LABORATORIOS DE ARQUITECTURA EN ORACLE CLOUD INFRASTRUCTURE

Manual de consola — Observabilidad y confiabilidad del servicio

Alarmas con umbral, destinatario y acción escrita sobre un servicio que ya está corriendo

Módulo 07

Primera pasada 60–75 minutos la primera pasada, más el ciclo de prueba de la alarma

Costo aproximado prácticamente 0 USD en observabilidad; lo único que cobra por volumen son los registros. El laboratorio observado midió 0,13 USD de cómputo en el ensayo completo

Vía consola de Oracle Cloud, pantalla por pantalla

Este documento reproduce desde la consola lo que el laboratorio automatiza con Terraform. Los dos caminos llegan al mismo sitio: el código sirve para repetirlo, la consola para entenderlo.

El laboratorio corre sobre un tenancy de prueba desechable. Las decisiones de acceso que aquí se toman no son una referencia de configuración segura para un ambiente con datos; cuando alguna se aparta de la buena práctica, el manual lo dice en el sitio donde se aparta.

Este manual se hace **desde la consola de OCI**, haciendo clic. No usa Terraform y usa la CLI solo para verificar lo que la consola ya mostró. Está escrito para alguien que administra infraestructura, conoce AWS o Azure y nunca ha entrado a OCI.

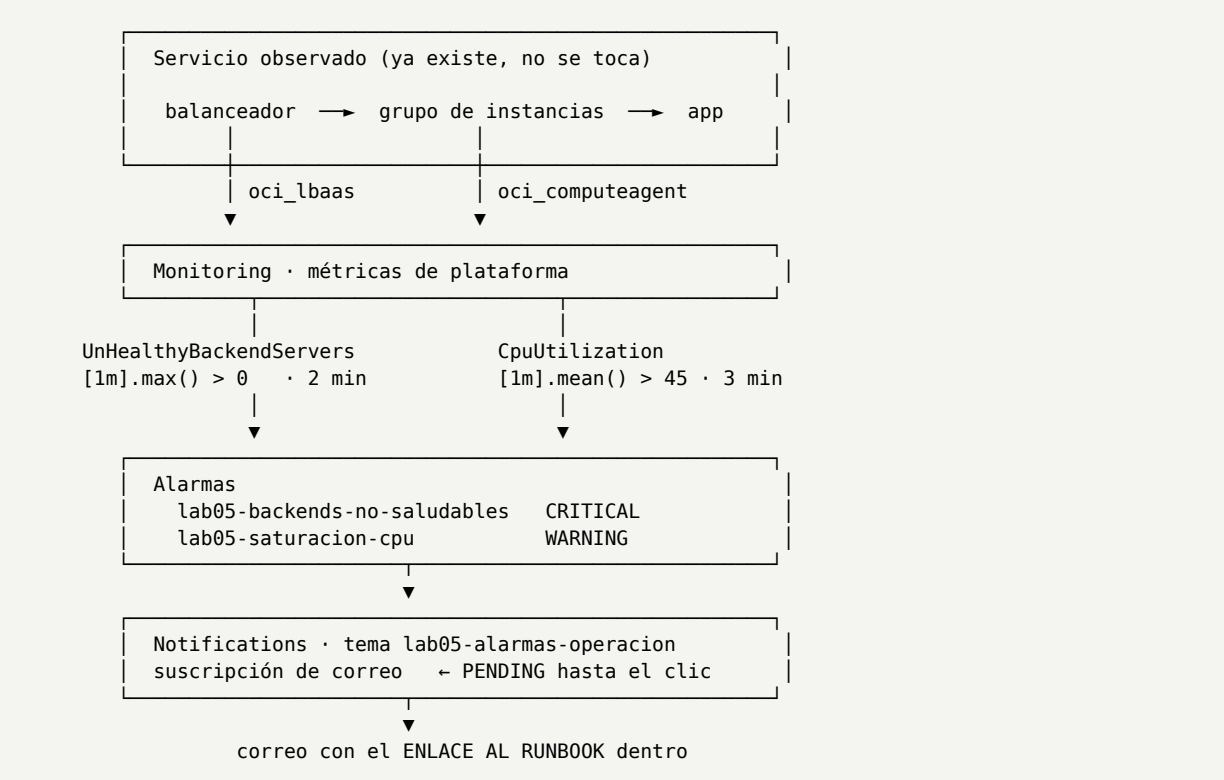
1. Qué se construye y para qué

Este laboratorio **no crea infraestructura**. Se monta encima de un servicio que ya está corriendo: un balanceador con un grupo de instancias detrás, con autoescalamiento configurado. Lo que se construye es la capa que convierte una métrica en una alarma con dueño.

La diferencia importa. Una métrica es un número. Una alarma es un número, un umbral, un destinatario y una acción escrita. Sin esas cuatro cosas hay una gráfica bonita y un teléfono que suena de madrugada sin que nadie sepa qué hacer.

Al terminar, tres piezas quedan funcionando:

1. Un **tema de notificaciones** con una suscripción de correo confirmada.
2. Dos **alarmas de métrica** con umbral, ventana y severidad distintas, que escriben en ese tema.
3. Los **registros de servicio** (logs) habilitados y consultables, para bajar del síntoma al detalle cuando la alarma no se explica sola.



La decisión que este laboratorio ayuda a tomar

No es «qué herramienta de monitoreo compramos». Es esta:

> *¿Qué merece despertar a alguien de madrugada y qué puede esperar al día siguiente?*

Todo el laboratorio está construido para hacer visible esa pregunta con un caso concreto. La demostración central es un ciclo completo que se midió de punta a punta:

Momento	Estado de la alarma
En reposo	`OK`
Con carga: CPU en 62 % contra un umbral de 45 %	**`FIRING`**
Después, sin que nadie intervenga	**`OK` — se cerró sola**

La alarma sonó, el autoescalamiento agregó capacidad, la CPU por instancia bajó y la alarma se cerró **sin que nadie hiciera nada**. Esa secuencia es el argumento del laboratorio: una alarma que se resuelve sola no debía despertar a nadie, pero tampoco debía desaparecer sin dejar rastro. Distinguir lo que se atiende *ahora* de lo que hay que *saber* es lo que separa a un equipo que apaga incendios de uno que opera.

El entregable no es la alarma: es la **matriz de señales** del capítulo 13.

2. Equivalencias con otras nubes

Solo los servicios que aparecen en este manual.

Para qué	AWS	Azure	OCI
Métricas de plataforma	CloudWatch Metrics	Azure Monitor Metrics	Monitoring
Explorar una métrica	CloudWatch Metrics console	Metrics Explorer	Metrics Explorer / Service Metrics
Lenguaje de consulta de métricas	Metric Math	Consultas de Azure Monitor	MQL (Monitoring Query Language)
Alarma sobre una métrica	CloudWatch Alarm	Alert rule (metric)	Alarm definition
Ventana antes de disparar	Evaluation periods × period	Aggregation granularity + Frequency	Interval + Trigger delay minutes
A quién le llega	SNS topic + subscription	Action group	Notifications: topic + subscription
Confirmación del correo	SNS pide confirmar por correo	El grupo de acción no siempre la pide	**Siempre** la pide: PENDING → ACTIVE
Tableros	CloudWatch Dashboards	Azure Dashboards / Workbooks	Dashboards

Registros de servicio	CloudWatch Logs	Diagnostic settings → Log Analytics	Logging (service logs)
Registros de red	VPC Flow Logs	NSG flow logs	Flow Logs (a través de Logging)
Consultar registros	CloudWatch Logs Insights	Log Analytics (KQL)	Logging Search
Eventos de la nube	EventBridge	Event Grid	Events Service
Registro de auditoría	CloudTrail	Activity Log	Audit (automático, sin habilitar nada)

Tres diferencias que cuestan tiempo si se llega con el modelo mental de otra nube:

- **La alarma y el recurso observado pueden vivir en compartimentos distintos**, y en la práctica conviene que así sea. En la pantalla de creación de la alarma hay **dos** campos de compartimento y no significan lo mismo. Es el error de configuración más frecuente de este laboratorio.
- **El compartimento no es un grupo de recursos de Azure ni una cuenta de AWS.** Es un contenedor lógico con jerarquía y políticas propias, dentro de un solo tenancy.
- **Audit está siempre encendido.** No hay que habilitar un «trail». Los registros de red y de servicio sí hay que habilitarlos uno por uno, y por eso tienen su propio capítulo.

3. Prerrequisitos

3.1 Lo que debe existir antes de empezar

Requisito	Detalle
Un servicio corriendo con métricas	Un balanceador con un grupo de instancias detrás. Este manual observa ese servicio; no lo crea
Dos compartimentos	Uno para la observabilidad (`lab-05-observabilidad`) y el del servicio observado
El plugin de monitoreo activo	En las instancias del grupo. Sin él, el espacio de nombres `oci_computeagent` no publica nada
Una dirección de correo a la que pueda entrar	Va a tener que abrirla y hacer clic en un enlace
Permisos de la consola	Ver abajo

Los dos compartimentos podrían ser el mismo, pero se separan a propósito: la operación mira todo desde un solo lugar aunque los recursos vivan repartidos. Ese es el modelo que sirve cuando hay más de un ambiente.

3.2 Permisos

Si usted es administrador del tenancy, puede saltarse esta sección. Si no, el grupo al que pertenece necesita al menos estas sentencias, expresadas en el lenguaje de políticas de OCI:

```
Allow group Operacion to manage alarms in compartment lab-05-observabilidad
Allow group Operacion to manage ons-topics in compartment lab-05-observabilidad
Allow group Operacion to read metrics in compartment lab-laboratorio
Allow group Operacion to manage log-groups in compartment lab-05-observabilidad
Allow group Operacion to manage log-content in compartment lab-05-observabilidad
Allow group Operacion to read instance-family in compartment lab-laboratorio
Allow group Operacion to read load-balancers in compartment lab-laboratorio
```

Dos detalles fáciles de pasar por alto:

- `read metrics` va sobre el compartimento **del servicio observado**, no sobre el de las alarmas. Sin ese permiso la pantalla de creación de la alarma se ve completa pero el desplegable de métricas aparece vacío.
- `manage ons-topics` es necesario para que la alarma pueda escribir en el tema.

Consola de OCI Identity & Security › Policies

3.3 Verificar el plugin de monitoreo

Las métricas de CPU del espacio de nombres `oci_computeagent` las publica un plugin del agente que corre dentro de cada instancia. Si está deshabilitado, no hay métrica; si no hay métrica, la alarma se crea igual y nunca suena.

Consola de OCI Compute › Instances › (una instancia del grupo) › Oracle Cloud Agent

1. Abra una de las instancias que están detrás del balanceador.
2. Entre a la pestaña del agente de Oracle Cloud.
3. Confirme que el plugin de monitoreo de la instancia de cómputo está habilitado.
4. Repita en una segunda instancia. Si el grupo crece por autoescalamiento, las instancias nuevas heredan la configuración de la plantilla del grupo: si la plantilla lo tiene apagado, las nuevas nacen ciegas.

Una alarma sobre una métrica que no existe se crea sin ningún error. La consola acepta cualquier nombre de métrica. Si el nombre está mal escrito, si el espacio de nombres no es el correcto o si el plugin está apagado, la alarma queda creada, se ve perfecta en la lista y nunca dispara.

Es la peor forma de fallar, porque parece que funciona. Por eso el capítulo 6 explora la métrica en el gráfico **antes** de crear la alarma: si el número no se ve dibujado en pantalla, la alarma no va a servir.

3.4 Cuotas y límites en una cuenta de prueba

Servicio	Qué esperar
Monitoring	Sin costo relevante. Las alarmas no cobran por existir
Notifications	Cuota mensual de envíos de correo holgada para un laboratorio. [VALIDAR] la cifra vigente en la página de precios
Logging	**Cobra por lo que ingiere y almacena.** Es lo único de este manual con costo por volumen
Dashboards	Sin costo

El ensayo completo del laboratorio midió **0,13 USD** de cómputo en total, y la capa de observabilidad no movió esa cifra de forma apreciable.

3.5 Una advertencia sobre el ambiente observado

El servicio que se observa tiene el puerto 22 y el bastión abiertos a `0.0.0.0/0`. **Está así a propósito, por ser un ambiente desechable** que se destruye al final del día y no tiene datos. No es una recomendación.

En un ambiente con datos, el acceso administrativo se restringe a los bloques CIDR concretos desde los que se administra, el bastión lleva su propia lista de clientes permitidos y el puerto 22 no se expone a internet en ninguna subred. Lo señalamos aquí porque más adelante, en los registros de red, ese `0.0.0.0/0` va a aparecer en forma de tráfico que nadie esperaba.

4. El tema de notificaciones

El tema es el buzón al que las alarmas escriben. Las alarmas no envían correos: publican en un tema, y el tema reparte a sus suscriptores. Esa indirección es la que permite cambiar destinatarios sin tocar ninguna alarma.

Consola de OCI Observability & Management › Notifications › Topics › Create Topic

Campo	Valor
Name	`lab05-alarmas-operacion`
Description	`Alarmas de plataforma del laboratorio de observabilidad`
Compartment	el compartimento de observabilidad

1. Abra el menú de navegación de la consola. Si no encuentra **Notifications** en el menú, escríbalo en la barra de búsqueda de la consola: en algunas versiones el servicio aparece además bajo la categoría de integración de aplicaciones. Es el mismo servicio.
2. Verifique en el selector de compartimento de la izquierda que está parado en el compartimento de observabilidad. Todo lo que cree a continuación nace ahí.
3. Presione **Create Topic**.

4. En **Name**, escriba `lab05-alarmas-operacion`. El nombre admite letras, números, guiones y guiones bajos; no admite espacios.
5. En **Description**, escriba una línea que diga de quién son estas alarmas. En seis meses, con cuatro temas creados, la descripción es lo único que los distingue.
6. Agregue etiquetas de forma libre. Para un laboratorio son útiles tres:

Etiqueta	Valor
`Proyecto`	`LaboratorioObservabilidad`
`Owner`	quien lo creó
`Efimero`	`si`

La tercera es la que permite encontrar y borrar después todo lo que era desechable. Un recurso sin etiqueta de dueño sobrevive a todas las limpiezas.

1. Presione **Create**. El tema aparece en la lista en estado activo, con su OCID visible en la página de detalle: algo de la forma `ocid1.onstopic.oc1..aaaaEJEMPLO`.

Un tema por audiencia, no por alarma. La tentación es crear un tema por alarma. Termina en veinte temas y nadie sabe quién está suscrito a qué.

El criterio útil es la audiencia: un tema para lo que atiende el turno de operación, otro para lo que va a seguridad, otro para lo que le interesa a quien mira el presupuesto. Las alarmas cambian; las audiencias, casi nunca.

5. La suscripción de correo, y el clic que no se puede saltar

Este capítulo tiene un solo paso que importa, y es el que más veces se salta.

Consola de OCI Observability & Management › Notifications › Topics › lab05-alarmas-operacion › Create Subscription

Campo	Valor
Protocol	Email
Email	la dirección que va a recibir las alarmas

1. Entre al tema que acaba de crear.
2. En el panel de recursos de la izquierda, seleccione la lista de suscripciones.
3. Presione **Create Subscription**.
4. En **Protocol**, elija el protocolo de correo electrónico. El desplegable trae también

mensajería, funciones y llamadas HTTPS; para este laboratorio es correo.

5. Escriba la dirección de destino. Por ejemplo, `operacion@ejemplo.com`.
6. Presione **Create**.
7. Mire la columna de estado en la lista de suscripciones. Dice **Pending**.

El estado que decide si todo esto sirve

IMPORTANTE · La suscripción queda en PENDING hasta que alguien hace clic

Al crear la suscripción, OCI envía un correo de confirmación con un enlace. **Hasta que alguien abre ese correo y hace clic, la suscripción sigue en `PENDING` y no recibe absolutamente nada.**

Y aquí está la trampa: la alarma dispara igual. En la consola se ve todo perfecto —la alarma existe, está habilitada, pasa a `FIRING`, el tema tiene su suscripción— y la bandeja de entrada sigue vacía. No hay ningún mensaje de error en ninguna pantalla que le diga que le falta el clic.

Es la forma más común de montar una observabilidad que parece funcionar y no funciona. Si de este manual solo se acuerda de una cosa, que sea esta.

1. Abra el correo. El remitente es el servicio de notificaciones de Oracle Cloud y el cuerpo trae un enlace de confirmación.
2. Haga clic en el enlace. Se abre una página que confirma la suscripción.
3. Vuelva a la consola y refresque la lista. El estado debe decir **Active**.

Estado	Qué significa
`PENDING`	**No va a llegar nada.** Falta el clic
`ACTIVE`	Listo. Las alarmas de este tema llegan a ese correo

1. Si el correo no aparece en unos minutos, revise la carpeta de no deseados. Los filtros corporativos son especialmente aficionados a los correos con enlaces de confirmación.
2. Si aun así no llega, el menú de acciones de la suscripción permite reenviar la confirmación. Úselo antes de borrar y volver a crear: una suscripción borrada y recreada genera otro correo idéntico, y termina con dos pendientes.

Verificación por CLI

La consola ya lo dice, pero este es el comando que conviene dejar anotado, porque es el que se corre sin abrir el navegador antes de una jornada importante:

```
COMP07="ocid1.compartment.oc1..aaaaEJEMPLO"

oci ons subscription list --compartment-id "$COMP07" \
  --query 'data[].{correo:endpoint,estado:"lifecycle-state"}' --output table
```

En el ensayo de este laboratorio, esta verificación fue el único punto pendiente de toda la preparación, y se resolvió confirmando la suscripción el mismo día en que se creó. La regla que salió de ahí: **crear la suscripción y confirmarla son el mismo paso**, no dos.

Una prueba de 30 segundos que ahorra una madrugada

1. En la página del tema, use la acción de publicar un mensaje de prueba.
2. Escriba cualquier texto en el cuerpo y envíelo.
3. El correo debe llegar en menos de un minuto.

Si llega, la cadena tema → suscripción → bandeja de entrada está probada y ya no va a ser la causa de que una alarma no se vea. Si no llega, el problema está aquí y no en la alarma que todavía no ha creado. Probar la cadena por partes, en orden, es lo que evita diagnosticar tres cosas a la vez.

6. La métrica antes de la alarma: el explorador y su lenguaje de consulta

Antes de crear la alarma hay que ver la métrica dibujada. Este capítulo no crea nada: prepara la consulta que el siguiente capítulo va a usar.

6.1 Ver la métrica

Consola de OCI Observability & Management › Monitoring › Metrics Explorer

Campo	Valor
Compartment	el compartimento **del servicio observado**
Metric namespace	`oci_computeagent`
Metric name	`CpuUtilization`
Interval	`1m`
Statistic	`Mean`

1. Abra el explorador de métricas.
2. En **Compartment**, seleccione el compartimento donde viven las instancias. No el de observabilidad. Si se equivoca aquí, el desplegable de métricas aparece vacío y parece un problema de permisos.
3. En **Metric namespace**, elija `oci_computeagent`. Los espacios de nombres son la forma en que OCI separa las métricas por servicio: `oci_computeagent` para lo que publica el agente dentro de la instancia, `oci_lbaas` para el balanceador, `oci_vcn` para la red, y así.
4. En **Metric name**, elija `CpuUtilization`.

5. Deje **Interval** en `1m` y **Statistic** en el promedio.
6. Actualice el gráfico.
7. **Debe ver una línea con datos.** Si el gráfico está vacío:

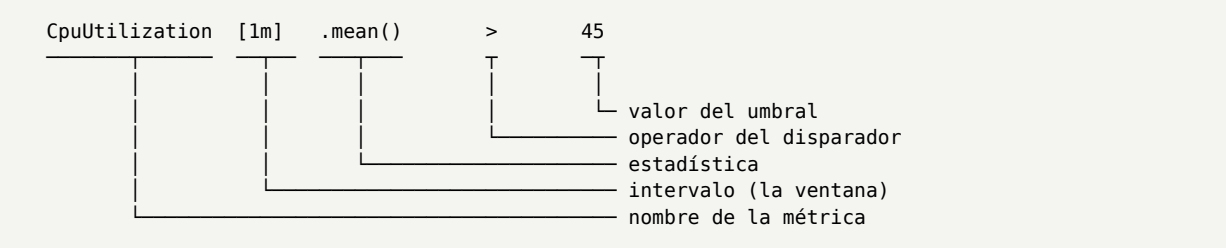
Lo que ve	Qué revisar
Gráfico vacío	El compartimento del paso 2
Gráfico vacío y el compartimento es el correcto	El plugin de monitoreo (sección 3.3)
El servicio se levantó hace poco	Espere 10 minutos. Las métricas tardan en publicarse

1. Fíjese en que hay **varias líneas**, una por instancia. Cada combinación de dimensiones es una serie distinta. Esto va a importar en el capítulo 9.

6.2 Cómo se arma la expresión

La consola tiene dos formas de decir lo mismo. En el modo básico usted llena campos; al activar el modo avanzado, la consola le muestra la consulta que esos campos producen, escrita en MQL, el lenguaje de consulta de métricas de OCI.

Vale la pena mirarla una vez, porque es la que va a leer cuando algo no dispare:



Y la equivalencia con los campos de la consola:

Pieza de la consulta	Campo en la consola	Qué decide
`CpuUtilization`	Metric name	**Qué** se mide
`[1m]`	Interval	Sobre **cuánto tiempo** se agrupan los datos crudos
`.mean()`	Statistic	**Cómo** se resumen los puntos de esa ventana
`> 45`	Operator + Value	**Cuándo** se considera que la condición se cumple
—	Trigger delay minutes	**Cuánto tiempo** debe cumplirse antes de sonar

6.3 El intervalo

El intervalo es la ventana sobre la que se agrupan los datos crudos antes de aplicar la estadística. Con `[1m]`, cada punto de la serie resume un minuto de mediciones.

Intervalo	Cuándo conviene
`1m`	Señales que hay que ver rápido: CPU, salud de backends, errores
`5m`	Señales ruidosas por naturaleza, donde el minuto suelto miente
`1h`	Tendencias: capacidad, almacenamiento, consumo

Un intervalo más corto no es «mejor»: es más sensible. Y más sensible, en una señal ruidosa, significa más correos que nadie va a leer.

Hay un piso que no se puede bajar: el intervalo no puede ser más fino que la frecuencia con la que el servicio publica la métrica. Pedir `[10s]` sobre una métrica que se publica cada minuto no da más resolución; da huecos.

6.4 La estadística

La estadística resuelve los puntos de la ventana en un solo número. La elección cambia por completo lo que la alarma significa.

Estadística	Qué responde	Ejemplo de uso
`mean()`	¿Cómo estuvo en promedio?	CPU sostenida
`max()`	¿Llegó a pasar aunque fuera una vez?	Backends caídos, errores
`min()`	¿Estuvo siempre por encima de…?	Verificaciones de disponibilidad
`sum()`	¿Cuánto en total?	Peticiones, bytes
`count()`	¿Cuántos puntos hubo?	Detectar métricas que dejan de publicarse
`rate()`	¿Qué tan rápido cambia?	Crecimiento de una cola
`percentile()`	¿Cómo le fue al peor tramo?	Latencia p95, p99

La regla práctica, y es la que se aplica en las dos alarmas de este manual:

- Para **causas** —CPU, memoria, disco— se usa `mean()`. Un pico de un minuto no es un problema; un promedio sostenido sí.
- Para **síntomas** —backends caídos, errores— se usa `max()`. Si pasó una sola vez, ya pasó, y el usuario ya lo sintió.

La latencia promedio esconde justamente lo que hay que ver. Un promedio de 200 ms puede ser un servicio impecable o uno donde el 5 % de los usuarios espera 8 segundos. El promedio no los distingue; el percentil 95 sí.

Para cualquier señal de experiencia de usuario —latencia, tiempo de respuesta, tiempo de proceso— la estadística por defecto debería ser un percentil alto y no el promedio. Es el cambio de una sola casilla que más mejora un modelo de observabilidad heredado.

6.5 Dimensiones y agrupación

Una métrica no es una línea: es un conjunto de líneas, una por combinación de dimensiones. `CpuUtilization` trae, entre otras, el identificador del recurso y su nombre visible.

En la consulta, las dimensiones se filtran entre llaves:

```
CpuUtilization[1m]{resourceDisplayName = "lab01-app"}.mean() > 45
```

Y hay una decisión de fondo, que es la que explica por qué la alarma de este laboratorio se cierra sola:

- **Sin agrupar** (lo que hace este manual), la alarma evalúa la condición **por cada instancia**. Si una instancia pasa del umbral, la alarma dispara.
- **Agrupando** con `.grouping()`, todas las series se colapsan en una sola y la condición se evalúa sobre el conjunto.

Como aquí se evalúa por instancia y el promedio de CPU **por instancia** baja cuando el grupo crece, el autoescalamiento apaga la alarma sin que nadie intervenga. Si se agrupara por suma, el total no bajaría al crecer el grupo y la alarma no se cerraría nunca. La misma métrica, el mismo umbral, y un comportamiento opuesto.

6.6 La función que detecta la ceguera

MQL tiene una función que casi nadie usa y que resuelve el estado más peligroso de todos:

```
CpuUtilization[5m].absent()
```

Dispara cuando la métrica **deja de llegar**. Es la alarma sobre la alarma: avisa de que el plugin se apagó, de que la instancia desapareció o de que alguien cambió el nombre de algo.

No la vamos a crear en este laboratorio, pero conviene saber que existe, porque el capítulo 14 muestra por qué hace falta: una alarma sin datos se ve exactamente igual que una alarma tranquila.

7. La alarma de saturación de CPU

Primera alarma. Es una **causa**: avisa de que algo puede estar por degradarse, no de que el usuario ya lo esté sintiendo. Por eso su severidad es `WARNING`.

Consola de OCI Observability & Management › Monitoring › Alarm Definitions › Create Alarm

7.1 Definición de la alarma

Campo	Valor

de este manual no termina cuando la alarma aparece creada.

1. Elija el espacio de nombres `oci_computeagent` y la métrica `CpuUtilization`. Si el desplegable aparece vacío, vuelva al paso 6: casi siempre es el compartimento.
2. Deje **Interval** en `1m` y **Statistic** en el promedio.
3. No agregue dimensiones. Queremos que la alarma evalúe instancia por instancia, por la razón de la sección 6.5.
4. Si quiere confirmar la consulta, active el modo avanzado. Debe leerse:

```
CpuUtilization[1m].mean() > 45
```

7.3 La regla del disparador

Campo	Valor
Operator	greater than
Value	45
Trigger delay minutes	3

1. En **Operator**, elija «mayor que».
2. En **Value**, escriba `45`.

El número no es arbitrario. El servicio observado tiene autoescalamiento configurado con un umbral de **55 %**. La alarma se pone **por debajo**: en 45. Así la alarma suena primero y el autoescalamiento la apaga después. Si la alarma estuviera por encima del umbral de escalamiento, la plataforma reaccionaría antes de que la CPU llegara al umbral de la alarma y **la alarma no sonaría nunca**.

1. En el retardo del disparador, escriba `3` minutos. Es la traducción del valor ISO-8601 `PT3M`: la condición tiene que sostenerse tres minutos seguidos antes de que la alarma pase a `FIRING`.

Sin ese retardo, cualquier pico de treinta segundos genera un correo. En dos semanas nadie los lee, y entre esos correos que nadie lee va a estar el que sí importaba.

7.4 A dónde llega

Campo	Valor
Destination service	Notifications
Compartment	el compartimento de observabilidad

Topic	`lab05-alarmas-operacion`

1. En la sección de destino, elija el servicio de notificaciones.
2. Seleccione el compartimento de observabilidad y el tema `lab05-alarmas-operacion`. Si no aparece, revise que el selector de compartimento sea el correcto.
3. Deje el formato de mensaje en la opción de **mensaje formateado** —la que corresponde al valor `ONS_OPTIMIZED`—. Las otras opciones entregan el evento en crudo o en JSON, que sirven cuando el destino es una función o un sistema de tiquetes, no un ser humano.

7.5 El cuerpo del mensaje — el paso que de verdad baja el tiempo de recuperación

1. En el cuerpo de la alarma, escriba un texto que sirva a alguien con sueño:

```
La CPU del servicio superó el 45 % durante 3 minutos.

QUE HACER (runbook): https://<su-repo>/runbooks/RUNBOOK-saturacion-cpu.md

Antes de escalar manualmente, verificar si el autoescalamiento ya reaccionó:
si el grupo creció y la carga es legítima, esta alarma se cierra sola y NO
requiere acción — solo queda registrada para la revisión semanal.
```

Tres cosas, y ninguna es decorativa:

- **El enlace al runbook viaja dentro del correo.** Quien lo recibe de madrugada no tiene que buscar en ningún wiki. Esto es lo que baja el tiempo de recuperación: no el tablero, no la herramienta, sino que la instrucción viaje con la alarma.
- **La primera instrucción es descartar, no diagnosticar.** «¿La plataforma ya lo está resolviendo sola?» es la pregunta que ahorra la mitad de las madrugadas.
- **Dice explícitamente que puede no requerir acción.** Un runbook que solo contempla la rama «hay que hacer algo» entrena al equipo a hacer algo siempre.

El cuerpo del mensaje es el runbook de emergencia. Si mañana el repositorio de runbooks no abre —porque es privado, porque cambió la ruta, porque la VPN no levanta—, lo único que queda es el texto del correo.

Escríbalo pensando en eso: tres líneas que digan qué significa la alarma, qué revisar primero y a quién llamar si eso no alcanza. El enlace es el detalle; el texto es el respaldo del enlace.

7.6 Repetición y cierre

Campo	Valor
Repeat notification	Habilitado

Notification frequency	30 minutos

1. Habilite la repetición de la notificación y póngale 30 minutos. Mientras la alarma siga disparada, llega un recordatorio cada media hora.

Más frecuente cansa y se convierte en ruido. Menos frecuente deja que algo se olvide durante un turno entero.

1. Deje habilitada la alarma.
2. Agregue las mismas etiquetas del tema.
3. Presione **Save alarm**.

La alarma aparece en la lista. En la página de detalle verá su OCID (`ocid1.alarm.oc1..aaaaEJEMPLO`) y un gráfico con la métrica y la línea del umbral dibujada encima. **Ese gráfico es la primera validación**: si la línea del umbral está muy por encima de todo lo que se ve, la alarma nunca va a sonar; si está por debajo del ruido de fondo, va a sonar siempre.

8. La alarma de backends no saludables

Segunda alarma. Esta es un **síntoma**: si hay servidores que no pasan la verificación de salud, el usuario ya está viendo errores o lentitud. Por eso su severidad es mayor aunque el número sea más pequeño.

Consola de OCI Observability & Management › Monitoring › Alarm Definitions › Create Alarm

Campo	Valor
Alarm name	`lab05-backends-no-saludables`
Alarm severity	Critical
Compartment (de la métrica)	el compartimento del servicio observado
Metric namespace	`oci_lbaas`
Metric name	`UnHealthyBackendServers`
Interval	`1m`
Statistic	`Max`
Operator	greater than
Value	`0`
Trigger delay minutes	`2`

Destination	el tema `lab05-alarmas-operacion`

1. Cree una alarma nueva con el nombre `lab05-backends-no-saludables`.
2. En severidad, elija **Critical**.
3. En la descripción de la métrica, vuelva a cambiar el compartimento al del servicio observado.
4. Elija el espacio de nombres `oci_lbaas` y la métrica `UnHealthyBackendServers`.
5. Deje el intervalo en `1m` y elija la estadística **máximo**, no el promedio. Con el promedio, un backend caído entre cuatro daría 0,25 y no pasaría el umbral de cero. Con el máximo, basta con que uno esté caído en cualquier punto de la ventana.
6. Operador «mayor que», valor `0`.
7. Retardo del disparador: `2` minutos. Más corto que el de CPU, porque el impacto es inmediato. Pero no cero: cuando el grupo crece, una instancia recién creada aparece como no saludable durante unos segundos hasta que la aplicación levanta, y sin esos dos minutos el autoescalamiento generaría una alarma crítica cada vez que hace su trabajo.
8. La consulta, en modo avanzado, debe leerse:

```
UnHealthyBackendServers[1m].max() > 0
```

1. Destino: el mismo tema `lab05-alarmas-operacion`.
2. Cuerpo del mensaje:

```
Hay servidores detrás del balanceador que no pasan la verificación de salud.
El usuario final puede estar viendo errores o lentitud.

QUE HACER (runbook): https://<su-repo>/runbooks/RUNBOOK-backends-caidos.md
```

1. Guarde.

Por qué esta es CRITICAL y la de CPU es WARNING

	CPU al 45 %	Un backend caído
Qué dice	Una causa posible	Un síntoma que ya se está viviendo
Magnitud del número	Grande	1
¿El usuario lo nota?	Todavía no	Sí
¿Despierta a alguien?	No	Sí
Severidad	WARNING	CRITICAL

La severidad la define el impacto, no la magnitud de la métrica. Es la confusión más frecuente al heredar un modelo de monitoreo: alguien pone CRITICAL en la CPU porque 90 % suena más grave que 1, y termina con un turno de guardia que se despierta por cosas que se resuelven solas.

Nadie llama al soporte para decir «su CPU está al 80 %». Llaman porque no les carga una pantalla.

9. Umbral, ventana y retardo: ajustar sin fabricar ruido

Las alarmas ya existen. Este capítulo es sobre editarlas, que es lo que de verdad se hace durante las primeras semanas de operación.

Consola de OCI Observability & Management › Monitoring › Alarm Definitions › (la alarma) › Edit

9.1 Los tres números y qué mueve cada uno

Número	En la consola	Si lo baja	Si lo sube
Umbral	Value	Suena antes y más veces	Suena menos, y puede no sonar nunca
Intervalo	Interval	Más sensible al pico corto	Más estable, más lenta
Retardo	Trigger delay minutes	Dispara más rápido	Filtra los picos pasajeros

El tiempo total hasta el correo es aproximadamente la suma de tres cosas:

```
tiempo hasta el correo  ≈   publicación de la métrica
                          + intervalo
                          + retardo del disparador
                          + entrega del correo (< 1 min)
```

Con los valores de este manual —`1m` de intervalo y 3 minutos de retardo— la alarma de CPU tarda del orden de cuatro a cinco minutos desde que la carga empieza hasta que el correo llega. **Ese número hay que medirlo, no estimarlo**, y el capítulo 14 explica cómo.

9.2 El umbral y la ventana deciden juntos si la alarma sirve

IMPORTANTE · Un umbral sin ventana es una máquina de ruido

Un umbral solo responde «¿cuánto es demasiado?». La ventana responde «¿durante cuánto tiempo?», y es la que separa una alarma útil de un generador de correos.

CPU por encima de 45 % **un instante** es un dato normal: pasa cada vez que arranca un proceso.

CPU por encima de 45 % **sostenida tres minutos** es otra cosa, y ya se puede escribir una acción para ella.

Si le dieran a elegir un solo ajuste para mejorar un modelo de alarmas heredado, sería este: casi todas las alarmas ruidosas que uno se encuentra tienen un umbral razonable y una ventana de cero.

9.3 Ajustes según el síntoma

Esta tabla viene del laboratorio y resuelve casi todo lo que aparece al ajustar:

Síntoma	Ajuste
Tarda demasiado en sonar	Baje el umbral, o baje el retardo a 2 minutos
Suena por picos que no eran nada	Suba el retardo antes que el umbral
Suena y nadie hace nada, siempre	El problema no es el umbral: es que no tiene acción escrita. Escríbala, o baje la severidad
Nunca suena	Revise que el umbral esté por debajo del de autoescalamiento, y el compartimento de la métrica
No se cierra nunca	El grupo llegó a su tamaño máximo. Es otra conversación, y es válida

Los dos últimos son los interesantes.

Que **no suene nunca** casi siempre es configuración: umbral inalcanzable, compartimento equivocado o métrica que no se publica.

Que **no se cierre nunca** casi nunca es configuración: normalmente significa que la plataforma hizo todo lo que podía y no alcanzó. Esa alarma sí merece a alguien despierto.

9.4 Que la alarma se cierre sola es tan importante como que dispare

Una alarma tiene dos transiciones, no una: entra en `FIRING` y vuelve a `OK`. La segunda casi nunca se diseña, y es la que decide si el equipo le va a hacer caso a la primera.

- **Si se cierra sola**, quien la recibió sabe que la plataforma reaccionó. La alarma queda registrada para la revisión semanal y nadie se levanta.
- **Si no se cierra sola nunca**, cada alarma queda abierta para siempre. El panel se llena de rojo permanente, y el rojo permanente es indistinguible del rojo nuevo. A las tres semanas nadie mira el panel.

Para que se cierre sola hacen falta dos condiciones, y las dos se configuran en esta pantalla:

1. Que la condición **pueda** dejar de cumplirse. Una alarma sobre un contador acumulado —

`sum()` de errores desde siempre— nunca baja. Si la métrica solo crece, la alarma no se cierra jamás: hay que medir tasa, no total.

2. Que la agrupación **deje** bajar el número. Es el caso de la sección 6.5: sin agrupar, la CPU por instancia baja al crecer el grupo y la alarma se cierra; agrupando por suma, el total no baja y la alarma se queda encendida.

CUIDADO · Una alarma que no se cierra sola entrena al equipo a ignorarla

No es un problema estético. Es el mecanismo exacto por el que un modelo de observabilidad se muere: las alarmas que quedan encendidas ensucian el panel, el panel deja de mirarse, y cuando llega la alarma que sí importaba, llega a un panel que nadie mira hace meses.

Cuando revise una alarma heredada, hágale dos preguntas en vez de una: **¿en qué condiciones dispara?** y **¿en qué condiciones se apaga?** La segunda casi nunca tiene respuesta escrita.

9.5 Ventanas de silencio para mantenimientos

La pantalla de la alarma permite suprimir notificaciones durante un rango de fechas. Sirve para un mantenimiento planificado.

Dos advertencias de uso:

- Supresión **con fecha de fin**, siempre. Una supresión indefinida es una alarma apagada que todo el mundo cree encendida.
- La alarma **sigue evaluando** durante la supresión; lo que se silencia es la notificación. En el histórico va a poder ver que disparó, lo cual es exactamente lo que se quiere para el informe posterior.

10. El tablero: ver varias señales juntas

Un tablero no reemplaza a una alarma. La alarma es para lo que no se está mirando; el tablero es para cuando ya se está mirando. Confundirlos produce equipos que tienen pantallas enormes y se enteran por el cliente.

10.1 El panel rápido: métricas por servicio

Consola de OCI Observability & Management › Monitoring › Service Metrics

Campo	Valor
Compartment	el compartimento del servicio observado
Metric namespace	`oci_computeagent`

1. Seleccione el compartimento y el espacio de nombres.
2. La página dibuja de una vez todas las métricas de ese espacio de nombres, cada una en su gráfico: CPU, memoria, red, disco.
3. Cambie el espacio de nombres a `oci_lbaas` y va a ver las del balanceador: peticiones, backends saludables y no saludables, bytes.
4. En el menú de opciones de cada gráfico hay una acción para **crear una alarma a partir de esa consulta**. Es el camino corto: en vez de llenar la pantalla de creación desde cero, se parte de una consulta que ya está dibujada y que, por estar dibujada, se sabe que tiene datos.

10.2 El estado de las alarmas

Consola de OCI Observability & Management › Monitoring › Alarm Status

1. La página lista las alarmas con su estado actual. Es la pantalla que se deja abierta en una jornada de operación.
2. Entre a una alarma. En su página de detalle está el histórico de transiciones: cuándo pasó a `FIRING`, cuándo volvió a `OK`. Ese histórico es la evidencia del ciclo completo y es lo que se guarda para la revisión semanal.

10.3 Un tablero propio

Consola de OCI Observability & Management › Dashboards

1. Cree un tablero nuevo en el compartimento de observabilidad. [VALIDAR] la ruta exacta del menú en la versión de consola que tenga: en algunas versiones los tableros se alcanzan desde el propio servicio de monitoreo.
2. Agregue widgets. Para este servicio, cuatro alcanzan:

Widget	Métrica	Para qué
CPU por instancia	`oci_computeagent` · `CpuUtilization` · `mean`	La causa
Backends no saludables	`oci_lbaas` · `UnHealthyBackendServers` · `max`	El síntoma
Peticiones al balanceador	`oci_lbaas` · conteo de peticiones	El tráfico
Tamaño del grupo	métrica del grupo de instancias	Si la plataforma reaccionó

1. El cuarto es el que casi nadie pone y el que más explica: viendo la CPU y el tamaño del grupo en la misma pantalla, el ciclo «sube la carga, crece el grupo, baja la CPU» se lee de un vistazo, sin narración.

Cuatro widgets que se miran valen más que veinte que se ignoran. El tablero útil cabe en una pantalla y responde cinco preguntas: ¿está arriba?, ¿va rápido?, ¿está fallando?, ¿aguanta lo que viene?, ¿cuánto cuesta?

La quinta es la que casi siempre falta, y es la que más agradece quien firma el presupuesto.

11. Los registros: logs de servicio y flow logs

Las métricas dicen **que** algo pasa. Los registros dicen **qué** pasó. La cadena completa de un diagnóstico es: alarma → métrica → registro → recurso. Sin el tercer eslabón, cuando algo no se resuelve solo, hay que entrar a las máquinas a mirar archivos, y eso a las tres de la mañana no escala.

En OCI los registros se organizan en **grupos de registro** (log groups) que contienen **registros** (logs). Los de servicio se habilitan uno por uno: no vienen encendidos.

11.1 Crear el grupo de registro

Consola de OCI Observability & Management › Logging › Log Groups › Create Log Group	
Campo	**Valor**
Compartment	el compartimento de observabilidad
Name	`lab05-logs`
Description	`Registros del laboratorio de observabilidad`

1. Entre a los grupos de registro y presione **Create Log Group**.
2. Nómbrelo `lab05-logs`.
3. Presione **Create**.

El grupo es un contenedor: no ingiere nada por sí mismo y no cobra. Sirve para aplicar permisos y retención a un conjunto de registros de una vez.

11.2 Habilitar los flow logs de la subred

Los flow logs registran **quién habló con quién** dentro de la red: origen, destino, puerto, si se aceptó o se rechazó. Es el registro que responde la pregunta «¿esto llegó a entrar?» sin tener que entrar a ninguna máquina.

Consola de OCI Observability & Management › Logging › Logs › Enable service log	
Campo	**Valor**

Resource compartment	el compartimento del servicio observado
Service	el de registros de flujo de red (Flow Logs)
Resource	la subred donde están las instancias de aplicación
Log category	la categoría que incluye todos los registros
Log name	`lab05-flowlogs-app`
Log group	`lab05-logs`
Retention	`30` días

1. En la lista de registros, presione la acción de habilitar un registro de servicio.
2. Seleccione el compartimento del servicio observado.
3. En el desplegable de servicio, elija el de registros de flujo de red. Ese desplegable es, además, el catálogo de qué servicios de OCI pueden publicar registros: vale la pena abrirlo una vez y mirarlo entero.
4. En el recurso, elija la subred de aplicación. Los flow logs se habilitan por subred o por VCN, no por instancia.
5. Elija la categoría que incluye todos los registros.
6. Nombre el registro `lab05-flowlogs-app`.
7. Abra las opciones avanzadas y seleccione el grupo `lab05-logs`. Si no lo hace, la consola va a crear o elegir otro grupo y el registro va a terminar donde no lo va a buscar.
8. Deje la retención en 30 días. Es suficiente para investigar un incidente, y la retención es el principal multiplicador de la factura de registros.
9. Presione **Enable log**.

CUIDADO · Los registros son lo único de este manual que cobra por volumen

Las alarmas y el tema de notificación no tienen costo relevante. Los registros cobran por lo que ingieren y por lo que almacenan, y los flow logs de una subred con tráfico real generan bastante.

Tres decisiones que controlan el costo, en orden de impacto: **qué subredes** se registran, **qué retención** se les pone, y si se envían o no a un almacenamiento más barato para archivo. Habilitar flow logs en toda la VCN «por si acaso» es la forma más común de sorprenderse con la factura.

Para un laboratorio efímero no importa. Para el ambiente real, decídalo antes.

11.3 Habilitar los registros del balanceador

1. Repita la operación con el balanceador como recurso. Sus categorías de registro son dos y sirven para cosas distintas:

Categoría	Qué trae
Acceso	Una línea por petición: ruta, código de respuesta, tiempo
Error	Los fallos del propio balanceador

1. El registro de acceso es el que permite responder «¿cuántos 5xx hubo entre las 14:10 y las 14:20 y en qué rutas?», que es la pregunta que sigue a casi cualquier alarma de errores.

11.4 Lo que ya está registrado sin hacer nada

Registro	Estado	Qué trae
Audit	Siempre encendido	Toda llamada a la API: quién creó, borró o modificó qué
Flow logs	Hay que habilitarlo	Tráfico de red aceptado y rechazado
Registros de servicio	Hay que habilitarlos, uno por servicio	Acceso, errores, operaciones
Registros de la aplicación	Hay que enviarlos con un agente	Lo que escribe su código

Audit merece una mención aparte, porque resuelve solo la pregunta más frecuente después de una caída: **¿qué cambió?** La causa más probable de una caída total no es una avería: es un cambio. Audit está encendido desde el primer día del tenancy y nadie tiene que acordarse de habilitarlo.

12. Consultar los registros

Habilitar registros que nadie sabe consultar es gastar dinero en almacenamiento. Este capítulo es el que convierte el gasto en capacidad de diagnóstico.

Consola de OCI Observability & Management › Logging › Search

1. Abra la búsqueda de registros.
2. Seleccione el rango de tiempo. Empiece por la última hora: los rangos amplios sobre registros grandes tardan y casi nunca hacen falta al principio.
3. Seleccione qué registros buscar. Puede elegir un compartimento entero, un grupo de registro o un registro concreto. Para empezar, elija `lab05-flowlogs-app`.

4. La consola arma una consulta base y muestra los resultados más recientes. Verifique que hay líneas. **Si no hay ninguna después de diez minutos**, el registro está habilitado pero no está recibiendo: revise que la subred elegida sea la que tiene tráfico.

12.1 El lenguaje de consulta

La consulta base tiene esta forma:

```
search "<compartimento>/<grupo de registro>/<registro>"
```

Y se encadenan operaciones con barras verticales, igual que en una tubería de shell:

```
search "lab-laboratorio/lab05-logs/lab05-flowlogs-app"
  | sort by datetime desc
```

Filtrar por un campo:

```
search "lab-laboratorio/lab05-logs/lab05-flowlogs-app"
  | where data.destinationPort = 80
  | sort by datetime desc
```

Agrupar y contar, que es lo que responde «¿quién está generando este tráfico?»:

```
search "lab-laboratorio/lab05-logs/lab05-flowlogs-app"
  | where data.destinationPort = 80
  | summarize count() by data.sourceAddress
```

Los campos propios de cada registro van bajo el prefijo `data`. Para los flow logs, los más usados son la dirección de origen, la de destino, el puerto de destino, el protocolo y el resultado (si el paquete se aceptó o se rechazó). **[VALIDAR] el nombre exacto del campo de resultado en su versión**: cambió entre versiones del formato de flow logs, y la forma segura de averiguarlo es expandir una línea de resultado en la consola y leer los nombres reales.

Ese, de hecho, es el método general y vale para cualquier registro nuevo:

1. Corra la consulta base sin filtros.
2. Expanda una línea de resultado.
3. Lea los nombres de campo que trae.
4. Escriba el filtro con esos nombres.

Es más rápido que buscar el esquema en la documentación, y no se equivoca.

12.2 Tres consultas que vale la pena dejar guardadas

Tráfico rechazado en la última hora. Responde «¿alguien está tocando puertas que no debería?»:

```
search "lab-laboratorio/lab05-logs/lab05-flowlogs-app"
  | where <campo de resultado> = 'REJECT'
  | summarize count() by data.sourceAddress
```

En el servicio observado, con el puerto 22 abierto a `0.0.0.0/0`, esta consulta devuelve tráfico de escaneo desde internet a las pocas horas de levantar el ambiente. Es la demostración más concreta de por qué esa apertura es aceptable en un laboratorio desechable y no lo es en un ambiente con datos.

Códigos de respuesta del balanceador, sobre el registro de acceso:

```
search "lab-laboratorio/lab05-logs/<registro de acceso del balanceador>"
  | summarize count() by <campo del codigo de respuesta>
```

Qué cambió en la última hora, sobre Audit. Es la primera consulta después de cualquier caída inesperada:

```
search "lab-laboratorio/_Audit"
  | sort by datetime desc
```

1. Cada consulta útil se guarda. La consola permite guardar búsquedas; una búsqueda guardada con un nombre claro es, en la práctica, un runbook de una sola línea.

12.3 La cadena completa, con un ejemplo

Así se ve el diagnóstico cuando las tres capas están puestas:

```
1. Llega el correo          alarma lab05-backends-no-saludables · CRITICAL
        |
2. Métrica                  UnHealthyBackendServers: de 0 a 2 a las 14:12
        |
3. Registro de acceso       5xx desde las 14:11, en dos backends
        |
4. Audit                    a las 14:09 alguien modificó una regla de red
        |
5. Recurso                  se revierte el cambio · el servicio vuelve
```

Sin el paso 4, el equipo habría reiniciado instancias durante veinte minutos. **La causa más probable de una caída total es un cambio, no una avería**, y el registro que lo dice ya estaba encendido.

13. La matriz: qué despierta a alguien y qué no

Este es el entregable del laboratorio. Las alarmas de los capítulos 7 y 8 son dos ejemplos; la matriz es el método.

La idea es simple y difícil de sostener: **una alarma sin acción escrita no es una alarma, es ruido.** Y el ruido tiene un costo que se paga tarde: dentro de seis meses nadie mira las alarmas, incluidas las buenas.

13.1 Síntoma y causa

La distinción que ordena todo lo demás:

	Qué dice	Ejemplos	¿Despierta a alguien?
Síntoma	El usuario lo está sintiendo	No responde · va lento · da error	**Sí**
Causa	Por qué puede estar pasando	CPU · memoria · disco · conexiones	Casi nunca

La mayoría de los equipos empieza por las causas, porque son las fáciles de medir: la nube las regala. Pero nadie llama al soporte para decir que una CPU está al 80 %.

13.2 Las cuatro preguntas por señal

Por cada señal candidata se responden cuatro cosas, en este orden:

1. **¿Qué umbral?**
2. **¿Quién responde?**
3. **¿Qué hace?** (el runbook, aunque sea de tres líneas)
4. **¿Esto despierta a alguien de madrugada?**

La cuarta es la que hace trabajar a la matriz. En cuanto alguien dice «sí, despierta» y la casilla de acción está vacía, esa fila es ruido — y tiene exactamente dos salidas válidas: **escribirle la acción, o bajarle la severidad.** Dejarla como está no es una de ellas.

Estado	Cuándo
Lista	Existe, con umbral acordado, responsable y acción
Ruido	Despierta a alguien sin responsable o sin acción escrita
Por implementar	No existe todavía
Incompleta	Existe pero le falta umbral acordado, responsable o acción

13.3 El catálogo de partida

Dieciséis señales candidatas en diez categorías. Las dos que este manual construyó son la S-06 y la S-07.

ID	Categoría	Señal	Síntoma o causa	Umbral sugerido
S-01	Disponibilidad	El servicio responde desde fuera (prueba sintética)	Síntoma	2 fallos seguidos
S-02	Latencia	Latencia p95 de la API	Síntoma	Acordar con el compromiso de servicio
S-03	Errores	Tasa de respuestas 5xx	Síntoma	> 1 % durante 5 min
S-04	Errores	Fallos de integración con sistemas externos	Síntoma	> N por hora
S-05	Tráfico	Caída abrupta del volumen de transacciones	Síntoma	< 50 % de lo normal para esa hora
S-06	Saturación	CPU de la capa de aplicación	Causa	**> 45 % sostenido 3 min**
S-07	Disponibilidad	Servidores no saludables tras el balanceador	Síntoma	**> 0 durante 2 min**
S-08	Saturación	Conexiones o memoria de la base de datos	Causa	> 80 % del máximo
S-09	Capacidad	Almacenamiento de la base de datos	Causa	> 75 % (avisa con semanas, no con horas)
S-10	Capacidad	El grupo lleva rato en su tamaño máximo	Causa	En el máximo más de 15 min
S-11	Negocio	Transacciones detenidas en proceso	Síntoma	> N detenidas más de 4 h
S-12	Negocio	Tiempo de proceso p95 por canal	Síntoma	Acordar con el compromiso de servicio
S-13	Costo	Consumo del mes contra el presupuesto	Causa	50 % · 75 % · 90 % · proyección
S-14	Seguridad	Cambios en identidades y reglas de red	Causa	Cualquier cambio, siempre
S-15	Seguridad	Certificado TLS próximo a vencer	Causa	Faltan menos de 30 días
S-16	Confiabilidad	Tiempo medio de recuperación del último mes	Indicador	No es alarma: se revisa cada mes

Dos filas donde conviene detenerse:

- **S-13, el costo.** Casi nadie la tiene como señal de operación, y es la que más agradece quien firma el presupuesto. Se implementa con presupuestos y alertas, no con alarmas de métrica.
- **S-11 y S-12, las de negocio.** Son las únicas que detectan que el servicio está roto **para el cliente** aunque toda la infraestructura esté verde. Salen de una consulta a la base de datos, no de una métrica de plataforma.

13.4 Cómo se lee el resultado

Con la matriz llena, tres lecturas:

1. **Señales que despiertan sin acción escrita.** Si el número no es cero, esa es la primera semana de trabajo — y es la más barata de todas, porque consiste en *escribir, no en construir*.
2. **Cobertura por categoría.** Las categorías vacías son puntos ciegos. *Disponibilidad* y *Errores* no se pueden quedar vacías: son las dos que el cliente nota primero.
3. **Señales que despiertan 24x7.** Si son más de seis o siete, es una promesa que ningún equipo pequeño sostiene. Mejor menos alarmas y que se les haga caso.

13.5 El número que casi nadie mide

Cuatro indicadores rodean esta conversación, y el más útil es el que menos se mide:

Número	Qué mide	Trampa habitual
Disponibilidad	% del tiempo que el servicio responde	Se mide desde dentro, donde casi siempre responde
MTTD (detección)	Desde que empieza el problema hasta que alguien se entera	Casi nadie lo mide, y es donde está el margen
MTTR (recuperación)	Desde que se detecta hasta que se resuelve	Se confunde con «desde que se reportó»
Frecuencia	Cuántas veces al mes	Tres caídas de 5 min molestan más que una de 15

Si el cliente avisa antes que la alarma, el problema no fue la caída: fue la observabilidad. Y ese es un problema que se arregla con configuración, no con más infraestructura.

Y sobre los «nueves», la conversación honesta:

Disponibilidad	Tiempo caído al mes	Qué exige en la práctica

99 %	~7 h	Monitoreo básico y atención en horario
99,5 %	~3,6 h	Alarmas con dueño; guardia de alguna forma
99,9 %	~43 min	Guardia real, redundancia, runbooks probados
99,95 %	~22 min	Automatización de la recuperación
99,99 %	~4 min	Multi-zona, despliegues sin corte, inversión considerable

Cada nueve adicional cuesta del orden de diez veces más. Prometer 99,9 % sin guardia ni redundancia no es un compromiso: es una esperanza.

14. Validación de extremo a extremo

Nada de lo anterior está validado hasta que la alarma complete el ciclo: `OK` → `FIRING` → `OK`. Este capítulo lo provoca y lo mide.

14.1 Antes de generar carga

Consola de OCI Observability & Management › Monitoring › Alarm Status

1. Confirme que las dos alarmas aparecen y que su estado es `OK`.
2. Confirme que la suscripción está en `ACTIVE`, no en `PENDING`. Si se saltó el capítulo 5, este es el momento en que el laboratorio deja de funcionar.
3. Anote la hora. Va a necesitarla.

IMPORTANTE · «Sin datos» no es calma: es ceguera, y se ve idéntica

Una alarma puede estar en tres situaciones, no en dos: **tranquila** (recibe datos y la condición no se cumple), **disparada**, y **sin datos** (no está viendo nada).

La tercera es la peligrosa, porque en la lista se ve exactamente igual que la primera. Una alarma sobre una métrica que dejó de publicarse es tan silenciosa como una que está funcionando perfectamente.

Para distinguirlas: abra la alarma y mire el gráfico de su página de detalle. Si hay línea, está viendo datos. Si el gráfico está vacío con el servicio arriba, la alarma está ciega. Y si esto le importa de verdad, cree la alarma de ausencia de la sección 6.6.

La verificación equivalente por CLI, que es la que se deja corriendo en una pantalla durante una jornada de operación:

```
COMP07="ocid1.compartment.oc1..aaaaEJEMPLO"

oci monitoring alarm-status list-alarms-status --compartment-id "$COMP07" \
  --query 'data[].{alarma:"display-name",estado:status}' --output table
```

Dos detalles del comando, que costaron una corrección en el laboratorio:

- El subcomando es `list-alarms-status`, no `list`. `oci monitoring alarm-status list` **no existe**.
- Para la salud de los backends el comando es `oci lb backend-set-health get`. `oci lb backend-health list` **tampoco existe**.

14.2 Provocar el ciclo

1. Genere carga contra el balanceador. Cualquier generador sirve mientras sostenga la CPU por encima del umbral durante varios minutos; lo importante es que la carga sea suficiente para pasar de 45 % y no tanta como para saturar de inmediato.
2. Quédese mirando la pantalla de estado de alarmas y cronometre.

Esto es lo que debe pasar, y en este orden:

```
t0         empieza la carga
 │
 ├─ ~1 min   el agente publica el primer punto por encima de 45 %
 │
 ├─ +3 min   se cumple el retardo del disparador → la alarma pasa a FIRING
 │
 ├─ <1 min   el correo llega a la bandeja de entrada
 │
 ├─ ~7 min   el autoescalamiento agrega instancias
 │           (medido en el laboratorio de elasticidad: 410 s para pasar de 2 a 4)
 │
 ├─ ~12 min  la capacidad nueva ya atiende tráfico
 │           (medido: 707 s)
 │
 └─ la CPU por instancia baja de 45 % → la alarma vuelve a OK, sola
```

1. Cuando el estado cambie a `FIRING`, **abra el correo**. Debe haber llegado en menos de un minuto. Mire el cuerpo: trae el enlace al runbook. Eso es lo que va a leer alguien de madrugada, y es la pieza que de verdad baja el tiempo de recuperación.
2. **No haga nada más.** Deje correr.
3. El grupo de instancias crece, la CPU por instancia baja y la alarma vuelve a `OK` sin intervención.

14.3 Los números medidos

Este es el resultado real del ensayo de este laboratorio, y es el que se cita en vez de estimar:

Momento	Resultado
Alarma en reposo	`OK`
Con carga: CPU en **62 %** contra un umbral de **45 %**	**`FIRING`**
Tras bajar la carga, sin intervención	**`OK` — se cerró sola**

Del laboratorio de elasticidad, que es lo que explica los tiempos del diagrama:

Evento	Medido
CPU bajo carga	76 %
El grupo crece de 2 a 4 instancias	410 s
La capacidad nueva atiende tráfico	707 s
El grupo baja de 6 a 5 al cortar la carga	251 s

VALIDACIÓN · El laboratorio quedó bien si se cumplen estas cinco

1. La suscripción está en `ACTIVE`, no en `PENDING`. 2. Las dos alarmas aparecen con estado `OK` y su gráfico de detalle tiene línea (no están ciegas). 3. Bajo carga, la alarma de CPU pasa a `FIRING`. 4. **El correo llega**, y trae el enlace al runbook dentro del cuerpo. 5. **La alarma vuelve a `OK` sola**, sin que nadie toque nada.

La quinta es la que no se puede saltar. Una alarma que dispara pero no se cierra solo demostró la mitad del mecanismo, y es la mitad fácil.

14.4 Evidencia que conviene guardar

Evidencia	Por qué
Captura del correo con el enlace al runbook visible	Es la imagen del laboratorio
Captura del panel con una alarma disparada	Sirve si la demostración en vivo falla
Captura del histórico mostrando el cierre automático	La prueba de que se cerró sola
El tiempo medido desde la carga hasta el disparo	Es el número que se dice en voz alta, y no se estima

Antes de compartir cualquier captura del correo, revise que no aparezcan direcciones de correo ni identificadores internos.

15. Qué puede salir mal

Los errores de esta tabla están documentados en el laboratorio. No son hipótesis.

Síntoma	Causa	Arreglo
La alarma dispara y no llega ningún correo	La suscripción está en `PENDING`: nadie hizo clic en el enlace de confirmación	Abrir el correo y confirmar. Si no llegó, revisar no deseados y reenviar la confirmación desde el menú de la suscripción
El correo de confirmación nunca llegó	Filtro corporativo	Revisar no deseados; probar con otra dirección. No borrar y recrear la suscripción: genera pendientes duplicadas
La alarma se ve perfecta y nunca dispara	El compartimento de la métrica quedó en el de la alarma	Editar la alarma y cambiar el compartimento **de la descripción de la métrica** al del servicio observado
La alarma nunca dispara y el compartimento es correcto	Nombre de métrica o espacio de nombres equivocado	Verificar en el explorador de métricas que la métrica existe y se dibuja. Una alarma sobre una métrica inexistente se crea sin error
El desplegable de métricas aparece vacío	Falta `read metrics` en el compartimento observado, o el compartimento es el equivocado	Revisar la política y el selector de compartimento
No hay métricas de CPU en ninguna instancia	El plugin de monitoreo del agente está deshabilitado	Habilitarlo en la instancia y en la plantilla del grupo. Es la causa número uno de que el autoescalamiento tampoco reaccione
El servicio acaba de levantarse y no hay métricas	Las métricas tardan en publicarse	Esperar 10 minutos antes de concluir que algo está mal
La CPU no pasa del 48 % por más carga que se le ponga	El servidor de prueba usa hilos y el bloqueo global del intérprete serializa el bucle de CPU: en 1 OCPU el techo es la mitad exacta	Usar un generador de carga que bifurque procesos. Con umbral de 55 % la alarma no sonaría nunca; con 45 % sí
Se corta la carga y la CPU sigue arriba	Los procesos hijos del generador quedaron huérfanos	Matar también a los hijos. Sin eso, el escalamiento hacia adentro no llega nunca y la alarma no se cierra
La alarma no se cierra nunca	El grupo de instancias llegó a su tamaño máximo	Es una conversación válida, no un error: la plataforma hizo todo lo que podía. Revisar el tope de capacidad

La alarma no se cierra nunca y el grupo no está en su tope	La consulta agrupa de forma que el número no puede bajar	Revisar la agrupación: sin agrupar, la CPU por instancia baja al crecer el grupo
La alarma se cierra tan rápido que no se alcanza a ver	El autoescalamiento reaccionó antes de lo previsto	Mirar el histórico en la página de detalle de la alarma: las transiciones quedan registradas
El estado dice «sin datos» y se parece a la calma	La alarma no está viendo nada	Abrir la alarma y mirar su gráfico. Revisar espacio de nombres, nombre de métrica y compartimento de la métrica
`oci monitoring alarm-status list` devuelve error	Ese subcomando no existe	Es `oci monitoring alarm-status list-alarms-status`
`oci lb backend-health list` devuelve error	Ese comando no existe	Es `oci lb backend-set-health get --load-balancer-id … --backend-set-name …`
El enlace del runbook en el correo no abre	El repositorio es privado	Publicarlo donde la operación pueda leerlo. Un runbook que no abre de madrugada no es un runbook
La búsqueda de registros no devuelve nada	El registro se habilitó hace poco, o sobre la subred equivocada	Esperar unos minutos; verificar que la subred elegida sea la que tiene tráfico
Un filtro de la búsqueda de registros no encuentra nada	El nombre del campo no es el que se supuso	Correr la consulta sin filtros, expandir una línea y leer los nombres reales

El error que más cuesta, aunque no aparezca como error

Una alarma huérfana: apunta a un servicio que ya no existe. Queda en estado «sin datos» para siempre, en un panel que alguien mira de reojo, y es indistinguible de una alarma sana.

Es exactamente el antipatrón del que habla este laboratorio: parece observabilidad, ocupa lugar en el panel, y no vigila nada. Por eso el capítulo siguiente importa más de lo que parece.

16. Limpieza

El orden importa. Borrar el tema antes que las alarmas deja alarmas apuntando a un destino que no existe.

1. **Deshabilite o borre las alarmas.**

Consola de OCI Observability & Management › Monitoring › Alarm Definitions

Seleccione cada alarma y bórrela. Si va a volver a usar el laboratorio, deshabilitarlas es suficiente: una alarma deshabilitada no evalúa ni notifica.

1. **Borre la suscripción y después el tema.**

Consola de OCI Observability & Management › Notifications › Topics

Entre al tema, borre la suscripción y luego el tema. Un tema con suscripciones se borra igual, pero dejar la suscripción huérfana en la cuenta de correo de alguien es una descortesía innecesaria.

1. **Deshabilite los registros de servicio.**

Consola de OCI Observability & Management › Logging › Logs

Deshabilite el flow log y los registros del balanceador. **Este es el paso que de verdad detiene un cobro**: mientras el registro esté habilitado, sigue ingiriendo.

1. **Borre el grupo de registro.** No se borra si todavía tiene registros dentro: por eso va después del paso 3.

1. **Borre el tablero**, si creó uno.

1. **El servicio observado se destruye con su propio procedimiento.** Este manual no lo creó y no debe borrarlo desde aquí.

Qué se queda cobrando si la limpieza se hace mal

Lo que queda	¿Cobra?	Consecuencia
Un registro de servicio habilitado	**Sí**, por ingesta y almacenamiento	Es lo único de este manual con costo continuo apreciable
Un grupo de registro vacío	No	Ruido en la consola
Una alarma habilitada apuntando a un servicio destruido	No de forma apreciable	Queda «sin datos» para siempre: el antipatrón
Un tema con suscripción activa	No de forma apreciable	Alguien sigue suscrito a algo que ya no existe

El servicio observado (instancias, balanceador)	**Sí, y es lo caro**	Se destruye con su propio procedimiento

El orden de magnitud, medido: el ensayo completo del laboratorio —incluyendo el servicio observado— costó **0,13 USD** de cómputo. La capa de observabilidad no movió esa cifra. Lo que puede crecer, si se deja habilitado sobre una red con tráfico real, son los registros.

17. Documentación oficial

Tema	Enlace
Monitoring — inicio	https://docs.oracle.com/en-us/iaas/Content/Monitoring/home.htm
Conceptos de Monitoring	https://docs.oracle.com/en-us/iaas/Content/Monitoring/Concepts/monitoringoverview.htm
Lenguaje de consulta de métricas (MQL)	https://docs.oracle.com/en-us/iaas/Content/Monitoring/Reference/mql.htm
Gestión de alarmas	https://docs.oracle.com/en-us/iaas/Content/Monitoring/Tasks/managingalarms.htm
Notifications — inicio	https://docs.oracle.com/en-us/iaas/Content/Notification/home.htm
Temas y suscripciones	https://docs.oracle.com/en-us/iaas/Content/Notification/Tasks/managingtopicsandsubscriptions.htm
Logging — inicio	https://docs.oracle.com/en-us/iaas/Content/Logging/home.htm
Registros de servicio	https://docs.oracle.com/en-us/iaas/Content/Logging/Concepts/service_logs.htm
Flow logs de VCN	https://docs.oracle.com/en-us/iaas/Content/Network/Concepts/vcnflowlogs.htm
Dashboards	https://docs.oracle.com/en-us/iaas/Content/Dashboards/home.htm
Audit	https://docs.oracle.com/en-us/iaas/Content/Audit/home.htm
Políticas de Monitoring	https://docs.oracle.com/en-us/iaas/Content/Identity/Reference/monitoringpolicyreference.htm
Políticas de Notifications	https://docs.oracle.com/en-us/iaas/Content/Identity/Reference/notificationpolicyreference.htm

Anexo · Qué falta para que esto sea un modelo

completo

Lo que este manual construye es la cadena mínima: **señal → umbral → destinatario → acción**. Es el mínimo viable, y el mínimo viable es justamente lo que se puede tener funcionando en dos semanas.

Un modelo completo para producción añade al menos:

Pieza	Para qué
Prueba sintética desde fuera de la nube	Enterarse antes que el cliente. Es la señal S-01, y da disponibilidad y MTTD de una vez
Trazas distribuidas	Saber cuál de los servicios de una cadena es el lento
Registros centralizados y correlacionados	Bajar del síntoma al detalle sin entrar a cada máquina
Tablero ejecutivo	Un resumen que alguien de dirección mire sin traducción
Revisión periódica de alarmas	Que la matriz no se vuelva obsoleta en seis meses

Y para un equipo que hoy no mide nada, el orden que da resultados más rápido no empieza por ninguna herramienta nueva:

1. **Una prueba sintética desde fuera**, cada minuto, del recorrido más importante.
2. **Registrar los incidentes en una hoja de cálculo**: cuándo empezó, cuándo se detectó, cuándo se resolvió, qué lo causó. Cuatro columnas.
3. **Una revisión mensual de 30 minutos** con esa hoja delante.

En dos meses eso produce los números que hoy no existen — y sin esos números, cualquier discusión sobre compromisos de servicio es una discusión de opiniones.