LABORATORIOS DE ARQUITECTURA EN ORACLE CLOUD INFRASTRUCTURE

Manual de consola — Arquitectura elástica

Un balanceador público y un grupo de instancias que crece de 2 a 6 bajo carga y vuelve a bajar solo, construido a mano en la consola de OCI

Módulo 03

Primera pasada 2,5 a 3 horas la primera pasada, incluyendo esperas de provisión y una medición completa del ciclo

Costo aproximado menos de 3 USD al mes con el laboratorio encendido unas horas al día; 0,13 USD de cómputo medidos en el ensayo, con proyección mensual de 2,10 USD contra un presupuesto de 150 USD

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 construye el laboratorio completo **desde la consola de OCI**, haciendo clic. No usa Terraform ni la CLI para crear nada. Los pocos comandos que aparecen son verificaciones opcionales que complementan lo que se ve en pantalla.

El lector esperado es un ingeniero de infraestructura que conoce AWS o Azure y no conoce OCI. Cada pantalla que hay que tocar tiene su paso.

Todo lo que sigue se hace en un **compartimento del laboratorio**, desechable, sobre un tenancy de prueba. Nunca sobre un ambiente con datos.

1. Qué se construye y para qué

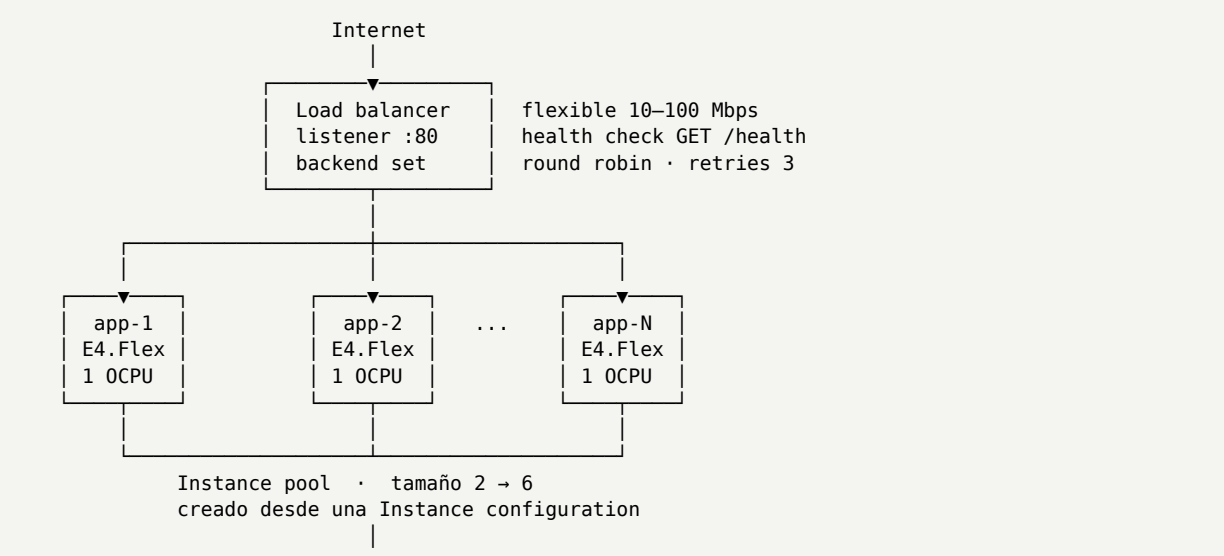
El laboratorio responde una pregunta concreta: **¿cuánto tarda realmente la plataforma en darme capacidad nueva cuando la carga sube, y cuánto en quitármela cuando baja?**

La respuesta no se estima: se mide. Y el número medido —casi doce minutos hasta que la capacidad nueva atiende tráfico— es lo que permite decidir si una arquitectura elástica sirve para el patrón de carga que usted tiene, o si su pico llega y se va antes de que la elasticidad alcance a reaccionar.

De esa medición salen dos decisiones que el laboratorio ayuda a tomar:

1. **Escalar horizontal contra escalar vertical.** Redimensionar una máquina virtual cuesta una ventana de indisponibilidad. Agregar instancias a un grupo, no. Pero agregar instancias tarda, y ese retraso tiene que caber dentro de su tolerancia.
2. **Cuánta capacidad base dejar encendida.** Si el ciclo de escalamiento tarda doce minutos, el mínimo del grupo no se define por el promedio de carga: se define por lo que debe poder absorber sin ayuda durante esos doce minutos.

Los componentes



```
              métrica CpuUtilization (oci_computeagent)
                                 │
                        ┌────────▼─────────┐
                        │   Autoscaling    │  CPU > 55 %  →  +2  (máx. 6)
                        │   configuration  │  CPU < 20 %  →  −1  (mín. 2)
                        └──────────────────┘  enfriamiento 300 s

   Red: VCN 10.30.0.0/16 · subred pública 10.30.1.0/24 · internet gateway
   FinOps: presupuesto mensual de 150 USD con avisos al 50 / 75 / 90 %
           y un aviso adicional por gasto proyectado
```

La aplicación que corre en cada instancia es un servidor HTTP escrito con la librería estándar de Python, sin dependencias externas. Expone tres rutas:

Ruta	Qué hace
`GET /`	Devuelve el nombre del host que atendió, sus CPUs y su tiempo de vida. Sirve para ver en vivo que detrás del balanceador hay servidores distintos.
`GET /health`	Responde `200 ok`. Es lo que consulta el health check del balanceador.
`GET /burn?ms=N`	Quema CPU durante N milisegundos. Es lo que hace subir la métrica y disparar el escalamiento.

No descarga nada de internet al arrancar. Eso es deliberado: hace predecible el tiempo que tarda una instancia nueva en empezar a servir tráfico, que es justamente uno de los tramos que el laboratorio mide.

Lo que este laboratorio simplifica a propósito

La capa de aplicación está en una **subred pública**, con IP pública en cada instancia. En una arquitectura real va en subred privada, detrás del balanceador, y el acceso administrativo pasa por un bastión. Aquí se simplifica para que el laboratorio quepa en el tiempo disponible y para que el diagnóstico sea directo cuando algo falla.

Dígalo en voz alta antes de que alguien lo pregunte. Un laboratorio que esconde sus simplificaciones pierde credibilidad cuando lo notan; uno que las declara, la gana.

2. Equivalencias con otras nubes

Solo los servicios que aparecen en este manual.

Concepto	AWS	Azure	OCI
Aislamiento administrativo y de costo	Cuenta / OU	Resource group / Subscription	**Compartment**
Red virtual	VPC	Virtual Network (VNet)	**VCN** (Virtual Cloud Network)
Segmento de red	Subnet	Subnet	**Subnet**

Salida a internet	Internet Gateway	(implícito con IP pública)	**Internet Gateway**
Tabla de rutas	Route table	Route table	**Route table**
Filtro de red por subred	Network ACL	NSG a nivel de subred	**Security list**
Filtro de red por interfaz	Security group	NSG a nivel de NIC	**Network Security Group (NSG)**
Balanceador de capa 7	Application Load Balancer	Application Gateway	**Load Balancer** (shape flexible)
Grupo de destinos	Target group	Backend pool	**Backend set**
Sonda de salud	Health check	Health probe	**Health check policy**
Plantilla de máquina	Launch template	VMSS model	**Instance configuration**
Grupo de máquinas idénticas	Auto Scaling group	Virtual Machine Scale Set	**Instance pool**
Política de escalamiento	Scaling policy (step / target tracking)	Autoscale settings	**Autoscaling configuration**
Script de arranque	User data	Custom data	**cloud-init** (initialization script)
Métricas de la instancia	CloudWatch agent	Azure Monitor agent	**Oracle Cloud Agent**, plugin de monitoreo
Métrica de CPU	`CPUUtilization`	`Percentage CPU`	`CpuUtilization` en el namespace `oci_computeagent`
Presupuesto con avisos	AWS Budgets	Cost Management budgets	**Budgets** con *alert rules*

Tres diferencias que sorprenden a quien viene de otra nube:

- **El compartimento no es una etiqueta, es una frontera.** Las políticas de acceso, los límites de servicio, el análisis de costo y el borrado masivo se apoyan en él. Crear el laboratorio en el compartimento raíz hace que todo eso sea más difícil después.
- **Los presupuestos viven en el compartimento raíz**, aunque apunten a un compartimento hijo. Si intenta crearlos parado en el compartimento del laboratorio, no los verá.
- **El grupo de instancias y la política de escalamiento son dos recursos distintos.** En AWS el Auto Scaling group lleva la política adentro. En OCI el *instance pool* solo mantiene un tamaño; quien cambia ese tamaño es una *autoscaling configuration* aparte, que se crea después y se puede desactivar sin tocar el grupo.

3. Prerrequisitos

3.1 Cuenta y permisos

Requisito	Detalle
Tenancy	Un tenancy de prueba o una cuenta de desarrollo. Nunca un ambiente productivo.
Usuario	Administrador del tenancy, o un usuario con políticas sobre `virtual-network-family`, `instance-family`, `load-balancers`, `instance-configurations`, `instance-pools`, `auto-scaling-configurations` y `usage-budgets` en el compartimento del laboratorio.
Presupuestos	El permiso de `usage-budgets` debe estar **en el tenancy**, no en el compartimento del laboratorio.
Región	Una región donde exista el shape `VM.Standard.E4.Flex` y el balanceador flexible. La medición de referencia se hizo en `us-chicago-1`.
Llave SSH	Un par de llaves. Necesitará el contenido de la **llave pública** para pegarlo en la configuración de instancia.

3.2 Límites de servicio — el paso que más gente se salta

Este es, con diferencia, el motivo más común por el que un laboratorio sobre una cuenta de prueba no llega a la fecha. Los aumentos de límite **no son inmediatos**: se piden con días de anticipación.

Consola de OCI Governance & Administration › Limits, Quotas and Usage

Límite a revisar	Valor necesario
Cores de `VM.Standard.E4.Flex`	Al menos `6` (6 instancias × 1 OCPU en el tamaño máximo del grupo)
Balanceadores flexibles	Al menos `1`
Block Volume (GB)	Al menos `300` (6 × 50 GB de volumen de arranque)
Instance pools y autoscaling	Habilitados en la región

1. Abra **Limits, Quotas and Usage**.
2. En **Service**, elija `Compute`. Filtre por el ámbito de la región y busque el contador de cores del shape E4. Confirme que el límite no es `0` y que el disponible cubre los 6 cores.
3. Cambie **Service** a `LBaaS` y confirme el cupo de balanceadores flexibles.
4. Cambie **Service** a `Block Volume` y confirme los GB disponibles.
5. Si algún límite aparece en `0`, use **Request a service limit increase** el mismo día. No continúe asumiendo que se aprobará a tiempo.

3.3 Compartimento del laboratorio

Consola de OCI Identity & Security › Compartments › Create Compartment

Campo	Valor
Name	`lab-01-elasticidad`

Description	`Laboratorio de arquitectura elastica`
Parent Compartment	El compartimento raíz del tenancy, o un compartimento padre de laboratorios
Tags	`Proyecto` = `laboratorio-elastico` · `Taller` = `01-elasticidad` · `Efimero` = `si`

1. Abra **Compartments** y pulse **Create Compartment**.
2. Llene **Name** y **Description** con los valores de la tabla.
3. En **Tag namespace**, deje *None* para usar etiquetas de forma libre y agregue las tres parejas clave/valor de la tabla.
4. Pulse **Create Compartment**.
5. Abra el compartimento recién creado y copie su **OCID**. Lo va a necesitar en el capítulo del presupuesto. Tendrá una forma parecida a `ocid1.compartment.oc1..aaaaEJEMPLO`.

Las etiquetas no son decoración: son lo que permite filtrar el costo del laboratorio en **Cost Analysis** sin separarlo a mano. Aplíquelas desde el principio, porque el análisis de costo necesita cerca de 24 horas de datos acumulados para mostrar algo útil.

3.4 Qué debe tener a mano antes de empezar

- El OCID del compartimento.
- El contenido de la llave pública SSH (el texto completo, no la ruta al archivo).
- Un correo electrónico para recibir las alertas de presupuesto.
- El archivo de configuración de arranque de la aplicación, que está transcrito completo en el capítulo 9 de este manual.

4. Presupuesto y alertas de costo

Este capítulo va **primero**, antes de crear un solo recurso. Por dos razones.

La primera es práctica: el crédito de una cuenta de prueba es finito y se consume solo. Un balanceador olvidado un fin de semana cuesta más que todo el laboratorio bien apagado.

La segunda es que el presupuesto **es material del laboratorio**. Es el control de FinOps más barato que existe: se activa en cinco minutos, no necesita proyecto ni herramienta adicional, y su alerta de proyección avisa antes de que el presupuesto se supere, no después. Lo que no hace es frenar el gasto; el alcance exacto está al final del capítulo.

Consola de OCI Billing & Cost Management › Budgets › Create Budget

Campo	Valor
Budget scope	Target compartment

Target compartment	`lab-01-elasticidad`
Name	`lab01-presupuesto-laboratorio`
Description	`Laboratorio de arquitectura elastica`
Monthly budget amount	`150`
Budget alert rule (la primera, en el mismo formulario)	Threshold metric `Actual` · Threshold type `Percentage` · Threshold `50`
Email recipients	Su correo
Email message	`Laboratorio de arquitectura elastica: consumo por encima del 50% del presupuesto.`

1. Cambie el selector de compartimento de la esquina izquierda al **compartimento raíz** del tenancy. Los presupuestos se crean ahí aunque vigilen un compartimento hijo. Si no lo hace, el botón de creación no le dejará avanzar o el presupuesto quedará donde no debe.
2. Abra **Budgets** y pulse **Create Budget**.
3. En el ámbito del presupuesto, elija la opción que apunta a un **compartimento** (la alternativa apunta a una etiqueta de seguimiento de costo, que sirve cuando el gasto está repartido en varios compartimentos).
4. Seleccione `lab-01-elasticidad` como compartimento vigilado.
5. Llene **Name** y **Description**.
6. En el monto mensual escriba `150`.
7. En la sección de la regla de alerta, elija **Threshold metric** = `Actual`, **Threshold type** = `Percentage` y **Threshold** = `50`.
8. Escriba su correo en los destinatarios y el mensaje de la tabla.
9. Pulse **Create**.

Faltan tres avisos más. Se agregan sobre el presupuesto ya creado.

Consola de OCI Billing & Cost Management › Budgets › `lab01-presupuesto-laboratorio` › Alert Rules › Create Alert Rule

Campo	Valor (aviso del 75 %)
Name	`lab01-alerta-75pct`
Threshold metric	`Actual`
Threshold type	`Percentage`
Threshold	`75`
Email recipients	Su correo

Email message	`Laboratorio de arquitectura elastica: consumo por encima del 75% del presupuesto.`

1. Abra el presupuesto y, en el menú de recursos de la izquierda, entre a las reglas de alerta.
2. Pulse **Create Alert Rule** y llene los campos de la tabla.
3. Repita el paso anterior para el aviso del **90 %**, con nombre `lab01-alerta-90pct` y umbral `90`.
4. Cree una cuarta regla, distinta de las anteriores: con nombre `lab01-alerta-proyeccion`, **Threshold metric** = `Forecast`, **Threshold type** = `Percentage` y **Threshold** = `100`.
5. Verifique que las cuatro reglas aparecen listadas y habilitadas.

Las tres primeras avisan de gasto **ya ocurrido**. La cuarta avisa de gasto **proyectado**: dispara cuando la tendencia del mes indica que va a superar el presupuesto, aunque todavía no lo haya superado. Es la única de las cuatro que llega a tiempo de servir para algo.

De los tres umbrales, el útil es el del 50 %: llega cuando todavía hay margen para corregir. El del 90 % ya solo sirve para apagar cosas.

Una advertencia sobre el alcance de este control: los presupuestos de OCI **avisan, no cortan**. Son un límite informativo —un *soft limit*, en los términos de la documentación—: nada se apaga solo cuando se cruza un umbral. Y el aviso tampoco es instantáneo: OCI **evalúa las reglas de alerta cada 24 horas**, así que un descuido de la tarde puede tardar un día en llegar al correo. Por eso el control real sigue siendo el hábito de apagar el laboratorio todos los días y la revisión periódica de **Cost Analysis**.

En el ensayo de referencia, el laboratorio completo registró **0,13 USD** de cómputo y el presupuesto proyectó **2,10 USD** para el mes contra un límite de 150. El orden de magnitud importa: lo que arruina el crédito de una cuenta de prueba no es este laboratorio, es olvidarlo encendido.

5. Red: VCN, internet gateway y tabla de rutas

A partir de aquí, **todo se crea dentro del compartimento `lab-01-elasticidad`**. Cambie el selector de compartimento en la parte izquierda de cada pantalla antes de crear cada recurso. Es el error de navegación más frecuente en OCI y no avisa: el recurso queda creado, pero en otro sitio.

La consola ofrece un asistente (**Start VCN Wizard**) que crea de un golpe la VCN, una subred pública, una privada, el gateway y las rutas. **No lo use aquí.** Este laboratorio crea cada pieza por separado, porque el objetivo es ver qué hay debajo y porque la topología que necesitamos no es la del asistente.

5.1 La VCN

Consola de OCI Networking › Virtual Cloud Networks › Create VCN

Campo	Valor
Name	`lab01-vcn`
Create in compartment	`lab-01-elasticidad`
IPv4 CIDR Blocks	`10.30.0.0/16`
DNS resolution	Habilitada
DNS label	`lab01`
Tags	`Proyecto` = `laboratorio-elastico` · `Taller` = `01-elasticidad` · `Efimero` = `si`

1. Abra **Virtual Cloud Networks** y confirme que el compartimento seleccionado a la izquierda es `lab-01-elasticidad`.
2. Pulse **Create VCN** (no el asistente).
3. Escriba el nombre `lab01-vcn`.
4. En el bloque CIDR escriba `10.30.0.0/16`.
5. Deje habilitada la resolución de nombres y escriba la etiqueta DNS `lab01`.
6. Agregue las tres etiquetas de la tabla.
7. Pulse **Create VCN**.

El rango `10.30.0.0/16` no tiene nada de mágico, pero elija uno que no choque con sus redes existentes si algún día va a conectarlas. Cambiar el CIDR de una VCN después implica recrearla.

5.2 El internet gateway

Consola de OCI Networking › Virtual Cloud Networks › `lab01-vcn` › Internet Gateways › Create Internet Gateway

Campo	Valor
Name	`lab01-igw`
Create in compartment	`lab-01-elasticidad`

1. Abra la VCN recién creada.
2. En el menú de recursos de la izquierda, entre a **Internet Gateways**.
3. Pulse **Create Internet Gateway**, escriba el nombre y confirme.

El gateway por sí solo no hace nada. Sin una ruta que lo apunte, la subred sigue sin salida. Ese es el siguiente paso, y es el que más gente olvida.

5.3 La tabla de rutas

Consola de OCI Networking › Virtual Cloud Networks › `lab01-vcn` › Route Tables › Create Route Table

Campo	Valor
Name	`lab01-rt-public`
Create in compartment	`lab-01-elasticidad`
Target Type	Internet Gateway
Destination CIDR Block	`0.0.0.0/0`
Target	`lab01-igw`

1. En el menú de recursos de la VCN, entre a **Route Tables**.
2. Pulse **Create Route Table** y escriba el nombre `lab01-rt-public`.
3. En la regla de ruta, elija **Target Type** = *Internet Gateway*.
4. Escriba `0.0.0.0/0` como destino y seleccione `lab01-igw` como objetivo.
5. Pulse **Create**.

No use la tabla de rutas por defecto que la VCN trae. Una tabla propia hace explícito qué sale a internet y permite cambiar la topología después sin tocar lo que la VCN creó sola.

6. Security list: las cuatro reglas

La security list es el filtro de red que se aplica **a toda la subred**. OCI también tiene Network Security Groups, que se aplican por interfaz de red y son el mecanismo que se recomienda en ambientes reales. Aquí usamos una security list porque el laboratorio tiene una sola subred y una sola clase de máquina, y porque así la regla que se abre queda a la vista de todos en una sola pantalla.

Consola de OCI Networking › Virtual Cloud Networks › `lab01-vcn` › Security Lists › Create Security List

Campo	Valor
Name	`lab01-sl-app`
Create in compartment	`lab-01-elasticidad`

Las reglas quedan así:

Dirección	Origen / destino	Protocolo	Puerto	Para qué
Egress	`0.0.0.0/0`	All Protocols	—	Salida de las instancias

Ingress	`0.0.0.0/0`	TCP	`80`	Tráfico HTTP público hacia el balanceador
Ingress	`0.0.0.0/0`	TCP	`22`	SSH administrativo — ver el recuadro
Ingress	`10.30.1.0/24`	All Protocols	—	Tráfico interno: balanceador hacia instancias

1. En el menú de recursos de la VCN, entre a **Security Lists**.
2. Pulse **Create Security List** y escriba el nombre `lab01-sl-app`.
3. En **Allow Rules for Egress**, agregue una regla con **Destination Type** = *CIDR*, **Destination CIDR** = `0.0.0.0/0` e **IP Protocol** = *All Protocols*.
4. En **Allow Rules for Ingress**, agregue la primera regla: **Source Type** = *CIDR*, **Source CIDR** = `0.0.0.0/0`, **IP Protocol** = *TCP*, **Destination Port Range** = `80`. Deje el rango de puertos de origen vacío.
5. Agregue la segunda regla de ingreso: **Source CIDR** = `0.0.0.0/0`, **IP Protocol** = *TCP*, **Destination Port Range** = `22`. Lea el recuadro antes de aceptar este valor.
6. Agregue la tercera regla de ingreso: **Source CIDR** = `10.30.1.0/24`, **IP Protocol** = *All Protocols*. Sin esta regla el balanceador no alcanza a las instancias y todos los backends quedan en estado crítico.
7. Deje sin marcar la casilla de regla sin estado (*stateless*) en las cuatro. Las reglas con estado permiten automáticamente el tráfico de respuesta, que es lo que se quiere aquí.
8. Pulse **Create Security List**.

CUIDADO · El puerto 22 queda abierto a `0.0.0.0/0` a propósito

En este laboratorio el acceso administrativo está abierto a todo internet **de forma deliberada**, porque es un ambiente desechable, sin datos, que se destruye el mismo día y cuyo único contenido es una aplicación de demostración de treinta líneas. La decisión se toma para que el laboratorio no dependa de desde qué red se conecte quien lo opera.

En un ambiente con datos eso no se hace. La práctica correcta es: SSH restringido al CIDR de su red administrativa en `/32` cuando es una sola IP, instancias sin IP pública en subred privada, y el acceso por un servicio de bastión que registra cada sesión. Si va a dejar este laboratorio encendido más de unas horas, cambie la regla del 22 a su IP y nada más: en una cuenta de prueba recién abierta, los escaneos empiezan en minutos.

7. La subred pública

La subred se crea al final de la red porque referencia las tres piezas anteriores: la tabla de rutas, la security list y, a través de ellas, el gateway.

Consola de OCI Networking › Virtual Cloud Networks › `lab01-vcn` › Subnets › Create Subnet

8. Balanceador flexible, backend set y health check

El balanceador se crea **antes** que el grupo de instancias, porque el grupo se va a enganchar a su backend set. Y el listener se crea antes de enganchar el grupo, porque un backend set sin listener no recibe tráfico y el health check da resultados confusos.

El orden es: balanceador con su backend set y su health check → listener → (más adelante) grupo de instancias enganchado.

Consola de OCI Networking › Load Balancers › Load Balancer › Create Load Balancer

8.1 Detalles y ancho de banda

Campo	Valor
Load balancer name	`lab01-lb`
Choose visibility type	Public
Assign a public IP address	Ephemeral IP address
Choose the bandwidth	Flexible shape
Minimum bandwidth	`10` Mbps
Maximum bandwidth	`100` Mbps
Virtual cloud network	`lab01-vcn`
Subnet	`lab01-subnet-public`

1. Abra **Load Balancers** y confirme el compartimento `lab-01-elasticidad`.
2. Pulse **Create Load Balancer**. Cuando la consola pregunte el tipo, elija el balanceador de capa 7 (*Load Balancer*), no el balanceador de red.
3. Escriba el nombre y elija visibilidad **Public**.
4. Deje la IP pública **efímera**. Una IP reservada solo hace falta si va a apuntar un nombre DNS estable, que no es el caso de un laboratorio que se destruye a diario.
5. En el ancho de banda, elija el shape **flexible** y escriba `10` de mínimo y `100` de máximo. El mínimo es capacidad garantizada y es lo que se factura de base; el máximo es el techo hasta donde puede crecer solo. Un balanceador flexible con mínimo alto cuesta más aunque no pase tráfico.
6. Seleccione la VCN `lab01-vcn` y la subred `lab01-subnet-public`.
7. Pase a la siguiente pantalla.

8.2 Backend set y política de salud

Campo	Valor

Load balancing policy	Weighted Round Robin
Backends	Ninguno (los pone el grupo de instancias)
Health check · Protocol	`HTTP`
Health check · Port	`80`
Health check · URL path (URI)	`/health`
Health check · Status code	`200`
Health check · Interval in ms	`10000`
Health check · Timeout in ms	`3000`
Health check · Number of retries	`3`

1. En la política de balanceo, elija la de round robin ponderado. Con todos los backends del mismo peso equivale al round robin simple, que es lo que este laboratorio quiere: las peticiones se reparten en orden entre todas las instancias, incluidas las nuevas.
2. **No agregue backends.** El grupo de instancias los va a registrar y a dar de baja solo. Un backend agregado a mano aquí sobrevive al escalamiento y queda apuntando a una instancia que ya no existe.
3. En la política de health check, elija protocolo `HTTP` y puerto `80`.
4. Escriba `/health` como ruta y `200` como código esperado.
5. Escriba `10000` en el intervalo, `3000` en el tiempo de espera y `3` en los reintentos.
6. Pase a la siguiente pantalla.

El intervalo de 10 segundos es corto a propósito: hace que un backend nuevo aparezca rápido en pantalla. En producción se usan valores más conservadores, porque un health check agresivo multiplica las peticiones contra toda la flota y puede sacar de servicio un backend sano que solo estaba lento.

Estos tres números tienen consecuencia directa en el tiempo del ciclo: un backend nuevo necesita **tres respuestas correctas separadas 10 segundos** antes de recibir tráfico. Son 30 segundos como mínimo teórico, y forman parte de los casi doce minutos del capítulo 12.

8.3 Listener

Campo	Valor
Listener name	`lab01-listener-http`
Specify the type of traffic your listener handles	`HTTP`
Specify the port your listener monitors	`80`

Backend set	`lab01-bset`

1. Escriba el nombre del listener.
2. Elija tráfico `HTTP` y puerto `80`. Sin certificado: este laboratorio no hace TLS, y agregarlo no cambia nada de lo que se quiere medir.
3. Confirme que el backend set asociado es el que acaba de definir.
4. En la pantalla de registro (*logging*), deje los registros de acceso y de error como vengan. No hacen falta para el laboratorio y generan volumen.
5. Pulse **Submit** y espere. **El balanceador es lo más lento de crear de todo el laboratorio**: entre 5 y 8 minutos hasta que queda activo.
6. Cuando el estado pase a activo, anote la **IP pública** que aparece en la página de detalle. Es la dirección que va a usar para todo lo demás.

Si el backend set no quedó con el nombre `lab01-bset`, córrijalo ahora o anote el nombre real: lo va a necesitar al enganchar el grupo de instancias.

En este punto el balanceador está vivo y **sin backends**. Si abre su IP en un navegador, responderá con un error del propio balanceador. Es lo esperado: todavía no hay a quién mandarle el tráfico.

9. Instance configuration: la plantilla de la máquina

La *instance configuration* es la plantilla que el grupo usa para crear cada instancia nueva. Es el equivalente de un launch template de AWS o del modelo de un scale set de Azure.

Dos detalles de esta pantalla deciden si el laboratorio funciona o no: el **script de inicialización** y el **plugin de monitoreo**. Los dos están explicados abajo, y los dos son fallos medidos, no advertencias teóricas.

Consola de OCI Compute › Instance Configurations › Create instance configuration

Campo	Valor
Name	`lab01-instance-config`
Create in compartment	`lab-01-elasticidad`
Placement · Availability domain	El primero de la región
Image	Oracle Linux 9, la imagen más reciente
Shape	`VM.Standard.E4.Flex` · `1` OCPU · `8` GB de memoria
Boot volume size	`50` GB

Virtual cloud network	`lab01-vcn`
Subnet	`lab01-subnet-public`
Assign a public IPv4 address	Sí
SSH keys	Pegar la llave pública
Initialization script	El archivo `cloud-init.yaml` transcrito abajo
Oracle Cloud Agent · plugin de monitoreo de la instancia	Habilitado

1. Abra **Instance Configurations** en el menú de Compute y confirme el compartimento.
2. Pulse **Create instance configuration**. El formulario que abre es prácticamente el mismo de crear una instancia, pero **no crea nada**: solo guarda la plantilla.
3. Escriba el nombre `lab01-instance-config`.
4. En la sección de ubicación, deje el dominio de disponibilidad que la consola propone. Si la región tiene varios y quiere repartir el grupo, eso se configura después, en el grupo de instancias, no aquí.
5. En la imagen, pulse el botón de cambiar imagen y elija **Oracle Linux**, versión `9`, la compilación más reciente que ofrezca la lista.
6. En el shape, elija la familia **Virtual machine**, serie **AMD**, y el shape `VM.Standard.E4.Flex`. Ponga `1` OCPU y `8` GB de memoria.
7. En la red, seleccione la VCN `lab01-vcn` y la subred `lab01-subnet-public`, y marque la asignación de **dirección IPv4 pública**.
8. En el volumen de arranque, especifique `50` GB.
9. Pegue el contenido completo de su llave pública SSH en el campo de llaves.

9.1 El script de inicialización

1. Abra las opciones avanzadas del formulario y ubique la pestaña de administración, donde está el **initialization script**.
2. Elija pegar el contenido (la alternativa es subir un archivo; las dos sirven) y pegue **literalmente** el bloque completo que sigue, empezando por `#cloud-config`.

```
#cloud-config
# Aplicacion minima de demostracion del laboratorio.
#
#   GET /              -> nombre del host que atendio, hora y CPUs
#   GET /health        -> 200 OK. Lo consulta el health checker del balanceador.
#   GET /burn?ms=250   -> quema CPU durante ms milisegundos. Es lo que hace subir
#                         la metrica y disparar el autoescalamiento.
#
# Deliberadamente sin dependencias externas: solo la libreria estandar de Python.
```

```
# Una instancia nueva del grupo queda sirviendo trafico sin descargar nada de
# internet, que es justo lo que hace predecible el tiempo de arranque.

write_files:
  - path: /opt/lab/app.py
    permissions: "0755"
    content: |
      import os
      import socket
      import time
      from http.server import BaseHTTPRequestHandler, HTTPServer
      from socketserver import ForkingMixIn
      from urllib.parse import urlparse, parse_qs

      HOSTNAME = socket.gethostname()
      CPUS = os.cpu_count()
      STARTED = time.time()

      def burn(ms):
          end = time.perf_counter() + (ms / 1000.0)
          x = 0
          while time.perf_counter() < end:
              x += 1
          return x

      class Handler(BaseHTTPRequestHandler):
          protocol_version = "HTTP/1.1"

          def _send(self, code, body, ctype="text/plain; charset=utf-8"):
              data = body.encode("utf-8")
              self.send_response(code)
              self.send_header("Content-Type", ctype)
              self.send_header("Content-Length", str(len(data)))
              self.end_headers()
              self.wfile.write(data)

          def do_GET(self):
              parsed = urlparse(self.path)

              if parsed.path == "/health":
                  self._send(200, "ok")
                  return

              if parsed.path == "/burn":
                  qs = parse_qs(parsed.query)
                  ms = int(qs.get("ms", ["250"])[0])
                  ms = max(1, min(ms, 5000))
                  burn(ms)
                  self._send(200, "burned {}ms on {}".format(ms, HOSTNAME))
                  return

              uptime = int(time.time() - STARTED)
              self._send(200,
                  "Laboratorio de arquitectura elastica\n"
                  "Atendido por : {}\n"
                  "OCPUs        : {}\n"
                  "Uptime       : {}s\n"
                  "Hora         : {}\n".format(
                      HOSTNAME, CPUS, uptime,
                      time.strftime("%Y-%m-%d %H:%M:%S")))
```

```
          def log_message(self, *args):
              pass  # sin ruido en el journal

      # Servidor que BIFURCA procesos, no hilos. Ver el recuadro del manual.
      class ServidorConProcesos(ForkingMixIn, HTTPServer):
          max_children = 64
          daemon_threads = True

      if __name__ == "__main__":
          ServidorConProcesos(("0.0.0.0", 80), Handler).serve_forever()

  - path: /etc/systemd/system/lab-app.service
    permissions: "0644"
    content: |
      [Unit]
      Description=Aplicacion de demostracion del laboratorio
      After=network-online.target
      Wants=network-online.target

      [Service]
      ExecStart=/usr/bin/python3 /opt/lab/app.py
      Restart=always
      RestartSec=2

      [Install]
      WantedBy=multi-user.target

runcmd:
  # Oracle Linux trae firewalld activo y bloquea el 80. Sin esto el health check
  # nunca pasa y el backend queda permanentemente en CRITICAL.
  - firewall-cmd --permanent --add-port=80/tcp
  - firewall-cmd --reload
  - systemctl daemon-reload
  - systemctl enable --now lab-app.service
```

Tres cosas de este archivo merecen atención, y las tres son fallos que costaron tiempo antes de quedar documentados.

IMPORTANTE · El servidor bifurca procesos, no hilos — y esa es la diferencia entre 48 % y 76 % de CPU

La clase `ServidorConProcesos` hereda de `ForkingMixIn`, que crea un **proceso** por petición. La versión obvia, y la primera que se escribió, usaba `ThreadingMixIn`, que crea un **hilo**.

Con hilos el laboratorio no funciona. El bloqueo global del intérprete de Python (el *GIL*) serializa el bucle que quema CPU: por mucha concurrencia que llegue, solo se ocupa un núcleo. En una instancia de 1 OCPU —que son 2 vCPU— eso pone el techo exactamente en la mitad. **Medido: la CPU nunca pasó del 48 %**, el umbral de escalamiento está en 55 %, y el grupo no creció nunca por mucha carga que se le echara.

Cambiado a procesos, la misma carga llevó la CPU a **76 %** y de ahí hasta cerca del 99 %. Este es el fallo número 1 del ensayo, y es el que habría convertido una demostración en vivo en diez minutos de mirar una gráfica plana.

La lección general no es sobre Python: es que **la métrica que dispara el escalamiento tiene**

que poder llegar al umbral. Antes de confiar en una política de autoescalamiento, verifique que la carga que usted puede generar mueve de verdad la métrica que eligió. Si no la mueve, la política es decorativa.

La segunda: la línea de `firewall-cmd`. Oracle Linux trae `firewalld` activo y el puerto 80 cerrado. Sin abrirlo, la instancia arranca bien, el servicio corre bien, y el health check nunca pasa: el backend queda permanentemente en estado crítico y el balanceador nunca le manda tráfico. Es el síntoma más común y el más confuso, porque todo *parece* estar bien.

La tercera: la aplicación se instala como servicio de `systemd` con `Restart=always`. Si se cae, vuelve sola. Eso importa cuando una instancia del grupo tiene un mal momento bajo carga: sin reinicio automático, el health check la sacaría y el grupo escalaría para compensar algo que era un proceso caído.

9.2 El plugin de monitoreo

1. En las opciones avanzadas, abra la pestaña del **Oracle Cloud Agent**.
2. Confirme que el plugin de **monitoreo de la instancia de cómputo** está **habilitado**. En las imágenes de plataforma suele venir activo por defecto, pero confírmelo con los ojos: es gratis confirmarlo y caro descubrirlo después.
3. Pulse **Create** para guardar la configuración de instancia.

IMPORTANTE · Sin el plugin de monitoreo no hay métrica, y sin métrica el grupo nunca crece

El autoescalamiento por CPU no lee la CPU de la máquina: lee la métrica `CpuUtilization` del namespace `oci_computeagent`, que **publica el agente que corre dentro de la instancia**. Si el plugin está deshabilitado, no hay métrica, la política de escalamiento no tiene contra qué comparar y no dispara nunca. Sin error, sin aviso, sin nada en pantalla: simplemente no pasa nada.

Además, una instancia recién arrancada tarda varios minutos en publicar su primera métrica. Espere entre 5 y 10 minutos antes de concluir que algo está mal.

10. Instance pool: el grupo de instancias

El grupo toma la plantilla del capítulo anterior, crea N instancias idénticas y las mantiene. Si una se cae o se borra, la repone. Y —esta es la parte que importa— se engancha al backend set del balanceador, de modo que cada instancia que entra o sale del grupo se registra o se da de baja en el balanceador **sola**.

Consola de OCI Compute › Instance Pools › Create instance pool

Campo	Valor
Name	`lab01-pool`
Create in compartment	`lab-01-elasticidad`
Instance configuration	`lab01-instance-config`
Number of instances	`2`
Availability domain	El mismo de la configuración de instancia
Primary VNIC · Virtual cloud network	`lab01-vcn`
Primary VNIC · Subnet	`lab01-subnet-public`
Attach load balancer	Sí
Load balancer	`lab01-lb`
Backend set	`lab01-bset`
Port	`80`
VNIC	Primary VNIC

1. Abra **Instance Pools** y confirme el compartimento.
2. Pulse **Create instance pool** y escriba el nombre `lab01-pool`.
3. Seleccione `lab01-instance-config` como configuración de instancia. Si no aparece, revise que está mirando el compartimento correcto.
4. Pase a la pantalla de ubicación del grupo.
5. Escriba `2` en el número de instancias. Este es el tamaño con el que arranca; a partir de que exista la política de autoescalamiento, el tamaño lo gobierna ella.
6. Elija el dominio de disponibilidad y, como subred de la VNIC principal, `lab01-subnet-public`.
7. Active la opción de **enganchar un balanceador**.
8. Seleccione el balanceador `lab01-lb`, el backend set `lab01-bset`, el puerto `80` y la VNIC principal como origen de la dirección a registrar.
9. Agregue las etiquetas del laboratorio.
10. Revise y pulse **Create**.

El grupo pasa a estado de aprovisionamiento y crea las dos instancias. Tardan unos minutos en arrancar, ejecutar el script de inicialización y empezar a responder.

TIP · El enganche al balanceador es en un solo sentido: no toque los backends a mano

Con el grupo enganchado al backend set, OCI agrega y quita backends por usted. **No agregue,**

edite ni borre backends desde la pantalla del balanceador. Un backend puesto a mano no desaparece cuando el grupo reduce su tamaño: queda apuntando a una instancia que ya no existe, el health check lo marca crítico para siempre y la salud global del backend set nunca vuelve a verse limpia.

Si necesita sacar una instancia de rotación por un rato, el camino es cambiar el tamaño del grupo o desactivar la política de autoescalamiento, nunca editar backends.

10.1 Verificar que el grupo quedó sirviendo

1. Abra la página de detalle del grupo y confirme que aparece en estado de ejecución con dos instancias.
2. Abra el balanceador y entre a su backend set. Debe ver **dos backends**, y al cabo de unos minutos los dos en estado correcto. Recuerde que el health check necesita tres respuestas buenas separadas 10 segundos.
3. Abra `http://<IP-del-balanceador>/` en el navegador. Debe ver el nombre del host que atendió.
4. Recargue varias veces. **El nombre del host debe alternar** entre las dos instancias. Si siempre responde el mismo, uno de los dos backends no está sano.

Verificación equivalente por línea de comandos, si la prefiere:

```
LB=<IP-del-balanceador>

curl -s "http://$LB/health"      # -> ok
curl -s "http://$LB/"            # -> muestra el host que atendio
curl -s "http://$LB/" ; curl -s "http://$LB/"   # el hostname debe alternar
```

Si los backends no se ponen sanos, no siga al capítulo siguiente. Un grupo que no sirve tráfico tampoco va a escalar de forma útil, y estará depurando dos cosas a la vez. El capítulo 15 tiene la tabla de diagnóstico.

11. Autoescalamiento: las dos mitades

Aquí es donde el grupo deja de ser un tamaño fijo. La *autoscaling configuration* es un recurso aparte que se engancha al grupo y le cambia el tamaño según una métrica.

La política tiene **dos reglas**, y las dos son obligatorias para que el laboratorio tenga sentido: una que agrega capacidad cuando la CPU sube, y otra que la quita cuando baja.

Consola de OCI Compute › Autoscaling Configurations › Create autoscaling configuration

11.1 Detalles y grupo asociado

Campo	Valor
Name	`lab01-autoscaling`
Create in compartment	`lab-01-elasticidad`
Resource type	Instance pool
Instance pool	`lab01-pool`

1. Abra **Autoscaling Configurations** y confirme el compartimento.
2. Pulse **Create autoscaling configuration** y escriba el nombre.
3. Seleccione el grupo `lab01-pool` como recurso a escalar.
4. Pase a la pantalla de la política.

11.2 La política

Campo	Valor
Autoscaling policy type	Metric-based autoscaling
Policy name	`lab01-policy-cpu`
Cool down in seconds	`300`
Performance metric	CPU utilization
Scale-out rule · operador	Greater than (>)
Scale-out rule · umbral	`55` %
Scale-out rule · instancias a agregar	`2`
Scale-in rule · operador	Less than (<)
Scale-in rule · umbral	`20` %
Scale-in rule · instancias a quitar	`1`
Minimum number of instances	`2`
Maximum number of instances	`6`
Initial number of instances	`2`

1. Elija el tipo de política **basada en métrica**. La alternativa es basada en horario, que sirve para cargas predecibles por calendario y no es lo que se mide aquí.
2. Escriba el nombre de la política.
3. En el enfriamiento, escriba `300` segundos. **Es el mínimo que la consola acepta**, y también el mínimo práctico: por debajo, el grupo oscila —sube, baja, vuelve a subir— y ese

es exactamente el antipatrón que este laboratorio existe para no mostrar.

4. Elija **CPU utilization** como métrica de rendimiento.
5. En la regla de escalamiento hacia afuera, ponga operador *mayor que*, umbral `55` y `2` instancias a agregar.
6. En la regla de escalamiento hacia adentro, ponga operador *menor que*, umbral `20` y `1` instancia a quitar.
7. Escriba `2` como mínimo, `6` como máximo y `2` como número inicial de instancias. El número inicial debe estar entre el mínimo y el máximo, y conviene que coincida con el tamaño con que creó el grupo.
8. Deje la configuración **habilitada**.
9. Revise y pulse **Create**.

11.3 Por qué esos números

Umbral de salida en 55 %. Tiene que estar por debajo del techo real que su carga puede producir. Con el servidor de hilos el techo medido era 48 % y el umbral de 55 % nunca se cruzaba; con procesos, la carga llega a 76 % y lo cruza con margen. El umbral se elige mirando la métrica bajo carga, no en abstracto.

Paso de salida de +2. Agregar de dos en dos hace el cambio visible y llega antes al tamaño donde la carga se reparte. Agregar de a uno, con un enfriamiento de 300 segundos entre acciones, multiplica el tiempo hasta tener capacidad suficiente.

Umbral de entrada en 20 %. Lejos del de salida, a propósito. Umbrales cercanos producen oscilación: el grupo crece, la CPU baja al repartirse, cruza el umbral de bajada, el grupo encoge, la CPU vuelve a subir. Esa banda muerta entre 20 % y 55 % es lo que mantiene el sistema quieto.

Paso de entrada de −1. Más pequeño que el de salida, también a propósito. Se baja con más cautela de la que se sube, para no recortar capacidad justo antes de un pico nuevo. Equivocarse escalando hacia arriba cuesta unos centavos; equivocarse escalando hacia abajo cuesta una caída de servicio.

11.4 El scale-in es la mitad que casi nadie configura

Casi todas las configuraciones de autoescalamiento que uno encuentra en la práctica tienen solo la regla de subida. Es entendible: la regla de subida es la que protege el servicio, y es la que alguien escribió el día que el sitio se cayó.

Pero un grupo que sube y no baja es un grupo que **queda permanentemente en su tamaño máximo** después del primer pico. Toda la promesa de la elasticidad —pagar por lo que se usa— depende de la regla que casi nadie escribe.

En este laboratorio, la bajada se puede ver porque el generador de carga golpea a través del

balanceador. Como el balanceador reparte en round robin, la misma carga total se divide entre más instancias a medida que el grupo crece: más instancias repartiéndose el mismo trabajo significa menos CPU por instancia. El grupo sube porque la CPU sube, y después la propia subida hace bajar la CPU. Ese es el ciclo completo, no la mitad.

Medido tras cortar la carga: la CPU cayó a **0,6 %** y el primer paso de bajada, de 6 a 5 instancias, ocurrió **251 segundos después** (4 minutos y 11 segundos). Los pasos siguientes van uno a uno, con 300 segundos de enfriamiento entre cada uno, hasta llegar al mínimo de 2. Bajar de 6 a 2 toma varios ciclos y es notablemente más lento que subir: es la consecuencia directa de haber elegido un paso de bajada de 1 y uno de subida de 2.

12. El ciclo completo, medido

Este capítulo no crea nada. Es el resultado del laboratorio: la medición real del ciclo, y de dónde sale cada tramo.

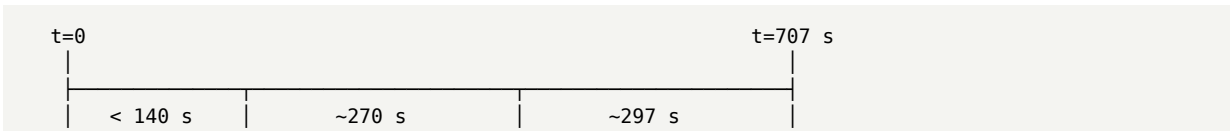
12.1 Los números

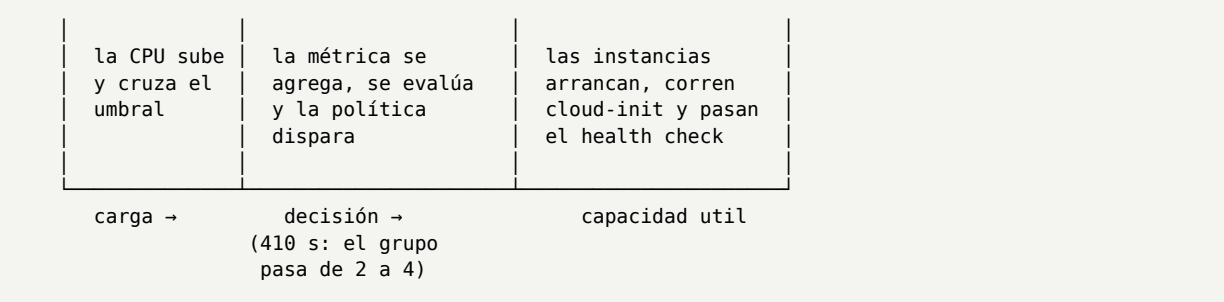
Medición sobre esta misma configuración —`VM.Standard.E4.Flex` de 1 OCPU, grupo 2–6, subida con CPU > 55 % en pasos de +2, bajada con CPU < 20 % en pasos de −1, enfriamiento de 300 s — con una carga de **40 procesos concurrentes pidiendo 1500 ms de CPU cada uno**, generada desde fuera de la región.

Evento	Desde el inicio de la carga
Inicio de la carga	`0`
La CPU cruza el umbral (llega a **76 %**)	menos de `140 s`
El grupo crece de 2 a 4	**`410 s`** (6 min 50 s)
Los backends nuevos sirven tráfico	**`707 s`** (11 min 47 s)
El grupo llega a su máximo de 6	antes de cortar la carga
CPU tras cortar la carga	`0,6 %`
Primer paso de bajada, de 6 a 5	**`251 s` después de cortar** (4 min 11 s)

El número que hay que decir en voz alta es 11 minutos 47 segundos. Casi doce minutos desde que la carga empieza hasta que la capacidad nueva atiende peticiones.

12.2 De dónde sale cada tramo





Tramo 1 — menos de 140 segundos: la carga tiene que llegar a la métrica. La CPU de la máquina sube casi de inmediato, pero lo que importa no es la CPU: es la métrica. El agente de la instancia publica `CpuUtilization` en intervalos de un minuto, de modo que el primer punto que refleja la carga aparece hasta un minuto después de que la carga empiece. Medido: menos de 140 segundos hasta ver la métrica por encima del umbral en **Metrics Explorer**.

Tramo 2 — del cruce del umbral a los 410 segundos: el servicio decide. Aquí ocurren tres cosas. La métrica se agrega a nivel del grupo, no de una instancia. El servicio de autoescalamiento evalúa esa serie contra el umbral y no actúa con un solo punto: exige que la condición se sostenga, que es lo que evita que un pico de tres segundos dispare una acción. Y, si hubo una acción de escalamiento reciente, el enfriamiento de 300 segundos bloquea la siguiente. En la medición, el grupo cambió de 2 a 4 instancias a los **410 segundos**.

Tramo 3 — de los 410 a los 707 segundos: la capacidad nueva tiene que estar lista. Este tramo, de casi cinco minutos, es el que más sorprende y el que menos se puede recortar. Contiene, en orden: el aprovisionamiento de las dos instancias nuevas, el arranque de Oracle Linux 9, la ejecución del script de inicialización —escribir la aplicación, abrir el puerto 80 en `firewalld`, habilitar y arrancar el servicio—, el registro de cada instancia como backend del balanceador, y **tres health checks correctos separados 10 segundos** antes de que el backend reciba tráfico. Solo ese último paso son 30 segundos de mínimo teórico. El resto es arranque de sistema operativo.

Los tramos internos no se cronometraron por separado; el reparto entre arranque y health check es una lectura de la configuración, no una medición. **[VALIDAR]** si va a citar el desglose fino del tercer tramo; el total de 707 s sí está medido.

La bajada, 251 segundos: llama la atención que sea *menor* que el enfriamiento de 300 segundos. La explicación es que el reloj del enfriamiento corre desde la **última acción de escalamiento**, no desde el momento en que la carga se detiene. Cuando la carga se cortó, el grupo llevaba ya un rato en 6 instancias sin cambios, así que el enfriamiento estaba cumplido y lo único que faltaba era que la métrica bajara y se evaluara.

12.3 Qué se hace con este número

El criterio que este laboratorio propone es sencillo: **si el ciclo completo cabe en menos de ocho minutos, una demostración en vivo funciona; si no, hay que arrancar la carga**

antes de empezar. Con 11 minutos 47 segundos, la respuesta es la segunda. La carga se arranca unos quince minutos antes y lo que se muestra es el ciclo ya en marcha.

Trasladado a una decisión de arquitectura, el mismo número dice esto: **la capacidad base del grupo no se define por la carga promedio, se define por lo que hay que poder absorber sin ayuda durante doce minutos.** Si su pico de tráfico llega y se va en cinco minutos, el autoescalamiento reactivo no lo va a atender: llegará cuando el pico ya pasó, y le cobrará las instancias nuevas mientras la carga baja. Para ese patrón las alternativas son otras: capacidad base mayor, escalamiento por horario si el pico es predecible, o una arquitectura donde el pico se absorba en una cola en vez de en instancias.

Esa conversación es el producto real del laboratorio. La demostración solo es la prueba de que los números son verdad.

12.4 Cómo reproducir la medición

Necesita tres cosas a la vez: una fuente de carga, una vista del grupo y del balanceador, y un reloj.

1. Abra **Metrics Explorer** en otra pestaña, con namespace `oci_computeagent` y métrica `CpuUtilization`, agrupada por el identificador del recurso. Esta es la métrica que la política está leyendo: si aquí no ve nada, la política tampoco.
2. Abra la página del grupo de instancias y la del backend set del balanceador.
3. Genere carga contra `http://<IP-del-balanceador>/burn?ms=1500` desde varias conexiones simultáneas. Cuarenta concurrentes es lo que produjo los números de arriba.
4. Anote la hora de inicio. Anote la hora de cada cambio: cruce del umbral, cambio de tamaño del grupo, instancia nueva en ejecución, backend nuevo sano.
5. Corte la carga y anote la hora. Anote la hora del primer paso de bajada.
6. **Repita la medición dos veces.** La primera pasada incluye el arranque en frío de la métrica y no es representativa.

Un detalle que costó una medición perdida: **al cortar el generador de carga, asegúrese de que se cortó de verdad.** En el ensayo, los procesos hijos del generador quedaron huérfanos y siguieron golpeando el balanceador después de un corte aparentemente limpio. La CPU se mantuvo al 80 %, el paso de bajada no llegó nunca, y el tiempo se fue en buscar un problema de autoescalamiento que no existía. Verifique en **Metrics Explorer** que la CPU efectivamente cayó antes de empezar a contar.

13. Cambiar la aplicación sin bloquear el borrado

Tarde o temprano va a querer cambiar el script de inicialización: corregir la aplicación, cambiar el

shape, agregar un paquete. Y aquí hay una trampa que en la consola se resuelve con un orden de operaciones concreto.

Una instance configuration no se puede editar. Al abrirla verá que no hay botón de edición de sus parámetros: la plantilla es inmutable. Cambiarla significa crear una nueva.

Y una instance configuration no se puede borrar mientras un grupo la referencia. El borrado falla.

Puestas las dos restricciones juntas, el orden equivocado deja el laboratorio trabado: si intenta borrar la configuración vieja para poner la nueva, el borrado falla porque el grupo la está usando; y si intenta cambiar el grupo sin tener la nueva creada, no hay a qué apuntarlo.

IMPORTANTE · Cree la nueva, reapunte el grupo, y solo entonces borre la vieja

Este es el mismo problema que en Terraform se resuelve con `create_before_destroy` en la configuración de instancia, y fue el fallo número 6 del ensayo: sin esa marca, Terraform intentaba borrar primero la configuración que el grupo referencia y el `apply` se quedaba trabado.

En la consola no hay una marca que lo haga por usted. **Lo hace usted, en este orden:**

1. Crear una **configuración de instancia nueva**, con el cambio, y un nombre distinto. 2. **Editar el grupo de instancias** para que apunte a la nueva configuración. 3. Reemplazar las instancias existentes para que tomen el cambio. 4. **Solo entonces**, borrar la configuración vieja.

Invertir el orden no rompe nada de forma permanente, pero le va a costar el rato de entender por qué un recurso que "ya no usa nadie" se niega a borrarse.

El procedimiento completo, paso a paso:

Consola de OCI Compute › Instance Configurations › Create instance configuration

Campo	Valor
Name	`lab01-instance-config-v2`
El resto	Idéntico al capítulo 9, con el cambio que quiere introducir

1. Cree la configuración nueva repitiendo el capítulo 9 completo, con el nombre `lab01-instance-config-v2` y el cambio que necesita. Sí, hay que volver a llenar todo el formulario: no hay clonado del formulario en esta pantalla. Si va a iterar mucho sobre el script de inicialización, guárdelo en un archivo aparte y súbalo en vez de pegarlo.

Consola de OCI Compute › Instance Pools › `lab01-pool` › Edit

Campo	Valor

Instance configuration	`lab01-instance-config-v2`

1. Abra el grupo `lab01-pool` y pulse **Edit**.
2. Cambie la configuración de instancia a `lab01-instance-config-v2` y guarde.
3. Entienda qué acaba de pasar: **las instancias que ya existen no cambian.** El grupo usará la configuración nueva para las instancias que cree de ahora en adelante. Las viejas siguen corriendo la aplicación vieja.
4. Para que el cambio llegue a todas, hay dos caminos. El brusco: reducir el tamaño del grupo hasta el mínimo y volver a subirlo, de modo que las instancias nuevas nazcan con la configuración nueva. El ordenado: terminar las instancias una por una desde la página del grupo, dejando que el grupo las reponga, y esperar entre cada una a que el backend nuevo esté sano antes de tocar la siguiente. En un laboratorio, el brusco basta y es más rápido; en algo con tráfico real, el ordenado es el único aceptable.
5. Verifique que el grupo volvió a su tamaño y que todos los backends están sanos.

Consola de OCI Compute › Instance Configurations › `lab01-instance-config` › Delete

1. Solo ahora, borre la configuración vieja. Si el borrado falla con un mensaje de recurso en uso, es que algún grupo todavía la referencia: vuelva al paso 2 y confirme que el cambio quedó guardado.

Una nota sobre el tamaño del grupo y el autoescalamiento: mientras la política de autoescalamiento esté habilitada, cualquier tamaño que usted fije a mano será corregido por ella en la siguiente evaluación. Si necesita que el grupo se quede quieto durante una intervención, **deshabilite primero la configuración de autoescalamiento** y vuelva a habilitarla al terminar. Es una casilla en la página de la configuración, y es reversible.

14. Validación de extremo a extremo

Recorra esta lista antes de dar el laboratorio por bueno. En orden: cada punto asume que el anterior pasó.

14.1 La red

Consola de OCI Networking › Virtual Cloud Networks › `lab01-vcn`

1. La VCN existe, con CIDR `10.30.0.0/16`.
2. La subred `lab01-subnet-public` existe, es pública y usa la tabla de rutas `lab01-rt-public` y la security list `lab01-sl-app`.

3. La tabla de rutas tiene una regla `0.0.0.0/0` hacia el internet gateway.
4. La security list tiene las cuatro reglas del capítulo 6, incluida la de `10.30.1.0/24`.

14.2 El balanceador y el grupo

Consola de OCI Networking › Load Balancers › `lab01-lb`

1. El balanceador está activo y tiene una IP pública.
2. El backend set `lab01-bset` tiene tantos backends como instancias haya en el grupo.
3. **Todos** los backends están en estado correcto. Uno solo en estado crítico indica que esa instancia no está sirviendo, y el grupo va a escalar para compensar algo que es un problema de arranque.
4. Abra `http://<IP-del-balanceador>/` y recargue cinco veces. El nombre del host debe alternar entre todas las instancias vivas.

Equivalente por línea de comandos:

```
# Salud del backend set. Ojo: el subcomando es backend-set-health get.
# «oci lb backend-health list» no existe, aunque suene plausible.
oci lb backend-set-health get \
  --load-balancer-id "<ocid-del-balanceador>" \
  --backend-set-name "lab01-bset" \
  --query 'data.{estado:status,total:"total-backend-count",criticos:"critical-state-backend-
names"}'

# Tamaño actual del grupo
oci compute-management instance-pool get \
  --instance-pool-id "<ocid-del-grupo>" --query 'data.size'
```

14.3 La métrica

Consola de OCI Observability & Management › Monitoring › Metrics Explorer

Campo	Valor
Compartment	`lab-01-elasticidad`
Metric namespace	`oci_computeagent`
Metric name	`CpuUtilization`
Interval	1 minuto
Statistic	Mean
Agrupar por	`resourceId`

1. Abra **Metrics Explorer** y arme la consulta con los valores de la tabla.
2. Debe ver una serie por cada instancia del grupo. Si no ve ninguna, el plugin de monitoreo está deshabilitado o las instancias todavía no publicaron su primer punto. Espere entre 5 y 10 minutos antes de concluir que hay un problema.
3. Con carga aplicada, las series deben cruzar el 55 %. Si se quedan pegadas cerca del 50 %, revise el recuadro del capítulo 9 sobre procesos e hilos: es exactamente ese síntoma.

14.4 El ciclo

VALIDACIÓN · Cómo saber que el laboratorio quedó bien

El laboratorio está bien cuando esta secuencia completa ocurre sin que usted toque nada:

1. Aplica carga contra el balanceador. 2. La métrica `CpuUtilization` cruza el 55 % en **Metrics Explorer**, y llega a un valor del orden de **76 %**. Si se detiene cerca del 48 %, el servidor está usando hilos y no procesos. 3. El grupo pasa de 2 a 4 instancias. Medido: **410 segundos** desde el inicio de la carga. 4. Los backends nuevos aparecen en el balanceador y se ponen sanos. Medido: **707 segundos** desde el inicio de la carga, casi doce minutos. 5. El grupo sigue creciendo hasta su máximo de **6** si la carga se mantiene. 6. Corta la carga y verifica en **Metrics Explorer** que la CPU cayó de verdad. 7. El grupo baja de 6 a 5. Medido: **251 segundos** después de cortar. 8. El grupo sigue bajando, de a una instancia cada ciclo, hasta el mínimo de 2.

Si los ocho pasos ocurren, el laboratorio está completo. Si el octavo no ocurre, tiene la mitad que casi nadie configura sin configurar, y la factura lo va a notar.

14.5 El presupuesto

Consola de OCI Billing & Cost Management › Budgets › `lab01-presupuesto-laboratorio`

1. El presupuesto existe, apunta al compartimento `lab-01-elasticidad` y tiene las cuatro reglas de alerta.
2. En **Cost Analysis**, filtre por el compartimento del laboratorio y agrupe por servicio. Necesita cerca de 24 horas de datos acumulados para mostrar algo: si acaba de crear todo, vuelva mañana.
3. Confirme que el gasto observado está en el orden esperado. En el ensayo de referencia fueron **0,13 USD** de cómputo, con una proyección mensual de **2,10 USD** contra el límite de 150.

15. Qué puede salir mal

Todos los síntomas de esta tabla ocurrieron de verdad durante la construcción y el ensayo del

laboratorio. Ninguno es hipotético.

Síntoma	Causa	Arreglo
La CPU bajo carga se queda pegada cerca del **48 %** y el grupo nunca crece	El servidor usa hilos, y el bloqueo global del intérprete de Python serializa el bucle que quema CPU. En 1 OCPU (2 vCPU) el techo es la mitad exacta	Usar el script de inicialización del capítulo 9, que bifurca **procesos**. La CPU pasa a 76–99 %. Es el fallo 1 del ensayo
Se corta el generador de carga y la CPU sigue alta; el paso de bajada no llega nunca	Los procesos hijos del generador quedaron huérfanos y siguen pidiendo	Verificar en **Metrics Explorer** que la CPU cayó de verdad antes de empezar a contar el tiempo de bajada. Matar también los procesos hijos. Es el fallo 2 del ensayo
Borrar la instance configuration falla con un error de recurso en uso	El grupo de instancias todavía la referencia	Crear la nueva, reapuntar el grupo, y solo entonces borrar la vieja. Capítulo 13. Es el fallo 6 del ensayo
El backend queda en estado **crítico** con la instancia en ejecución	`firewalld` bloquea el puerto 80 en Oracle Linux	El script de inicialización abre el 80. Verificar con `sudo firewall-cmd --list-ports` dentro de la instancia
Ninguna instancia responde y el script no parece haber corrido	El script de inicialización falló	Entrar por SSH y leer `/var/log/cloud-init-output.log`
La instancia responde por SSH pero no por HTTP	El servicio de la aplicación no arrancó	`sudo systemctl status lab-app` dentro de la instancia
Tiempo de espera agotado desde su equipo, pero las instancias están sanas	Falta la regla de ingreso al puerto 80 en la security list	Revisar el capítulo 6
El balanceador no alcanza a las instancias; todos los backends críticos	Falta la regla de ingreso desde `10.30.1.0/24`	Es la tercera regla del capítulo 6, y es la que más se olvida
Metrics Explorer no muestra ninguna serie para las instancias	El plugin de monitoreo del Oracle Cloud Agent está deshabilitado, o la instancia acaba de arrancar	Habilitarlo en la configuración de instancia. Esperar 5–10 minutos tras el arranque antes de concluir que falla
La política de autoescalamiento existe pero nunca dispara	No hay métrica que leer, o el enfriamiento de 300 s sigue corriendo desde la última acción	Verificar primero la métrica. Después, revisar cuándo fue la última acción de escalamiento
El grupo crece pero el balanceador sigue mandando todo a las instancias viejas	El health check todavía no ha pasado las tres verificaciones	Esperar. Con intervalo de 10 s y 3 reintentos son 30 segundos como mínimo

Un backend queda permanentemente crítico y apunta a una instancia que ya no existe	Se agregó un backend a mano en el balanceador	Borrarlo. Con el grupo enganchado, los backends los administra OCI
El grupo vuelve solo al tamaño que usted no quería	La política de autoescalamiento lo está corrigiendo	Deshabilitar la configuración de autoescalamiento durante la intervención y volver a habilitarla
No aparece la opción de crear el presupuesto, o el presupuesto no vigila nada	Está parado en el compartimento del laboratorio en vez del raíz	Los presupuestos se crean en el compartimento **raíz**, aunque apunten a un hijo
La creación de instancias falla por falta de cupo	Límite de servicio insuficiente para el shape	**Limits, Quotas and Usage**. El aumento no es inmediato: pídalo con días
`oci lb backend-health list` devuelve un error de comando desconocido	Ese subcomando no existe	Es `oci lb backend-set-health get`. Fallo 4 del ensayo

16. Limpieza

El laboratorio se destruye todos los días. No es una recomendación de higiene: es la diferencia entre gastar centavos y gastar el crédito de la cuenta de prueba. Un balanceador flexible y seis instancias olvidadas un fin de semana cuestan más que toda la semana de preparación bien apagada.

Predicar FinOps en el capítulo 4 y dejar el laboratorio encendido se nota.

16.1 El orden correcto

El orden importa porque cada recurso bloquea el borrado del anterior.

```
1. Deshabilitar la autoscaling configuration
      └─ si no, repone instancias mientras usted las borra
2. Borrar la autoscaling configuration
3. Borrar el instance pool
      └─ termina sus instancias y las quita del backend set
4. Borrar la instance configuration
      └─ ahora sí se deja: ya nadie la referencia
5. Borrar el load balancer
      └─ es lo que cobra por hora aunque no pase trafico
6. Borrar la subred, la security list, la route table y el
   internet gateway
7. Borrar la VCN
8. (Opcional) Borrar el compartimento
      └─ solo cuando este vacio; el borrado tarda
```

Consola de OCI Compute › Autoscaling Configurations › `lab01-autoscaling`

1. Abra la configuración de autoescalamiento y **deshabilítela** antes de tocar nada más. Si borra instancias con la política activa, el grupo las repone y usted persigue su propia cola.
2. Espere unos segundos y bórrela.

Consola de OCI Compute › Instance Pools › `lab01-pool` › Terminate

1. Abra el grupo y termínelo. Esto apaga y borra todas sus instancias, y las da de baja del backend set. Espere a que el grupo desaparezca de la lista antes de seguir.

Consola de OCI Compute › Instance Configurations › `lab01-instance-config` › Delete

1. Borre la configuración de instancia. Ahora sí se deja, porque ya no hay grupo que la referencie.

Consola de OCI Networking › Load Balancers › `lab01-lb` › Delete

1. Borre el balanceador. **Este es el recurso que hay que borrar sí o sí**: cobra por hora por su ancho de banda mínimo garantizado, pase o no pase tráfico por él.

Consola de OCI Networking › Virtual Cloud Networks › `lab01-vcn`

1. Dentro de la VCN, borre en este orden: la subred, la security list, la tabla de rutas y el internet gateway. Una subred con VNICs adentro no se borra; si se resiste, es que quedó una instancia viva o el balanceador todavía existe.
2. Borre la VCN.
3. Opcionalmente, borre el compartimento. Solo se deja borrar cuando está vacío, y el borrado tarda un rato en completarse. Si va a repetir el laboratorio, consérvelo.

16.2 Qué se queda cobrando si lo hace mal

CUIDADO · Lo que sigue corriendo el reloj después de un borrado incompleto

- **El balanceador flexible.** Cobra por su ancho de banda mínimo garantizado las 24 horas, haya tráfico o no. Es el recurso más caro de este laboratorio y el más fácil de olvidar, porque no aparece en la lista de instancias de cómputo. - **Los volúmenes de arranque huérfanos.** Terminar una instancia sin marcar que se borre su volumen de arranque deja 50 GB de almacenamiento facturándose por cada instancia. Seis instancias son 300 GB de nada. Revise **Block Storage › Boot Volumes** en el compartimento después de limpiar. - **Las IP públicas reservadas.** Si cambió la IP efímera del balanceador por una reservada, esa IP sigue facturándose después de borrar el balanceador, hasta que la libere. - **El presupuesto y sus alertas no cuestan nada.**

Déjelos puestos. Son lo que le va a avisar si algo de lo anterior se le quedó encendido.

Después de limpiar, vuelva a **Cost Analysis** filtrando por el compartimento del laboratorio y confirme que el gasto diario cae a cero en las siguientes 24 horas. Los datos de consumo tardan horas en consolidarse, así que la cifra del día del borrado seguirá subiendo un rato. El que importa es el día siguiente.

17. Documentación oficial

Los enlaces apuntan a la sección general de cada servicio. Las rutas profundas de `docs.oracle.com` cambian con las versiones de la consola; desde la página general siempre se llega.

Tema	Enlace
Red virtual (VCN, subredes, gateways)	https://docs.oracle.com/en-us/iaas/Content/Network/Concepts/overview.htm
Security lists	https://docs.oracle.com/en-us/iaas/Content/Network/Concepts/securitylists.htm
Network Security Groups	https://docs.oracle.com/en-us/iaas/Content/Network/Concepts/networksecuritygroups.htm
Load Balancer	https://docs.oracle.com/en-us/iaas/Content/Balance/Concepts/balanceoverview.htm
Cómputo: instancias, configuraciones y grupos	https://docs.oracle.com/en-us/iaas/Content/Compute/Concepts/computeoverview.htm
Gestión de instancias	https://docs.oracle.com/en-us/iaas/Content/Compute/Concepts/instancemanagement.htm
Autoescalamiento de instance pools	https://docs.oracle.com/en-us/iaas/Content/Compute/Tasks/autoscalinginstancepools.htm
Oracle Cloud Agent y sus plugins	https://docs.oracle.com/en-us/iaas/Content/Compute/Tasks/manedatoricloudagent.htm
Monitoring y Metrics Explorer	https://docs.oracle.com/en-us/iaas/Content/Monitoring/Concepts/monitoringoverview.htm
Métricas del servicio de cómputo	https://docs.oracle.com/en-us/iaas/Content/Compute/References/computemetrics.htm
Budgets	https://docs.oracle.com/en-us/iaas/Content/Billing/Concepts/budgetsoverview.htm
Cost Analysis	https://docs.oracle.com/en-us/iaas/Content/Billing/Concepts/costanalysisoverview.htm

Límites de servicio	https://docs.oracle.com/en-us/iaas/Content/General/Concepts/servicelimits.htm
Compartimentos	https://docs.oracle.com/en-us/iaas/Content/Identity/Tasks/managingcompartments.htm

Si alguno de estos enlaces devuelve un error, busque el nombre del servicio en `docs.oracle.com/en-us/iaas/` y entre por el índice. Es más rápido que adivinar la ruta.