

# El Curso de Ingeniero Cloud (The Cloud Engineer Course)

De cero a ingeniero cloud listo para trabajar — un curso B4LCILC

13 de agosto de 2026

## El Curso de Ingeniero Cloud (The Cloud Engineer Course)

**De cero conocimiento a estar genuinamente listo para trabajar como ingeniero cloud.**

### Cómo funciona este curso

Usted se está formando para *ser* ingeniero cloud: la persona que construye servidores con código, mantiene los sistemas vivos a las 3 a.m., automatiza el trabajo aburrido hasta hacerlo desaparecer, y puede mirar un sistema roto y descubrir con calma por qué. Ese es un oficio práctico, y este curso lo trata como tal. Pasará mucho más tiempo tecleando que leyendo — cada módulo tiene laboratorios (labs), y los labs son el curso. Leer sobre la nube (cloud) construye reconocimiento; construir en la nube construye una carrera.

El curso tiene cinco etapas. **Etapas 0 (Semanas 1–4): Fundamentos** — cómo funcionan realmente las computadoras y las redes, y Linux, el sistema operativo sobre el que corre la nube. **Etapas 1 (Semanas 5–12): Cloud esencial, con las manos en la masa** — los cinco servicios de AWS que aparecen en toda descripción de puesto, aprendidos como labs que usted construye y desmonta. **Etapas 2 (Semanas 13–20): Automatización** — scripting, Git, Terraform, CI/CD y contenedores: las habilidades que separan a un ingeniero de alguien que solo hace clics en la consola. **Etapas 3 (Semanas 21–28): Operaciones** — monitoreo, incidentes, seguridad y costos: las habilidades por las que realmente le pagan. **Etapas 4 (Meses 8–12): Preparación laboral** — tres proyectos de portafolio, tres certificaciones y entrenamiento para entrevistas.

Reglas para todo el curso: estudie **1.5–2 horas al día, seis días a la semana** — la constancia vence a la intensidad, y este es un oficio que se aprende por repetición diaria, como un instrumento musical. Cada módulo termina con una **tabla de Vocabulario** (las palabras que debe dominar), un **drill de decirlo en voz alta** (frases que repite hasta que le salgan con naturalidad — las entrevistas de trabajo son habladas, no escritas), **Ejercicios** prácticos, y un **Hito** que condiciona el avance: no siga adelante hasta que pueda cumplirlo, porque cada etapa se apoya en la anterior. Desde la Semana 1, lleve un **Diario de Ingeniería**: cada lab, cada mensaje de error, cada arreglo, un párrafo honesto. Para el Mes 10, ese diario se convierte en la materia prima de su portafolio y en sus historias de entrevista.

Una promesa a cambio: nada en este curso asume que usted sabe algo. Cada término se define la primera vez que aparece. Si puede usar un navegador web y está dispuesto a teclear comandos que se sentirán extraños durante las primeras dos semanas, tiene todos los prerrequisitos que existen.

## La descripción de puesto para la que se está formando

En agosto de 2026 recopilamos descripciones reales de puestos de Cloud Engineer a partir de plantillas de empleadores y guías de contratación (Arc.dev, Wiz, DevsData, XOPA, Betterteam — lista completa en Fuentes). Quite los nombres de las empresas y los mismos requisitos aparecen una y otra vez. Esta tabla es su contrato con el curso — cada punto que pide un empleador corresponde a la etapa que lo enseña:

| Lo que piden las descripciones de puesto reales (casi textual)                                                                                                                                                                                                  | Dónde lo enseña este curso        |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------|
| “Knowledge of Linux/Unix operating systems” (conocimiento de sistemas operativos Linux/Unix)                                                                                                                                                                    | Etapas 0, Módulo 1                |
| “Expertise in cloud networking including VPCs, subnets, load balancers, DNS” (dominio de redes cloud incluyendo VPCs, subredes, balanceadores de carga, DNS)                                                                                                    | Etapas 0 Módulo 2 + Etapa 1 Lab 3 |
| “Demonstrated expertise in core AWS services, including EC2, S3, RDS, VPC, IAM” (dominio demostrado de los servicios centrales de AWS, incluidos EC2, S3, RDS, VPC, IAM)                                                                                        | Etapas 1 (Labs 1–5)               |
| “Design, develop, and deploy cloud infrastructure using infrastructure as code tools such as Terraform, CloudFormation” (diseñar, desarrollar y desplegar infraestructura cloud con herramientas de infraestructura como código como Terraform, CloudFormation) | Etapas 2, Módulo 7                |
| “Proficiency in scripting languages such as Python, Bash, PowerShell, or Go” (competencia en lenguajes de scripting como Python, Bash, PowerShell o Go)                                                                                                         | Etapas 2, Módulo 6                |
| “Build and maintain CI/CD pipelines using tools like Jenkins, GitLab CI, or GitHub Actions” (construir y mantener pipelines de CI/CD con herramientas como Jenkins, GitLab CI o GitHub Actions)                                                                 | Etapas 2, Módulo 8                |
| “Experience with containerization technologies including Docker and Kubernetes” (experiencia con tecnologías de contenedores incluyendo Docker y Kubernetes)                                                                                                    | Etapas 2, Módulo 8                |
| “Monitor infrastructure health and performance using cloud-native monitoring                                                                                                                                                                                    | Etapas 3, Módulo 9                |

| Lo que piden las descripciones de puesto reales (casi textual)                                                                                                                                                                                                                                                                                                                                                                                                                    | Dónde lo enseña este curso                                                    |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------|
| tools” (monitorear la salud y el rendimiento de la infraestructura con herramientas de monitoreo nativas de la nube)                                                                                                                                                                                                                                                                                                                                                              |                                                                               |
| “Participate in incident response, including log analysis”; “troubleshooting and analytical skills” (participar en la respuesta a incidentes, incluido el análisis de logs; habilidades de diagnóstico y análisis)                                                                                                                                                                                                                                                                | Etapa 3, Módulo 10                                                            |
| “Implement and enforce security controls including encryption, identity and access management”; “least-privilege access” (implementar y hacer cumplir controles de seguridad incluyendo cifrado, gestión de identidades y accesos; acceso de mínimo privilegio)                                                                                                                                                                                                                   | Etapa 1 Lab 5 + Etapa 3 Módulo 10                                             |
| “Manage cloud costs through rightsizing resources, implementing auto-scaling, resource tagging” (gestionar costos cloud mediante ajuste de tamaño de recursos, auto-scaling y etiquetado de recursos)                                                                                                                                                                                                                                                                             | Etapa 3, Módulo 10                                                            |
| “Maintaining, testing and implementing disaster recovery procedures” (mantener, probar e implementar procedimientos de recuperación ante desastres)                                                                                                                                                                                                                                                                                                                               | Etapa 3 + Proyecto de Portafolio 3                                            |
| “AWS certifications preferred” (certificaciones AWS preferidas)                                                                                                                                                                                                                                                                                                                                                                                                                   | Calendario de certificaciones de la Etapa 4 (CCP – SAA – Terraform Associate) |
| “Provide technical guidance and documentation”; “good communication and collaboration skills” (proporcionar guía técnica y documentación; buenas habilidades de comunicación y colaboración)                                                                                                                                                                                                                                                                                      | Diario + runbooks + preparación de entrevistas de la Etapa 4                  |
| <p>Contexto salarial, para que sepa hacia qué está trabajando: los salarios de ingeniero cloud en EE. UU. se agrupan en torno a una <b>mediana cercana a \$104,000</b>, con un rango típico de <b>\$85K–\$140K</b> y roles senior/especialistas de AWS anunciados muy por encima. Los puestos de nivel de entrada existen bajo muchos nombres — cloud support associate, junior cloud engineer, cloud operations engineer — y este curso apunta exactamente a sus requisitos.</p> |                                                                               |

## ETAPA 0 — FUNDAMENTOS (Semanas 1-4)

### Módulo 1 (Semanas 1-2): Cómo funcionan las computadoras, y Linux — el idioma de los servidores

**La gran idea:** un **servidor** es simplemente una computadora cuyo trabajo es servir a otras computadoras. La laptop frente a usted y las máquinas que hacen funcionar Netflix difieren en tamaño y confiabilidad, no en naturaleza: ambas son una **CPU** (la parte que hace el trabajo), **memoria/RAM** (espacio de trabajo rápido de corto plazo, borrado al reiniciar), **disco** (almacenamiento lento de largo plazo que sobrevive a los reinicios) y una **tarjeta de red** (la conexión con todo lo demás), todo coordinado por un **sistema operativo (OS)**. Su laptop probablemente corre Windows o macOS. Los servidores corren abrumadoramente **Linux** — un OS gratuito y de código abierto que es estable, automatizable con scripts y controlado enteramente por comandos escritos. Esa última parte es el punto: no se pueden automatizar los clics del mouse, pero sí los comandos, y la ingeniería cloud es automatización. Así que su primera fluidez es la **terminal**.

**La terminal, desmitificada.** La terminal (o “shell” — el programa dentro de ella suele ser **Bash**) es una conversación de texto con la computadora. Usted escribe un comando; ella responde. Eso es todo. El \$ que verá en los ejemplos es el **prompt** — el shell diciendo “su turno”. En Linux todo es un archivo, los archivos viven en un solo árbol que empieza en la raíz /, y su carpeta personal es /home/sunombre (apodo: ~). Una **ruta (path)** es la dirección de un archivo en ese árbol: /home/anna/notes.txt.

**Consiga un Linux para practicar (elijá uno, diez minutos):** en Windows, instale **WSL** (Windows Subsystem for Linux — un Ubuntu Linux real dentro de Windows: <https://learn.microsoft.com/en-us/windows/wsl/install>); en una Mac, la app Terminal integrada es lo bastante parecida para empezar (macOS es un primo de Unix); o espere a la Semana 5, cuando alquilará gratis un servidor Linux real de AWS. WSL es la mejor respuesta para la mayoría.

**Los 25 comandos principales — esta tabla es las Semanas 1-2. Teclee cada uno de ellos, muchas veces:**

| Comando | Qué hace                                                           | Ejemplo             |
|---------|--------------------------------------------------------------------|---------------------|
| pwd     | Print working directory — “¿dónde estoy?”                          | pwd                 |
| ls      | Listar los archivos de aquí (-l detalle largo, -a incluir ocultos) | ls -la              |
| cd      | Change directory — moverse por el árbol                            | cd /var/log         |
| mkdir   | Crear un directorio (carpeta)                                      | mkdir projects      |
| touch   | Crear un archivo vacío                                             | touch notes.txt     |
| cp      | Copiar un archivo (-r para carpetas)                               | cp a.txt backup.txt |

| Comando                     | Qué hace                                                                                                 | Ejemplo                             |
|-----------------------------|----------------------------------------------------------------------------------------------------------|-------------------------------------|
| <code>mv</code>             | Mover o renombrar                                                                                        | <code>mv old.txt new.txt</code>     |
| <code>rm</code>             | Remove — permanente, sin papelera. Respételo.                                                            | <code>rm notes.txt</code>           |
| <code>cat</code>            | Imprimir el contenido completo de un archivo                                                             | <code>cat notes.txt</code>          |
| <code>less</code>           | Leer un archivo largo página por página (q para salir)                                                   | <code>less /var/log/syslog</code>   |
| <code>head / tail</code>    | Primeras / últimas líneas de un archivo. <code>tail -f</code> observa un log en vivo — un clásico de ops | <code>tail -f app.log</code>        |
| <code>grep</code>           | Buscar un patrón en texto — el comando de ops más usado de todos                                         | <code>grep "ERROR" app.log</code>   |
| <code>find</code>           | Encontrar archivos por nombre/tamaño/antigüedad                                                          | <code>find / -name "*.conf"</code>  |
| <code>echo</code>           | Imprimir texto (a menudo hacia archivos o variables)                                                     | <code>echo "hello"</code>           |
| <code>nano</code>           | Un editor de texto amigable dentro de la terminal                                                        | <code>nano notes.txt</code>         |
| <code>man</code>            | El manual de cualquier comando (q para salir)                                                            | <code>man grep</code>               |
| <code>sudo</code>           | Ejecutar un comando como el administrador todopoderoso (“root”). Con respeto.                            | <code>sudo apt update</code>        |
| <code>apt</code>            | Instalar/actualizar software (familia Ubuntu/Debian)                                                     | <code>sudo apt install httpd</code> |
| <code>chmod</code>          | Cambiar los permisos de un archivo                                                                       | <code>chmod 644 notes.txt</code>    |
| <code>chown</code>          | Cambiar el dueño de un archivo                                                                           | <code>sudo chown anna file</code>   |
| <code>ps</code>             | Listar los procesos en ejecución ( <code>ps aux</code> para todos)                                       | <code>ps aux</code>                 |
| <code>top (o htop)</code>   | Tablero en vivo de CPU/memoria/procesos — “¿por qué está lento el servidor?” empieza aquí                | <code>top</code>                    |
| <code>df -h / du -sh</code> | Espacio libre en disco / espacio que usa una carpeta — “el disco está lleno”                             | <code>df -h</code>                  |

| Comando | Qué hace                                                                                                            | Ejemplo                  |
|---------|---------------------------------------------------------------------------------------------------------------------|--------------------------|
| ssh     | empieza aquí<br>Iniciar sesión en la terminal de otra máquina por la red — la puerta de entrada del ingeniero cloud | ssh anna@server-ip       |
| curl    | Hacer una solicitud web desde la terminal — “¿el sitio está arriba?”                                                | curl https://example.com |

**Permisos, la versión de 60 segundos.** Cada archivo tiene un dueño y un modo como `rwxr-xr--`: tres tripletas — dueño, grupo, todos — de **read** (leer), **write** (escribir), **execute** (ejecutar). En números: `r=4, w=2, x=1`, así que `chmod 755 script.sh` significa “el dueño puede hacer todo ( $7=4+2+1$ ); todos los demás pueden leerlo y ejecutarlo ( $5=4+1$ )”. Cuando un programa “no tiene permiso”, este es el sistema diciendo que no — y ahora usted puede leer por qué.

**SSH, la versión de 60 segundos.** `ssh` abre una terminal remota segura en otra máquina — desde su laptop hacia un servidor en Virginia como si estuviera sentado frente a él. En lugar de contraseñas, los profesionales usan un **par de llaves (key pair)**: una llave privada (un archivo secreto en su laptop, nunca compartido) y una llave pública (colocada en el servidor). Encajan como llave y cerradura. Cada servidor de AWS que lance en la Etapa 1 le entregará exactamente esto.

**Procesos y servicios:** un **proceso** es un programa en ejecución; un **servicio** (o “daemon”) es un proceso que corre para siempre en segundo plano — servidores web, bases de datos. `systemctl status nginx` le pregunta a Linux “¿el servicio nginx está sano?” — una frase que usted tecleará profesionalmente durante años.

### Vocabulario:

| Término                 | Definición                                                                                                                       |
|-------------------------|----------------------------------------------------------------------------------------------------------------------------------|
| Servidor                | Una computadora cuyo trabajo es servir a otras computadoras. En la nube, una que usted alquila.                                  |
| CPU / RAM / disco       | El trabajador, el espacio de trabajo temporal rápido (borrado al reiniciar) y el almacenamiento permanente lento.                |
| Sistema operativo (OS)  | El software que hace funcionar la máquina y aloja los programas. Los servidores corren Linux.                                    |
| Linux / distribución    | El OS de servidores gratuito y de código abierto. Una “distro” (Ubuntu, Amazon Linux, Debian) es una variante empaquetada de él. |
| Terminal / shell / Bash | La interfaz de texto hacia el OS / el programa                                                                                   |

| Término                     | Definición                                                                                                                              |
|-----------------------------|-----------------------------------------------------------------------------------------------------------------------------------------|
|                             | que interpreta sus comandos / el nombre del shell estándar.                                                                             |
| Prompt                      | El \$ — el shell esperando su comando.                                                                                                  |
| Directorio / ruta (path)    | Una carpeta / la dirección completa de un archivo en el árbol único que empieza en /.                                                   |
| Root (dos significados)     | La cima del árbol de archivos (/) y el usuario administrador todopoderoso. El contexto le dice cuál es cuál.                            |
| sudo                        | “Superuser do” — ejecutar un comando con poder de administrador.                                                                        |
| Permisos (rwx)              | Reglas por archivo sobre quién puede leer, escribir, ejecutar — mostradas como tripletas para dueño/grupo/todos.                        |
| Proceso / servicio (daemon) | Un programa en ejecución / uno que corre para siempre en segundo plano (servidores web, bases de datos).                                |
| SSH / par de llaves         | Inicio de sesión remoto seguro en la terminal de otra máquina / los archivos de llave privada+pública que reemplazan a las contraseñas. |
| Log                         | Un archivo de texto donde el software escribe lo que pasó — el primer lugar donde mirar cuando algo se rompe.                           |
| Gestor de paquetes          | El instalador del OS (apt, yum) — software por comando, no por página de descargas.                                                     |

### Videos de este módulo (enlaces verificados):

| Video                                               | Canal           | Duración | Enlace                                                                                                                    |
|-----------------------------------------------------|-----------------|----------|---------------------------------------------------------------------------------------------------------------------------|
| Linux Operating System — Crash Course for Beginners | freeCodeCamp    | ~2 h     | <a href="https://www.youtube.com/watch?v=ROjZy1WbCIA">https://www.youtube.com/watch?v=ROjZy1WbCIA</a>                     |
| WSL install guide (referencia, no video)            | Microsoft Learn | —        | <a href="https://learn.microsoft.com/en-us/windows/wsl/install">https://learn.microsoft.com/en-us/windows/wsl/install</a> |

**Dígallo en voz alta hasta que salga natural:** “Let me SSH in and check the logs.” (Déjeme entrar por SSH y revisar los logs.) · “Grep the log for the error, then tail -f it while we retry.” (Busque el error en el log con grep, y luego hágale tail -f mientras reintentamos.) · “It’s a permissions problem — who owns the file and what’s the mode?” (Es un problema de permisos

— ¿quién es el dueño del archivo y cuál es el modo?) · “Check top — is it CPU, memory, or disk?” (Revise top — ¿es CPU, memoria o disco?)

**Ejercicios:** (1) En su terminal Linux, construya un pequeño árbol de proyecto con `mkdir` y `touch`, copie y mueva cosas, luego bórralo — narrando cada comando en voz alta. (2) Cree `hello.sh` con el contenido `echo "hello from $(whoami)"`, hágalo ejecutable con `chmod +x`, ejecútelo con `./hello.sh`. (3) Ejecute `tail -f` sobre un archivo de log (en Ubuntu: `sudo tail -f /var/log/syslog`) y observe llegar las líneas. (4) Diario: explique a un amigo imaginario por qué los servidores corren Linux, en tres frases.

**Hito:** sin notas, usted puede navegar a cualquier parte del árbol de archivos, crear/copiar/mover/borrar archivos, explicar `chmod 755` y usar `grep` para encontrar una palabra en un archivo. Si algo de eso requiere buscar información, pase dos días más aquí. Este módulo es la base estructural de todo lo demás.

## Módulo 2 (Semanas 3–4): Fundamentos de redes + su cuenta de AWS

**La gran idea:** una red es computadoras pasándose sobres con dirección. Cada máquina recibe una **dirección IP** (como `172.31.8.14` — una dirección postal numérica). Los datos se trocean en **paquetes** (sobres) y se enrutan salto a salto hacia la dirección de destino. Al llegar, un número de **puerto** dice para qué programa es el sobre — el mismo edificio, miles de puertas numeradas: el puerto **22** es SSH, el **80** es web sin cifrar (**HTTP**), el **443** es web cifrada (**HTTPS**), el **5432** es PostgreSQL. “Abrir el puerto 443” significa “permitir sobres dirigidos a la puerta 443”.

**Privado vs público:** su casa y toda red cloud reutilizan **rangos de IP privadas** (`10.x.x.x`, `172.16–31.x.x`, `192.168.x.x`) que solo funcionan dentro de la red local; una **IP pública** es alcanzable desde toda la internet. Esta división es la base de la seguridad cloud: lo que no necesita mirar hacia internet no recibe ninguna dirección pública.

**DNS — la guía telefónica de internet.** Los humanos usan nombres (`example.com`); los paquetes necesitan números. El **DNS** traduce: su máquina le pregunta a un servidor DNS “¿cuál es la IP de `example.com`?”, recibe el número y luego se conecta. La mitad de las caídas misteriosas involucran al DNS; el chiste de la industria “it’s always DNS” (siempre es el DNS) existe porque a menudo es cierto. `nslookup example.com` realiza la consulta a mano.

**HTTP — cómo habla la web.** Un **cliente** (navegador) envía una solicitud — un **método** (GET = obtener, POST = enviar) más una ruta — y el **servidor** responde con un **código de estado**: 200 OK, 301 movido, 403 prohibido, 404 no encontrado, 500 error del servidor, 502/503 “el servidor detrás de mí está roto/sobrecargado”. Memorice esos seis; como ingeniero los leerá a diario. `curl -I https://example.com` le muestra en vivo una línea de estado y los encabezados.

**Firewalls:** una lista de reglas que decide qué paquetes pueden pasar, por origen, destino y puerto — “permitir 443 desde cualquier lugar; permitir 22 solo desde la oficina; denegar el resto”. En AWS el firewall por servidor se llama **security group** (grupo de seguridad), y los mal configurados son el agujero de seguridad #1 del principiante. **Latencia** (retraso, en milisegundos) y **ancho de banda** (capacidad por segundo) completan el vocabulario: la distancia crea latencia, y por eso las nubes tienen regiones en todo el mundo.

**Ahora, su cuenta de AWS — la segunda mitad de este módulo.** Vaya a <https://aws.amazon.com/free> y cree una cuenta Free Tier (correo, teléfono, tarjeta de crédito/débito para identidad — la tarjeta no recibe cargos significativos si usted sigue la disciplina de desmontaje de este curso). El **Free Tier** (capa gratuita) da asignaciones mensuales de lo básico (incluidas 750 horas/mes de un servidor EC2 pequeño en su primer año) y cada lab de este curso está diseñado para caber dentro de él. Luego, antes que nada, tres pasos de seguridad no negociables — hacerlos es el primer ejercicio de su carrera en seguridad:

1. **MFA en el usuario root.** El usuario root es la llave maestra de la cuenta. Añada autenticación multifactor (una app autenticadora en el teléfono) en IAM → Security credentials. Luego deje de usar root para el trabajo diario.
2. **Cree un usuario IAM administrador** (el Lab 5 explica IAM a fondo; por ahora: consola → IAM → Users → crear usuario con acceso de administrador) e inicie sesión como ese usuario de ahora en adelante.
3. **Una alerta de facturación.** Consola → Billing → Budgets → cree un **presupuesto de gasto cero** (la plantilla de AWS que le envía un correo en el momento en que algo factura, lo que sea). Active también las alertas de uso del Free Tier. Un ingeniero que no puede controlar los costos es un pasivo; usted lleva 20 minutos con su cuenta y ya va por delante de muchos profesionales. (Referencia: <https://docs.aws.amazon.com/cost-management/latest/userguide/budgets-managing-costs.html>)

### Vocabulario:

| Término              | Definición                                                                                                     |
|----------------------|----------------------------------------------------------------------------------------------------------------|
| Dirección IP         | La dirección de red numérica de una máquina, p. ej. 172 . 31 . 8 . 14.                                         |
| Paquete              | Un sobre de datos con dirección; todo el tráfico son flujos de ellos.                                          |
| Puerto               | Una puerta numerada en una máquina que identifica para qué programa es el tráfico: 22 SSH, 80 HTTP, 443 HTTPS. |
| IP privada / pública | Una dirección válida solo dentro de una red local / una alcanzable desde toda la internet.                     |
| DNS                  | El sistema que traduce nombres (example.com) en direcciones IP. “It’s always DNS.”                             |
| HTTP / HTTPS         | El protocolo solicitud-respuesta de la web / el mismo, cifrado con TLS.                                        |
| Código de estado     | El resumen de la respuesta del servidor: 200 OK, 404 no encontrado, 500 error del servidor, 503 sobrecargado.  |
| Firewall             | La lista de reglas que decide qué paquetes pasan, por origen, destino, puerto.                                 |
| Security group       | El firewall por servidor de AWS. Configurar uno mal es el agujero clásico del principiante.                    |

| Término                             | Definición                                                                                       |
|-------------------------------------|--------------------------------------------------------------------------------------------------|
| Latencia / ancho de banda           | Retraso (ms) / capacidad (por segundo). La distancia crea latencia.                              |
| Cliente / servidor (roles)          | El que pregunta y el que responde en cualquier conversación de red.                              |
| AWS Free Tier                       | La asignación mensual gratuita de una cuenta nueva de AWS — el presupuesto entero de este curso. |
| Usuario root                        | La identidad maestra de la cuenta de AWS. Póngale MFA, y luego deje de usarla.                   |
| Alerta de facturación / presupuesto | El correo automático cuando el gasto cruza un umbral. El suyo está en \$0.                       |

### Videos de este módulo:

| Video                                        | Canal                         | Duración | Enlace                                                                                                |
|----------------------------------------------|-------------------------------|----------|-------------------------------------------------------------------------------------------------------|
| What is AWS?                                 | Amazon Web Services (oficial) | ~2 min   | <a href="https://www.youtube.com/watch?v=a9_D53WsUs">https://www.youtube.com/watch?v=a9_D53WsUs</a>   |
| Top 50+ AWS Services Explained in 10 Minutes | Fireship                      | ~10 min  | <a href="https://www.youtube.com/watch?v=JlBIYCM48to">https://www.youtube.com/watch?v=JlBIYCM48to</a> |
| AWS Networking Basics — VPC & Subnets        | KodeKloud                     | ~30 min  | <a href="https://www.youtube.com/watch?v=QM63dyA_4Pc">https://www.youtube.com/watch?v=QM63dyA_4Pc</a> |

**Dívalo en voz alta:** “What’s the IP, and is it public or private?” (¿Cuál es la IP, y es pública o privada?) · “Is port 443 open in the security group?” (¿El puerto 443 está abierto en el security group?) · “Curl it — what status code do you get?” (Hágale curl — ¿qué código de estado obtiene?) · “Did DNS resolve? Check with nslookup before blaming the server.” (¿El DNS resolvió? Verifique con nslookup antes de culpar al servidor.)

**Ejercicios:** (1) `ping google.com` (anote la latencia), `nslookup google.com` (note que el DNS devuelve varias IPs), `curl -I https://aws.amazon.com` (lea la línea de estado y tres encabezados, y luego busque qué significa cada uno). (2) Encuentre la IP privada de su máquina (`ip addr` en Linux) y su IP pública (busque “what is my IP”) — explique en su diario por qué difieren. (3) Complete la configuración de la cuenta de AWS: MFA, usuario administrador, presupuesto de gasto cero — capture una imagen del presupuesto para su diario. (4) Dibuje de memoria: laptop — consulta DNS — solicitud HTTPS por el puerto 443 — firewall — servidor web. Tres veces, hasta que sea automático.

**Hito — fin de la Etapa 0:** usted puede narrar qué pasa cuando escribe una URL y presiona Enter — DNS, IP, puerto, firewall, solicitud HTTP, código de estado — en menos de dos

minutos, usando cada término correctamente; y su cuenta de AWS existe con MFA y una alerta de presupuesto de \$0. Ahora sabe más sobre cómo funciona internet que la mayoría de las personas que lo usan para ganarse la vida. La Etapa 1 es donde empieza a construir sobre ello.

---

## ETAPA 1 — CLOUD ESENCIAL, CON LAS MANOS EN LA MASA (Semanas 5-12)

**Cómo funciona esta etapa.** Cada uno de los cinco servicios de abajo es un **Lab**: un objetivo, un esquema de pasos (lo bastante detallado para seguirlo, lo bastante corto para que usted tenga que pensar — pensar es el aprendizaje), lo que aprendió, y — siempre — el **teardown** (desmontaje). La disciplina del teardown importa por partida doble: lo mantiene dentro del Free Tier, y “no deja corriendo nada que no se necesite” es un reflejo profesional que los entrevistadores realmente sondean. Reconstruya cada lab al menos dos veces: una siguiendo el esquema, otra de memoria. La segunda construcción es donde el conocimiento pasa a sus manos. Presupueste aproximadamente una semana y media por lab; use la holgura para las averías, porque las cosas se romperán, y depurarlas es la mejor enseñanza que este curso no puede guionar.

Primero, tres ideas que enmarcan todo lo que está a punto de construir. **Regiones y Availability Zones**: una Región es un grupo geográfico de centros de datos de AWS (elija una cercana a usted y quédese en ella — los recursos de una región son invisibles desde otra, la confusión #1 de “¿dónde quedó mi servidor?”). Una **Availability Zone (AZ, zona de disponibilidad)** es un centro de datos aislado dentro de la región; los sistemas serios corren en dos para que la falla de un edificio no los tumbé. **La consola vs la CLI**: la consola (<https://console.aws.amazon.com>) es el panel web de control de AWS — excelente para aprender y mirar; la **AWS CLI** (aws en su terminal) hace todo lo que hace la consola, de forma automatizable. Empezará en la consola y se graduará a la CLI, porque la Etapa 2 automatiza todo lo que aquí hace a mano.

### Lab 1 (Semanas 5-6): EC2 — su primer servidor

**EC2 (Elastic Compute Cloud)** alquila máquinas virtuales, llamadas **instancias**. Este es el acto primigenio de la ingeniería cloud: un servidor Linux real, en internet, en 60 segundos, gratis.

**Objetivo**: lanzar un servidor Linux, entrar por SSH, hacer que sirva una página web al mundo, y luego destruirlo.

**Esquema de pasos**: 1. Consola → EC2 → Launch instance. Póngale nombre. Elija **Amazon Linux 2023** como la **AMI** (Amazon Machine Image — el disco plantilla desde el que arranca su servidor) y **t2.micro** o **t3.micro** como el **tipo de instancia** (el tamaño; estos son Free Tier). 2. Cree un **par de llaves**; se descarga un archivo `.pem` de llave privada. Esa es su llave SSH del Módulo 1 — protéjala, hágale `chmod 400`. 3. En la configuración de red, permita SSH (puerto 22) *solo desde “My IP”* — ahora usted sabe exactamente qué significa esta regla de security group — y permita HTTP (puerto 80) desde cualquier lugar. 4. Lance, espere el estado “running”, copie la IP pública, y luego desde su terminal: `ssh -i mykey.pem ec2-`

user@<public-ip>. Respire: usted está dentro de una computadora en un centro de datos de AWS. 5. En el servidor: `sudo dnf install -y nginx && sudo systemctl start nginx && sudo systemctl enable nginx`. Luego navegue a `http://<public-ip>` — ese es su servidor web respondiéndole al mundo. 6. Reemplace la página por defecto: `echo "<h1>Built by me, on EC2</h1>" | sudo tee /usr/share/nginx/html/index.html`. Refresque. Capture la imagen para el diario. 7. Husmee como un ingeniero: `top, df -h, sudo tail -f /var/log/nginx/access.log` mientras refresca la página — observe sus propias visitas llegar al log.

**Lo que aprendió:** AMIs, tipos de instancia, pares de llaves, security groups en acción, SSH a un servidor real, instalar y ejecutar un servicio de Linux, leer sus logs — es decir, los movimientos físicos diarios del oficio.

**Teardown:** EC2 — seleccionar instancia — Instance state — **Terminate**. Confirme que diga “terminated”. El Free Tier da 750 horas/mes de una instancia micro, así que incluso dejarla encendida no habría facturado — pero desmóntela de todos modos. El hábito es el punto.

## Lab 2 (Semana 7): S3 — archivos que nunca desaparecen

**S3 (Simple Storage Service)** es almacenamiento de objetos: un balde (bucket) sin fondo, con durabilidad de once nueves, para archivos. Es la respuesta por defecto a “¿dónde ponemos los archivos?” y, mal configurado, la fuente de las fugas de datos más famosas de la historia — por eso este lab es mitad almacenamiento, mitad seguridad.

**Objetivo:** crear un bucket, manejarlo desde la CLI, alojar un pequeño sitio web estático, y entender exactamente qué significa “bucket público”.

**Esquema de pasos:** 1. Consola — S3 — Create bucket (los nombres son globalmente únicos — `sunombre-lab-2026` funciona). Note que **Block Public Access** está activado por defecto. Suba cualquier archivo por la consola; descárguelo de vuelta. 2. Instale la AWS CLI localmente y ejecute `aws configure` con una **access key** (clave de acceso) que cree para su usuario IAM administrador (una access key es un usuario+contraseña programático para la API — trátela como una contraseña, nunca la ponga en el código; interiorizará esta regla en el Lab 5). 3. Desde su terminal: `aws s3 ls · aws s3 cp notes.txt s3://sunombre-lab-2026/ · aws s3 sync ./myfolder s3://sunombre-lab-2026/backup/`. Sienta la diferencia: la consola es estar de visita; la CLI es ingeniería. 4. Sitio web estático: cree un segundo bucket, habilite el alojamiento de sitio web estático, suba un `index.html`, y añada la **bucket policy** de lectura pública documentada (un documento JSON de permisos — léalo línea por línea: *quién* puede hacer *qué* a *cuál* bucket). Su página está ahora en internet sin ningún servidor. 5. Explore las **clases de almacenamiento** (Standard — Infrequent Access — Glacier: más baratas por GB, más lentas/costosas de recuperar) y configure una **regla de ciclo de vida (lifecycle rule)** (“mover los objetos a IA después de 30 días”) — su primera probada de gestión de costos automatizada.

**Lo que aprendió:** almacenamiento de objetos vs discos, la CLI y las access keys, políticas de bucket y acceso público (deliberado, no accidental), los niveles de almacenamiento como palanca de costos.

**Teardown:** vacíe ambos buckets, borre ambos buckets, y — importante — desactive cualquier access key que no esté usando. La asignación gratuita de S3 es pequeña (5 GB) pero estos archivos son kilobytes; la disciplina, otra vez, es el punto.

### Lab 3 (Semanas 8-9): VPC — la red que es suya

**VPC (Virtual Private Cloud, nube privada virtual)** es su porción privada y cercada de la red de AWS. Hasta ahora usó la predeterminada sin mirarla; los ingenieros construyen la propia, porque la frase “*la base de datos está en una subred privada sin ruta a internet*” es la mitad de la seguridad cloud, y usted está a punto de hacerla realidad con sus propias manos.

**Objetivo:** construir una red de dos capas — subred pública para un servidor web, subred privada para una futura base de datos — y demostrar que la mitad privada es inalcanzable desde internet.

**Esquema de pasos:** 1. Consola → VPC → Create VPC. Dele el bloque de direcciones 10.0.0.0/16 — **notación CIDR**, donde /16 significa “los primeros 16 bits están fijos, el resto es mío”: 65,536 direcciones privadas. 2. Cree dos **subredes**: 10.0.1.0/24 (pública, en la AZ-a) y 10.0.2.0/24 (privada, en la AZ-b). Una subred es un bloque más pequeño dentro de la VPC, que vive en exactamente una AZ. 3. Cree un **Internet Gateway** (la puerta de la VPC hacia internet) y adjúntelo. Cree una **tabla de rutas (route table)** con la regla 0.0.0.0/0 → internet gateway (“todo lo que no sea local va a la puerta de internet”) y asóciela *solo* con la subred pública. La subred privada conserva únicamente la ruta local — esa ausencia de ruta *es* la seguridad. 4. Lance una instancia micro de EC2 en cada subred (la pública con IP pública, la privada sin ella). 5. La prueba: SSH a la instancia pública — funciona. Intente la dirección de la instancia privada desde su laptop — se cuelga para siempre, y ahora usted puede decir con precisión por qué. Luego haga SSH *desde* la instancia pública *hacia* la privada (es alcanzable desde dentro de la VPC): la caja pública está actuando como un **bastion host** (bastión), un patrón estándar de la industria que usted acaba de descubrir construyéndolo. 6. Concepto extra para investigar y anotar: un **NAT gateway** permite que las máquinas privadas llamen *hacia afuera* (para actualizaciones) mientras siguen siendo inalcanzables desde fuera — pero factura por hora, así que lea sobre él, no lo construya.

**Lo que aprendió:** CIDR, subredes, tablas de rutas, internet gateways, la división público/privado, bastion hosts — el punto exacto de redes de toda descripción de puesto cloud, como memoria muscular.

**Teardown:** termine primero ambas instancias, luego borre la VPC (que arrastra consigo subredes, tablas de rutas y el gateway). Verifique en EC2 que nada diga “running”.

### Lab 4 (Semana 10): RDS — una base de datos que no hay que cuidar como a un bebé

**RDS (Relational Database Service)** es una base de datos gestionada: AWS opera el motor de base de datos (PostgreSQL, MySQL...) y se encarga de respaldos, parcheo y failover, mientras usted es dueño de los datos y las consultas. “Servicio gestionado” es el trato central de la nube — ceder algo de control a cambio de mucho trabajo no diferenciado — y este lab es donde usted siente ese trato.

**Objetivo:** lanzar una base de datos PostgreSQL en una subred privada, conectarse a ella desde una instancia EC2, y entender los respaldos y el multi-AZ — y luego desmontarla pronto, porque RDS es el lab más fácil de dejar corriendo por accidente.

**Esquema de pasos:** 1. Reconstruya rápido la VPC del Lab 3 (segunda construcción de memoria — esto es repetición espaciada deliberada), añadiendo una segunda subred *privada* en otra AZ, porque RDS requiere un **subnet group** (grupo de subredes) que abarque dos AZ. 2. Consola — RDS — Create database — PostgreSQL — **plantilla Free tier** (esto preselecciona `db.t3.micro/db.t4g.micro`, una sola AZ). Configure la contraseña maestra. Colóquela en su VPC, **Public access: No**, en un security group que permita el puerto 5432 *solo desde el security group del servidor web* — una regla que referencia a otra regla en lugar de una IP. Elegante, y estándar. 3. Lance una micro EC2 en la subred pública, instale el cliente `postgresql`, y conéctese: `psql -h <rds-endpoint> -U postgres`. El **endpoint** es un nombre DNS, no una IP — AWS puede mover la máquina subyacente, y el nombre la sigue. (El Módulo 2 ya está pagando la renta.) 4. En el prompt de `psql`: cree una tabla, inserte tres filas, consúltelas de vuelta. Hoy no necesita profundidad en SQL; necesita haberlo tocado. 5. Recorra, sin habilitar: la configuración de **respaldos automatizados** (restauración a un punto en el tiempo a partir de snapshots nocturnos + logs), y la opción **Multi-AZ** (una copia en espera en vivo en otra AZ con **failover** automático — aproximadamente el doble de costo, y por eso prod dice sí y los labs dicen no). Tome un **snapshot** manual, encuéntralo en la consola, entienda que podría restaurar un clon desde él. 6. Pregunta de diario que vale diez minutos: ¿qué, exactamente, está haciendo AWS por usted aquí que de otro modo usted haría a las 2 a.m.? (Parcheo, respaldos, failover, hardware.) Esa respuesta es la respuesta de entrevista sobre servicios gestionados.

**Lo que aprendió:** bases de datos gestionadas, subnet groups, reglas de security group a security group, endpoints, snapshots, multi-AZ/failover — más una demostración en vivo de pagar con control para comprar confiabilidad.

**Teardown:** borre la instancia RDS (rechace el snapshot final por ser un lab; note que prod sí tomaría uno), borre el snapshot manual (¡los snapshots facturan por almacenamiento!), termine la EC2, borre la VPC. Revise su tablero de facturación al día siguiente — leerlo semanalmente es un hábito de la Etapa 3 que empieza ahora.

## Lab 5 (Semanas 11–12): IAM — quién puede hacer qué

**IAM (Identity and Access Management, gestión de identidades y accesos)** decide qué personas y qué programas pueden hacer qué a qué recursos. No factura nada, no aprovisiona nada — y es el servicio más auditado, más sondeado en entrevistas y más implicado en brechas de todo AWS. Los puntos de seguridad de las descripciones de puesto (“least-privilege access controls”, “IAM policies”) significan este lab.

**Objetivo:** crear usuarios, grupos, políticas y — el crucial — un **rol**, e interiorizar el mínimo privilegio sintiendo cómo AWS le niega el acceso.

**Esquema de pasos:** 1. Conceptos primero, cinco minutos: un **usuario** es una identidad para un humano o un programa; un **grupo** agrupa usuarios; una **política (policy)** es un documento JSON que otorga permisos (“allow `s3:GetObject on arn:aws:s3:::my-bucket/*`”); un **rol** es una identidad con políticas pero *sin contraseña* que una parte de confianza *asume*

temporalmente — así es como los servidores y servicios obtienen permisos sin ningún secreto almacenado. 2. Cree una usuaria `readonly` en un grupo con la política gestionada por `AWS ReadOnlyAccess`. Inicie sesión como ella en una ventana privada del navegador: puede ver todo, pero cada botón de crear/borrar falla con una denegación explícita. Lea uno de esos errores completo — aprender a interpretar “not authorized to perform X on Y” es una habilidad diaria del trabajo. 3. Escriba su primera política personalizada en el editor JSON: permita `s3:ListBucket` y `s3:GetObject` sobre un bucket específico. Adjúntela a una usuaria nueva; verifique que puede leer ese bucket y *nada más*. Acaba de implementar el mínimo privilegio, no solo definirlo. 4. El rol, la recompensa: cree un rol para EC2 con `AmazonS3ReadOnlyAccess`, lance una instancia micro con ese rol adjunto, entre por SSH y ejecute `aws s3 ls` — funciona **sin ninguna access key en la máquina**. La instancia está asumiendo el rol y recibiendo credenciales de corta duración automáticamente. Este es el patrón de seguridad más importante de AWS: *roles para las máquinas, nunca claves en las máquinas*. Dígalo dos veces. 5. Audite su propia cuenta como un profesional: ¿root tiene MFA (Semana 3)? ¿Alguna access key con más de 90 días? ¿Algún usuario tiene más permisos de los que usa? Ejecute el **credential report** de IAM y léalo. Este ritual de diez minutos, mensual, es la lista de higiene de IAM de la Etapa 3 naciendo.

**Lo que aprendió:** usuarios/grupos/políticas/roles, leer y escribir JSON de políticas, mínimo privilegio comprobado por experimento, roles-no-claves, y su primera auditoría de seguridad.

**Teardown:** borre los usuarios de prueba y sus credenciales, termine la instancia, quédese con el conocimiento del rol para siempre.

### Vocabulario de la Etapa 1 (una tabla, los cinco labs):

| Término                                          | Definición                                                                                                                 |
|--------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------|
| Región / Availability Zone                       | Grupo geográfico de centros de datos de AWS / un centro de datos aislado dentro de él. Los sistemas serios abarcan dos AZ. |
| EC2 / instancia                                  | El servicio de alquiler de máquinas virtuales / un servidor alquilado.                                                     |
| AMI                                              | Amazon Machine Image — el disco plantilla desde el que arranca una instancia.                                              |
| Tipo de instancia                                | El tamaño/especificación que usted eligió (t3.micro = diminuta, Free Tier).                                                |
| Security group                                   | Firewall por recurso: qué puertos, desde qué orígenes.                                                                     |
| Par de llaves                                    | Las llaves SSH privada/pública para alcanzar su instancia.                                                                 |
| S3 / bucket / objeto                             | Almacenamiento de objetos / un contenedor con nombre / un archivo almacenado.                                              |
| Clase de almacenamiento / regla de ciclo de vida | Nivel precio-velocidad para objetos / regla automática que los mueve a niveles más baratos con la edad.                    |

| Término                                 | Definición                                                                                                                                     |
|-----------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------|
| Bucket policy                           | Documento JSON que declara quién puede hacer qué a un bucket. Los públicos salen en los titulares.                                             |
| AWS CLI / access key                    | La interfaz de terminal hacia AWS / las credenciales programáticas para ella (trátelas como una contraseña).                                   |
| VPC / subred                            | Su porción de red privada / un bloque más pequeño de ella que vive en una AZ, pública o privada.                                               |
| CIDR                                    | Notación de bloques de direcciones:<br>10.0.0.0/16 = “los primeros 16 bits fijos, 65,536 direcciones más”.                                     |
| Internet gateway / tabla de rutas       | La puerta de la VPC hacia internet / las reglas que deciden a dónde se envía el tráfico. Sin ruta = inalcanzable = seguro.                     |
| NAT gateway                             | Permite que las máquinas privadas llamen hacia afuera sin ser alcanzables desde fuera. Factura por hora — conózcalo, no lo deje ocioso.        |
| Bastion host                            | La máquina pública endurecida a través de la cual usted hace SSH para llegar a las privadas.                                                   |
| RDS / endpoint                          | Servicio de base de datos relacional gestionada / el nombre DNS al que usted se conecta.                                                       |
| Snapshot / Multi-AZ / failover          | Copia en un punto en el tiempo / copia en espera en vivo en una segunda AZ / el cambio automático hacia ella.                                  |
| Usuario / grupo / política / rol de IAM | Identidad / conjunto de identidades / concesión de permisos en JSON / identidad asumible sin contraseña — cómo las máquinas obtienen permisos. |
| Mínimo privilegio                       | La regla de oro: los permisos mínimos que hacen el trabajo, nada más.                                                                          |
| Servicio gestionado                     | AWS opera el trabajo no diferenciado (parcheo, respaldos, failover); usted conserva los datos y las decisiones.                                |

### Videos de esta etapa:

| Video                | Canal          | Duración | Enlace                          |
|----------------------|----------------|----------|---------------------------------|
| Getting Started with | AWS Developers | 27 min   | <a href="https://">https://</a> |

| Video                                 | Canal                                                                   | Duración | Enlace                                                                                                |
|---------------------------------------|-------------------------------------------------------------------------|----------|-------------------------------------------------------------------------------------------------------|
| EC2                                   | (oficial)                                                               |          | <a href="https://www.youtube.com/watch?v=nJ-djerESW0">www.youtube.com/watch?v=nJ-djerESW0</a>         |
| Introduction to Amazon S3             | Amazon Web Services (oficial)                                           | ~5 min   | <a href="https://www.youtube.com/watch?v=ecv-19sYL3w">https://www.youtube.com/watch?v=ecv-19sYL3w</a> |
| AWS Networking Basics — VPC & Subnets | KodeKloud (vuelva a verlo después del Lab 3 — aterrizas distinto ahora) | ~30 min  | <a href="https://www.youtube.com/watch?v=QM63dyA_4Pc">https://www.youtube.com/watch?v=QM63dyA_4Pc</a> |
| The AWS Shared Responsibility Model   | Digital Cloud Training                                                  | 4 min    | <a href="https://www.youtube.com/watch?v=ESPBBEK-cvo">https://www.youtube.com/watch?v=ESPBBEK-cvo</a> |

**Dígalos en voz alta:** “It’s in a private subnet; there’s no route to the internet gateway.” (Está en una subred privada; no hay ruta al internet gateway.) · “The security group only allows 5432 from the web tier’s security group.” (El security group solo permite 5432 desde el security group de la capa web.) · “The instance uses a role — there are no keys on the box.” (La instancia usa un rol — no hay claves en la máquina.) · “Terminate it when you’re done; nothing idles in this account.” (Termínala cuando acabe; nada queda ocioso en esta cuenta.)

**Hito — fin de la Etapa 1:** la construcción de desafío. En una sola sesión, de memoria: VPC con subredes pública/privada → servidor web EC2 (público, con rol adjunto, sirviendo una página) → RDS PostgreSQL (privada, alcanzable solo desde el servidor web) → un bucket de S3 que la instancia puede leer mediante su rol — y luego desmontar todo limpio. Menos de tres horas significa que está listo para la Etapa 2. Además: **empiece la preparación del CLF-C02 ahora** (el curso completo de freeCodeCamp — <https://www.youtube.com/watch?v=7HKot-brXFE> — a 1.25× como repaso; la edición clásica de 14 horas es <https://www.youtube.com/watch?v=NhDYbskXRgc>), y presente el examen **AWS Cloud Practitioner** alrededor de la Semana 14–16.

## ETAPA 2 — AUTOMATIZACIÓN (Semanas 13–20)

**La idea de esta etapa:** todo lo que hizo con clics en la Etapa 1, ahora lo hará con código. El insulto suave de la industria para hacer clics en la consola es **ClickOps**; la frase de las descripciones de puesto para lo que lo reemplaza es “*design, develop, and deploy cloud infrastructure using infrastructure as code*” (diseñar, desarrollar y desplegar infraestructura cloud usando infraestructura como código). Esta etapa es la diferencia entre alguien que ha *usado* AWS y alguien a quien una empresa le pagará por *operarlo*.

## Módulo 6 (Semanas 13-15): Scripting — Bash y Python para ops, más Git

El **scripting en Bash** es poner los comandos del Módulo 1 en un archivo para que la computadora los repita a la perfección. Aprenda, en este orden: **variables** (`NAME="web-1"`), **sustitución de comandos** (`TODAY=$(date +%F)`), **if** (`if [ -f "$FILE" ]; then ... fi`), **bucles** (`for f in *.log; do gzip "$f"; done`), **códigos de salida** (todo comando devuelve 0 en caso de éxito, distinto de cero en caso de falla; `&&` encadena sobre el éxito — los scripts toman decisiones con esto), y la lectura de **argumentos** (`$1`, `$2`). Eso es el 90% del Bash real de ops. Escriba su primer script genuinamente útil esta semana: `backup.sh` — empaquete un directorio con `tar`, nombre el archivo con la fecha de hoy, hágale `aws s3 cp` a un bucket, borre los archivos locales de más de 7 días, y haga `echo` de una línea de éxito/falla a un archivo de log. Ese único script es variables, sustitución, condicionales, códigos de salida y la CLI en un solo artefacto — y un punto en su currículum (“automaticé respaldos a S3”).

**Python** es el hermano mayor de Bash: mejor para todo lo que involucre lógica, datos o hablar con APIs. Usted necesita un Python de ops funcional, no profundidad de ingeniería de software: variables y tipos, listas y diccionarios, `if/for`, funciones, leer/escribir archivos, manejo de errores con `try/except`, e instalar librerías con `pip`. Luego conozca **boto3**, la librería de AWS para Python: `import boto3; ec2 = boto3.client("ec2"); ec2.describe_instances()` — de repente su conocimiento de la Etapa 1 es programable. Escriba `audit.py`: liste cada instancia EC2 de la cuenta con su nombre, tipo, estado y hora de lanzamiento, e imprima una línea de advertencia para cualquier cosa corriendo más de 24 horas. Felicidades — acaba de escribir una herramienta real de control de costos por la que los empleadores pagan.

**Git** es el sistema de control de versiones en el que vive todo el trabajo de ingeniería. Conceptos: un **repositorio** (una carpeta cuya historia completa queda registrada), un **commit** (una instantánea guardada con un mensaje), un **branch** (rama; una línea de trabajo paralela), un **remote** (la copia en GitHub) y un **pull request** (pedir que una rama sea revisada y fusionada). El ciclo diario son seis comandos: `git init`/`git clone`, `git status`, `git add`, `git commit -m "message"`, `git push`, `git pull`. Cree una cuenta de GitHub hoy, cree un repo llamado `cloud-journey`, y de ahora en adelante **cada script y configuración que escriba en este curso se hace commit ahí**. En diez meses, ese historial de commits es prueba pública y fechada de todo lo que este curso afirma que usted puede hacer.

### Vocabulario:

| Término              | Definición                                                                                            |
|----------------------|-------------------------------------------------------------------------------------------------------|
| Script               | Un archivo de comandos ejecutados de arriba abajo — el átomo de la automatización.                    |
| Variable / argumento | Un valor con nombre en un script / un valor pasado al ejecutarlo ( <code>\$1</code> ).                |
| Código de salida     | La señal de éxito (0) o falla (distinto de cero) de todo comando — cómo los scripts toman decisiones. |
| Cron                 | El programador de Linux: ejecutar un script cada noche a las 2 a.m., para siempre.                    |

| Término                        | Definición                                                                                                          |
|--------------------------------|---------------------------------------------------------------------------------------------------------------------|
| Python / pip                   | El lenguaje de programación favorito del mundo ops / su instalador de paquetes.                                     |
| boto3                          | La librería de Python para manejar AWS — la consola, como código.                                                   |
| try/except                     | El “intenta esto; si falla, haz esto otro” de Python — cómo los scripts fallan con gracia.                          |
| Git / repositorio / commit     | Control de versiones / la historia registrada de un proyecto / una instantánea guardada con un mensaje.             |
| Branch / merge                 | Una línea de trabajo paralela / traerla de vuelta a la línea principal.                                             |
| GitHub / remote / pull request | El sitio de alojamiento / la copia en la nube de su repo / una solicitud revisada y discutida de fusionar una rama. |
| README                         | El documento de portada de un repo que explica qué es y cómo ejecutarlo. Los reclutadores los leen.                 |

### Videos de este módulo:

| Video                                    | Canal        | Duración                                   | Enlace                                                                                                |
|------------------------------------------|--------------|--------------------------------------------|-------------------------------------------------------------------------------------------------------|
| Learn Python — Full Course for Beginners | freeCodeCamp | ~4.5 h (véalo a lo largo de las 3 semanas) | <a href="https://www.youtube.com/watch?v=rfscVS0vtbw">https://www.youtube.com/watch?v=rfscVS0vtbw</a> |

**Dígalo en voz alta:** “I’ll script it — it’ll be wrong once, then never again.” (Lo haré con un script — se equivocará una vez, y luego nunca más.) · “Check the exit code before the next step runs.” (Revise el código de salida antes de que corra el siguiente paso.) · “Commit it with a message that says why, not what.” (Hágale commit con un mensaje que diga por qué, no qué.) · “It’s in the repo — pull the latest.” (Está en el repo — baje lo último con pull.)

**Ejercicios:** (1) `backup.sh`, como se especifica arriba, programado cada noche con cron en una instancia EC2 de lab. (2) `audit.py`, como se especifica arriba. (3) Ambos con commit en `cloud-journey` con un README que explique cada uno. (4) Rompa su propio script a propósito (una ruta equivocada, un bucket faltante) y hágalo fallar *ruidosa y claramente* — el manejo de errores es un lenguaje de amor de ops.

**Hito:** su script de respaldo sobrevive a ser ejecutado dos veces seguidas y con una carpeta faltante (sin crash, mensaje claro); su auditoría en Python corre contra su cuenta real; su repo muestra una semana de commits.

## Módulo 7 (Semanas 16-17): Terraform — infraestructura como código

**La gran idea:** en lugar de crear recursos a punta de clics, usted escribe archivos de texto que *declaran* lo que debe existir — “una VPC, dos subredes, una instancia, estas etiquetas” — y **Terraform** hace que AWS coincida con el archivo. Declarativo, no imperativo: usted enuncia el destino, no las vueltas del camino. Por qué toda descripción de puesto lo pide: los archivos viven en Git, así que la infraestructura se vuelve **revisada** (pull requests antes de los cambios), **repetible** (el mismo archivo construye dev, staging y prod de forma idéntica), **reversible** (se revierte revirtiendo el commit) y **auditable** (el historial dice quién cambió qué, cuándo, por qué). La consola es un taller; Terraform es una fábrica.

**El ciclo central que ejecutará cientos de veces:** `terraform init` (descarga el **provider** de AWS — el plugin que traduce sus archivos en llamadas a la API) — `terraform plan` (un ensayo en seco que imprime exactamente qué se crearía/cambiaría/destruiría — lea cada línea; los ingenieros que aplican sin leer los plans causan caídas) — `terraform apply` (hacerlo realidad) — `terraform destroy` (deshacerlo — el teardown como un solo comando, lo cual a estas alturas le parecerá hermoso). Terraform registra lo que construyó en un **state file** (archivo de estado) — su memoria de la realidad; piérdalo o edite la realidad a mano a sus espaldas (“drift”, deriva) y vendrá el dolor. Los recursos se declaran en **HCL**, un lenguaje de configuración legible:

```
resource "aws_instance" "web" {
  ami          = "ami-0abcdef1234567890"
  instance_type = "t3.micro"
  tags = { Name = "web-1", Project = "lab" }
}
```

**El lab (esto es el módulo):** reconstruya el desafío de la Etapa 1 en Terraform, incrementalmente. (1) Inicialice un repo `terraform-labs`; escriba la configuración de solo un bucket S3 etiquetado; plan, lea, apply, revise la consola — ahí está; destroy. (2) Haga crecer la configuración: VPC, dos subredes, internet gateway, tabla de rutas — la construcción del Lab 3, ahora en ~60 líneas de HCL. (3) Añada el security group y una instancia EC2 con `variables.tf` (región, tamaño de instancia) y `outputs.tf` (imprimir la IP pública tras el apply). (4) Cambie el tipo de instancia en el archivo y vuelva a aplicar — observe a Terraform calcular la diferencia y cambiar *solo eso*. Este comportamiento guiado por diffs es toda la magia. (5) Destruya todo, confirme que la consola está vacía, haga commit y etiquete el repo `v1.0`. La documentación de referencia vive en <https://developer.hashicorp.com/terraform> — guárdela en marcadores; leer la documentación del provider es la mitad del trabajo real con Terraform.

### Vocabulario:

| Término                      | Definición                                                                                              |
|------------------------------|---------------------------------------------------------------------------------------------------------|
| IaC (Infrastructure as Code) | Declarar la infraestructura en archivos de texto versionados que una herramienta convierte en realidad. |
| Terraform / HCL              | La herramienta de IaC multi-cloud dominante / su lenguaje de configuración.                             |
| Provider                     | El plugin que traduce sus archivos en                                                                   |

| Término                             | Definición                                                                                                                             |
|-------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------|
| <code>plan / apply / destroy</code> | llamadas a la API de una nube (aquí: AWS).<br>El diff de ensayo / hacerlo realidad / deshacerlo todo. Lea cada plan.                   |
| State file                          | El registro de Terraform de lo que construyó — su memoria de la realidad. Protéjalo.                                                   |
| Drift                               | La realidad cambió a espaldas de Terraform (alguien hizo clic). El enemigo.                                                            |
| Variable / output / módulo          | Una entrada de configuración / un resultado impreso (una IP, una URL) / un trozo de configuración empaquetado y reutilizable.          |
| Declarativo vs imperativo           | Enunciar el destino vs guionar las vueltas. Terraform es declarativo.                                                                  |
| CloudFormation                      | El servicio de IaC propio de AWS — la misma idea, solo AWS. Las descripciones de puesto aceptan cualquiera; aprenda Terraform primero. |

### Videos de este módulo:

| Video                                                     | Canal               | Duración | Enlace                                                                                                |
|-----------------------------------------------------------|---------------------|----------|-------------------------------------------------------------------------------------------------------|
| Terraform explained in 15 mins                            | TechWorld with Nana | 18 min   | <a href="https://www.youtube.com/watch?v=l5k1ai_GBDE">https://www.youtube.com/watch?v=l5k1ai_GBDE</a> |
| Terraform Course — Automate your AWS cloud infrastructure | freeCodeCamp        | ~2.5 h   | <a href="https://www.youtube.com/watch?v=SLB_c_ayRMo">https://www.youtube.com/watch?v=SLB_c_ayRMo</a> |

**Dígalo en voz alta:** “Nothing changes in prod except through Terraform.” (Nada cambia en prod excepto a través de Terraform.) · “Show me the plan before you apply.” (Muéstreme el plan antes de aplicar.) · “That was clicked in manually — that’s drift; let’s import it or remove it.” (Eso se creó a mano con clics — eso es drift; vamos a importarlo o eliminarlo.) · “It’s in the module; reuse it, don’t rewrite it.” (Está en el módulo; reutilícelo, no lo reescriba.)

**Ejercicios:** el lab de arriba, más: (1) cree drift deliberadamente — cambie a mano la etiqueta de la instancia en la consola, ejecute `terraform plan`, y observe a Terraform notarlo; anote en el diario qué propuso. (2) Escriba un párrafo en lenguaje llano en el README del repo: “por qué esto vence a los clics”. Si no puede escribirlo, todavía no lo tiene.

**Hito:** desde un directorio vacío, usted puede levantar el stack de VPC + instancia con `init/plan/apply` y eliminarlo con `destroy`, explicando en voz alta qué está haciendo cada comando — sin notas.

## Módulo 8 (Semanas 18–20): CI/CD, Docker y un primer vistazo a Kubernetes

**CI/CD** (Continuous Integration / Continuous Delivery, integración y entrega continuas) es la cinta transportadora robótica conectada a su repo de Git: en cada push, automáticamente verifica, prueba, construye y — cuando está configurado — despliega su cambio. El objetivo son lanzamientos pequeños, frecuentes y aburridos en lugar de raros y aterradores. **GitHub Actions** es el sistema de CI/CD integrado en GitHub: un **workflow** es un archivo YAML en `.github/workflows/` que dice “en cada push, ejecuta estos pasos en un **runner** fresco (una VM temporal)”. Lab: añada a `terraform-labs` un workflow que ejecute `terraform fmt -check` y `terraform validate` en cada push — suba un archivo deliberadamente mal formado y observe la **X** roja, arréglolo y observe la **✓** verde. Ahora tiene un pipeline; por pequeño que sea, es de la misma especie que los de toda descripción de puesto. (Segundo paso, en el Proyecto 1: un workflow que ejecute `terraform plan` en los pull requests, para que los cambios de infraestructura propuestos muestren su diff en la revisión.)

**Docker** empaqueta una aplicación con todo lo que necesita en un **container** (contenedor) — una lonchera sellada que corre de forma idéntica en su laptop, en EC2 y en cualquier otro lugar, poniendo fin a la antigua plaga del “en mi máquina funciona”. Una **imagen** es la receta congelada (construida desde un **Dockerfile**, un archivo de texto corto de pasos de construcción: empieza FROM una imagen base, COPY su app adentro, defina el comando de arranque); un **container** es una instancia en ejecución de una imagen; un **registry** (Docker Hub, o **ECR** de AWS) es donde las imágenes se suben (push) y bajan (pull). Los contenedores *no* son VMs: comparten el kernel Linux del host, así que arrancan en más o menos un segundo y usted puede correr docenas en una instancia micro. Lab: en una instancia EC2, instale Docker, `docker run hello-world`, luego `docker run -d -p 80:8080 <a sample web image>` y navegue hacia él; luego escriba un Dockerfile de cinco líneas que sirva su página estática del Lab 2 desde una imagen base `nginx`, constrúyalo, ejecútelo y súbalo a Docker Hub. Aprenda los verbos diarios: `docker ps`, `docker logs`, `docker exec -it <id> bash` (un shell *dentro* del contenedor), `docker stop`.

**Kubernetes (K8s)** — para este curso, una habilidad a nivel de lectura, etiquetada con honestidad. Cuando una empresa corre cientos de contenedores en muchas máquinas, algo debe programarlos, reiniciar los caídos, escalar los ocupados y enrutar el tráfico entre ellos: ese orquestador es Kubernetes. Aprenda el mapa de conceptos ahora — un **cluster** de **nodos** ejecuta **pods** (la unidad desplegable más pequeña, normalmente un contenedor); un **deployment** declara “mantén 3 réplicas de este pod vivas” y el cluster lo hace verdad continuamente (declarativo otra vez — Kubernetes es la filosofía de Terraform aplicada al software en ejecución); un **service** da a los pods una dirección estable. La oferta gestionada de AWS es **EKS**. Las descripciones de puesto junior quieren exactamente esta alfabetización más fluidez en Docker; la habilidad real de operar K8s (y la certificación CKA) es una meta fuerte para el Año 2, no para el Mes 5. Este curso le dice con honestidad cuál es cuál.

### Vocabulario:

| Término | Definición                                                                    |
|---------|-------------------------------------------------------------------------------|
| CI/CD   | El pipeline automatizado que verifica, construye, prueba y envía cada cambio. |

| Término                            | Definición                                                                                                                         |
|------------------------------------|------------------------------------------------------------------------------------------------------------------------------------|
| GitHub Actions / workflow / runner | El CI/CD integrado de GitHub / el archivo YAML que define un pipeline / la VM temporal que lo ejecuta.                             |
| YAML                               | El formato de configuración basado en indentación de los pipelines y de Kubernetes. La indentación es significado — cuídela.       |
| Docker / imagen / container        | El kit de herramientas de contenedores / la receta congelada por capas / una instancia en ejecución de ella.                       |
| Dockerfile                         | El archivo de texto corto de pasos que construye una imagen.                                                                       |
| Registry / ECR                     | Donde las imágenes se suben y bajan / el registry de AWS.                                                                          |
| Kubernetes (K8s) / cluster / nodo  | El orquestador de contenedores / su grupo de máquinas / una máquina en él.                                                         |
| Pod / deployment / service         | La unidad desplegable más pequeña / “mantén N réplicas vivas”, aplicado continuamente / una dirección estable delante de los pods. |
| EKS                                | El plano de control de Kubernetes gestionado de AWS.                                                                               |
| Rollback                           | Volver rápido a la versión anterior cuando un lanzamiento se porta mal — la red de seguridad que CI/CD hace rutinaria.             |

### Videos de este módulo:

| Video                                 | Canal               | Duración | Enlace                                                                                                |
|---------------------------------------|---------------------|----------|-------------------------------------------------------------------------------------------------------|
| DevOps CI/CD Explained in 100 Seconds | Fireship            | 2 min    | <a href="https://www.youtube.com/watch?v=scEDHsr3APg">https://www.youtube.com/watch?v=scEDHsr3APg</a> |
| What is DevOps? REALLY understand it  | TechWorld with Nana | ~15 min  | <a href="https://www.youtube.com/watch?v=OyWAtQ6wYNM">https://www.youtube.com/watch?v=OyWAtQ6wYNM</a> |
| Docker in 100 Seconds                 | Fireship            | 2 min    | <a href="https://www.youtube.com/watch?v=Gjnup-PuquQ">https://www.youtube.com/watch?v=Gjnup-PuquQ</a> |
| Docker Tutorial for Beginners [FULL   | TechWorld with Nana | 3 h      | <a href="https://www.youtube.com/">https://www.youtube.com/</a>                                       |

| Video                           | Canal               | Duración | Enlace                                      |
|---------------------------------|---------------------|----------|---------------------------------------------|
| COURSE in 3 Hours]              |                     |          | watch?v=3c-iBn73dDE                         |
| Kubernetes explained in 15 mins | TechWorld with Nana | ~16 min  | https://www.youtube.com/watch?v=VnvRFRk_51k |

**Dígalo en voz alta:** “Don’t merge until the pipeline is green.” (No fusione hasta que el pipeline esté en verde.) · “It’s containerized — same image in dev and prod.” (Está en contenedores — la misma imagen en dev y en prod.) · “Exec into the container and check its logs.” (Entre al contenedor con exec y revise sus logs.) · “K8s keeps three replicas up; kill one and watch it come back.” (K8s mantiene tres réplicas arriba; mate una y véala volver.)

**Ejercicios:** los dos labs de arriba, más: (1) escriba en el diario una respuesta de un párrafo a “VM vs container vs serverless — ¿cuándo cada uno?” (Serverless in 100 Seconds de Fireship — [https://www.youtube.com/watch?v=W\\_VV2Fx32\\_Y](https://www.youtube.com/watch?v=W_VV2Fx32_Y) — completa la tercera opción). (2) Termine la instancia del lab de Docker; confirme que sus imágenes siguen vivas en el registry — note que el *artefacto* ahora sobrevive al *servidor*, que es toda la cosmovisión moderna del despliegue en una sola observación.

**Hito — fin de la Etapa 2:** su GitHub muestra: un repo de scripts de ops en Bash + Python que funcionan, un repo de Terraform que construye y destruye un stack real, un workflow de Actions en verde, y un Dockerfile subido a un registry — y usted puede explicar cada archivo en una entrevista. Ese perfil de GitHub ya no es el de un estudiante. Es el de un ingeniero junior.

## ETAPA 3 — OPERACIONES (Semanas 21-28)

**La idea de esta etapa:** construir sistemas lo hace contratable; *operarlos* es el trabajo real. Los puntos de descripción de puesto que esta etapa responde son la mitad operativa — “monitor infrastructure health” (monitorear la salud de la infraestructura), “participate in incident response, including log analysis” (participar en la respuesta a incidentes, incluido el análisis de logs), “identifying, analyzing, and resolving infrastructure vulnerabilities” (identificar, analizar y resolver vulnerabilidades de infraestructura), “manage cloud costs” (gestionar los costos cloud). La semana de un ingeniero cloud es mayormente esta etapa.

### Módulo 9 (Semanas 21-24): Monitoreo, logging e incidentes

**Monitoreo — saber antes que los usuarios.** CloudWatch es el servicio de observabilidad integrado de AWS. Tres primitivas: **métricas** (números en el tiempo — % de CPU, % de disco, conteo de solicitudes, conteo de errores), **alarmas** (una regla que vigila una métrica: “si CPU > 80% durante 5 minutos — notificar”) y **dashboards** (tableros; métricas organizadas en una pantalla). Las notificaciones fluyen a través de **SNS** (Simple Notification Service — un topic al que usted publica, y los suscriptores reciben correo/aviso). El oficio está en *qué* alarmar: avise a un humano solo por lo que necesita un humano, o la gente aprende a ignorar el localizador

(**alert fatigue**, fatiga de alertas — el modo de falla que ha precedido a muchas caídas famosas). Las cuatro señales doradas que vale la pena memorizar: **latencia, tráfico, errores, saturación** — qué tan lento, qué tan ocupado, qué tan roto, qué tan lleno.

**Logging — saber por qué.** Las métricas dicen que algo anda mal; los logs dicen qué pasó. **CloudWatch Logs** los centraliza: un **agente** en cada instancia envía archivos como `/var/log/nginx/access.log` a **log groups**, donde **Logs Insights** le permite consultar entre máquinas (“cuenta las respuestas 5xx por minuto de la última hora”). La centralización importa porque los servidores ahora son desechables (usted lo demostró en el Módulo 8) — los logs deben sobrevivir a las máquinas que los escribieron.

**Lab (Semanas 21–22):** levante un pequeño servidor web monitoreado, todo en Terraform (su stack del Módulo 7, extendido): EC2 + nginx + el agente de CloudWatch; un topic de SNS que le envía correos; una alarma por CPU alta y otra por falla del status check de la instancia. Luego atáquese a sí mismo: entre por SSH y ejecute un quemador de CPU (`yes > /dev/null &` unas cuantas veces); observe la métrica subir, la alarma dispararse, el correo llegar; mate los procesos; observe la recuperación. Luego consulte sus propios logs de acceso en Logs Insights. Acaba de presenciar el ciclo completo detectar—notificar—diagnosticar—resolver en un sistema que usted construyó.

**Respuesta a incidentes — la mitad humana.** Un **incidente** es un “el sistema no está bien” no planificado; los niveles de **severidad** (sev-1 = clientes caídos, todos a bordo) fijan la escala de la respuesta. El ciclo profesional: **detectar** (la alarma, no un correo de un cliente, idealmente) → **triaje** (qué tan grave, a quién se necesita) → **mitigar** (detener la hemorragia *primero* — rollback, reinicio, failover; la causa raíz viene después) → **resolver** → **post-mortem**: una revisión escrita *sin culpas* de qué pasó, por qué, y qué prevendrá una repetición. Sin culpas no es blandura; es ingeniería: las personas castigadas esconden información, y la información escondida causa caídas repetidas. **On-call** (guardia) es la rotación de quién lleva el localizador; las descripciones de puesto lo mencionan, los entrevistadores preguntan por él, y su respuesta honesta después de este módulo es “lo he simulado y conozco el ciclo”.

**Runbooks — el punto de documentación, hecho concreto.** Un runbook es una receta paso a paso para una situación operativa, escrita para que una persona estresada a las 3 a.m. pueda seguirla: síntomas → verificaciones (comandos exactos) → arreglos (comandos exactos) → escalamiento (a quién despertar si no funciona). **Lab (Semanas 23–24):** escriba dos runbooks en su repo — “servidor web caído” y “disco llenándose” — y luego *ensáyelos*: rompa la cosa, siga su propio documento al pie de la letra, y corrija cada paso que resultó vago. Luego ejecute un **game day** completo: haga que un amigo (o una IA) rompa su stack de lab en secreto; a usted le llega el aviso, diagnostica desde métricas y logs, mitiga, y escribe el post-mortem en su diario. Ese post-mortem es una historia de entrevista, y de las buenas.

## Vocabulario:

| Término                      | Definición                                                                              |
|------------------------------|-----------------------------------------------------------------------------------------|
| CloudWatch                   | El servicio de monitoreo de AWS: métricas, alarmas, dashboards, logs.                   |
| Métrica / alarma / dashboard | Un número en el tiempo / una regla que se dispara sobre él / una pantalla con varios de |

| Término                     | Definición                                                                                                      |
|-----------------------------|-----------------------------------------------------------------------------------------------------------------|
| SNS                         | ellos.<br>El servicio de notificaciones al que publican las alarmas — correo, SMS, localizador.                 |
| Señales doradas             | Latencia, tráfico, errores, saturación — los cuatro números que describen la salud de cualquier servicio.       |
| Alert fatigue               | Demasiados avisos no accionables — avisos ignorados → avisos reales perdidos. El enemigo del diseño de alarmas. |
| Log group / Logs Insights   | Donde aterrizan los logs centralizados / el lenguaje de consulta sobre ellos.                                   |
| Incidente / severidad       | Una degradación no planificada / su gravedad clasificada (sev-1 = la peor).                                     |
| Triage / mitigar / resolver | Evaluar rápido / detener la hemorragia primero / arreglar de verdad.                                            |
| Post-mortem                 | La revisión escrita sin culpas: qué, por qué, qué previene una repetición.                                      |
| Runbook                     | La receta a prueba de las 3 a.m. para una situación: síntomas, verificaciones, arreglos, escalamiento.          |
| On-call / game day          | La rotación del localizador / un incidente de práctica deliberado.                                              |
| MTTR                        | Mean time to recovery (tiempo medio de recuperación) — la métrica de ops que los equipos maduros optimizan.     |

**Dívalo en voz alta:** “Did we find out from the alarm or from a customer?” (¿Nos enteramos por la alarma o por un cliente?) · “Mitigate first — root cause after we’re stable.” (Mitigar primero — la causa raíz después de estar estables.) · “Is there a runbook for this? There will be by tomorrow.” (¿Hay un runbook para esto? Lo habrá para mañana.) · “What did the post-mortem conclude, and what’s the prevention item?” (¿Qué concluyó el post-mortem, y cuál es la acción de prevención?)

**Hito:** su post-mortem del game day existe, es sin culpas, y nombra una prevención concreta que usted luego implementó de verdad (una alarma añadida, un runbook corregido).

## Módulo 10 (Semanas 25–28): Operaciones de seguridad y gestión de costos

**Operaciones de seguridad — higiene, no heroísmo.** Primero el **modelo de responsabilidad compartida** (vuelva a ver: <https://www.youtube.com/watch?v=ESPBBEK-cvo>): AWS asegura la nube en sí; *usted* asegura lo que pone en ella — y la mayoría de las brechas reales son errores de configuración del cliente, no fallas de AWS. Su lista operativa de seguridad, practicada hasta el aburrimiento:

1. **Higiene de IAM** (mensual): MFA en todas partes; nada de access keys de larga vida donde un rol serviría; ejecute el credential report; borre usuarios y claves sin uso; cuestione cada \* en cada política. Usted construyó este reflejo en el Lab 5 — ahora es una entrada de calendario.
2. **Parcheo**: el software sin parches es el origen de la mayoría de las intrusiones. En sus instancias: aplique actualizaciones con un calendario (y sepa que **SSM Patch Manager** lo automatiza para toda la flota); en el mundo de los contenedores, parchear a menudo significa reconstruir la imagen desde una base actualizada — conecte eso con el Módulo 8.
3. **Respaldos — probados**: un respaldo no probado es una esperanza, no un plan. Snapshots de RDS, **versioning** de S3 (cada sobrescritura conserva la versión anterior — el control anti-descuidos y anti-ransomware). Lab: haga snapshot de su base de datos de lab, *restáurela* en una instancia nueva, verifique las filas, desmonte. Ahora sí puede decir “probado” en las entrevistas.
4. **Guardrails (barandillas)**: active **CloudTrail** (el log de auditoría de cada llamada a la API en la cuenta — quién hizo qué, cuándo) y hojee los eventos de ayer una vez; habilite la prueba gratuita de **GuardDuty** (detección automatizada de amenazas) y lea qué vigila. Conozca la frase **cifrado en reposo y en tránsito** y sepa que en AWS es mayormente una casilla respaldada por **KMS** — lo mínimo esperable, siempre activo.

**Gestión de costos — la habilidad que hace que los juniors parezcan seniors.** El punto de la descripción de puesto es textual: “*manage cloud costs through rightsizing resources, implementing auto-scaling, resource tagging*” (gestionar los costos cloud mediante ajuste de tamaño de recursos, auto-scaling y etiquetado de recursos). Cómo corre el medidor: el cómputo factura por segundo que está *encendido* (ocioso ≠ gratis — “lo dejamos corriendo” es el desperdicio clásico), el almacenamiento por GB-mes, y el **egress** (datos que *salen* de AWS) por GB mientras que los datos que entran son gratis — la famosa sorpresa de la factura. Las palancas, en el orden en que un junior puede jalarlas: **apáguelo** (sistemas de dev por la noche; su hábito de teardown, industrializado) → **rightsizing** (la mayoría de los servidores están sobredimensionados; revise el historial de CPU en CloudWatch y encoja) → **etiquete todo** (Project, Owner, Environment — gasto sin etiquetar es gasto sin responsable; imponga las etiquetas en su Terraform) → **niveles de almacenamiento** (las reglas de ciclo de vida del Lab 2) → sepa que la **capacidad reservada/Savings Plans** (compromiso de 1–3 años, ahorro de 30–70%) y **Spot** (hasta 90% de descuento, interrumpible) existen para workloads estables y por lotes respectivamente. Herramientas: **Cost Explorer** (el gasto de la cuenta, graficado — léalo semanalmente, para siempre) y **AWS Budgets** (su alerta de \$0 de la Semana 3, ahora entendida como el miembro más pequeño de una familia seria). La disciplina más amplia se llama **FinOps** — vale dos minutos: [https://www.youtube.com/watch?v=Y-c\\_xw9bHFW](https://www.youtube.com/watch?v=Y-c_xw9bHFW).

## Vocabulario:

| Término                              | Definición                                                                                              |
|--------------------------------------|---------------------------------------------------------------------------------------------------------|
| Modelo de responsabilidad compartida | AWS asegura la nube; usted asegura lo que pone en ella. La mayoría de las brechas son la segunda mitad. |
| CloudTrail                           | El log de auditoría de la cuenta: cada llamada                                                          |

| Término                               | Definición                                                                                                                             |
|---------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------|
| GuardDuty                             | a la API, por quién, cuándo.<br>La detección automatizada de amenazas de AWS sobre logs y tráfico.                                     |
| Parcheo / SSM                         | Aplicar actualizaciones de seguridad / el Systems Manager de AWS, que lo automatiza para toda la flota.                                |
| S3 versioning                         | Conservar cada versión sobrescrita — el control anti-descuidos y anti-ransomware.                                                      |
| KMS / cifrado en reposo y en tránsito | El servicio de llaves de AWS / datos codificados en disco y en la red. Siempre activo.                                                 |
| Egress                                | Datos saliendo de AWS — facturados por GB; la entrada es gratis. La sorpresa clásica de la factura.                                    |
| Rightsizing                           | Encoger recursos sobredimensionados a la necesidad medida. Dinero gratis.                                                              |
| Tagging                               | Etiquetar cada recurso con dueño/proyecto/entorno para que todo gasto sea atribuible.                                                  |
| Savings Plans / Spot                  | Compromiso de 1–3 años por 30–70% de descuento en carga estable / capacidad sobrante reclamable con hasta 90% de descuento para lotes. |
| Cost Explorer / Budgets               | Las gráficas de gasto que usted lee semanalmente / las alertas que significan que nunca se entera por la factura.                      |
| FinOps                                | La disciplina de hacer el gasto cloud visible, asignado y continuamente optimizado.                                                    |

**Dígalos en voz alta:** “Who has access to prod, and when did we last review the list?” (¿Quién tiene acceso a prod, y cuándo revisamos la lista por última vez?) · “When did we last *restore* a backup?” (¿Cuándo fue la última vez que *restauramos* un respaldo?) · “What’s untagged, and what died but is still billing?” (¿Qué está sin etiquetar, y qué murió pero sigue facturando?) · “It’s over-provisioned — the CPU history says we can halve it.” (Está sobredimensionado — el historial de CPU dice que podemos reducirlo a la mitad.)

**Ejercicios:** (1) El lab de prueba de restauración de arriba. (2) Un archivo de lista mensual de seguridad en su repo, y luego ejecútela de verdad contra su cuenta. (3) En Cost Explorer, encuentre la cosa más cara en la historia de su cuenta y explíquela en una frase de diario. (4) Añada etiquetas por defecto a cada recurso en su repo de Terraform.

**Hito — fin de la Etapa 3:** ejecute una autoauditoría completa — lista de seguridad, revisión de costos, prueba de alarmas, restauración de respaldo — y escriba el informe de una página.

Ahora usted puede hacer el *trabajo*, no solo la construcción. Lo que queda es demostrárselo a extraños: la Etapa 4.

---

## ETAPA 4 — PREPARACIÓN LABORAL (Meses 8-12)

### Módulo 11 (Meses 8-10): Tres proyectos de portafolio

Las certificaciones dicen que usted estudió; los proyectos demuestran que puede construir. Tres, cada uno en su propio repo de GitHub, cada uno con un diagrama de arquitectura, un README escrito para un gerente de contratación (qué, por qué, cómo ejecutarlo, cuánto cuesta) y un script de teardown. Construya — capture/grabe — *destruya* — el repo es el artefacto, no una factura corriendo.

#### Proyecto 1 — App web de tres capas, totalmente automatizada (la pieza central).

Terraform construye todo: VPC con subredes públicas/privadas en dos AZ; un **Application Load Balancer** (el dispositivo que reparte el tráfico — nuevo para usted, y el siguiente paso natural desde el Lab 3) delante de un **Auto Scaling Group** de instancias web (AWS añade/quita instancias según la carga — investiguelo, conéctelo); RDS en las subredes privadas; S3 para los recursos estáticos; alarmas de CloudWatch y un runbook. GitHub Actions ejecuta `terraform plan` en cada pull request y `apply` al fusionar — un pipeline de infraestructura real. Este único proyecto demuestra diez de los catorce puntos de la tabla de correspondencias; espere que cada entrevista lo recorra, y ensaye narrarlo en cinco minutos.

**Proyecto 2 — Pipeline de datos serverless (amplitud).** Sin servidores en absoluto: un archivo que aterriza en un bucket de S3 dispara una **función Lambda** (su Python, ejecutado por AWS por evento, facturado por invocación) que lo procesa — parsear un CSV, resumirlo, escribir los resultados en un segundo bucket o en una tabla de DynamoDB — con las fallas capturadas, registradas en CloudWatch y alarmadas hacia su bandeja de entrada. Desplieguelo con Terraform. Muestra Python, pensamiento orientado a eventos y amplitud más allá de EC2. Dos minutos de orientación primero: [https://www.youtube.com/watch?v=W\\_VV2Fx32\\_Y](https://www.youtube.com/watch?v=W_VV2Fx32_Y).

**Proyecto 3 — Vitrina de ops estilo producción (el diferenciador).** Tome el Proyecto 1 y opérelolo como si importara: un dashboard de CloudWatch con las señales doradas; alarmas con una política de avisos documentada; tres runbooks; un procedimiento de respaldo y restauración probado con su evidencia; una pasada de endurecimiento de seguridad (auditoría de IAM, notas de parcheo, CloudTrail activado) documentada; un análisis de costos (“este stack cuesta \$X/mes; aquí están los tres cambios que lo reducirían a la mitad”); y un post-mortem de game day. Casi ningún candidato junior tiene esto. Responde a la única pregunta que las entrevistas realmente hacen — “¿se le puede confiar producción a esta persona?” — con documentos en lugar de adjetivos.

### Módulo 12 (Meses 8-12, en paralelo): El calendario de certificaciones

Las certificaciones abren los filtros de los reclutadores y estructuran su estudio. La ruta estándar de 2026 para este rol, con preparación realista al ritmo de este curso — note cómo

cada examen cae justo después de las etapas del curso que enseñan su contenido, y por eso estos tiempos son más cortos que los de internet:

| Orden | Certificación                                      | Qué demuestra                                                                                             | Tiempo de preparación desde donde usted estará                                                                                                                                                                                                                       | Cuándo presentarlo |
|-------|----------------------------------------------------|-----------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------|
| 1     | <b>AWS Certified Cloud Practitioner (CLF-C02)</b>  | Vocabulario cloud, facturación, responsabilidad compartida                                                | 3–4 semanas (las guías dicen lo mismo para principiantes; las Etapas 0–1 cubren la mayor parte)                                                                                                                                                                      | ~ Mes 4            |
| 2     | <b>AWS Solutions Architect Associate (SAA-C03)</b> | Diseñar arquitecturas AWS reales — la credencial por la que las descripciones de puesto realmente filtran | 6–8 semanas de preparación enfocada (estimación estándar de la industria después del CCP)                                                                                                                                                                            | Meses 8–9          |
| 3     | <b>HashiCorp Terraform Associate (003)</b>         | Fluidez en IaC, verificada                                                                                | 2–3 semanas — después del Módulo 7 y el Proyecto 1 usted está mayormente repasando (preparación oficial: <a href="https://developer.hashicorp.com/terraform/tutorials/certification-003">https://developer.hashicorp.com/terraform/tutorials/certification-003</a> ) | Meses 10–11        |

Cuarta opcional, si apunta a roles con título de ops: **AWS SysOps Administrator / CloudOps Associate** — su contenido es literalmente la Etapa 3. Para el CLF-C02, el curso completo de freeCodeCamp (<https://www.youtube.com/watch?v=7HKot-brXFE>, edición clásica <https://www.youtube.com/watch?v=NhDYbskXRgc>) más exámenes de práctica es el camino gratuito bien transitado; para el SAA-C03, combine un curso con muchos exámenes de práctica cronometrados — el examen se basa en escenarios y la resistencia importa.

## Módulo 13 (Meses 11–12): Currículum, entrevistas y las diez preguntas

**El currículum:** una página. Empiece con las habilidades (AWS, Terraform, Python/Bash, Docker, CI/CD, CloudWatch — refleje las palabras exactas de la descripción de puesto; los filtros automáticos comparan palabras clave) y los tres proyectos, cada uno como dos viñetas de la forma *hice X con Y logrando Z*: “Deployed a three-tier web app on AWS with Terraform and GitHub Actions; zero-downtime deploys via ALB + Auto Scaling.” Liste las certificaciones con fechas. Enlace el GitHub — y luego asuma que de verdad lo abrirán, porque los buenos lo hacen, y el suyo ahora recompensa la visita.

**Preparación de entrevistas:** tres sabores para ensayar — trivia (las tablas de vocabulario de las Etapas), escenarios (los labs que realmente ha hecho) y conductuales (las historias de su diario, contadas en forma **STAR**: Situación, Tarea, Acción, Resultado). Practique *en voz alta*, a diario, durante dos semanas — idealmente con un entrevistador de IA, sin piedad. Las diez preguntas de abajo cubren los clásicos; las respuestas de ejemplo son deliberadamente compactas — expanda cada una con sus propios detalles de lab, porque “...y cuando construí esto, lo que realmente pasó fue...” es la frase que lo separa de los candidatos que solo leyeron.

### Las diez preguntas, con respuestas sólidas:

1. **“Explain the difference between a public and a private subnet.” (Explique la diferencia entre una subred pública y una privada.)** *La tabla de rutas de una subred pública tiene una ruta a un internet gateway, así que sus recursos pueden ser alcanzados desde internet; una subred privada no tiene esa ruta. Las capas web van en públicas, las bases de datos en privadas, y el acceso administrativo llega a las máquinas privadas a través de un bastión o de SSM. En mis labs demostré que la instancia privada era inalcanzable desde mi laptop pero alcanzable desde la capa pública — la ruta ausente es la seguridad.*
2. **“What is IAM, and what is least privilege?” (¿Qué es IAM, y qué es el mínimo privilegio?)** *IAM controla qué identidades pueden realizar qué acciones sobre qué recursos, mediante usuarios, grupos, roles y políticas JSON. Mínimo privilegio significa otorgar lo mínimo que hace el trabajo — el usuario de mi script de auditoría tiene acceso de solo lectura a exactamente un servicio, y mis instancias EC2 usan roles en lugar de claves almacenadas, así que no hay credenciales de larga vida que filtrar.*
3. **“An EC2 web server stopped responding. Walk me through your troubleshooting.” (Un servidor web EC2 dejó de responder. Guíeme por su proceso de diagnóstico.)** *Primero el alcance: ¿un servidor o todo — reviso el dashboard de CloudWatch y las alarmas. Luego las capas en orden: status checks de la instancia; security group — ¿el 80/443 está realmente abierto?; SSH adentro — ¿nginx está corriendo (systemctl status), el disco está lleno (df -h), la CPU está clavada (top)?; luego los logs (tail del log de errores). Mitigar primero — reiniciar el servicio o reemplazar la instancia — y la causa raíz después de estar estables. Ese es el orden que ensayo en mis runbooks.*
4. **“Why Terraform instead of clicking in the console?” (¿Por qué Terraform en lugar de hacer clics en la consola?)** *La consola no deja un registro revisable y no puede reconstruir nada. Terraform es declarativo y versionado: los cambios pasan por pull requests, la misma configuración construye entornos idénticos, un mal cambio se*

revierte revirtiendo un commit, y `plan` muestra el diff exacto antes de que pase nada. Mi pipeline ejecuta `plan` en cada PR para que el diff sea parte de la revisión.

5. **“What’s the difference between a container and a virtual machine?” (¿Cuál es la diferencia entre un contenedor y una máquina virtual?)** Una VM virtualiza el hardware y carga un OS completo, tardando minutos en arrancar; un contenedor comparte el kernel del host y empaqueta solo la app y sus dependencias, arrancando en más o menos un segundo. Los contenedores dan comportamiento idéntico entre entornos — la misma imagen corre en mi laptop y en EC2. Las VMs siguen importando para el aislamiento y para correr workloads no Linux.
6. **“How would you reduce our AWS bill?” (¿Cómo reduciría nuestra factura de AWS?)** En orden de esfuerzo: encontrar lo que corre y no debería — instancias ociosas, volúmenes sin adjuntar, snapshots viejos; Cost Explorer y las etiquetas lo hacen visible. Luego rightsizing contra el historial de CloudWatch. Luego apagar los entornos de dev fuera de horario. Luego mover los datos viejos de S3 a niveles más baratos con ciclo de vida. Luego comprometer los workloads estables en Savings Plans y poner los lotes interrumpibles en Spot. Y vigilar el egress — los datos de salida son la línea sorpresa clásica.
7. **“What happens when you type a URL and press Enter?” (¿Qué pasa cuando usted escribe una URL y presiona Enter?)** El DNS resuelve el nombre a una IP; el navegador abre una conexión TCP al puerto 443 y negocia TLS; envía un HTTP GET; del otro lado eso llega a un balanceador de carga, que reenvía a una instancia sana en una subred pública; la app puede consultar una base de datos en una subred privada; la respuesta regresa con un código de estado — 200 si está sana — y el navegador la renderiza. Puedo profundizar en cualquier salto, y los he construido todos.
8. **“A teammate’s change broke production. What happens next?” (El cambio de un compañero rompió producción. ¿Qué pasa después?)** Mitigar primero — rollback a través del pipeline; recuperarse vence a diagnosticar en el momento. Luego un post-mortem sin culpas: qué pasó, por qué el sistema lo permitió, y qué barandilla faltaba — una prueba, una revisión de plan, una alarma. Lo de sin culpas importa en la práctica: quien teme el castigo esconde información, y la información escondida causa la repetición.
9. **“What is the shared responsibility model?” (¿Qué es el modelo de responsabilidad compartida?)** AWS asegura la nube — centros de datos, hardware, hipervisor; el cliente asegura lo que está en ella — datos, IAM, reglas de red, parcheo, configuraciones de cifrado. La mayoría de las brechas cloud reales están del lado del cliente, errores de configuración como buckets públicos o claves filtradas. Así que ‘AWS es seguro’ y ‘nosotros somos seguros en AWS’ son frases distintas, y la segunda es mi trabajo.
10. **“Tell me about a problem you debugged.” (Cuénteme sobre un problema que usted depuró.)** (El suyo, del diario, en forma STAR. Esqueleto de ejemplo:) Durante mi game day de monitoreo, mi alarma se disparó por saturación de CPU (S). Tenía que encontrar y detener la causa antes de que los ‘usuarios’ lo notaran (T). Las métricas mostraron el inicio del pico; Logs Insights lo ligó a procesos desbocados; entré por SSH, confirmé con top, los maté y observé la recuperación en el dashboard (A). Resuelto en once minutos, y el post-mortem produjo una alarma de conteo de procesos que luego implementé en Terraform (R).

**Mecánica de la búsqueda de empleo:** postule desde el Mes 11 — después del SAA, no espere a estar “listo”, porque las entrevistas son entrenamiento. Títulos objetivo: cloud engineer (junior/associate), cloud support engineer, cloud operations engineer, junior DevOps engineer, AWS support associate. Cada rechazo con entrevista es una lección gratis; anote en el diario qué le preguntaron.

### Vocabulario:

| Término                                     | Definición                                                                                                                                 |
|---------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------|
| Portafolio                                  | Prueba pública y documentada de que usted puede construir — para este oficio, repos de GitHub con diagramas y READMEs.                     |
| Application Load Balancer (ALB)             | El repartidor de tráfico de AWS: distribuye las solicitudes entre instancias sanas, descarta las enfermas.                                 |
| Auto Scaling Group                          | Mantiene N instancias vivas y ajusta N según la carga — autorreparación y elasticidad en uno.                                              |
| Lambda / serverless                         | Código que AWS ejecuta por evento, facturado por invocación — sin servidores que gestionar en absoluto.                                    |
| DynamoDB                                    | La tabla NoSQL serverless de AWS — combina de forma natural con Lambda.                                                                    |
| STAR                                        | Situación, Tarea, Acción, Resultado — la forma de una buena historia de entrevista.                                                        |
| ATS                                         | Applicant tracking system (sistema de seguimiento de candidatos) — el filtro de palabras clave que su currículum de una página debe pasar. |
| CLF-C02 / SAA-C03 / Terraform Associate 003 | Sus tres exámenes: vocabulario, arquitectura, IaC.                                                                                         |
| SysOps / CloudOps Associate                 | El associate opcional de AWS enfocado en ops — la Etapa 3, examinada.                                                                      |

**Hito — fin del curso:** tres repos que un extraño puede entender, tres certificaciones agendadas o aprobadas, diez respuestas ensayadas en voz alta, postulaciones activas. Usted no está “esperando entrar al cloud”. Usted es un ingeniero cloud junior con evidencia, entrevistándose.

---

## El plan de estudios en una página

| Cuándo      | Enfoque            | Prueba práctica | Prueba externa |
|-------------|--------------------|-----------------|----------------|
| Semanas 1–2 | Cómo funcionan las | Operaciones de  | —              |

| Cuándo        | Enfoque                                                          | Prueba práctica                                           | Prueba externa                  |
|---------------|------------------------------------------------------------------|-----------------------------------------------------------|---------------------------------|
|               | computadoras;<br>Linux, terminal,<br>permisos, SSH               | archivos con scripts;<br>primer shell script              |                                 |
| Semanas 3–4   | Redes: IP, DNS,<br>puertos, HTTP,<br>firewalls; cuenta de<br>AWS | Cuenta + MFA +<br>presupuesto de \$0                      | —                               |
| Semanas 5–6   | Lab 1: EC2                                                       | Servidor web en<br>internet, desmontado                   | —                               |
| Semana 7      | Lab 2: S3 + CLI                                                  | Sitio estático; fluidez<br>en la CLI                      | —                               |
| Semanas 8–9   | Lab 3: VPC                                                       | Red pública/privada,<br>prueba del bastión                | —                               |
| Semana 10     | Lab 4: RDS                                                       | Base de datos<br>privada, snapshot,<br>teardown           | —                               |
| Semanas 11–12 | Lab 5: IAM;<br>reconstrucción del<br>desafío de la Etapa 1       | Stack completo de<br>memoria, <3 h                        | Agendar el CCP                  |
| Semanas 13–15 | Bash, Python/boto3,<br>Git                                       | backup.sh, audit.py,<br>historial del repo                | <b>Examen CCP (~Mes<br/>4)</b>  |
| Semanas 16–17 | Terraform                                                        | Stack como código,<br>plan/apply/destroy                  | —                               |
| Semanas 18–20 | CI/CD, Docker,<br>alfabetización en K8s                          | Pipeline en verde;<br>imagen en el registry               | —                               |
| Semanas 21–24 | CloudWatch, logs,<br>incidentes, runbooks                        | Alarma disparada +<br>post-mortem del<br>game day         | —                               |
| Semanas 25–28 | Ops de seguridad;<br>gestión de costos                           | Informe de<br>autoauditoría;<br>prueba de<br>restauración | —                               |
| Meses 8–10    | Proyectos de<br>Portafolio 1–3                                   | Tres repos<br>documentados                                | <b>SAA-C03 (Meses 8–<br/>9)</b> |
| Meses 10–11   | Preparación del<br>Terraform Associate                           | —                                                         | <b>Terraform<br/>Associate</b>  |
| Meses 11–12   | Currículum,<br>entrevistas,<br>postulaciones                     | Diez respuestas, en<br>voz alta                           | Primeras ofertas                |

**Unas palabras de cierre de su profesor.** Doce meses es poco para un cambio de carrera y mucho para un hábito diario, así que aquí está el trato honesto: las personas que terminan este

curso no son las más inteligentes — son las que teclearon los comandos también en los días de cansancio. Cada mensaje de error con el que se tope es el plan de estudios funcionando, no fallando; el ingeniero que ha roto y arreglado cien cosas pequeñas es exactamente lo que un gerente de contratación quiere decir con “experiencia”. Mantenga el diario, desmonte lo que construya, nunca afirme en una entrevista lo que sus repos no puedan respaldar — y dentro de un año, cuando el localizador suene a las 3 a.m., sentirá algo inesperado debajo de la adrenalina: competencia. Vaya a construir.

---

## Fuentes

Investigación de descripciones de puesto (agosto de 2026): [Arc.dev](#) — [AWS Cloud Engineer Job Description](#) · [Wiz](#) — [Cloud Engineer Job Description Guide](#) · [DevsData](#) — [AWS Cloud Engineer JD Template](#) · [XOPA](#) — [Cloud Engineer JD Template 2026](#) · [Betterteam](#) — [Cloud Engineer Job Description](#). Ruta de certificaciones y tiempos de preparación: [StudyTech](#) — [AWS Certification Roadmap 2026](#) · [Cloud Evolvers](#) — [Cloud Engineer Roadmap 2026](#) · [HashiCorp](#) — [Terraform Associate 003 prep](#). Contexto salarial (mediana en EE. UU. ≈ \$104K, rango \$85K–\$140K): [Wiz](#), arriba. Todos los enlaces de YouTube verificados mediante metadatos de YouTube al momento de la redacción.

*Un curso B4LCILC — volumen complementario de “El Curso del Líder Cloud (The Cloud Leader Course)”.*

---

## LECCIONES NUEVAS — EDICIÓN DE AGOSTO DE 2026

*En agosto de 2026 se añadieron lecciones nuevas a los tres cursos interactivos de [b4lcilc.org](#) — los cursos de Líder Cloud, Ingeniero Cloud y Arquitecto Cloud. Este apéndice pone al día la edición de lectura con condensaciones fieles de ese material nuevo, en la misma voz que el resto de este libro.*

*Unas palabras sobre lo que usted tiene en las manos. Estas páginas son la edición de lectura: las ideas, las metáforas y las frases que vale la pena decir en voz alta. Las versiones interactivas completas en línea traen bastante más — cada lección nueva incluye quizzes que comprueban su comprensión antes de dejarlo avanzar, laboratorios prácticos (incluida una terminal de práctica que se comporta como la real pero no puede costar dinero), videos, drills de vocabulario y — para el material del capstone — ejercicios de diseño guiados con listas de requisitos y respuestas modelo resueltas. Lea estas condensaciones para meterse en la cabeza la forma de cada idea; después, cuando pueda, haga las versiones interactivas para que el conocimiento se quede.*

*Las adiciones caen en tres grupos. Para el curso de **Líder Cloud**: una introducción amable a la terminal, la ley de privacidad propia de Tailandia, una guía del líder para propuestas de IA y facturas de IA, y un ejercicio capstone que ensambla el curso entero en una sola página. Para el curso de **Ingeniero Cloud**: la computación serverless — el patrón que convierte los pequeños scripts operativos en funciones que no cuestan nada ejecutar. Para el curso de **Arquitecto***

**Cloud:** la aritmética de las promesas de disponibilidad, el patrón de Generación Aumentada por Recuperación (Retrieval-Augmented Generation) por el que los clientes ahora preguntan cada semana, y un ejercicio completo de costeo — ponerle precio a un diseño antes de que alguien lo construya.

---

## Para el curso de Ingeniero Cloud

### Qué es serverless en realidad

**Todo lo que usted ha automatizado hasta ahora sigue suponiendo una máquina:** el cron job del laboratorio de este módulo corre dos minutos al día en una máquina virtual que usted paga todo el mes, parcha, monitorea y un día depura a las 3 de la madrugada. Serverless es la respuesta de la nube a ese desajuste. Sigue habiendo servidores — el nombre es marketing — pero son problema del proveedor. Usted entrega una función, el proveedor la ejecuta cuando algo la dispara, y usted paga por invocación, por milisegundo. El modelo se llama **FaaS — Function as a Service (función como servicio): Lambda en AWS, Azure Functions en Azure**. Los conceptos se transfieren nombre por nombre.

**La anatomía cabe en una mano.** Un **handler** es la función que la plataforma llama, y recibe un **evento** — una descripción en **JSON** (JavaScript Object Notation) de lo que sea que ocurrió: un archivo aterrizó en un bucket, llegó una petición HTTP, se disparó un horario. Preste atención a lo último: “se disparó un horario” es exactamente lo que hace cron — salvo que una Lambda con horario no necesita ninguna máquina encendida entre ejecuciones. Convertir uno de los scripts operativos de este módulo es un clásico primer proyecto serverless.

**La economía es la razón por la que el modelo existe, y corta en ambos sentidos.** El nivel gratuito de AWS incluye un millón de invocaciones de Lambda al mes, para siempre — su función de laboratorio podría correr cada minuto del año y no costar nada, haciendo de serverless el hogar más barato para el pegamento de las operaciones. Pero un precio por milisegundo que es un error de redondeo a bajo volumen se vuelve dinero real a alto volumen; una función que corre constantemente suele salir más barata como container o como VM. La habilidad profesional es saber de qué lado de esa línea se sienta una carga de trabajo.

**Dos advertencias honestas.** Primera, el **cold start** (arranque en frío): una función que no ha corrido recientemente tarda unos cientos de milisegundos en despertar — irrelevante para un reporte nocturno, una consideración de diseño para una API sensible a la latencia. Segunda, una función igualmente corre como una identidad: Lambda asume un rol de IAM (Identity and Access Management, gestión de identidad y acceso) exactamente como lo hace una instancia EC2, y una función con permisos de más es el mismo error de seguridad en un envoltorio más pequeño.

---