que revisarlo cada trimestre, como el resto
                                                clase 214
```

Y una comprobación honesta que conviene hacer:

```
¿cuántos incidentes reales ha detectado este centro en el
último año?
¿y cuántos se detectaron por otra vía —un usuario, una
factura, un socio—?
→ la segunda cifra dice qué no cubre la detección
→ en la clase 200, la exfiltración de 14 meses la
  encontró un inventario, no una alerta
```

Y la lista de comprobación de la clase:

- ■ las recomendaciones se ordenan por alcance, no por puntuación
- ■ se usan los caminos de ataque para priorizar
- ■ lo arreglado se impide por política para que no vuelva
- ■ toda exención tiene motivo, dueño y caducidad
- ■ el marco de cumplimiento elegido es el que aplica
- ■ cada control tiene dueño y evidencia
- ■ lo manual está documentado aparte, sin inflar el porcentaje
- ■ los conectores se eligieron por valor y volumen
- ■ lo que alimenta reglas está en plan analítico
- ■ las reglas se calibraron en modo observación antes
- ■ cada regla tiene procedimiento enlazado
- ■ se simulan técnicas periódicamente y se mide qué proporción se detecta
- ■ las acciones automáticas son reversibles y registradas
- ■ hay forma rápida de desactivar la automatización
- ■ se mide cuántos incidentes se detectaron por otra vía
- ■ el coste de protección e ingesta se revisa cada trimestre

Y el cierre que enlaza con la clase siguiente: con seguridad, observabilidad y datos en pie, quedan el coste y la resiliencia, que en esta nube tienen herramientas propias y una para provocar fallos de forma controlada. Es la materia de la clase 227.

Ejemplo trabajado

CloudShop monta la seguridad operativa de Azure. Lo que sigue son las recomendaciones ordenadas por alcance en vez de por puntos, la simulación que reveló que la mitad de las técnicas no se detectaban, y la factura que hubo que recortar.

La postura, al empezar:

```
recomendaciones abiertas                          1.847
puntuación de seguridad                             47 %

ordenadas por PUNTOS, las cinco primeras
  1  cifrado en discos de máquinas               +7,2 pts
     412 recursos, casi todos de desarrollo
  2  copias de seguridad habilitadas             +6,1 pts
  3  versión de TLS mínima                       +5,4 pts
  4  agente de supervisión instalado             +4,8 pts
  5  cifrado en tránsito en almacenes            +4,1 pts

ordenadas por ALCANCE, las cinco primeras
  1  una máquina expuesta a internet con puerto de
     administración abierto, y desde su identidad se
     alcanza la base de pedidos
  2  3 identidades con Propietario en producción
                                                 clase 218
  3  2 cuentas de almacenamiento públicas con datos de
     clientes
  4  una base de datos con acceso público y contraseña de
     administrador antigua
  5  una identidad de canalización que puede asignar
     papeles

→ ninguna de las cinco de alcance estaba entre las 40
  primeras por puntos
```

Y el camino de ataque que la herramienta calculó:

```
internet
  → máquina de salto con puerto de administración abierto
    → su identidad administrada tiene Colaborador en la
      suscripción
      → alcanza la base de pedidos y el almacén de copias
        → 2,3 M de registros de clientes

pasos                                                 4
recursos comprometibles                          11.400
tiempo estimado de explotación                  minutos

→ ese único camino concentraba más riesgo que las 1.800
  recomendaciones restantes juntas
```

El orden de trabajo que se siguió:

```
semana 1   los 5 caminos de ataque: cerrados
           puerto de administración cerrado; acceso por
           servicio de administración temporal  clase 256
           identidades reducidas de ámbito      clase 218
           almacenes públicos cerrados
semana 2   políticas de denegación para que no vuelvan
                                                clase 217
semanas 3-9 remediación por lotes del resto
           tareas automáticas para lo que se puede
semana 10  exenciones escritas para lo que no compensa
           41 exenciones, todas con motivo, dueño y fecha

puntuación de seguridad          47 % → 81 %
recomendaciones abiertas       1.847 → 214
caminos de ataque con datos al final    5 → 0
```

Y la observación:

```
la puntuación subió 34 puntos
y los 34 puntos NO son la medida de lo que mejoró
→ lo que mejoró fue cerrar 5 caminos, que valían 0,9 puntos
```

```
  entre los cinco                                  ley 17
```

El cumplimiento, medido con honestidad:

```
marco elegido   el que exige el contrato con el mayor
                cliente empresarial; no se activaron los
                otros cuatro que ofrecía la herramienta

resultado presentado a auditoría
  controles del marco                              114
    automatizados y conformes                       71
    automatizados con excepciones documentadas      22
    manuales, con evidencia adjunta                 18
    NO cumplidos                                     3
      · rotación de claves de cifrado propias
      · segregación de funciones en dos procesos
      · registro de acceso físico (fuera de alcance:
        es del proveedor)

  y un plan con fecha para los 3

→ lo que la herramienta mostraba como «94 % conforme»
  ocultaba que 18 controles eran manuales y que 3 no se
  cumplían
```

La detección: la simulación que dolió.

```
se montaron 61 reglas de detección
la mayoría, plantillas del proveedor

la simulación, con 14 técnicas conocidas
  técnica                                  ¿detectada?
  fuerza bruta contra identidad                 sí
  inicio de sesión desde país inhabitual        sí
  creación de identidad con permisos altos      sí
  exfiltración a almacén externo                NO  ←
  desactivación de registro de auditoría        sí
  acceso a secretos desde identidad inusual     NO  ←
  enumeración de recursos                       NO  ←
  uso de credencial de emergencia               sí
  movimiento lateral por identidad administrada NO  ←
  borrado masivo de recursos                    sí
  cambio de política de seguridad               sí
  consultas de nombres anómalas                 NO  ←
  ejecución en contenedor con privilegios       NO  ←
  persistencia por credencial federada nueva    NO  ←

  detectadas                                 7 de 14
```

Y el análisis de las siete que no:

```
3  la regla existía y no se disparaba
   · umbral demasiado alto (enumeración: exigía 500
     llamadas en 1 min; el ataque hizo 80 en 10 min)
   · consulta con un error de campo
   · regla en estado deshabilitado desde su creación

3  no había regla
   · movimiento lateral por identidad administrada
   · persistencia por credencial federada nueva
   · ejecución en contenedor con privilegios

1  la señal NO SE INGERÍA
   · consultas de nombres: el conector estaba desactivado
     por coste                                clase 200
```

Y las correcciones:

```
umbrales ajustados con datos reales
reglas escritas para las 3 técnicas sin cobertura
conector de nombres activado, con plan básico para el
  volumen y regla sobre un resumen agregado
  → coste 340 €/mes en vez de 2.100 €
y simulación trimestral, con la cifra publicada

segunda simulación, 3 meses después     13 de 14
```

```
  la que falta: exfiltración a almacén externo por un canal
  que se decidió no cubrir; aceptada por escrito, con la
  mitigación de que el perímetro de datos lo impide
                                              clase 200
```

El ruido, calibrado:

```
al activar las 61 reglas de golpe
  incidentes el primer día                          890
  reales                                              2

calibración
  modo observación durante 3 semanas
  exclusiones legítimas
    · el escáner de vulnerabilidades propio
    · la cuenta de copias, que accede a todo por diseño
    · los rangos de las 11 oficinas
    · una integración de un socio
  reglas retiradas por no accionables                14

incidentes al día                               890 → 6
  que resultan ser algo                               1,4
  tiempo hasta la primera acción                8 min
```

El coste, recortado:

```
planes de protección por recurso          4.100 €     1.900 €
  → activados en todo; se dejaron donde hay datos o
    exposición
ingesta de señales de seguridad           5.800 €     1.240 €
  → registros de red: de analítico a básico, con
    resúmenes agregados para las reglas
  → identidad y actividad: analítico, sin cambios
retención                                 1.100 €       380 €
████████████████████████████████████████████████████████████████
total                                    11.000 €     3.520 €
```

La comprobación honesta, al año:

```
incidentes reales detectados por el centro             9
incidentes detectados por OTRA vía                     4
  · una factura anómala (una prueba de carga olvidada)
  · un socio que avisó de un correo sospechoso
  · un inventario de destinos de salida       clase 200
  · un usuario que reportó un comportamiento raro

→ 4 de 13: casi un tercio no lo vio la detección
→ y las cuatro vías son las de siempre: coste, terceros,
  inventarios y personas
```

La lección que esta clase deja: ordenar por puntuación habría hecho al equipo cifrar cuatrocientos doce discos de desarrollo mientras un camino de cuatro pasos llevaba de internet a dos millones trescientos mil registros de clientes. Y la simulación reveló que siete de catorce técnicas no se detectaban, aunque las reglas existían: tres tenían umbrales o errores, tres no existían y una dependía de una señal que se había desactivado por coste.

Laboratorio guiado

Ejecuta desde la raíz:

```
python classes/part-18-azure-production-architecture/226-defender-for-cloud-policy-y-sentinel/lab.py
```

El laboratorio selecciona el motor de práctica security y produce labresult.json. El escenario, sus comprobaciones y el artefacto esperado corresponden a esta clase; no requiere credenciales y deja explícito qué debe revalidarse en un sandbox real.

1. Lee exercise.steps y formula una predicción antes de ejecutar.
2. Ejecuta la práctica y verifica todos los elementos de checks.
3. Provoca el caso de negativetest y explica la señal observada.
4. Materializa azure-security-operations en evidence/ usando la plantilla indicada.
5. Para proveedor real, sigue sandbox.requires, registra costo y ejecuta destroy.

Evidencia esperada

El artefacto principal es un control con amenaza, mitigación y evidencia. Además, la entrega debe incluir el comando ejecutado, la salida estructurada y una conclusión que no exceda lo observado.

Reto verificable

Construye azure-security-operations para el caso CloudShop. Incluye una alternativa descartada, un supuesto que pueda falsarse, una prueba de fallo y una decisión de rollback.

■ Criterio de aceptación

- [ ] lab.py termina con código 0 y genera JSON válido.
- [ ] La entrega conecta al menos tres requisitos con mecanismos verificables.
- [ ] Existe una prueba positiva y una prueba negativa con evidencia.
- [ ] Seguridad, costo y operación aparecen como decisiones, no como anexos.
- [ ] Se declara una limitación y una condición que obligaría a revisar el diseño.
- [ ] Otra persona puede repetir el recorrido sin conocimiento tácito.

■■ Errores frecuentes

Síntoma	Causa probable	Corrección
El equipo trabaja mucho en seguridad y el riesgo real no baja	Las recomendaciones se priorizan por la puntuación agregada	Ordena por alcance desde el punto comprometido y usa los caminos de ataque calculados.
Una recomendación resuelta reaparece meses después	Se corrigió el recurso pero nada impide crearlo mal otra vez	Tras remediar, impide por política que vuelva a ocurrir.
El informe dice que se cumple el 94 % y la auditoría encuentra huecos	El porcentaje incluye controles manuales sin evidencia y oculta los no cumplidos	Presenta el desglose por tipo de control, con dueño y evidencia, y un plan con fecha para los no cumplidos.
Las reglas de detección existen y el ataque no se detecta	Umbrales mal puestos, consultas con errores o reglas deshabilitadas	Simula técnicas conocidas periódicamente y publica qué proporción se detecta; es la única medida real.
El centro genera cientos de incidentes al día y nadie los mira	Las reglas se activaron sin calibrar ni excluir lo legítimo	Despliega en modo observación, excluye lo conocido, retira lo no accionable y mide la proporción que resulta ser algo.
La factura de seguridad es enorme	Protección activada en todos los recursos e ingesta de señales voluminosas en plan analítico	Protege donde hay datos o exposición, ingiere en plan analítico solo lo que alimenta reglas y usa resúmenes agregados para lo voluminoso.

■ Seguridad, ética y costo

Trabaja con cuentas propias o sandboxes autorizados. No publiques secretos, identificadores reales ni datos personales. Antes de crear recursos pagos define presupuesto, etiquetas y comando de destrucción; después verifica que no queden recursos huérfanos. Los ejemplos locales enseñan contratos, pero no certifican cumplimiento ni disponibilidad de producción.

■ Preguntas de comprobación

1. ¿Qué pregunta responde cada una de las tres herramientas y cómo se relacionan?
2. ¿Por qué la puntuación de seguridad es mala guía de prioridad?
3. ¿Qué debe acompañar a cada control de cumplimiento para que la medida sea honesta?
4. ¿Qué mide de verdad si una detección funciona?

5. ¿Qué acciones de respuesta conviene automatizar y cuáles no?

Referencias

- Microsoft (2025). Microsoft Defender for Cloud: secure score and attack paths. <https://learn.microsoft.com/en-us/azure/defender-for-cloud/secure-score-security-controls>
- Microsoft (2025). Regulatory compliance in Defender for Cloud. <https://learn.microsoft.com/en-us/azure/defender-for-cloud/regulatory-compliance-dashboard>
- Microsoft (2025). Microsoft Sentinel: data connectors and analytics rules. <https://learn.microsoft.com/en-us/azure/sentinel/overview>
- MITRE (2025). ATT&CK for cloud. <https://attack.mitre.org/matrices/enterprise/cloud/>
- Microsoft (2025). Sentinel cost optimization and data tiers. <https://learn.microsoft.com/en-us/azure/sentinel/billing>
- Documentación oficial vigente del servicio implementado; registra URL y fecha de consulta.

Descargar: Parte 18 en PDF · Recorrido de Azure en PDF · Manual integral

Anterior	Índice	Siguiente
← 225 · Azure Monitor, Application Insights y OpenTelemetry	Parte 18 · Programa	227 · Cost Management, Advisor, resiliencia y Chaos Studio →

Evaluación — 226 Defender for Cloud, Policy y Sentinel

Preguntas

1. Explica el problema que resuelve esta capacidad sin mencionar un proveedor.
2. Identifica una frontera de responsabilidad y su propietario.
3. Compara dos alternativas mediante confiabilidad, seguridad, costo y operación.
4. ¿Qué demuestra el laboratorio y qué no puede demostrar?
5. Propón un caso límite o una prueba de fallo.

Reto verificable

Entrega azure-security-operations con diagrama o configuración, evidencia de ejecución, alternativa descartada, riesgo residual y criterio de rollback.

Criterio de aceptación

- [ ] Resultado reproducible por una segunda persona.
- [ ] Evidencia vinculada a cada afirmación técnica.
- [ ] Prueba negativa o de fallo incluida.
- [ ] Costos y permisos expresados con unidades y alcance.
- [ ] Limitaciones declaradas sin presentar la simulación como producción.

Escala

Nivel	Evidencia
A — competente	Cumple todo y defiende trade-offs con datos.
B — en desarrollo	Cumple lo esencial; falta profundidad en una dimensión.
C — inicial	Artefacto parcial o conclusión sin evidencia suficiente.
0	Sin entrega reproducible.

227 — Cost Management, Advisor, resiliencia y Chaos Studio

Parte: 18 — Azure: arquitectura empresarial y operación en producción Nivel: avanzado · Horas estimadas: 4
Laboratorio: finops · Estado: EXECUTABLECORE

Propósito

Cerrar la operación de Azure con las dos cosas que se dejan para el final y deciden si el sistema es sostenible: el control del coste y la comprobación de la resiliencia. La clase aplica el método de la clase 214 con las herramientas de esta nube, evalúa las recomendaciones automáticas con escepticismo —el asesor propone reservas sobre cargas que había que retirar—, y usa la inyección controlada de fallos para comprobar lo que la clase 215 exige comprobar.

Resultados de aprendizaje

Al finalizar podrás:

1. Atribuir el coste con etiquetas impuestas en la creación.
2. Evaluar las recomendaciones automáticas antes de aplicarlas.
3. Comprometer capacidad solo sobre la base estable y vigilar su uso.
4. Inyectar fallos de forma controlada para comprobar la resiliencia.
5. Cerrar el ciclo: cada hallazgo, una acción con dueño y fecha.

Conceptos centrales

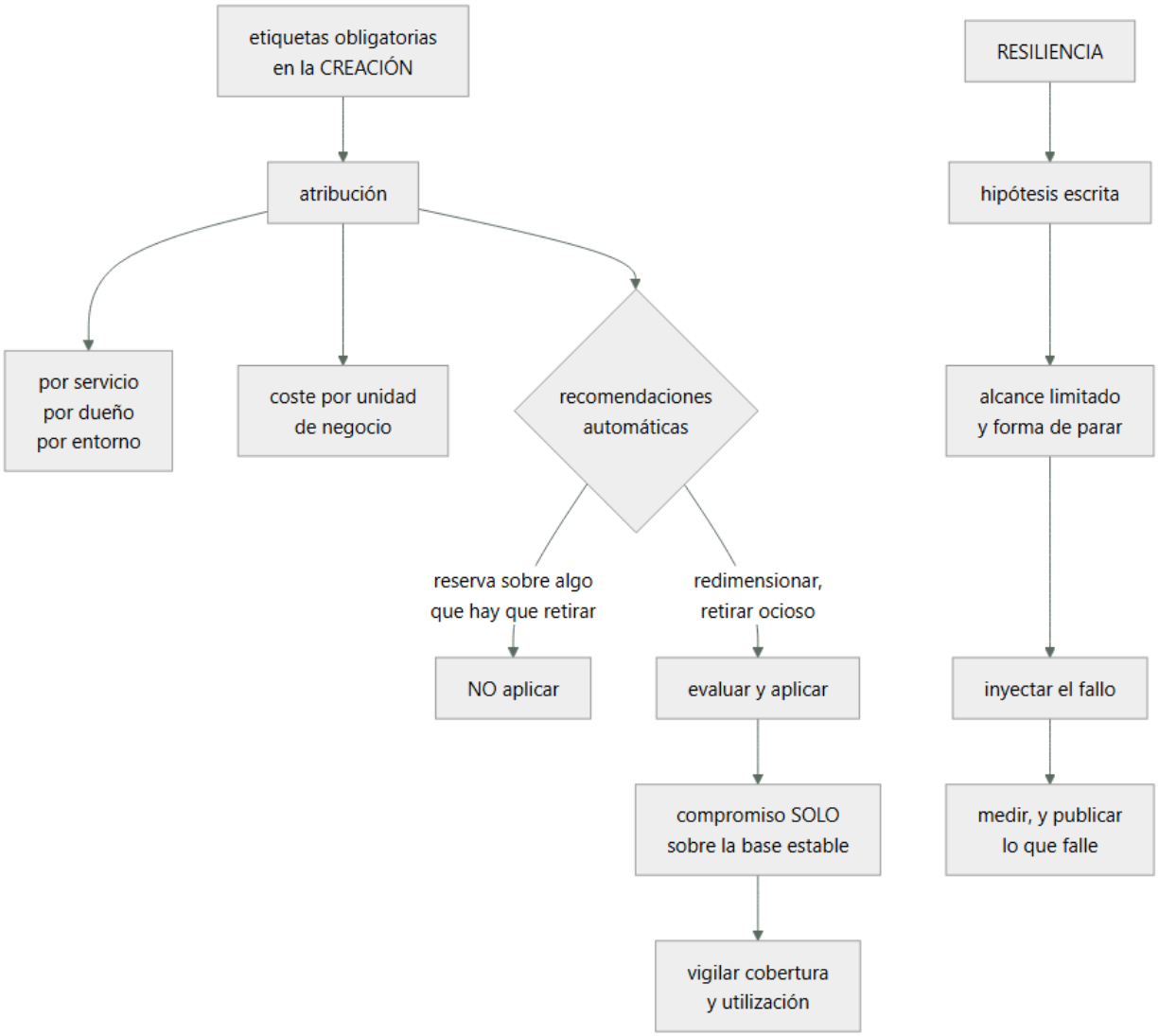
Concepto	Comprensión verificable
ámbito de facturación	Nivel al que se agrega el gasto: cuenta de facturación, perfil, sección y suscripción.
etiqueta heredada	Etiqueta que se propaga del grupo de recursos al recurso. No todos los servicios la respetan.
reserva de capacidad	Compromiso de uso a uno o tres años a cambio de descuento. Se aplica a la base estable.
plan de ahorro	Compromiso de gasto por hora, más flexible que la reserva y con menos descuento.
experimento de caos	Inyección controlada de un fallo, con hipótesis, alcance limitado y forma de parar.
recomendación automática	Sugerencia de la plataforma sobre coste, fiabilidad o rendimiento. Se evalúa, no se aplica.

Modelo mental

La arquitectura Azure nace del tenant y las políticas heredables, y llega al workload mediante identidades administradas y redes privadas.

Aplicado a esta clase, separa siempre cuatro planos: intención (qué necesita el usuario), configuración (qué declaramos), estado observado (qué existe de verdad) y evidencia (cómo sabemos que cumple). Confundirlos produce diseños que se ven correctos en un diagrama pero fallan al operar.

■ Flujo de razonamiento



Desarrollo

1. Atribuir el coste en esta nube

El método es el de la clase 214 y aquí hay particularidades que hay que conocer.

```
LA JERARQUÍA DE FACTURACIÓN
  cuenta de facturación → perfil → sección → suscripción
  → el gasto se agrega por ahí, y por eso la separación
    por suscripción de la clase 217 es también una
    decisión de facturación

LAS ETIQUETAS
  obligatorias en la creación, con valores cerrados, por
  política                                    clase 217
  → y la herencia del grupo de recursos NO la respetan
    todos los servicios
  → hay que comprobar cuáles, y para esos, etiquetar el
    recurso directamente
```

Y el gasto que no se puede etiquetar, que aquí es apreciable:

```
transferencia entre zonas y regiones
servicios compartidos: centro de red, cortafuegos,
  resolutor, área de observabilidad
y los recursos gestionados por la plataforma

→ se reparten por regla escrita y acordada
→ y hay que decir cuánto se reparte y con qué criterio
```

```
                                                clase 214
```

Las cinco vistas de la clase 214, con lo que suele aparecer en Azure:

```
POR SERVICIO      el 80 % en 3 o 4 líneas
POR DUEÑO         y cuánto no lo tiene
POR ENTORNO       preproducción por encima del 25 % de
                  producción es señal
POR TIPO DE USO   dentro de cada servicio grande
POR UNIDAD DE
  NEGOCIO         la única cifra comparable

y las partidas que sorprenden en esta nube
  unidades de petición de la base distribuida  clase 223
  ingesta de observabilidad y de seguridad clases 225, 226
  puntos privados e inspección del cortafuegos clase 219
  transferencia entre zonas
  y los planes de servicio con instancias encendidas y sin
    tráfico                                    clase 221
```

Y los presupuestos con acción, igual que antes:

```
por suscripción y por servicio, sobre previsión
en entornos no productivos, con acción real
en producción, solo aviso
y detección de anomalías por servicio y por dueño
                                                clase 214
```

2. Las recomendaciones automáticas, con escepticismo

La plataforma propone mejoras de coste, fiabilidad, rendimiento y seguridad. Son útiles y no se aplican sin evaluar.

```
LAS DE COSTE, y su trampa
  «compra una reserva para estas máquinas»
  → calcula el ahorro suponiendo que ESAS máquinas seguirán
    existiendo tres años
  → y no sabe que estaban en la lista de retirada
  → comprometerse con lo que hay que apagar es la peor
    compra posible                                ley 25

  «redimensiona esta máquina infrautilizada»
  → mira el uso medio, no el pico ni la estacionalidad
  → hay que comprobar el percentil alto antes  clase 212

  «retira este recurso ocioso»
  → suele ser correcta, y aun así: apagar antes de borrar
                                                clase 214
```

Y el orden correcto, que evita comprar lo que hay que tirar:

```
1  RETIRAR lo que no se usa
2  REDIMENSIONAR lo sobredimensionado
3  APAGAR por horario lo no productivo
4  Y SOLO ENTONCES comprometer capacidad

→ comprometer primero congela el desperdicio durante años
```

Los compromisos, con la decisión entre los dos tipos:

```
RESERVA DE CAPACIDAD
  se compromete un tipo y tamaño concreto
  + descuento mayor
  − rígida: si se cambia de familia, el descuento se pierde
    o hay que intercambiarla

PLAN DE AHORRO
  se compromete un gasto por hora
  + flexible: se aplica a lo que haya
  − descuento menor

CRITERIO
  infraestructura estable y previsible   → reserva
  cargas que van a cambiar de forma      → plan de ahorro
  y en ambos casos, SOLO sobre la base estable medida con
  meses de histórico                            clase 143
```

Y lo que hay que vigilar después:

```
COBERTURA     qué proporción del uso está cubierta
UTILIZACIÓN   qué proporción de lo comprometido se usa
  → por debajo del 95 %, es dinero tirado
  → y baja sola cuando se apaga algo o se cambia de
    familia

→ alerta si la utilización cae por debajo del umbral
→ y revisión antes de cada renovación
```

Y las recomendaciones de fiabilidad, que merecen más atención que las de coste:

```
«esta base no tiene redundancia de zona»
«este recurso no tiene copias de seguridad»
«esta configuración no sobrevive a la pérdida de una zona»

→ estas se leen con la aritmética de la clase 185: ¿cambian
  el techo de disponibilidad del flujo crítico?
→ y si sí, se aplican; si no, se aceptan por escrito
```

3. Inyectar fallos de forma controlada

La clase 215 exige comprobar la resiliencia. La inyección controlada de fallos es la forma de hacerlo sin esperar a un incidente.

```
QUÉ PERMITE INYECTAR
  apagar o reiniciar máquinas y nodos
  presión de CPU, memoria o disco
  latencia y pérdida en la red
  bloqueo del acceso a un servicio dependiente
  conmutación forzada de una base
  caída de una zona completa
  y fallos de una dependencia de la plataforma
```

Y la disciplina, que es la de la clase 215:

```
HIPÓTESIS ESCRITA, antes
  «al perder una zona, el flujo de compra seguirá
   funcionando con latencia menor de 600 ms y sin errores;
   la búsqueda quedará degradada»
  → lo valioso es dónde falla la hipótesis

ALCANCE LIMITADO
  un porcentaje de instancias, una zona, un servicio
  → y el experimento debe poder pararse en segundos

VENTANA Y OBSERVACIÓN
  con negocio avisado y con alguien tomando notas
  → y con tráfico real, no de madrugada la primera vez

ESCALA
  1  en papel: recorrer el procedimiento
  2  en preproducción
  3  en producción con alcance pequeño
  4  en producción completo
```

Los experimentos que más encuentran, por orden:

```
1  PÉRDIDA DE UNA ZONA
   → comprueba que las réplicas están repartidas de verdad
   → y que la capacidad restante aguanta   clases 185, 212

2  DEPENDENCIA QUE RESPONDE LENTO, no que cae
   → el fallo gris; la mayoría de los diseños no lo
     resisten                                  clase 185
   → y es donde se descubre que una dependencia declarada
     blanda es dura                            clase 201

3  CONMUTACIÓN DE BASE DE DATOS
   → cronometra y mide la pérdida real       clase 223

4  PRESIÓN DE RECURSOS
   → descubre límites de memoria mal puestos  clase 213

5  PÉRDIDA DE UN SERVICIO DE PLATAFORMA
```

```
  → el almacén de secretos, el resolutor, el registro de
    imágenes
  → y estos son los que nadie prueba y los que paran todo
                                                clase 215
```

Y la disciplina posterior, que es lo que hace útil el ejercicio:

```
cada hallazgo → una acción con dueño y fecha
el procedimiento se corrige EN EL MOMENTO
y el experimento se repite hasta que pase entero

→ un experimento que encuentra problemas y no se repite ha
  servido para la mitad                        clase 215
```

Y una advertencia sobre el permiso y el alcance:

```
la herramienta necesita permisos para romper cosas
→ y esos permisos son peligrosos si quedan permanentes
→ elevación temporal para ejecutar experimentos
                                                clase 218
→ y el alcance del experimento, restringido por etiqueta o
  por grupo de recursos
```

4. Cerrar el ciclo

Coste y resiliencia comparten una propiedad: son las dos cosas que se degradan solas si nadie las mira.

```
EL COSTE SE DEGRADA PORQUE
  se crean recursos y no se retiran               ley 25
  se sobredimensiona por si acaso
  se acumulan entornos y copias
  y nadie tiene el gasto entre sus señales

LA RESILIENCIA SE DEGRADA PORQUE
  el sistema crece y las comprobaciones no      clase 216
  se añaden dependencias sin recalcular el techo clase 185
  y los procedimientos envejecen                  ley 22
```

Y el ritmo que este programa propone:

```
SEMANAL
  revisión de anomalías de coste abiertas
  recursos ociosos detectados y retirados

MENSUAL
  coste por unidad de negocio y su tendencia
  cobertura y utilización de compromisos
  cumplimiento de las iniciativas             clase 217

TRIMESTRAL
  un experimento de resiliencia, ejecutado
  simulación de técnicas de ataque            clase 226
  prueba del acceso de emergencia             clase 218
  revisión de accesos
  y revisión de exenciones vencidas

ANUAL
  ejercicio de pérdida de región completo     clase 215
  y revisión de las decisiones registradas    clase 190
```

Y las señales que dicen si el sistema está sano:

```
COSTE
  proporción de gasto atribuido           > 90 %
  coste por unidad de negocio             tendencia
  utilización de compromisos              > 95 %
  recursos ociosos vivos                  tendencia a cero

RESILIENCIA
  proporción de pruebas negativas que pasan
  plazo de recuperación MEDIDO, no declarado
  proporción de técnicas simuladas detectadas clase 226
  y experimentos ejecutados frente a planificados
```

Y la lista de comprobación de la clase:

- [ ] las etiquetas obligatorias se imponen en la creación

- se comprobó qué servicios no heredan etiquetas
- el coste no etiquetable se reparte por regla escrita
- hay presupuestos con acción en entornos no productivos
- hay detección de anomalías por servicio y por dueño
- se retiró y redimensionó ANTES de comprometer capacidad
- los compromisos cubren solo la base estable medida
- se vigilan cobertura y utilización, con alerta
- las recomendaciones automáticas se evalúan, no se aplican
- las de fiabilidad se leen con la aritmética del techo
- hay experimentos de caos con hipótesis y forma de parar
- el alcance de los experimentos está restringido
- los permisos para inyectar fallos son temporales
- cada hallazgo tiene dueño y fecha, y el experimento se repite
- existe un calendario semanal, mensual, trimestral y anual

Y el cierre que enlaza con la clase siguiente: con todo lo de esta parte montado, queda ponerlo junto en un sistema productivo y comprobarlo. Es la materia de la clase 228, que además cierra la parte 18.

Ejemplo trabajado

CloudShop cierra la operación de su plataforma en Azure. Lo que sigue es la reserva que casi se compra sobre máquinas que había que retirar, y el experimento de pérdida de zona que encontró seis problemas, ninguno de infraestructura.

El coste, al revisar:

```
total mensual                              31.400 €

por servicio
  base distribuida                          5.740   18 %
  cómputo (planes y contenedores)           4.900   16 %
  observabilidad y seguridad                4.780   15 %
  transferencia y red                       4.210   13 %
  bases relacionales                        3.100   10 %
  almacenamiento                            2.840    9 %
  resto                                     5.830   19 %

por dueño
  atribuido                                19.100   61 %
  sin atribuir                             12.300   39 %
```

Y el detalle de lo no atribuido:

```
transferencia y servicios compartidos          6.100
recursos sin etiqueta                          3.240
  → de ellos, 1.870 € en servicios que NO HEREDAN las
    etiquetas del grupo de recursos
  → el equipo creía que las heredaban todos
recursos ociosos                               2.960
```

Y el hallazgo de la herencia:

```
se comprobó servicio por servicio cuáles heredan
  heredan                                        61 %
  NO heredan                                     39 %
    · cuentas de almacenamiento en algunas operaciones
    · discos creados por conjuntos de escalado
    · direcciones públicas creadas automáticamente
    · instantáneas y copias

corrección
  política de tipo MODIFICAR que añade las etiquetas al
  crear, en vez de confiar en la herencia      clase 217
  → 3.240 € pasaron a estar atribuidos en 3 semanas
```

La reserva que no se compró.

```
la recomendación automática decía
  «reserva a 3 años para 22 máquinas de la familia D»
  ahorro estimado                         4.100 €/mes
  coste del compromiso                  38 % del total

lo que el equipo comprobó antes de comprar
```

```
  ¿qué son esas 22 máquinas?
    14  nodos del clúster antiguo, en la lista de retirada
        del trimestre siguiente
     5  máquinas del sistema heredado de facturación, con
        migración planificada a 18 meses          clase 223
     3  máquinas de un proyecto terminado en 2024, que
        nadie había apagado                        ley 25

→ 17 de 22 no existirían en un año
→ comprar la reserva habría congelado el desperdicio
  durante tres

lo que se hizo, en el orden correcto
  1  RETIRAR      las 3 del proyecto terminado
                  y 6 más detectadas al inventariar
                                        -1.240 €/mes
  2  REDIMENSIONAR  11 máquinas con uso p95 muy por debajo
                    de su tamaño         -680 €/mes
  3  APAGAR POR HORARIO  no producción
                                        -2.100 €/mes
  4  Y ENTONCES comprometer
     plan de ahorro (no reserva) sobre la base estable
     medida con 4 meses de histórico
     motivo del plan y no la reserva: el clúster antiguo se
     retira y la familia cambiará       clase 143
                                        -1.900 €/mes

ahorro total                            -5.920 €/mes
frente a los 4.100 € de la recomendación, y sin
comprometerse con lo que había que tirar
```

El experimento de pérdida de zona.

```
HIPÓTESIS, escrita
  1  el flujo de compra seguirá funcionando
  2  la latencia p99 subirá por debajo de 600 ms
  3  no habrá errores para el usuario
  4  la búsqueda quedará degradada, no caída
  5  la recuperación al restaurar la zona será automática

ALCANCE   la zona 3 de las 3, en producción, con el 30 %
          del tráfico de un día normal
PARADA    detener el experimento restaura en < 60 s
OBSERVACIÓN  5 personas, 1 tomando notas

EJECUCIÓN, martes 15:00
```

Y lo que ocurrió:

```
15:00:00  se corta la zona 3

15:00:12  el balanceador retira los destinos de esa zona
          ✓ predicción 1: el flujo sigue

15:00:40  latencia p99                      410 ms
          ✓ predicción 2

15:01:10  aparecen errores 503 en el 3 % de las peticiones
          ✗ predicción 3
          causa   el plan de servicio tenía 6 instancias,
                  2 por zona; al perder 2, las 4 restantes
                  quedaron al 91 % de su codo  clase 186
                  y el escalado tardó 4 minutos
          → faltaba margen para perder una zona de tres
                                              clase 212

15:02:30  la BÚSQUEDA cayó por completo
          ✗ predicción 4
          causa   sus 3 réplicas estaban las tres en la
                  zona 3
                  el reparto entre zonas no estaba
                  configurado                clase 213
          → y nadie lo sabía porque nunca se había perdido
            una zona
```

```
15:04:00  la base distribuida siguió sirviendo
          ✓  (redundancia de zona activada)

15:04:20  el almacén de SECRETOS dejó de responder para 2
          servicios
          ✗ no estaba en la hipótesis
          causa   esos 2 servicios cacheaban el secreto al
                  arrancar; las instancias nuevas no
                  pudieron obtenerlo porque el punto
                  privado del almacén estaba en la zona 3
                                              clase 219

15:09:00  el equipo decide parar
15:09:40  zona restaurada

15:10-15:24  la recuperación NO fue automática
          ✗ predicción 5
          2 servicios quedaron con instancias marcadas como
          no sanas y hubo que reiniciarlas a mano
```

Los seis hallazgos:

#	hallazgo	tipo	acción
1	sin margen para perder una zona	capacidad	6 → 9 instancias
2	las 3 réplicas del buscador en la misma zona	reparto	restricción de reparto
3	punto privado del almacén de secretos en una sola zona	red	3 puntos, uno por zona
4	2 servicios cachean el secreto y no lo renuevan	código	renovación periódica
5	recuperación no automática	proceso	comprobación de salud corregida
6	nadie sabía qué hacer al ver los 503: no había procedimiento	proceso	escrito y probado

de infraestructura	0
de configuración, reparto, código y proceso	6

Y la segunda ejecución, dos meses después:

latencia p99 máxima	380 ms	✓
errores para el usuario	0	✓
búsqueda	degradada	✓
almacén de secretos	disponible	✓
recuperación	automática	✓
duración total del impacto	0 min	

El calendario que quedó:

semanal	anomalías de coste; ociosos retirados	
mensual	coste por pedido; cobertura y utilización; cumplimiento de iniciativas	
trimestral	1 experimento de resiliencia	
	simulación de técnicas de ataque	clase 226
	prueba de acceso de emergencia	clase 218
	revisión de accesos y de exenciones	
anual	ejercicio de pérdida de región	clase 215
	revisión de decisiones registradas	clase 190

El resultado, seis meses después:

coste mensual	31.400 €	23.100 €
coste atribuido	61 %	94 %
coste por pedido	0,071 €	0,049 €
utilización del compromiso	—	98 %
recursos ociosos	2.960 €	0 €
réplicas repartidas entre zonas	parcial	total
experimentos ejecutados	0	3
hallazgos con dueño y fecha	—	14

```
cerrados                                  —          13
```

La lección que esta clase deja: la recomendación automática proponía comprometerse tres años con veintidós máquinas de las que diecisiete no iban a existir en uno, y hacer las cosas en el orden correcto —retirar, redimensionar, apagar, y solo entonces comprometer— dio un ahorro mayor sin atarse a nada. Y el experimento de pérdida de zona encontró seis problemas, ninguno de infraestructura: los tres más graves fueron réplicas mal repartidas, un punto privado en una sola zona y dos servicios que cacheaban un secreto y no lo renovaban.

Laboratorio guiado

Ejecuta desde la raíz:

```
python classes/part-18-azure-production-architecture/227-cost-management-advisor-resiliencia-y-chaos-studio/lab.py
```

El laboratorio selecciona el motor de práctica finops y produce labresult.json. El escenario, sus comprobaciones y el artefacto esperado corresponden a esta clase; no requiere credenciales y deja explícito qué debe revalidarse en un sandbox real.

1. Lee exercise.steps y formula una predicción antes de ejecutar.
2. Ejecuta la práctica y verifica todos los elementos de checks.
3. Provoca el caso de negativetest y explica la señal observada.
4. Materializa azure-optimization en evidence/ usando la plantilla indicada.
5. Para proveedor real, sigue sandbox.requires, registra costo y ejecuta destroy.

Evidencia esperada

El artefacto principal es un cálculo trazable con unidad, supuesto y sensibilidad. Además, la entrega debe incluir el comando ejecutado, la salida estructurada y una conclusión que no exceda lo observado.

Reto verificable

Construye azure-optimization para el caso CloudShop. Incluye una alternativa descartada, un supuesto que pueda falsarse, una prueba de fallo y una decisión de rollback.

■ Criterio de aceptación

- [ ] lab.py termina con código 0 y genera JSON válido.
- [ ] La entrega conecta al menos tres requisitos con mecanismos verificables.
- [ ] Existe una prueba positiva y una prueba negativa con evidencia.
- [ ] Seguridad, costo y operación aparecen como decisiones, no como anexos.
- [ ] Se declara una limitación y una condición que obligaría a revisar el diseño.
- [ ] Otra persona puede repetir el recorrido sin conocimiento tácito.

Errores frecuentes

Síntoma	Causa probable	Corrección
Parte del gasto sigue sin atribuir pese a tener etiquetas obligatorias	Se confía en la herencia del grupo de recursos y hay servicios que no la respetan	Comprueba qué servicios heredan y usa una política de tipo modificar que etiquete al crear.
Se compra un compromiso y la utilización cae a los pocos meses	Se comprometió capacidad sobre cargas que iban a retirarse o a cambiar de forma	Retira, redimensiona y apaga primero; compromete solo la base estable medida y usa plan de ahorro si la forma va a cambiar.
Aplicar una recomendación de redimensionado degrada el servicio	La recomendación usa el uso medio y no el pico ni la estacionalidad	Comprueba el percentil alto y el comportamiento en campaña antes de aplicar.
Al perder una zona el servicio se degrada más de lo previsto	No hay margen de capacidad para perder una zona de tres	Dimensiona para que la capacidad restante quede por debajo del codo, y comprueba con un experimento.
Todas las réplicas de un servicio están en la misma zona	No hay restricción de reparto y el planificador las agrupó	Declara restricciones de reparto por zona y compruébalo inyectando la pérdida de una.
Un experimento de caos se convierte en un incidente	Alcance amplio, sin hipótesis ni forma de parar, y permisos permanentes para romper cosas	Hipótesis escrita, alcance restringido por etiqueta, parada en segundos y permisos por elevación temporal.

Seguridad, ética y costo

Trabaja con cuentas propias o sandboxes autorizados. No publiques secretos, identificadores reales ni datos personales. Antes de crear recursos pagos define presupuesto, etiquetas y comando de destrucción; después verifica que no queden recursos huérfanos. Los ejemplos locales enseñan contratos, pero no certifican cumplimiento ni disponibilidad de producción.

Preguntas de comprobación

1. ¿Por qué no todas las etiquetas se heredan y qué hacer al respecto?
2. ¿En qué orden hay que actuar antes de comprometer capacidad?
3. ¿Cuándo conviene una reserva y cuándo un plan de ahorro?
4. ¿Qué cinco experimentos de resiliencia encuentran más problemas?
5. ¿Qué ritmo de revisión propone esta clase y qué se hace en cada periodo?

Referencias

- Microsoft (2025). Azure Cost Management and Billing. <https://learn.microsoft.com/en-us/azure/cost-management-billing/>
- Microsoft (2025). Azure Advisor recommendations. <https://learn.microsoft.com/en-us/azure/advisor/advisor-overview>
- Microsoft (2025). Azure savings plans and reservations. <https://learn.microsoft.com/en-us/azure/cost-management-billing/savings-plan/savings-plan-compute-overview>
- Microsoft (2025). Azure Chaos Studio. <https://learn.microsoft.com/en-us/azure/chaos-studio/chaos-studio-overview>
- Microsoft (2025). Reliability and availability zones in Azure. <https://learn.microsoft.com/en-us/azure/reliability/availability-zones-overview>
- Documentación oficial vigente del servicio implementado; registra URL y fecha de consulta.

Descargar: Parte 18 en PDF · Recorrido de Azure en PDF · Manual integral

Evaluación — 227 Cost Management, Advisor, resiliencia y Chaos Studio

Preguntas

1. Explica el problema que resuelve esta capacidad sin mencionar un proveedor.
2. Identifica una frontera de responsabilidad y su propietario.
3. Compara dos alternativas mediante confiabilidad, seguridad, costo y operación.
4. ¿Qué demuestra el laboratorio y qué no puede demostrar?
5. Propón un caso límite o una prueba de fallo.

Reto verificable

Entrega azure-optimization con diagrama o configuración, evidencia de ejecución, alternativa descartada, riesgo residual y criterio de rollback.

Criterio de aceptación

- [ ] Resultado reproducible por una segunda persona.
- [ ] Evidencia vinculada a cada afirmación técnica.
- [ ] Prueba negativa o de fallo incluida.
- [ ] Costos y permisos expresados con unidades y alcance.
- [ ] Limitaciones declaradas sin presentar la simulación como producción.

Escala

Nivel	Evidencia
A — competente	Cumple todo y defiende trade-offs con datos.
B — en desarrollo	Cumple lo esencial; falta profundidad en una dimensión.
C — inicial	Artefacto parcial o conclusión sin evidencia suficiente.
0	Sin entrega reproducible.

228 — Proyecto: CloudShop productivo en Azure

Parte: 18 — Azure: arquitectura empresarial y operación en producción Nivel: avanzado · Horas estimadas: 8
Laboratorio: capstone · Estado: EXECUTABLECORE

Propósito

Poner en producción el sistema completo de CloudShop en Azure con lo de las once clases anteriores, y comprobarlo con las pruebas negativas de toda la parte. La clase da el orden, el entregable y los criterios. Y cierra la parte 18: corrige las cinco predicciones de la clase 216 —tres acertadas y dos a medias—, actualiza el recuento de leyes, añade la ley 27 y escribe la hipótesis de la parte 19, contestando además la pregunta incómoda que la clase 216 se dejó planteada.

Resultados de aprendizaje

Al finalizar podrás:

1. Construir el sistema completo en el orden que evita rehacer.
2. Comprobar con las pruebas negativas de toda la parte.
3. Comparar el resultado con el equivalente en AWS, con cifras.
4. Corregir las cinco predicciones de la clase 216 con evidencia.
5. Escribir la hipótesis de la parte 19 en forma refutable.

Conceptos centrales

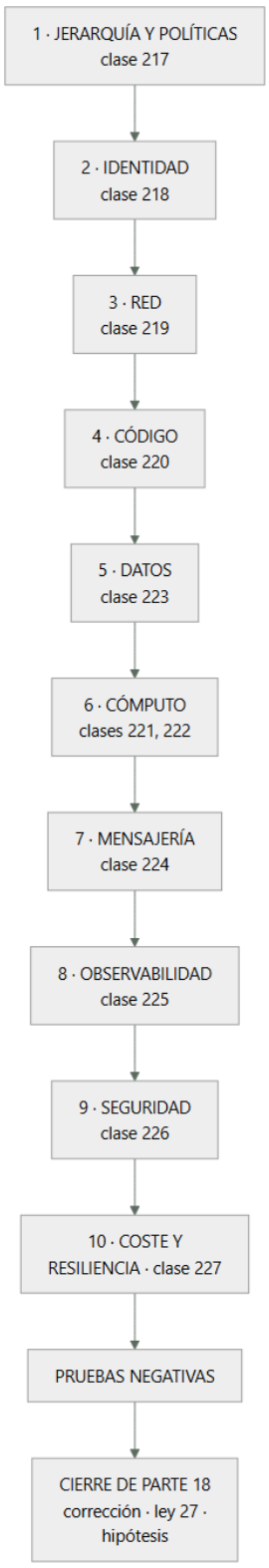
Concepto	Comprensión verificable
ley 27	Un control solo actúa sobre lo que cambia; lo que ya existe sigue incumpliendo hasta que alguien lo toque.
remediación	Tarea que corrige recursos existentes que incumplen. Es lo único que cierra la brecha que deja la ley 27.
equivalencia entre nubes	Correspondencia de conceptos. Alta en lo técnico, baja en el modelo operativo.
coste de traslado	Lo que cuesta llevar la misma arquitectura a otra nube. Se paga más en operación que en código.
prueba negativa de parte	Comprobación acumulada de las once clases, ejecutada sobre el sistema entero.
hipótesis de parte	Afirmación refutable escrita antes de estudiar, que la parte siguiente corrige con evidencia.

Modelo mental

La arquitectura Azure nace del tenant y las políticas heredables, y llega al workload mediante identidades administradas y redes privadas.

Aplicado a esta clase, separa siempre cuatro planos: intención (qué necesita el usuario), configuración (qué declaramos), estado observado (qué existe de verdad) y evidencia (cómo sabemos que cumple). Confundirlos produce diseños que se ven correctos en un diagrama pero fallan al operar.

■ Flujo de razonamiento



Desarrollo

1. El encargo, el orden y el entregable

El encargo. Llevar a producción la plataforma de pedidos de CloudShop en Azure, con usuarios reales, guardia, presupuesto y continuidad comprobada.

El orden, por coste de cambio:

```
1  JERARQUÍA Y POLÍTICAS                          clase 217
   grupos de administración, suscripciones por carga y
   entorno, iniciativas en auditoría antes de denegar
   → lo que llega tarde deja recursos incumpliendo
                                                    ley 27

2  IDENTIDAD                                      clase 218
   identidades administradas y federadas, sin secretos
   asignaciones en el ámbito del recurso
   elevación temporal y acceso de emergencia probado

3  RED                                            clase 219
   centro y radios, salida centralizada
   puntos privados con zonas DNS enlazadas a TODAS las
   redes

4  CÓDIGO                                         clase 220
   separado por ciclo de vida y ámbito, con módulos
   verificados y pilas de despliegue

5  DATOS                                          clase 223
   familia por patrones, consistencia por operación,
   índices recortados

6  CÓMPUTO                              clases 221, 222
   la opción gestionada que corresponda; clúster solo donde
   hace falta

7  MENSAJERÍA                                     clase 224
   el servicio que garantiza lo que hace falta, con
   idempotencia y fallidos vigilados

8  OBSERVABILIDAD                                 clase 225
   diagnóstico por política, instrumentación estándar

9  SEGURIDAD                                      clase 226
   postura por alcance, detección comprobada con simulación

10 COSTE Y RESILIENCIA                            clase 227
   atribución, compromisos tras retirar, experimentos
```

El entregable, con las piezas propias de esta parte:

```
1  el problema, con su cifra
2  jerarquía y iniciativas, con el estado de cumplimiento
3  inventario de identidades y su ALCANCE medido
4  topología de red, con las zonas DNS y su enlace
5  el código, con lo que gestiona cada pila
6  patrones de acceso y niveles de consistencia por
   operación
7  la lista de VALORES POR DEFECTO cambiados
8  observabilidad: qué se ingiere, en qué plan y qué cuesta
9  detección: qué técnicas se simulan y cuáles se detectan
10 coste atribuido y por unidad de negocio
11 experimentos de resiliencia y sus hallazgos
12 pruebas negativas, con los fallos publicados
13 lo que NO se hace, y por qué
```

Las pruebas negativas de la parte:

- quitar una política heredada desde una suscripción
- crear un recurso sin etiquetas obligatorias
- crear un recurso en una región no permitida
- obtener la identidad de otro espacio de nombres
- asumir una identidad federada desde otro repositorio
- activar un papel privilegiado sin aprobación
- entrar con la cuenta de acceso de emergencia
- alcanzar un radio desde otro radio
- resolver un nombre con punto privado desde cada red
- alcanzar un servicio con punto privado desde internet
- sacar datos a un destino no declarado
- desplegar quitando un recurso de la plantilla
- borrar la pila y comprobar que los datos sobreviven
- enviar el mismo mensaje 50 veces

- dejar un mensaje en la cola de fallidos y esperar la alerta
- perder una zona completa
- simular 14 técnicas de ataque
- provocar la condición de cada alerta
- desplegar el entorno de cero en una suscripción vacía

Y los criterios de evaluación, con el peso donde importa:

1	las políticas están en grupos de administración	3
2	se midió el cumplimiento antes de denegar	2
3	ninguna asignación amplia sin justificar	3
4	no quedan secretos de cliente sin excepción	3
5	las zonas DNS están enlazadas a todas las redes	3
6	la consistencia se decide por operación	2
7	la lista de valores por defecto cambiados es explícita	3
8	la instrumentación es estándar	2
9	se simulan técnicas y se publica la proporción	3
10	el coste atribuido supera el 90 %	2
11	hay experimentos de resiliencia ejecutados	3
12	las pruebas negativas se ejecutaron y hay fallos publicados	3

2. Cierre de la parte 18: corrección de las cinco predicciones

Las cinco predicciones de la clase 216, corregidas con la evidencia de las clases 217 a 227.

1. «los valores por defecto de Azure estarán mal elegidos en proporción parecida a los de AWS, pero fallarán en otro sitio: los de AWS son permisivos en red y almacenamiento; los de Azure lo serán en identidad y ámbito de permisos»

   PRIMERA MITAD CORRECTA: la lista de valores por defecto que hubo que cambiar es comparable a la de AWS. SEGUNDA MITAD FALLADA. Los peores valores por defecto de Azure no fueron de identidad: fueron de COSTE y de OBSERVABILIDAD. La consistencia fuerte del asistente costaba 1.720 €/mes; la política que indexa todos los campos, otros 1.020; y el 93 % de los recursos no enviaba ningún registro porque la configuración de diagnóstico no existe si no se crea. El problema de identidad fue real y grave, pero no era un valor por defecto: era un error de configuración que alguien tomó por comodidad.

2. «la jerarquía será el equivalente del plan de direcciones: se decide en una tarde, condiciona una década, y renumerarla costará meses»

   A MEDIAS, y la diferencia importa. Se decidió mal —47 suscripciones colgando de la raíz— y condicionó todo, como el plan de direcciones. Pero el mecanismo del coste es OTRO: mover las 61 suscripciones al grupo correcto fue instantáneo y gratis. Lo que costó cuatro meses fue corregir los recursos que ya existían, porque las políticas de denegación no actúan hacia atrás. En redes, lo caro es renumerar; aquí, lo caro es que el control nuevo no toca lo viejo. Y eso es una ley distinta.

3. «la identidad será el eje; el error más frecuente será un ámbito de asignación demasiado amplio, el equivalente del comodín de la clase 206»

   CORRECTA, y de forma casi literal. De 1.940 asignaciones, 820 estaban en la suscripción y 31 en grupos de administración; una sola identidad de canalización alcanzaba 14.200 recursos. Y en el clúster, la credencial federada atada solo al emisor permitía a cualquier cuenta de servicio de cualquier espacio de nombres obtenerla. El mecanismo cambia de nombre en cada nube y produce el mismo resultado.

4. «la mayoría de los conceptos se corresponderán uno a uno; lo que no se corresponderá es el modelo operativo, y trasladar la misma arquitectura costará más en operación

que en código»

CORRECTA. Se correspondieron: federación de identidad, puntos privados, centro y radios, infraestructura como código, contenedores, mensajería, observabilidad y coste. NO se correspondieron: el modelo de políticas, que aquí es mucho más fuerte y no actúa hacia atrás; el modelo de ámbitos de asignación; los cinco niveles de consistencia, que no tienen equivalente; los modos de despliegue; y la necesidad de crear el diagnóstico recurso a recurso. El código se tradujo en semanas; entender esas cinco diferencias operativas costó los tres primeros meses.

5. «los problemas volverán a ser de las leyes 25, 15 y 22; y si acierta por cuarta vez consecutiva, dejará de tener mérito y habrá que preguntarse por qué, sabiéndolo, sigue ocurriendo»

CORRECTA por cuarta vez, y toca contestar la pregunta. La evidencia: cuatro políticas quitadas por equipos sin que nadie se enterara; zonas DNS enlazadas a 4 de 43 redes durante siete meses; una configuración sin marcar como específica de ranura; el 93 % de los recursos sin diagnóstico; siete de catorce técnicas sin detectar con las reglas creadas; tres máquinas de un proyecto terminado en 2024.

Y la respuesta a la pregunta, que es lo que esta clase debe a la anterior:

```
las leyes 25, 15 y 22 no describen ERRORES
describen el ESTADO POR DEFECTO de cualquier sistema que
cambia

  lo provisional se queda        porque retirar es trabajo
                                 y nadie lo pide
  la señal no se mira            porque mirar es trabajo
                                 continuo y no urgente
  el procedimiento no funciona   porque ejecutarlo es
                                 trabajo y no hay incidente

→ no hace falta que nadie se equivoque para que se cumplan
→ hace falta que nadie haga algo, y eso ocurre siempre

y por eso saberlas no basta
  el equipo de este programa las conocía en las cuatro
  partes y las cuatro veces volvieron a ocurrir
  → lo único que las contrarresta es que la comprobación
    sea AUTOMÁTICA y PERIÓDICA
  → una persona que conoce la ley 15 sigue sin mirar el
    panel
  → una alerta de antigüedad, no
```

Marcador: tres correctas, dos a medias.

3. Recuento de leyes, ley 27 e hipótesis de la parte 19

El recuento de leyes, cerrada la parte 18.

```
ley 13  lo que no se mira deja de funcionar en silencio       50
ley 15  la señal existe y nadie la mira                       39
ley 22  un procedimiento nunca ejecutado no funciona          34
ley 14  el coste se decide al crear, no al pagar              31
ley 16  un control que estorba se rodea                       29
ley 20  lo que no tiene dueño se filtra y se desperdicia      28
ley 21  el acoplamiento vive en quién escribe                 23
ley 25  lo provisional sobrevive a su motivo                  18
ley 23  la capacidad la limita lo que ya se mantiene          16
ley 26  el valor por defecto sirve a la demostración          12
ley 24  lo que no está en el diagrama no se analiza           12
ley 17  se optimiza la medida, no el objetivo                 12
ley 19  la compensación hace invisible el fallo               10
ley 18  lo asíncrono traslada la garantía, no la elimina       8
```

Y la parte 18 obliga a escribir una ley nueva, que explica el fallo de la predicción 2:

```
LEY 27
  un control solo actúa sobre lo que cambia;
  lo que ya existe sigue incumpliendo hasta que alguien
  lo toque

apariciones en esta parte                                    5
  clase 217   mover 61 suscripciones al grupo correcto
              corrigió CERO recursos existentes; 14.200
              siguieron incumpliendo
  clase 219   la automatización de zonas DNS no enlazó las
              creadas antes; 39 radios quedaron fuera
  clase 220   el despliegue incremental nunca borra lo que
              deja de estar declarado: 1.140 huérfanos
  clase 222   activar la política de red no cambia lo ya
              desplegado, y en algunas configuraciones
              exige recrear el clúster
  clase 226   cerrar una recomendación no impide que el
              recurso siguiente nazca igual

y lo que la distingue
  la ley 25 dice que lo provisional se queda
  la ley 26 dice que el valor inicial está mal elegido
  la 27 dice algo distinto: que ARREGLAR LA REGLA no
  arregla lo que ya se hizo con la regla anterior

  → y el remedio no es una política mejor: es una TAREA DE
    REMEDIACIÓN, que es trabajo, y hay que planificarlo
  → toda implantación de un control necesita dos planes:
    el de lo nuevo y el de lo viejo
```

La hipótesis de la parte 19 (clases 229 a 240, Google Cloud en producción), escrita antes de estudiarla:

```
1. la tercera nube confirmará la equivalencia conceptual y
   volverá a diferir en el modelo operativo; y esta vez lo
   que más costará no será aprender lo distinto sino
   DESAPRENDER lo de las dos anteriores: los errores serán
   de trasladar suposiciones que aquí no valen

2. la jerarquía de proyectos y la red compartida serán otra
   vez la decisión que condiciona todo, con un matiz: aquí
   el proyecto es una frontera más fuerte que la suscripción,
   y eso hará que el error frecuente sea el CONTRARIO —
   demasiados proyectos, no demasiado pocos

3. la identidad volverá a ser el eje, y el error más
   frecuente volverá a ser el ámbito amplio: una asignación
   en la organización o en la carpeta en vez de en el
   recurso                                      ley 27, 218

4. los valores por defecto volverán a estar mal elegidos para
   producción, y aquí fallarán sobre todo en RED: la red
   global y los servicios con acceso público por defecto
   serán la fuente principal                      ley 26

5. y la predicción que ya no tiene mérito, con su corolario:
   los problemas del proyecto serán otra vez de las leyes
   25, 15 y 22. Lo que sí es refutable es esto: **la
   proporción de hallazgos que detecte una comprobación
   automática y periódica será mayor que en las partes 17 y
   18**, porque en esas dos aprendimos a montarlas. Si no lo
   es, el aprendizaje no se está trasladando
```

Y el cierre de la parte 18: de once clases, lo que más dinero movió no fue ninguna decisión de arquitectura sino dos casillas de un asistente de creación —el nivel de consistencia y la política de índices—, y lo que más riesgo concentró fue una sola asignación de permisos con el ámbito demasiado alto. La parte 19 hace el mismo recorrido en Google Cloud, empezando por su jerarquía de recursos y su red compartida. Es la clase 229.

4. Comparar las dos nubes, con cifras

Con el mismo sistema montado en dos nubes, se puede comparar de verdad. Y conviene, porque es la única forma de separar lo que es propio del proveedor de lo que es propio del método.

```
LO QUE RESULTÓ EQUIVALENTE
```

```
  el techo de disponibilidad se calcula igual   clase 185
  la aritmética de colas y concurrencia         clase 186
  la decisión de consistencia por operación     clase 187
  la idempotencia y la cola de fallidos    clases 210, 224
  la disciplina de despliegue escalonado
  la estructura del código por ciclo de vida
  y las pruebas negativas, una a una

LO QUE NO
  el modelo de gobierno: en Azure las políticas se heredan
    y son fuertes; en AWS las barreras y los controles
    están más repartidos
  el modelo de permisos: papel más ámbito frente a política
    con condiciones
  la observabilidad: aquí hay que crear el diagnóstico
    recurso a recurso
  y la consistencia configurable, que no tiene equivalente
```

Y la comparación de esfuerzo, medida:

```
                              AWS         Azure
tiempo hasta el primer
  despliegue productivo      9 semanas    7 semanas
  → más rápido la segunda vez, por el método, no por la
    nube

líneas de infraestructura
  como código                ~4.100      ~3.800

valores por defecto que hubo
  que cambiar                    24          19

pruebas negativas ejecutadas     19          19
  que fallaron la primera vez     7           8

coste mensual del mismo sistema  16.700 €   17.400 €
  → diferencia del 4 %, dentro del ruido

tiempo dedicado a aprender el
  modelo OPERATIVO               —      3 meses
  → y esto es lo que la predicción 4 acertaba
```

Y las dos conclusiones que se pueden defender:

```
1  el coste de un sistema equivalente es parecido
   → elegir nube por precio de lista es un error
   → lo que cambia el coste son las decisiones de diseño,
     no el proveedor                      clases 216, 227

2  el coste de OPERAR dos nubes no es el doble: es más
   dos modelos de gobierno, dos de permisos, dos de
   observabilidad, dos calendarios de actualización
   → y por eso la multinube se justifica por requisitos,
     no por preferencia                      clase 157
```

Y una advertencia sobre el traslado:

```
las suposiciones de la nube anterior son el mayor riesgo
  «esto se hereda» — aquí no
  «esto ya viene activado» — aquí no
  «el borrado no se propaga» — aquí sí
→ y por eso el orden de la parte empieza otra vez por lo
  que condiciona todo, en vez de por lo que ya se sabe
```

Y la lista de comprobación de la clase:

- ■ el sistema se construyó en el orden por coste de cambio
- ■ hay lista explícita de valores por defecto cambiados
- ■ el alcance de cada identidad está medido
- ■ las zonas DNS están enlazadas a todas las redes
- ■ la consistencia está decidida por operación
- ■ la instrumentación es estándar
- ■ se simulan técnicas y se publica la proporción detectada
- ■ hay plan de remediación de lo existente, no solo de lo nuevo

- las 19 pruebas negativas se ejecutaron y hay fallos publicados
- el entorno se despliega de cero desde el código
- el coste está atribuido por encima del 90 %
- está escrito lo que no se hace

Y el cierre que enlaza con la clase siguiente: la parte 19 repite el recorrido en Google Cloud, con el riesgo añadido de trasladar suposiciones que allí no valen. Empieza por la jerarquía de recursos y la red compartida, en la clase 229.

Ejemplo trabajado

El sistema de CloudShop en producción en Azure. Lo que sigue es la lista de valores por defecto cambiados, el resultado de las diecinueve pruebas negativas —de las que fallaron ocho— y la comparación con el mismo sistema en AWS.

La lista de valores por defecto cambiados:

servicio	por defecto	cambiado a
base distribuida	consistencia fuerte	acotada 5 s, sesión y eventual por petición
base distribuida	indexa todos los campos	7 de 61
base distribuida	1 región	2 (lectura)
grupo de seguridad	salida a internet permitida	ruta al cortafuegos
recursos	sin diagnóstico	por política
registros	retención global	por tabla
registros	plan analítico	básico donde procede
app service	acceso público	punto privado
app service	salida pública	integración de red
app service	config no específica de ranura	marcada por ranura
clúster	política de red desactivada	activada
clúster	red plana (asistente)	superpuesta
bus	bloqueo 30 s	120-300 s
bus	sin renovación de bloqueo	activada
flujo	retención 1 día	3 días
flujo	posición antes de procesar	después
despliegue	modo incremental sin ciclo de vida	pilas de despliegue
políticas	asignadas en suscripción	en grupos de administración
alertas	grupos por equipo	uno por turno
total		19
que funcionaban sin cambiarlos		19

Las diecinueve pruebas negativas: ocho fallaron.

✓	quitar una política heredada desde suscripción	imposible
✓	crear recurso sin etiquetas	rechazado
✓	crear en región no permitida	rechazado
✗	obtener la identidad de otro espacio de nombres → 1 de 7 credenciales federadas ataba solo el emisor: la del operador de base vectorial, añadida después de la revisión	clase 222
✓	asumir identidad federada desde otro repositorio	denegado
✗	activar papel privilegiado sin aprobación → 2 papeles se habían añadido como elegibles sin exigir aprobación, al crear un equipo nuevo	
✓	entrar con la cuenta de emergencia	45 s
✓	alcanzar un radio desde otro radio	denegado
✗	resolver nombre con punto privado desde cada red → 2 radios creados el mes anterior no tenían las zonas enlazadas: la automatización se había desplegado DESPUÉS de crearlos	ley 27, 219

```
✓  alcanzar servicio con punto privado desde internet
                                                  denegado
✗  sacar datos a un destino no declarado
   → funcionó desde el clúster: la política de red
     restringía entre pods, no la salida a internet
                                               clase 222
✓  desplegar quitando un recurso de la plantilla   borrado
✗  borrar la pila y comprobar los datos
   → una cuenta de almacenamiento de un servicio nuevo no
     tenía bloqueo de borrado ni estaba en la pila de datos
✓  mismo mensaje 50 veces                          1 efecto
✓  mensaje en cola de fallidos → alerta             38 s
✗  perder una zona completa
   → 12 min 40 s de degradación: dos servicios añadidos
     tras el último experimento no tenían restricción de
     reparto                                   clase 227
✗  simular 14 técnicas de ataque                  13 de 14
   → la que falta, aceptada por escrito
✗  provocar la condición de cada alerta
   → 4 de 58 no llegaron; 3 por umbral y 1 en estado de
     error                                     clase 225
✓  desplegar el entorno de cero                    94 min
```

Y el análisis de las ocho:

```
seis por elementos AÑADIDOS después de la última revisión
  · una credencial federada
  · dos papeles elegibles
  · dos radios sin zonas enlazadas
  · una cuenta de almacenamiento sin bloqueo
  · dos servicios sin restricción de reparto
dos por un control que no cubría lo que se creía
  · política de red que no restringe la salida
  · umbrales de alerta

→ y es el mismo diagnóstico de la clase 216: el sistema
  creció y las comprobaciones no crecieron con él
→ con un matiz nuevo: cuatro de los seis casos son de la
  ley 27, porque la automatización llegó después que el
  recurso
```

Y la corrección de método que salió:

```
toda automatización nueva se acompaña de una TAREA DE
REMEDIACIÓN sobre lo existente
y una comprobación periódica que compare el estado real
con el esperado, no solo en el momento de crear

→ y esa comprobación encontró, en los tres meses
  siguientes
    5 radios sin zonas enlazadas
    3 recursos sin bloqueo de borrado
    2 credenciales federadas mal atadas
  → todos, creados después de la automatización
    correspondiente
```

Las cifras del sistema en producción, tras tres meses:

```
p99 del flujo de compra                    < 500 ms    438 ms
disponibilidad observada                     99,8 %    99,84 %
coste mensual                             15.000 €   17.400 €
coste por pedido                            0,050 €    0,051 €
pérdida de una zona: degradación             0 min    12 min 40
  (tras corregir)                                        0 min
coste atribuido                               90 %      94 %
alertas por turno                             < 2        0,9
técnicas simuladas detectadas                 > 90 %    13/14
despliegue del entorno de cero                  —       94 min
```

La comparación con AWS, del mismo sistema:

```
p99 del flujo de compra                   412 ms      438 ms
disponibilidad observada                  99,86 %     99,84 %
coste mensual                          16.700 €    17.400 €
coste por pedido                        0,046 €     0,051 €
```

```
valores por defecto cambiados                24          19
pruebas negativas fallidas                    7           8
tiempo hasta producción                9 semanas   7 semanas
líneas de infraestructura                ~4.100      ~3.800
equipo dedicado a la plataforma            2,1         2,4
  → la diferencia está en el modelo operativo, no en la
    tecnología
```

Y las tres cosas que se decidió no hacer:

```
no operar las dos nubes en activo-activo: el coste de
  operar dos modelos de gobierno no compensa   clase 157
no migrar las cargas de AWS: funcionan
no unificar la observabilidad en un solo proveedor
  todavía: se revisa cuando el volumen lo justifique
```

La lección que este proyecto deja: los diecinueve valores por defecto cambiados funcionaban todos sin cambiarlos, y dos de ellos —dos casillas de un asistente— costaban dos mil setecientos euros al mes. De las ocho pruebas negativas fallidas, cuatro fueron por recursos creados antes de que existiera la automatización que los cubriría, que es la ley 27 en su forma más directa. Y comparado con AWS, el mismo sistema costó un 4 % más y necesitó un 14 % más de equipo: la diferencia está en operar dos modelos, no en la tecnología.

Laboratorio guiado

Ejecuta desde la raíz:

```
python classes/part-18-azure-production-architecture/228-proyecto-cloudshop-productivo-en-azure/lab.py
```

El laboratorio selecciona el motor de práctica capstone y produce labresult.json. El escenario, sus comprobaciones y el artefacto esperado corresponden a esta clase; no requiere credenciales y deja explícito qué debe revalidarse en un sandbox real.

1. Lee exercise.steps y formula una predicción antes de ejecutar.
2. Ejecuta la práctica y verifica todos los elementos de checks.
3. Provoca el caso de negativetest y explica la señal observada.
4. Materializa cloudshop-azure en evidence/ usando la plantilla indicada.
5. Para proveedor real, sigue sandbox.requires, registra costo y ejecuta destroy.

Evidencia esperada

El artefacto principal es un incremento integrado, demostrable y documentado. Además, la entrega debe incluir el comando ejecutado, la salida estructurada y una conclusión que no exceda lo observado.

Reto verificable

Construye cloudshop-azure para el caso CloudShop. Incluye una alternativa descartada, un supuesto que pueda falsarse, una prueba de fallo y una decisión de rollback.

■ Criterio de aceptación

- [ ] lab.py termina con código 0 y genera JSON válido.
- [ ] La entrega conecta al menos tres requisitos con mecanismos verificables.
- [ ] Existe una prueba positiva y una prueba negativa con evidencia.
- [ ] Seguridad, costo y operación aparecen como decisiones, no como anexos.
- [ ] Se declara una limitación y una condición que obligaría a revisar el diseño.
- [ ] Otra persona puede repetir el recorrido sin conocimiento tácito.

Errores frecuentes

Síntoma	Causa probable	Corrección
Se implanta un control y los recursos existentes siguen incumpliendo	Los controles actúan sobre creaciones y modificaciones, no hacia atrás	Toda implantación necesita dos planes: el de lo nuevo y una tarea de remediación de lo existente.
Una automatización cubre unos recursos y no otros	Los recursos anteriores a la automatización no pasaron por ella	Ejecuta la automatización sobre lo existente y añade una comprobación periódica que compare el estado real con el esperado.
Al trasladar la arquitectura a otra nube fallan cosas que en la anterior funcionaban	Se trasladaron suposiciones del modelo operativo anterior	Empieza por lo que condiciona todo en la nube nueva y comprueba explícitamente qué se hereda, qué viene activado y qué se propaga.
Las pruebas negativas pasan y aparecen huecos nuevos	El sistema creció y las comprobaciones no crecieron con él	Añade una prueba negativa por cada capacidad nueva y ejecútalas periódicamente, no solo al terminar.
Se elige la nube por precio de lista y no se ahorra	El coste lo deciden las decisiones de diseño, no el proveedor	Compara el coste del mismo sistema montado bien en ambas; la diferencia suele estar dentro del ruido.
Operar dos nubes consume mucho más equipo del previsto	Se contó el código y no los modelos de gobierno, permisos, observabilidad y actualizaciones	Justifica la multinube por requisitos y no por preferencia, contando el coste operativo real.

Seguridad, ética y costo

Trabaja con cuentas propias o sandboxes autorizados. No publiques secretos, identificadores reales ni datos personales. Antes de crear recursos pagos define presupuesto, etiquetas y comando de destrucción; después verifica que no queden recursos huérfanos. Los ejemplos locales enseñan contratos, pero no certifican cumplimiento ni disponibilidad de producción.

Preguntas de comprobación

1. ¿Cuál de las cinco predicciones de la clase 216 falló y en qué mitad?
2. ¿Qué dice la ley 27 y en qué se distingue de las leyes 25 y 26?
3. ¿Por qué saber las leyes 25, 15 y 22 no basta para evitar que se cumplan?
4. ¿Qué se correspondió entre las dos nubes y qué no?
5. ¿Cuántas de las pruebas negativas fallidas se deben a la ley 27?

Referencias

- Microsoft (2025). Azure Well-Architected Framework. <https://learn.microsoft.com/en-us/azure/well-architected/>
- Microsoft (2025). Cloud Adoption Framework: Azure landing zones. <https://learn.microsoft.com/en-us/azure/cloud-adoption-framework/ready/landing-zone/>
- Microsoft (2025). Azure Policy remediation tasks. <https://learn.microsoft.com/en-us/azure/governance/policy/how-to/remediate-resources>
- Beyer, B. y otros (2018). The Site Reliability Workbook. <https://sre.google/workbook/table-of-contents/>
- Basiri, A. y otros (2016). Chaos engineering. <https://ieeexplore.ieee.org/document/7503833>
- Documentación oficial vigente del servicio implementado; registra URL y fecha de consulta.

Descargar: Parte 18 en PDF · Recorrido de Azure en PDF · Manual integral

Evaluación — 228 Proyecto: CloudShop productivo en Azure

Preguntas

1. Explica el problema que resuelve esta capacidad sin mencionar un proveedor.
2. Identifica una frontera de responsabilidad y su propietario.
3. Compara dos alternativas mediante confiabilidad, seguridad, costo y operación.
4. ¿Qué demuestra el laboratorio y qué no puede demostrar?
5. Propón un caso límite o una prueba de fallo.

Reto verificable

Entrega cloudshop-azure con diagrama o configuración, evidencia de ejecución, alternativa descartada, riesgo residual y criterio de rollback.

Criterio de aceptación

- [ ] Resultado reproducible por una segunda persona.
- [ ] Evidencia vinculada a cada afirmación técnica.
- [ ] Prueba negativa o de fallo incluida.
- [ ] Costos y permisos expresados con unidades y alcance.
- [ ] Limitaciones declaradas sin presentar la simulación como producción.

Escala

Nivel	Evidencia
A — competente	Cumple todo y defiende trade-offs con datos.
B — en desarrollo	Cumple lo esencial; falta profundidad en una dimensión.
C — inicial	Artefacto parcial o conclusión sin evidencia suficiente.
0	Sin entrega reproducible.

Parte 20 — Plataformas cloud de datos, analítica, IA y agentes

248 — Bedrock, Azure AI Foundry y Vertex AI

Parte: 20 — Plataformas cloud de datos, analítica, IA y agentes Nivel: avanzado · Horas estimadas: 4 Laboratorio: decision · Estado: EXECUTABLECORE

Propósito

Elegir dónde y cómo ejecutar modelos fundacionales entre las tres nubes, con los criterios que de verdad diferencian: qué se hace con los datos que se envían, si el modelo puede cambiar bajo los pies, cómo se controla el gasto y cuánto cuesta cambiar de proveedor. La clase compara las plataformas gestionadas por lo que aportan y lo que atan, y da el patrón que reduce el coste de cambio.

Resultados de aprendizaje

Al finalizar podrás:

1. Comparar las plataformas gestionadas por criterios operativos, no por catálogo.
2. Comprobar qué hace cada proveedor con los datos enviados.
3. Fijar versiones y anticipar las retiradas de modelos.
4. Controlar el gasto con cuotas, presupuestos y enrutado.
5. Reducir el coste de cambiar de proveedor sin construir una abstracción inútil.

Conceptos centrales

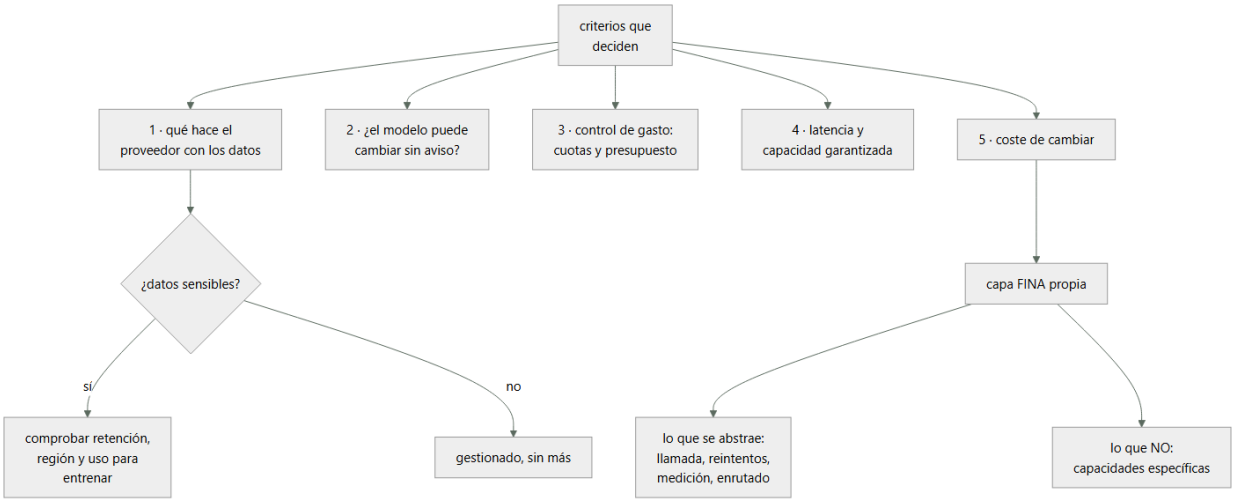
Concepto	Comprensión verificable
plataforma gestionada de modelos	Servicio que expone modelos propios y de terceros con una interfaz común, gobierno y facturación integrada.
retención de datos del proveedor	Qué guarda y durante cuánto tiempo lo que se le envía. Determina si se puede usar con datos sensibles.
versión fijada	Referencia a una versión concreta del modelo, para que no cambie sin aviso.
retirada de modelo	Fecha en que una versión deja de estar disponible. Ocurre y hay que planificarla.
capacidad reservada	Compromiso de rendimiento garantizado, frente al pago por uso con cuotas compartidas.
abstracción de proveedor	Capa que oculta las diferencias entre proveedores. Barata si es fina, inútil si intenta cubrirlo todo.

Modelo mental

Una plataforma de IA sigue siendo un sistema de datos: necesita procedencia, evaluación, límites de costo, seguridad y operación antes de una interfaz inteligente.

Aplicado a esta clase, separa siempre cuatro planos: intención (qué necesita el usuario), configuración (qué declaramos), estado observado (qué existe de verdad) y evidencia (cómo sabemos que cumple). Confundirlos produce diseños que se ven correctos en un diagrama pero fallan al operar.

■ Flujo de razonamiento



Desarrollo

1. Los cinco criterios que deciden

Las tres nubes ofrecen plataformas equivalentes en el catálogo. Lo que diferencia es operativo.

```
1  ¿QUÉ HACE EL PROVEEDOR CON LOS DATOS?
   ¿los guarda? ¿cuánto tiempo? ¿en qué región?
   ¿los usa para entrenar?
   ¿hay opción de retención cero?
   → y esto se comprueba EN EL CONTRATO, no en la página de
     producto                                  clase 251
   → es el criterio que más decisiones bloquea

2  ¿EL MODELO PUEDE CAMBIAR SIN AVISO?
   un alias que apunta a «la última versión» cambia solo
   → y el comportamiento cambia con él
   → hay que poder FIJAR la versión
   → y saber cuándo se retira                    ley 25

3  ¿CÓMO SE CONTROLA EL GASTO?
   cuotas por proyecto y por identidad
   presupuestos con alerta y con acción      clase 214
   → sin esto, una prueba mal hecha cuesta miles

4  ¿QUÉ CAPACIDAD Y QUÉ LATENCIA ESTÁN GARANTIZADAS?
   el pago por uso comparte cuota con otros clientes
   → y en momentos de demanda, hay limitación
   → la capacidad reservada garantiza rendimiento y cuesta
     aunque no se use

5  ¿CUÁNTO CUESTA CAMBIAR?
   ← y este suele ser el que menos se mira
```

Y lo que resulta equivalente entre las tres:

```
modelos de calidad parecida
precios del mismo orden
interfaces distintas pero conceptualmente iguales
integración con la identidad y la red de su nube
y herramientas de evaluación y de filtrado

→ y por eso elegir por catálogo de modelos es elegir por lo
  que menos diferencia                          clase 240
```

Y el criterio práctico:

```
¿ya se opera en una de las tres?
  → empieza por la suya: la identidad, la red, el registro
    y el coste ya están resueltos

¿hay un modelo concreto que solo está en una?
  → y ¿es realmente mejor para ESTE caso, medido?
```

```
                                                clase 250

¿hay requisito de residencia o de retención?
  → ese criterio manda sobre los demás      clase 251
```

2. Los datos, la versión y la retirada

Lo que hay que comprobar sobre los datos, con las preguntas concretas:

```
¿SE GUARDAN LAS PETICIONES Y LAS RESPUESTAS?
  y si sí, ¿cuánto tiempo y con qué acceso?
  → algunos proveedores guardan por defecto para
    supervisión de abuso, con retención de días
  → y hay opciones de retención cero, a veces con
    condiciones

¿SE USAN PARA ENTRENAR?
  en las plataformas empresariales, normalmente no
  → y hay que verlo escrito

¿EN QUÉ REGIÓN SE PROCESA?
  el modelo puede ejecutarse en otra región que el resto
  → y eso rompe un requisito de residencia    clase 177

¿QUÉ SE REGISTRA POR TU LADO?
  las peticiones a un modelo contienen lo que el usuario
  escribió
  → y eso puede incluir datos personales
  → los registros propios son un dato sensible clase 211
```

Y la decisión que resulta:

```
DATOS NO SENSIBLES        → gestionado, sin más
DATOS SENSIBLES           → gestionado CON retención cero y
                            región comprobada
DATOS QUE NO PUEDEN SALIR → modelo abierto alojado por uno
                            mismo               clase 245

y el patrón mixto
  seudonimizar antes de enviar
  → sustituir nombres, correos y números por marcadores
  → y restituirlos en la respuesta
  → resuelve muchos casos sin renunciar al gestionado
```

La versión, que es donde aparece la sorpresa:

```
✗ APUNTAR AL ALIAS «último»
  el proveedor actualiza y el comportamiento cambia
  → las instrucciones que funcionaban dejan de funcionar
  → y la calidad puede subir o bajar sin aviso

✓ FIJAR LA VERSIÓN
  y actualizarla como cualquier dependencia: evaluando
  antes                                      clase 250

Y LA RETIRADA
  las versiones se retiran, con aviso de meses
  → hay que tener la fecha en un calendario     ley 25
  → y una evaluación preparada para comparar la siguiente
  → porque la migración no es «cambiar el nombre»: hay que
    revalidar
```

Y lo que hay que tener listo para el día de la migración:

```
un conjunto de evaluación propio, con casos reales
                                                clase 250
las instrucciones versionadas
y una comparación automática entre la versión actual y la
  nueva
→ con eso, migrar es un día; sin eso, es un proyecto
```

3. Gasto, capacidad y límites

El gasto se descontrola con una facilidad notable, y por los mismos motivos de la ley 28.

```
LO QUE HAY QUE PONER ANTES DE EMPEZAR
```

```
  cuota por proyecto y por identidad
  presupuesto con alerta al 50, 80 y 100 %
  límite de testigos por petición
  y límite de peticiones por usuario

→ sin esto, un bucle mal escrito en una prueba consume el
  presupuesto del trimestre en una tarde
→ y ha pasado en todas las organizaciones que empiezan
```

Y las palancas, que son las de la clase 247:

```
enrutar: no llamar al modelo cuando no hace falta
respuestas cortas
caché de peticiones repetidas
contexto justo
modelo pequeño para lo fácil, grande para lo difícil
y caché de contexto, donde exista

→ y la medida que importa: coste por operación de negocio
  resuelta, no por petición                clases 214, 245
```

La capacidad, con la decisión de reservar o no:

```
PAGO POR USO
  + sin compromiso; se paga lo que se usa
  − cuota compartida: en picos de demanda global, hay
    limitación
  − y la limitación aparece como error, que hay que
    manejar

CAPACIDAD RESERVADA
  + rendimiento garantizado y latencia estable
  − se paga aunque no se use
  → y hay que dimensionarla con el tráfico real

CRITERIO
  camino crítico con usuarios esperando  → reservada
  procesos por lotes y usos internos     → por uso
  → y la mezcla es lo habitual
```

Y el manejo de la limitación, que hay que implementar:

```
reintentos con retroceso y dispersión
y un plazo total: si no se consigue en X, respaldo
                                              clase 245
y NUNCA reintentar sin límite: multiplica la carga en el
  peor momento                              clase 186
```

Y la vigilancia:

```
testigos de entrada y de salida, por caso de uso
coste por caso de uso y por identidad
peticiones limitadas por cuota
latencia por percentil, y hasta el primer testigo
errores del proveedor
y la proporción servida desde caché o desde el enrutado
```

4. El coste de cambiar, y la capa fina

La pregunta que casi nadie hace al empezar: ¿cuánto costaría cambiar de proveedor?

```
LO QUE ATA, por orden de coste
  1  LOS EMBEBIDOS y su índice
     cambiar de modelo de embebido obliga a reindexar todo
     → y con millones de fragmentos, eso es tiempo y dinero
                                              clase 247
  2  LAS INSTRUCCIONES ajustadas a un modelo concreto
     → lo que funciona en uno no funciona igual en otro
     → y hay que revalidar                  clase 250
  3  las capacidades específicas: llamada a herramientas,
     salidas estructuradas, caché de contexto
     → cada proveedor lo hace distinto
  4  la integración con el gobierno de esa nube
  5  y el código de llamada, que es lo más barato
```

Y la conclusión, que es la de la clase 158:

```
LO QUE MÁS ATA NO ES EL CÓDIGO
→ y por eso una abstracción que solo oculta la llamada no
  reduce el coste de cambio de forma apreciable
```

La capa fina que sí conviene, con lo que debe y no debe hacer:

```
LO QUE ABSTRAE
  la llamada al modelo, con reintentos y plazos
  la MEDICIÓN: testigos, coste, latencia, por caso de uso
  el enrutado entre modelos                clase 247
  el registro de peticiones y respuestas, con muestreo
  la aplicación de filtros de entrada y de salida
                                                clase 249
  y la versión de la instrucción usada

LO QUE NO ABSTRAE
  las capacidades específicas de cada proveedor
  → si se usan, se usan; y se registra la decisión
                                                clase 190

→ una capa de 200 líneas que hace lo primero vale mucho
→ una que intenta cubrir todas las capacidades de todos los
  proveedores acaba siendo el mínimo común y estorbando
                                                clase 158
```

Y las decisiones que reducen el coste de cambio de verdad:

```
MANTENER UN CONJUNTO DE EVALUACIÓN PROPIO
  → permite comparar cualquier modelo con datos propios
  → es lo que más reduce el coste de cambiar   clase 250

GUARDAR LOS TEXTOS ORIGINALES, no solo los embebidos
  → reindexar es posible; recuperar el original desde un
    vector, no

VERSIONAR LAS INSTRUCCIONES como código
  → y probarlas contra el conjunto de evaluación

Y EVITAR ATAR EL ÍNDICE VECTORIAL AL PROVEEDOR DEL MODELO
  → el índice puede vivir aparte y servirse a cualquiera
```

Y la lista de comprobación de la clase:

- ■ está comprobado por escrito qué hace el proveedor con los datos
- ■ la región de procesamiento cumple el requisito
- ■ los registros propios de peticiones se tratan como dato sensible
- ■ la versión del modelo está fijada
- ■ la fecha de retirada está en un calendario
- ■ hay conjunto de evaluación propio para comparar versiones
- ■ hay cuotas por proyecto e identidad
- ■ hay presupuesto con alerta y acción
- ■ hay límite de testigos y de peticiones por usuario
- ■ la limitación se maneja con retroceso y respaldo
- ■ la capacidad reservada cubre el camino crítico
- ■ se mide coste por operación de negocio, no por petición
- ■ hay capa fina propia que mide, enruta y filtra
- ■ los textos originales se conservan además de los embebidos
- ■ las instrucciones están versionadas y probadas

Y el cierre que enlaza con la clase siguiente: con el modelo elegido y controlado, aparece lo que más cambia el riesgo: darle herramientas para que actúe. Agentes, permisos y contención es la materia de la clase 249.

Ejemplo trabajado

CloudShop decide dónde ejecutar sus modelos fundacionales. Lo que sigue es la comparación por criterios operativos, la prueba que consumió el presupuesto del trimestre en una tarde, y la retirada de versión que pilló al equipo sin conjunto de evaluación.

La comparación, hecha por los cinco criterios:

```
criterio                    nube A     nube B     nube C
```

```
datos: retención por        30 días    0 días     30 días
  defecto                              (opción)   (opción
                                                  de 0)
datos: uso para entrenar    no         no         no
región de procesamiento     UE         UE         UE en 2
  garantizada                                     de 4
                                                  modelos
versión fijable             sí         sí         sí
aviso de retirada           6 meses    12 meses   6 meses
cuotas por proyecto         sí         sí         sí
capacidad reservada         sí         sí         sí
integración con su
  identidad y red           sí         sí         sí
modelos disponibles         propios y  propios y  propios
                            de terceros de terceros
```

```
y el criterio que decidió
  CloudShop opera principalmente en la nube A  clase 240
  → identidad, red, registro y coste ya resueltos
  → y los modelos son equivalentes para sus casos, medido
    con su conjunto de evaluación             clase 250

decisión
  nube A para el asistente interno y el de atención
  y un modelo abierto alojado para el caso con datos que no
  pueden salir                                ← ver abajo
```

Y el caso que obligó a alojar:

```
el equipo legal quería un asistente sobre los contratos con
socios
  → contienen condiciones comerciales confidenciales y
    cláusulas de no divulgación
  → tres de los 41 contratos prohíben expresamente enviar
    su contenido a terceros                   clase 251

decisión
  modelo abierto, alojado en la propia nube, con
  aceleradores
  coste                                  1.900 €/mes
  → frente a ~180 €/mes con el gestionado
  → y se aceptó, porque el requisito es contractual

y lo que se registró
  «si los tres contratos se renegocian y desaparece la
   cláusula, se revisa»                       clase 190
```

La prueba que consumió el presupuesto.

```
un equipo probaba un procesamiento de documentos
escribió un bucle que, ante un error, reintentaba sin
límite
y el error era un documento que el modelo no podía procesar

  duración del bucle                          4 horas
  peticiones                                410.000
  coste                                    18.400 €
  → el presupuesto trimestral del área era de 15.000 €

y no había
  cuota por proyecto
  presupuesto con alerta
  ni límite de peticiones

se detectó
  al día siguiente, en la revisión de coste     ley 15

correcciones
  cuota por proyecto y por identidad
  presupuesto con alerta al 50, 80 y 100 %, y acción al
  120 % en entornos no productivos        clase 214
  límite de testigos por petición
  límite de peticiones por identidad y hora
  y en la capa fina, reintentos con límite y retroceso
```

```
y la comprobación
  se repitió el escenario en preproducción
  → el bucle se detuvo a las 200 peticiones
  → coste                                        9 €
```

La retirada de versión.

```
el proveedor anunció la retirada de la versión que el
asistente usaba, con 6 meses de aviso

lo que el equipo tenía
  la versión, fijada                              ✓
  el aviso, en un correo que nadie leyó           ✗
  conjunto de evaluación propio                   ✗
  instrucciones versionadas                       ~

qué pasó
  se enteraron 3 semanas antes de la fecha
  → migraron a la versión nueva sin evaluar
  → y a los 4 días, las quejas del equipo de atención

  qué había cambiado
    la versión nueva daba respuestas más largas
      → coste por consulta +41 %
    y era más reacia a decir «no lo sé»
      → 3 casos de información inventada con fuente
        citada                                clase 250

correcciones
  1  conjunto de evaluación propio: 240 casos reales, con
     respuesta esperada                       clase 250
  2  las fechas de retirada, en el calendario del equipo
     con recordatorio a 3 meses                  ley 25
  3  instrucciones versionadas como código, probadas contra
     el conjunto
  4  y una comparación automática: al aparecer una versión
     nueva, se ejecuta el conjunto contra las dos y se
     publica la diferencia

y la migración siguiente
  duración                                     1 día
  diferencias detectadas antes de migrar           7
    → 5 de instrucción, corregidas
    → 2 de comportamiento, aceptadas y documentadas
```

La capa fina, y lo que hace:

```
214 líneas, en una biblioteca interna

  llamada al modelo con reintentos, retroceso y plazo
  MEDICIÓN por caso de uso
    testigos de entrada y salida
    coste
    latencia y tiempo hasta el primer testigo
  ENRUTADO
    respuestas preparadas → sin modelo
    consultas → base de datos
    modelo pequeño → casos simples
    modelo grande → el resto
  registro con muestreo del 5 %, sin datos personales
  filtros de entrada y de salida            clase 249
  y la versión de la instrucción, en cada llamada

lo que NO hace
  no abstrae la llamada a herramientas: se usa la del
  proveedor y se registró la decisión        clase 158
  no intenta unificar las salidas estructuradas

y el efecto
  el coste por caso de uso, visible desde el primer día
  y al comparar proveedores, el conjunto de evaluación
  ejecutable contra los dos en una tarde
```

El control de gasto, al año:

```
coste mensual por caso de uso
  asistente de atención                        610 €
  asistente interno de documentación           240 €
  clasificación de comentarios                  90 €
  asistente legal (alojado)                  1.900 €
  evaluaciones y pruebas                       180 €
  ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
  total                                      3.020 €

cuotas y presupuestos
  presupuesto por proyecto, con alerta          sí
  peticiones limitadas por cuota          41/mes
    → todas, en pruebas
  incidentes de coste                            0

capacidad reservada
  para el asistente de atención, en horario laboral
  → latencia p95 estable en 2,1 s
  → sin reserva, subía a 6 s en horas de demanda global
```

El resultado:

```
incidentes de coste                             1           0
coste del incidente                      18.400 €           —
migraciones de versión sin evaluar              1           0
duración de una migración              3 semanas       1 día
  con quejas                                   sí          no
conjunto de evaluación propio                  no   240 casos
cuotas y límites                               no          sí
coste mensual                           sin medir    3.020 €
coste por operación de negocio           sin medir   0,004 €
```

La lección que esta clase deja: la comparación entre proveedores no la decidió el catálogo de modelos, que era equivalente, sino que la nube A ya tenía resueltos la identidad, la red y el coste; y el único caso que se salió fue por una cláusula contractual de tres socios. Y la migración de versión, que debería haber sido un día, costó tres semanas y tres respuestas inventadas porque no había conjunto de evaluación propio: es lo que más reduce el coste de cambiar, y lo que menos se construye.

Laboratorio guiado

Ejecuta desde la raíz:

```
python classes/part-20-cloud-data-ai-platforms/248-bedrock-azure-ai-foundry-y-vertex-ai/lab.py
```

El laboratorio selecciona el motor de práctica decision y produce labresult.json. El escenario, sus comprobaciones y el artefacto esperado corresponden a esta clase; no requiere credenciales y deja explícito qué debe revalidarse en un sandbox real.

1. Lee exercise.steps y formula una predicción antes de ejecutar.
2. Ejecuta la práctica y verifica todos los elementos de checks.
3. Provoca el caso de negativetest y explica la señal observada.
4. Materializa genai-provider-adr en evidence/ usando la plantilla indicada.
5. Para proveedor real, sigue sandbox.requires, registra costo y ejecuta destroy.

Evidencia esperada

El artefacto principal es una matriz de decisión ponderada y un ADR. Además, la entrega debe incluir el comando ejecutado, la salida estructurada y una conclusión que no exceda lo observado.

Reto verificable

Construye genai-provider-adr para el caso CloudShop. Incluye una alternativa descartada, un supuesto que pueda falsarse, una prueba de fallo y una decisión de rollback.

■ Criterio de aceptación

- [ ] lab.py termina con código 0 y genera JSON válido.
- [ ] La entrega conecta al menos tres requisitos con mecanismos verificables.
- [ ] Existe una prueba positiva y una prueba negativa con evidencia.
- [ ] Seguridad, costo y operación aparecen como decisiones, no como anexos.
- [ ] Se declara una limitación y una condición que obligaría a revisar el diseño.
- [ ] Otra persona puede repetir el recorrido sin conocimiento tácito.

■■ Errores frecuentes

Síntoma	Causa probable	Corrección
Una prueba consume el presupuesto del trimestre	No hay cuotas, presupuestos ni límites, y un reintento sin tope	Pon cuotas por proyecto e identidad, presupuestos con alerta y acción, límites de testigos y peticiones, y reintentos con tope.
El comportamiento del asistente cambia sin que nadie toque nada	Se apunta al alias de última versión y el proveedor actualizó	Fija la versión y trátala como una dependencia: se actualiza evaluando antes.
La migración a una versión nueva se hace a ciegas y sale mal	No hay conjunto de evaluación propio con casos reales	Construye y mantén un conjunto de evaluación; es lo que convierte una migración en un día de trabajo.
Un requisito contractual impide usar el servicio gestionado	No se comprobó qué hace el proveedor con los datos ni dónde se procesan	Verifica retención, uso para entrenar y región en el contrato, y reserva el alojamiento propio para lo que no pueda salir.
En horas de demanda la latencia se dispara y aparecen errores	El pago por uso comparte cuota y hay limitación	Reserva capacidad para el camino crítico y maneja la limitación con retroceso y respaldo.
Cambiar de proveedor resulta mucho más caro de lo previsto	Lo que ata son los embebidos, las instrucciones y las capacidades específicas, no el código	Conserva los textos originales, versiona las instrucciones, mantén el conjunto de evaluación y construye una capa fina que mida y enrute, no una que lo abstraiga todo.

■ Seguridad, ética y costo

Trabaja con cuentas propias o sandboxes autorizados. No publiques secretos, identificadores reales ni datos personales. Antes de crear recursos pagos define presupuesto, etiquetas y comando de destrucción; después verifica que no queden recursos huérfanos. Los ejemplos locales enseñan contratos, pero no certifican cumplimiento ni disponibilidad de producción.

■ Preguntas de comprobación

1. ¿Cuáles son los cinco criterios que diferencian a los proveedores de verdad?
2. ¿Qué hay que comprobar sobre los datos y dónde se comprueba?
3. ¿Qué pasa si se apunta al alias de última versión?
4. ¿Qué ata más al cambiar de proveedor y qué es lo más barato de cambiar?
5. ¿Qué debe y qué no debe hacer una capa propia sobre el proveedor?

Referencias

- AWS (2025). Amazon Bedrock: data protection and model versions. <https://docs.aws.amazon.com/bedrock/latest/userguide/data-protection.html>
- Microsoft (2025). Azure AI Foundry: data privacy and model lifecycle. <https://learn.microsoft.com/en-us/azure/ai-foundry/>

- Google Cloud (2025). Vertex AI: generative AI data governance and model versions. <https://cloud.google.com/vertex-ai/generative-ai/docs/data-governance>
- Anthropic (2025). Claude API: models, versions and deprecations. <https://docs.anthropic.com/en/docs/about-claude/models/overview>
- NIST (2024). AI Risk Management Framework: Generative AI profile. <https://www.nist.gov/itl/ai-risk-management-framework>
- Documentación oficial vigente del servicio implementado; registra URL y fecha de consulta.

Descargar: Parte 20 en PDF · Manual integral

Anterior	Índice	Siguiente
← 247 · Modelos fundacionales, tokens, embeddings y RAG	Parte 20 · Programa	249 · Agentes, tools, memoria, permisos y guardrails →

Evaluación — 248 Bedrock, Azure AI Foundry y Vertex AI

Preguntas

1. Explica el problema que resuelve esta capacidad sin mencionar un proveedor.
2. Identifica una frontera de responsabilidad y su propietario.
3. Compara dos alternativas mediante confiabilidad, seguridad, costo y operación.
4. ¿Qué demuestra el laboratorio y qué no puede demostrar?
5. Propón un caso límite o una prueba de fallo.

Reto verificable

Entrega genai-provider-adr con diagrama o configuración, evidencia de ejecución, alternativa descartada, riesgo residual y criterio de rollback.

Criterio de aceptación

- [ ] Resultado reproducible por una segunda persona.
- [ ] Evidencia vinculada a cada afirmación técnica.
- [ ] Prueba negativa o de fallo incluida.
- [ ] Costos y permisos expresados con unidades y alcance.
- [ ] Limitaciones declaradas sin presentar la simulación como producción.

Escala

Nivel	Evidencia
A — competente	Cumple todo y defiende trade-offs con datos.
B — en desarrollo	Cumple lo esencial; falta profundidad en una dimensión.
C — inicial	Artefacto parcial o conclusión sin evidencia suficiente.
0	Sin entrega reproducible.

Parte 22 — Especializaciones, certificaciones y práctica profesional

273 — Mapeo AWS, Azure, Google Cloud, Kubernetes y FinOps

Parte: 22 — Especializaciones, certificaciones y práctica profesional Nivel: intermedio-avanzado · Horas estimadas: 4 Laboratorio: assessment · Estado: EXECUTABLECORE

Propósito

Traducir entre proveedores sin engañarse: dar el mapa de equivalencias entre las tres nubes, los contenedores y la disciplina de coste, y —más importante— dónde la equivalencia se rompe. La clase enseña el método de traducción por restricción en vez de por nombre, y las diferencias reales que este programa ha medido y que ninguna tabla de equivalencias recoge.

Resultados de aprendizaje

Al finalizar podrás:

1. Traducir entre proveedores por restricción, no por nombre de servicio.
2. Reconocer dónde la equivalencia aparente esconde una diferencia real.
3. Usar el mapa de las cinco capas para orientarse en una nube nueva.
4. Evaluar una traducción con las preguntas que la validan.
5. Aplicar el mismo método a los contenedores y a la disciplina de coste.

Conceptos centrales

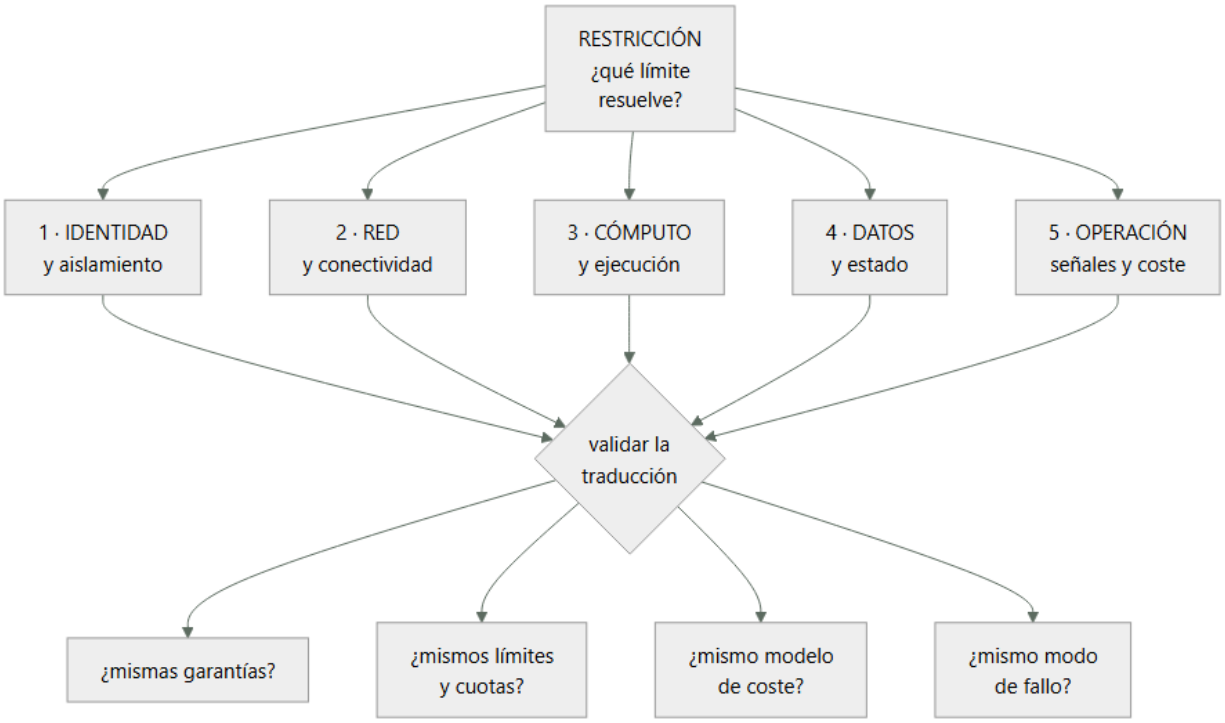
Concepto	Comprensión verificable
traducción por restricción	Buscar cómo la otra nube resuelve el mismo límite físico, en vez de buscar el servicio con nombre parecido.
equivalencia falsa	Dos servicios con el mismo propósito y garantías, límites o modelo de coste distintos.
unidad de aislamiento	El contenedor administrativo donde vive la separación: cuenta, suscripción o proyecto.
plano de control	La interfaz que crea y modifica recursos. Sus cuotas y su latencia difieren mucho entre nubes.
portabilidad aparente	Creer que usar la misma tecnología en tres nubes elimina las diferencias. No lo hace.
coste de traducción	Lo que cuesta operar en dos nubes: no es el doble de aprender, es el doble de operar.

Modelo mental

Una especialización combina fundamentos, evidencia de proyectos y juicio bajo restricciones; una insignia sin práctica no sustituye esa combinación.

Aplicado a esta clase, separa siempre cuatro planos: intención (qué necesita el usuario), configuración (qué declaramos), estado observado (qué existe de verdad) y evidencia (cómo sabemos que cumple). Confundirlos produce diseños que se ven correctos en un diagrama pero fallan al operar.

■ Flujo de razonamiento



Desarrollo

1. Traducir por restricción, no por nombre

Las tablas de equivalencias por nombre producen errores caros. El método que funciona empieza por la restricción.

```
EL MÉTODO
  1  ¿qué RESTRICCIÓN resuelve esto?
     → «necesito que un servicio pruebe su identidad ante
       otro sin credencial permanente»
  2  ¿cómo la resuelve la nube destino?
  3  ¿con qué garantías, límites y coste?
  4  ¿y con qué modo de fallo?

→ y solo entonces se escribe el nombre del servicio
→ empezar por el nombre es lo que produce las sorpresas
```

Y las cinco capas en las que conviene organizarse:

```
1  IDENTIDAD Y AISLAMIENTO
   quién es quién, qué puede, y dónde vive la separación
   → cuenta · suscripción · proyecto
2  RED Y CONECTIVIDAD
   direccionamiento, rutas, resolución de nombres, salida
   a internet, interconexión
3  CÓMPUTO Y EJECUCIÓN
   máquinas, contenedores, funciones, escalado
4  DATOS Y ESTADO
   objetos, bloques, relacional, no relacional, analítico,
   mensajería
5  OPERACIÓN
   señales, registros, políticas, coste, cuotas

→ en toda nube existen las cinco
→ y aprender una nube nueva es rellenar esta rejilla, no
  leer un catálogo
```

Y el mapa por restricción, que es lo transferible:

```
RESTRICCIÓN                        DÓNDE MIRAR EN CUALQUIER NUBE
«que un servicio pruebe quién es   identidad de carga de
sin credencial fija»               trabajo, federada
«separar entornos para que un       unidad de aislamiento
fallo no cruce»                     administrativa
```

```
«que el tráfico solo llegue a quien debe»          segmentación de red y reglas por identidad
«que un cambio no afecte a todos a la vez»         despliegue progresivo del proveedor o propio
«que el dato sobreviva a un administrador comprometido»   copia en otra unidad de aislamiento, inmutable
«saber quién gasta qué»                            etiquetas y jerarquía administrativa
«no pasar de N peticiones por segundo al plano de control»   cuotas del plano de control

→ y este mapa no cambia cuando cambian los productos
```

2. Dónde la equivalencia se rompe

Lo que ninguna tabla de equivalencias dice, y este programa ha medido.

```
1  LA UNIDAD DE AISLAMIENTO NO ES EQUIVALENTE
   cuenta, suscripción y proyecto se parecen y difieren en
     cuánto cuesta crear una nueva
     qué se hereda desde arriba
     y qué queda compartido aunque parezca separado
   → y de aquí salen los errores de aislamiento más caros
                                          clases 219, 231

2  LOS LÍMITES POR DEFECTO SON MUY DISTINTOS
   dos nubes con el «mismo» servicio de funciones tienen
   concurrencias por defecto de órdenes distintos
   → y en la región secundaria, los valores por defecto
     mandan                                clase 262, ley 26

3  EL MODELO DE COSTE CAMBIA LA ARQUITECTURA
   donde el tráfico entre zonas se cobra, la arquitectura
   correcta es otra
   donde el análisis se cobra por dato leído, el formato y
   la partición valen más que el motor    clase 243, ley 28
   → y una traducción literal puede multiplicar la factura

4  LA COHERENCIA DE LOS ALMACENES DE OBJETOS
   parecen iguales y sus garantías de lectura tras
   escritura, de listado y de sobrescritura no lo son
   → y esto rompe canales de datos silenciosamente
                                                ley 29

5  LA RESOLUCIÓN DE NOMBRES Y LA SALIDA A INTERNET
   quién resuelve, desde dónde, y qué pasa con las
   respuestas privadas difiere mucho          clase 195
   → es la fuente número uno de «funciona en una nube y no
     en la otra»

6  EL PLANO DE CONTROL
   su latencia y sus cuotas de peticiones difieren
   → y una herramienta de infraestructura como código que
     funciona en una nube se estrangula en otra
                                          clases 128, 262

7  Y LA SEMÁNTICA DE LOS PERMISOS
   denegar explícito, herencia, ámbito y evaluación no
   funcionan igual
   → una política traducida literalmente puede conceder
     más de lo que concedía                  clase 231
```

Y la regla que resume todo lo anterior:

```
LO QUE SE TRADUCE BIEN
  el propósito y la forma de la arquitectura

LO QUE NO SE TRADUCE
  garantías, límites, modelo de coste y modos de fallo

→ y esos cuatro son los que deciden si el sistema
  funciona
→ por eso la traducción se VALIDA midiendo, no leyendo
```

3. Contenedores y la portabilidad aparente

El caso especial que más expectativas defrauda.

```
LO QUE SÍ APORTA UNA CAPA COMÚN DE CONTENEDORES
  el empaquetado y el modelo de despliegue se parecen
  las descripciones de carga de trabajo son casi iguales
  y el conocimiento del equipo se traslada

LO QUE NO ELIMINA
  la identidad y su federación con la nube
  la red: direccionamiento, salida, balanceo, resolución
  el almacenamiento persistente y sus garantías
  el registro de imágenes y su disponibilidad
  las cuotas del plano de control
  el coste, muy distinto por nube
  y la operación: actualizaciones, versiones y su ritmo

→ la portabilidad es del ARTEFACTO, no del SISTEMA
→ y la parte que no se traslada es la que da los
  incidentes                                clase 185
```

Y el coste real de operar en dos nubes:

```
NO ES EL DOBLE DE APRENDER: ES EL DOBLE DE OPERAR
  dos inventarios                            clase 253
  dos modelos de permisos                    clase 231
  dos formas de red                          clase 194
  dos conjuntos de cuotas                    clase 262
  dos catálogos de alertas y dos guardias    clase 257
  dos formas de facturar                     clase 270
  y ensayos en las dos                       clase 261

→ y por eso multinube por defecto es caro
→ y multinube por una RAZÓN concreta —normativa,
  adquisición, un servicio que solo existe en una— se
  sostiene

y la pregunta que ordena la decisión
  «¿qué riesgo concreto estamos comprando con este coste?»
  → si la respuesta es «no depender de un proveedor», hay
    que cuantificar cuánto vale eso y compararlo
```

Y el mapa de la disciplina de coste, que también se traduce:

```
lo que existe en las tres, con otros nombres
  jerarquía administrativa para atribuir
  etiquetas y su obligatoriedad
  exportación detallada de facturación
  compromisos con descuento
  presupuestos y alertas
  y capacidad interrumpible

lo que cambia de verdad
  qué se cobra por transferencia entre zonas y regiones
  cómo se factura el almacenamiento frío y su
    recuperación
  cómo se cobra la analítica: por dato leído o por
    capacidad reservada
  y el retraso con que llega el detalle de coste

→ y esos cuatro cambian decisiones de diseño
                                            clase 270
```

4. Cómo se valida una traducción

Una traducción sobre el papel no vale nada. Estas son las comprobaciones que la convierten en fiable.

```
LAS CUATRO PREGUNTAS DE VALIDACIÓN
  1  ¿MISMAS GARANTÍAS?
     coherencia, durabilidad, orden, exactamente una vez
     → y aquí casi siempre hay una diferencia
  2  ¿MISMOS LÍMITES Y CUOTAS?
     por defecto y ampliables, por región
  3  ¿MISMO MODELO DE COSTE?
```

```
     qué se cobra por operación, por dato, por hora
  4  ¿MISMO MODO DE FALLO?
     ¿falla rápido o se degrada? ¿qué se ve cuando falla?

→ y las cuatro se responden con documentación y con una
  prueba
```

Y la prueba mínima antes de comprometerse:

```
PROTOTIPO DE UNA SEMANA
  el camino crítico de extremo a extremo
  con identidad real, red real y datos de tamaño
    realista
  midiendo latencia, coste por operación y comportamiento
    al fallar
  y provocando un fallo a propósito         clase 261

→ una semana de esto ahorra meses
→ y descubre las diferencias que ninguna tabla recoge
```

Y cómo se demuestra esta competencia:

```
LO QUE NO VALE
  «conozco las tres nubes»
  → y la parte 22 predijo que esto rendiría peor que
    conocer una a fondo                     clase 264

LO QUE VALE
  «migramos el canal de datos y descubrimos que las
   garantías de listado del almacén de objetos eran
   distintas; lo detectamos con una prueba de una semana
   antes de comprometernos»
  «traducimos la política y concedía más de lo que
   concedía la original; lo cogió la comprobación
   automática»
  «el mismo diseño costaba 3,2 veces más por el cobro de
   tráfico entre zonas, y rediseñamos la colocación»

→ efecto, mecanismo y cifra               clase 275
```

Y la lista de comprobación de la clase:

- ■ traduzco por restricción, no por nombre de servicio
- ■ uso la rejilla de cinco capas para orientarme
- ■ compruebo garantías, límites, coste y modo de fallo
- ■ sé que la unidad de aislamiento no es equivalente
- ■ reviso los valores por defecto, sobre todo en la región secundaria
- ■ compruebo la semántica de permisos, no solo su forma
- ■ verifico resolución de nombres y salida a internet
- ■ vigilo las cuotas del plano de control
- ■ no confundo portabilidad del artefacto con la del sistema
- ■ cuantifico el coste de operar en dos nubes
- ■ hago un prototipo de una semana antes de comprometerme
- ■ y provoco un fallo en ese prototipo

Y el cierre que enlaza con la clase siguiente: con el mapa y el método de traducción, queda la parte que examina esto por escrito y con tiempo limitado. Las preguntas de escenario y la estrategia de examen son la materia de la clase 274.

Ejemplo trabajado

CloudShop traduce tres piezas entre nubes. Lo que sigue son las cuatro diferencias que el prototipo de una semana encontró y ninguna tabla recogía, la política traducida que concedía de más, y la cuenta del coste real de operar en dos nubes.

Traducción 1 · El canal de datos.

```
qué se traducía
  ingesta a almacén de objetos, catálogo, y consultas
  analíticas                                clase 242

la tabla de equivalencias decía
  almacén de objetos      →  almacén de objetos
```

```
  catálogo               →  catálogo
  motor de consulta      →  motor de consulta

→ traducción aparentemente trivial
```

Y lo que el prototipo de una semana encontró:

```
DIFERENCIA 1 · garantías de listado
  el proceso dependía de listar un prefijo justo después
  de escribir
  → en el origen, el listado era inmediato
  → en el destino, podía tardar
  → y el trabajo posterior procesaba ficheros de menos
    SIN ERROR                                  ley 29

  cómo se detectó
    prueba con 40.000 ficheros escritos y listados de
    inmediato
    → 61 casos de listado incompleto
  cómo se arregló
    manifiesto explícito de ficheros, en vez de listar

DIFERENCIA 2 · modelo de coste de la consulta
  origen   por dato leído
  destino  por capacidad reservada
  → el mismo conjunto de 41 informes
    origen, tras optimizar               12 USD/informe
    destino con capacidad mínima         por debajo del
                                         umbral, pero con
                                         suelo mensual
  → por debajo de cierto volumen el destino era más caro
  → y por encima, mucho más barato
  → la decisión dependía del volumen, no del motor

DIFERENCIA 3 · cuota del plano de control
  la herramienta de infraestructura como código aplicaba
  341 recursos
  → en el destino se estrangulaba a mitad
  → aplicación fallida de forma intermitente  clase 128
  → se resolvió partiendo el estado y limitando el
    paralelismo

DIFERENCIA 4 · zonas horarias en las particiones
  la convención por defecto del catálogo difería
  → y las particiones diarias se desplazaban unas horas
  → los informes diarios cuadraban salvo en el límite del
    día                                    clase 243
```

Y la valoración:

```
coste del prototipo                    1 semana, 1 persona
diferencias encontradas                                  4
  que ninguna tabla de equivalencias recogía             4
  que habrían llegado a producción sin error visible     2

→ las dos silenciosas eran las peligrosas
```

Traducción 2 · La política que concedía de más.

```
política original
  permitir lectura de un prefijo concreto del almacén, a
  un rol de servicio, con denegación explícita para todo
  lo demás                                   clase 231

la traducción literal
  se mantuvo la concesión y se omitió la denegación
  explícita
  → porque en la nube destino la herencia y la evaluación
    funcionaban distinto y «parecía innecesaria»

lo que produjo
  el rol heredaba desde el nivel superior un permiso de
  lectura sobre el contenedor entero administrativo
  → y sin la denegación explícita, la herencia ganaba
  → el rol podía leer 214 conjuntos de datos en vez de 1
```

Y cómo se detectó:

```
no en la revisión humana: la revisión lo aprobó

lo cogió la comprobación automática de permisos efectivos
  → que compara «qué puede hacer realmente esta identidad»
    antes y después                          clase 217
  → 214 recursos accesibles frente a 1 esperado

→ la lección: se compara el EFECTO, no el texto de la
  política
→ y esa comprobación se añadió a la cadena para toda
  traducción
```

La cuenta del coste real de operar en dos nubes.

```
CloudShop tenía una razón concreta para la segunda nube
  un cliente del sector público exigía residencia en una
  región que el proveedor principal no ofrecía
                                           clase 280

y se midió lo que costaba mantenerla

  inventario y etiquetado, segunda nube      0,4 personas
  modelo de permisos y su auditoría          0,3
  red y conectividad                         0,3
  cuotas y capacidad                         0,2
  alertas, guardia y procedimientos          0,6
  facturación y atribución                   0,2
  ensayos                                    0,2
  actualizaciones y versiones                0,3
                                             ■■■■■■
                                             2,5 personas

  más el coste de infraestructura            41.000 USD/mes
  y el sobrecoste por menor volumen
    (compromisos peores)                     +9 %
```

Y la decisión que se tomó con esa cifra:

```
ingresos del cliente que exigía la segunda nube
                                        1,9 M USD/año
coste de mantenerla              ~2,5 personas + 492.000
                                        USD/año

→ se mantuvo, y con la cifra escrita en el registro de
  decisión                                clase 272
→ con condición de revisión: «si estos ingresos bajan de
  900.000, se revisa»

y lo que NO se hizo
  no se replicó el resto de la plataforma en la segunda
  nube «por si acaso»
  → solo lo que ese cliente necesitaba
  → 6 servicios de 41
```

Y la comparación con lo que se había propuesto al principio:

```
propuesta inicial   «seamos multinube en todo, para no
                    depender de un proveedor»
  coste estimado    ~7 personas + 1,4 M USD/año
  riesgo evitado    no cuantificado

y la pregunta que la desmontó
  «¿cuál es el escenario concreto del que nos protege, y
  cuánto nos costaría si ocurriera?»
  → el escenario realista era una subida de precios o una
    pérdida de servicio prolongada
  → y la protección real requería mantener el sistema
    EJECUTÁNDOSE en las dos, no solo poder migrar
  → lo que multiplicaba la cifra

→ se decidió: multinube donde hay una razón concreta;
  portabilidad razonable en lo demás
→ y «portabilidad razonable» se definió: contratos, datos
```

```
  exportables y decisiones registradas, no abstracciones
  propias                                   clase 267
```

Y la rejilla de cinco capas, rellenada para orientarse.

```
el equipo mantuvo una tabla de una página por nube

  capa            qué mirar             dónde difiere más
  identidad       unidad de aislamiento herencia y
                  y federación          denegación
  red             direccionamiento,     resolución de
                  salida, balanceo      nombres privados
  cómputo         escalado, arranque,   límites por
                  interrumpible         defecto
  datos           garantías del almacén listado y coste
                  y del relacional      por operación
  operación       señales, políticas,   retraso del
                  cuotas, coste         detalle de coste

→ una página por nube, actualizada cuando algo sorprende
→ y usada para entrar en una nube nueva en días
```

La lección que esta clase deja: la traducción del canal de datos parecía trivial y el prototipo de una semana encontró cuatro diferencias, dos de ellas silenciosas —un listado incompleto y un desplazamiento de particiones— que habrían llegado a producción sin producir ningún error. Y la política traducida pasó la revisión humana mientras concedía acceso a 214 conjuntos de datos en vez de uno: lo cogió comparar permisos efectivos, no leer el texto.

Laboratorio guiado

Ejecuta desde la raíz:

```
python classes/part-22-specializations-certifications-career/273-mapeo-aws-azure-google-cloud-kubernetes-y-finops/lab.py
```

El laboratorio selecciona el motor de práctica assessment y produce labresult.json. El escenario, sus comprobaciones y el artefacto esperado corresponden a esta clase; no requiere credenciales y deja explícito qué debe revalidarse en un sandbox real.

1. Lee exercise.steps y formula una predicción antes de ejecutar.
2. Ejecuta la práctica y verifica todos los elementos de checks.
3. Provoca el caso de negativetest y explica la señal observada.
4. Materializa certification-map en evidence/ usando la plantilla indicada.
5. Para proveedor real, sigue sandbox.requires, registra costo y ejecuta destroy.

Evidencia esperada

El artefacto principal es una evaluación por escenarios con rúbrica y evidencia trazable. Además, la entrega debe incluir el comando ejecutado, la salida estructurada y una conclusión que no exceda lo observado.

Reto verificable

Construye certification-map para el caso CloudShop. Incluye una alternativa descartada, un supuesto que pueda falsarse, una prueba de fallo y una decisión de rollback.

■ Criterio de aceptación

- [ ] lab.py termina con código 0 y genera JSON válido.
- [ ] La entrega conecta al menos tres requisitos con mecanismos verificables.
- [ ] Existe una prueba positiva y una prueba negativa con evidencia.
- [ ] Seguridad, costo y operación aparecen como decisiones, no como anexos.
- [ ] Se declara una limitación y una condición que obligaría a revisar el diseño.
- [ ] Otra persona puede repetir el recorrido sin conocimiento tácito.

Errores frecuentes

Síntoma	Causa probable	Corrección
Un servicio equivalente se comporta distinto en producción	Se tradujo por nombre y no se compararon garantías, límites, coste y modo de fallo	Traduce por restricción y valida las cuatro preguntas con documentación y con un prototipo del camino crítico.
Una política traducida concede más de lo que concedía	La herencia y la evaluación de permisos difieren y la denegación explícita se omitió	Compara permisos efectivos antes y después, no el texto de la política; automatiza esa comprobación en la cadena.
El canal de datos procesa registros de menos sin dar error	Las garantías de listado y de lectura tras escritura del almacén no son iguales	Usa manifiestos explícitos en vez de listar, y prueba con volumen realista escribiendo y listando de inmediato.
La infraestructura como código se aplica a medias e intermitentemente	Se alcanzan las cuotas de peticiones del plano de control, que difieren por nube	Parte el estado, limita el paralelismo y vigila la cuota del plano de control como una más del inventario.
El mismo diseño cuesta varias veces más en la otra nube	El modelo de coste cambia lo que es una buena arquitectura	Revisa qué se cobra por transferencia, por operación y por dato leído; rediseña colocación y formato antes de migrar.
Se adopta multinube por no depender de un proveedor y el coste se dispara	No se cuantificó el riesgo evitado ni el coste de operar dos nubes	Exige un escenario concreto y su coste; multinube donde hay una razón, portabilidad razonable en lo demás.

Seguridad, ética y costo

Trabaja con cuentas propias o sandboxes autorizados. No publiques secretos, identificadores reales ni datos personales. Antes de crear recursos pagos define presupuesto, etiquetas y comando de destrucción; después verifica que no queden recursos huérfanos. Los ejemplos locales enseñan contratos, pero no certifican cumplimiento ni disponibilidad de producción.

Preguntas de comprobación

1. ¿Por qué se traduce por restricción y no por nombre de servicio?
2. ¿Cuáles son las cinco capas de la rejilla y para qué sirven?
3. ¿Qué cuatro cosas no se traducen nunca y por qué deciden el resultado?
4. ¿Qué aporta y qué no aporta una capa común de contenedores?
5. ¿Qué comprueba un prototipo de una semana que ninguna tabla recoge?

Referencias

- AWS (2024). Compare AWS and Azure services. <https://learn.microsoft.com/azure/architecture/aws-professional/services>
- Google Cloud (2024). Google Cloud for AWS professionals. <https://cloud.google.com/docs/get-started/aws-comparison>
- CNCF (2024). Kubernetes conformance and portability. <https://www.cncf.io/certification/software-conformance/>
- FinOps Foundation (2024). FOCUS: FinOps cost and usage specification — coste comparable entre nubes. <https://focus.finops.org/>
- Brewer, E. (2012). CAP twelve years later: how the rules have changed — por qué las garantías no se traducen. <https://www.infoq.com/articles/cap-twelve-years-later-how-the-rules-have-changed/>
- Documentación oficial vigente del servicio implementado; registra URL y fecha de consulta.

Descargar: Parte 22 en PDF · Manual integral

Evaluación — 273 Mapeo AWS, Azure, Google Cloud, Kubernetes y FinOps

Preguntas

1. Explica el problema que resuelve esta capacidad sin mencionar un proveedor.
2. Identifica una frontera de responsabilidad y su propietario.
3. Compara dos alternativas mediante confiabilidad, seguridad, costo y operación.
4. ¿Qué demuestra el laboratorio y qué no puede demostrar?
5. Propón un caso límite o una prueba de fallo.

Reto verificable

Entrega certification-map con diagrama o configuración, evidencia de ejecución, alternativa descartada, riesgo residual y criterio de rollback.

Criterio de aceptación

- [ ] Resultado reproducible por una segunda persona.
- [ ] Evidencia vinculada a cada afirmación técnica.
- [ ] Prueba negativa o de fallo incluida.
- [ ] Costos y permisos expresados con unidades y alcance.
- [ ] Limitaciones declaradas sin presentar la simulación como producción.

Escala

Nivel	Evidencia
A — competente	Cumple todo y defiende trade-offs con datos.
B — en desarrollo	Cumple lo esencial; falta profundidad en una dimensión.
C — inicial	Artefacto parcial o conclusión sin evidencia suficiente.
0	Sin entrega reproducible.