



BLOCKSTELLART

# ROADMAP CLOUD ENGINEER



JOAN AMENGUAL



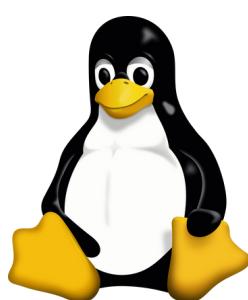
# Índice

- 1 Etapa 1 : Conocimientos Fundamentales de TI
- 2 Etapa 2 : Fundamentos de la Nube
- 3 Etapa 3 : Infraestructura como Código (IaC)
- 4 Etapa 4 : Contenedores y Orquestación de Contenedores
- 5 Etapa 5 : CI/CD y Prácticas DevOps
- 6 Etapa 6 : Monitoreo, Registro y Observabilidad
- 7 Etapa 7 : Seguridad en la Nube
- 8 Nuestro Bootcamp en cloud y DevOps
- 9 Mejor Bootcamp en Cloud de España

# Etapa 1

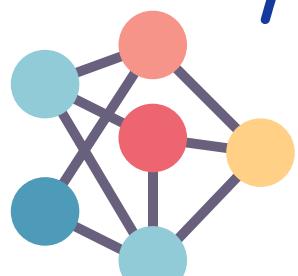
## Conocimientos Fundamentales de TI

Antes de entrar al mundo cloud, primero debes dominar los conceptos esenciales de tecnología. Estos pilares te ayudarán a entender cómo funciona todo por debajo en cualquier plataforma en la nube.



### Linux

Aprende a usar la terminal y gestionar un sistema operativo ampliamente usado en la nube



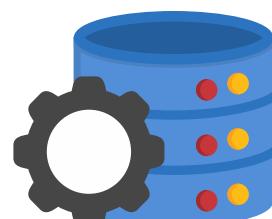
### Redes

Conoce cómo se comunican los sistemas: IPs, DNS, puertos, firewalls, etc.



### Programación

Mejora tu lógica con lenguajes clave como Python, Bash o JavaScript



### Bases de Datos

Entiende cómo se almacenan y consultan los datos, tanto relacionales como no relacionales.

# Etapa 1

## Administración de Linux

Para ser un buen ingeniero en la nube, necesitas una base sólida en sistemas operativos.

La mayoría de las cargas de trabajo en la nube se ejecutan en Linux, por eso es una habilidad esencial en este campo.

### **Habilidades clave de Linux para ingenieros cloud:**

#### **Uso de la línea de comandos:**

Aprende a moverte con soltura por la terminal. Domina comandos básicos para manejar archivos, revisar procesos y monitorear el sistema.

#### **Estructura del sistema de archivos:**

Comprende cómo está organizado el sistema de archivos de Linux y ubica directorios importantes.

#### **Permisos de archivos:**

Entiende la propiedad y los permisos (lectura, escritura, ejecución). Aprende a modificarlos con comandos como chmod y chown.

#### **Gestión de procesos:**

Aprende a iniciar, detener y supervisar procesos. Diferencia entre procesos en primer plano y segundo plano, prioridades y trabajos en ejecución.

#### **Scripting en Bash:**

Automatiza tareas repetitivas creando scripts. Usa variables, condicionales, bucles y funciones.

#### **Gestión de paquetes:**

Familiarízate con herramientas como apt, yum o dnf para instalar, actualizar o eliminar programas.

#### **Registros del sistema (logs):**

Identifica dónde se almacenan los logs del sistema y cómo analizarlos para detectar fallas o errores.



#### **Nota importante:**

En la nube, frecuentemente tendrás que acceder a máquinas virtuales usando SSH, configurar servicios o automatizar tareas. Para todo eso, necesitarás manejar bien Linux.

Incluso cuando uses servicios administrados, conocer cómo funciona el sistema operativo detrás te ayudará a entender mejor su comportamiento.

# Etapa I

## Fundamentos de Redes



Conocer redes es esencial en la ingeniería cloud, ya que toda comunicación entre servicios depende de una red correctamente configurada.

Cuando una aplicación en la nube deja de funcionar, los problemas de red suelen ser los responsables.

### Conceptos básicos de redes

#### Direcciones IP



Entiende la diferencia entre IPv4 e IPv6, y cómo se usan las notaciones CIDR y el subneteo para organizar redes.

#### DNS



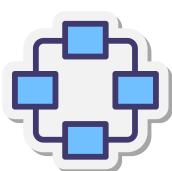
Aprende cómo se traducen los nombres de dominio a direcciones IP, y el papel de los servidores DNS.

#### Subneteo



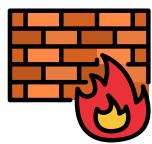
Descubre cómo dividir una red en segmentos más pequeños para una mejor organización y control.

#### Enrutamiento



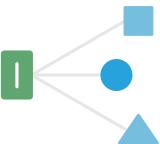
Comprende cómo viajan los datos entre redes hasta llegar a su destino.

#### Firewalls



Aprende a controlar el flujo de tráfico y a proteger los límites de una red.

#### Balanceo de carga



Entiende cómo se distribuye el tráfico entre varios servidores para mejorar el rendimiento y la disponibilidad.

#### VPN



Descubre cómo las redes privadas virtuales permiten conexiones seguras desde cualquier ubicación.

#### Dato clave:

Tener conocimientos sólidos en redes te permite diseñar arquitecturas cloud seguras y resolver problemas de conexión de forma eficaz.

Si una app no carga, sabrás si el problema está en el DNS, en el enrutamiento o en una mala configuración de seguridad.

# Etapa I

## Fundamentos de Programación



Aunque no necesitas ser desarrollador para trabajar en la nube, saber programar te abre muchas puertas.

La programación es clave para automatizar tareas, definir infraestructura como código y conectar con servicios cloud.

### ¿Por qué es importante saber programar en la nube?

- **Automatización:** Crea scripts para reducir tareas repetitivas y minimizar errores humanos.
- **Infraestructura como código:** Configura recursos cloud mediante código en lugar de hacerlo manualmente.
- **Conexión con APIs:** Interactúa con servicios en la nube a través de sus APIs.
- **Soluciones personalizadas:** Desarrolla utilidades que resuelvan necesidades específicas que otros servicios no cubren.
- **Procesamiento de datos:** Extrae, transforma y analiza datos desde servicios cloud.



### JavaScript / Node.js

Ideal para funciones serverless y aplicaciones nativas de la nube.



### Python

Fácil de aprender, con gran soporte en la nube y muchas librerías. Compatible con AWS, Azure y Google Cloud.

### Lenguajes recomendados



### Go (Golang)

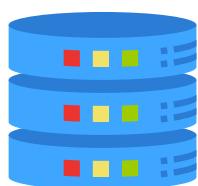
Usado en herramientas modernas como Docker, Kubernetes y Terraform.

### Conceptos clave a dominar:

- Variables y tipos de datos – Cómo almacenar y manejar información
- Estructuras de control – Uso de condicionales y bucles (if, for, while).
- Funciones – Organización del código en bloques reutilizables.
- Manejo de errores – Cómo responder correctamente ante fallos en el programa.
- Entrada/Salida de archivos – Leer y escribir archivos desde el código.
- Librerías y paquetes – Uso de componentes externos para extender funcionalidades.
- Solicitudes a APIs – Enviar peticiones HTTP para conectarse con servicios web.

# Etapa I

## Bases de Datos



Comprender cómo funcionan las bases de datos es esencial para cualquier ingeniero cloud, el almacenamiento de datos es un componente central en casi todas las aplicaciones.

Conocer los distintos tipos de bases de datos, sus casos de uso y cómo gestionarlas en la nube te permitirá diseñar soluciones más eficientes.

### Tipos de bases de datos

#### Bases de datos relacionales (SQL)

- Estructura en tablas, filas y columnas
- Claves primarias y foráneas
- Comandos básicos: SELECT, INSERT, UPDATE, DELETE
- Joins para combinar datos
- Índices para mejorar el rendimiento
- Transacciones para asegurar la integridad

En el cloud: Amazon RDS, Azure SQL, Google Cloud SQL



#### Bases de datos NoSQL

- Documentales: MongoDB, Amazon DocumentDB
- Clave-valor: Redis, DynamoDB
- Columnas: Cassandra, Bigtable
- Grafos: Neo4j, Amazon Neptune

Se usan cuando necesitas escalar a gran escala, esquemas flexibles o trabajar con grandes volúmenes de datos.

### Consideraciones al trabajar con bases de datos en la nube



#### Escalabilidad

Aprende a escalar verticalmente (más recursos) y horizontalmente (más instancias)



#### Alta disponibilidad

Usa réplicas, despliegues en múltiples zonas y mecanismos de conmutación por error



#### Optimización de rendimiento

Aprende a monitorear, analizar y ajustar el rendimiento de la base de datos



#### Seguridad

Aplica cifrado en tránsito y en reposo, aislamiento de red y control de acceso



#### Administradas vs autoadministradas

Decide entre usar servicios gestionados (como RDS) o gestionar la base por tu cuenta



#### Respaldo y recuperación

Implementa buenas estrategias de backup y recuperación ante fallos

## Etapa 2

# Fundamentos de la Nube

Una vez que dominas los conceptos básicos de TI, el siguiente paso es entender cómo funciona la nube.

Los fundamentos cloud te ayudarán a comprender qué es un servicio en la nube, cómo se estructura, y por qué es diferente del entorno tradicional.

### Tipos de servicios cloud

Comprende IaaS, PaaS y SaaS, sus usos principales y cómo se consumen los servicios.

### Modelos de despliegue

Explora los tipos de nube: pública, privada e híbrida, y sus ventajas para empresas.

### Elección de la plataforma en nube

Evalúa las necesidades del negocio para seleccionar entre AWS, Azure, Google Cloud u otras opciones.

### Competencias específicas de la plataforma: Inmersión profunda en la nube elegida

Desarrolla habilidades técnicas prácticas usando herramientas, servicios y buenas prácticas de tu proveedor seleccionado.



## Etapa 2

# Modelos de Servicio en la Nube

Cada modelo de servicio en la nube ofrece un nivel distinto de control y responsabilidad. Entender estas diferencias te ayudará a elegir la opción adecuada según tus necesidades.

## Infraestructura como Servicio (IaaS)

### ¿Qué es?

Proporciona recursos de cómputo virtualizados a través de internet.

### Tú gestionas:

Sistema operativo, middleware, aplicaciones y datos.

### El proveedor gestiona:

Servidores físicos, almacenamiento y red.

**Ejemplos:** Amazon EC2, Azure Virtual Machines, Google Compute Engine.

### ¿Cuándo usarlo?

Cuando necesitas el mayor control posible sobre la infraestructura, cumplir requisitos específicos o migrar sistemas existentes con pocos cambios.

## Plataforma como Servicio (PaaS)

### ¿Qué es?

Ofrece un entorno para desarrollar, ejecutar y administrar aplicaciones sin preocuparse por la infraestructura subyacente.

### Tú gestionas:

Las aplicaciones y los datos.

### El proveedor gestiona:

El sistema operativo, middleware y tiempo de ejecución.

### Ejemplos:

AWS Elastic Beanstalk, Azure App Service, Google App Engine.

### ¿Cuándo usarlo?

Cuando deseas centrarte en el desarrollo de software sin ocuparte de parches, servidores o escalado.

## Software como Servicio (SaaS)

### ¿Qué es?

Entrega aplicaciones listas para usar a través de internet, normalmente bajo suscripción.

### Tú gestionas:

Configuración y datos del usuario.

### El proveedor gestiona:

Todo lo demás (infraestructura, actualizaciones, mantenimiento, etc.).

### Ejemplos:

Microsoft 365, Google Workspace, Salesforce.

### ¿Cuándo usarlo?

Cuando solo necesitas usar una aplicación sin preocuparte por mantenimiento ni infraestructura.

## Etapa 2

# Modelos de Implementación en la Nube

Existen distintas formas de implementar servicios en la nube, y cada modelo tiene un nivel diferente de control, seguridad y costo.

### Nube Pública

Los servicios están disponibles para cualquier usuario a través de internet.

**Ejemplo:** AWS, Azure, Google Cloud.

**Ideal para:** Agilidad, escalabilidad rápida y costos reducidos.



### Nube Híbrida

Combina servicios públicos y privados. Permite mover datos o aplicaciones entre ambos entornos según se necesite.

**Ideal para:** Balancear flexibilidad con seguridad o requisitos regulatorios.

### Nube Privada

Infraestructura dedicada a una sola organización. Puede ser gestionada internamente o por un proveedor externo.

**Ideal para:** Mayor control, cumplimiento normativo y seguridad personalizada.

### Multi-Nube

Uso de servicios de múltiples proveedores cloud. Ayuda a evitar la dependencia de un solo proveedor o aprovechar lo mejor de cada uno.

**Ideal para:** Estrategias avanzadas de disponibilidad, rendimiento y resiliencia.

### Importante

Comprender estos modelos te permitirá tomar mejores decisiones sobre control, responsabilidad y costos según cada proyecto o empresa.

## Etapa 2

# Elige tu Plataforma Cloud



Una parte clave de tu camino como ingeniero en la nube es elegir en qué plataforma enfocarte primero.

El mercado cloud está dominado por tres grandes proveedores:

- Amazon Web Services (AWS)
- Microsoft Azure
- Google Cloud Platform (GCP)

 Actualmente, AWS lidera el mercado, lo que lo convierte en una excelente opción para comenzar.

### ¿Por qué es importante esta elección?

Elegir una plataforma cloud es una decisión que impactará tu aprendizaje y tu carrera profesional.

Aunque los conceptos básicos son parecidos entre proveedores, cada uno tiene sus propios servicios, nombres, herramientas y formas de gestión.

## Comparativa rápida



### Amazon – AWS

Con el mayor número de servicios y uso en empresas de todo tipo. Ideal por la gran demanda laboral de perfiles con experiencia en AWS.



### Microsoft – Azure

Integrado con herramientas como Windows, Active Directory y Office 365. Buena elección si tu entorno actual ya usa Microsoft.



### Google – GCP

Destaca en analítica de datos, inteligencia artificial y machine learning. Ideal para proyectos con enfoque en big data o IA.



### Recomendación:

Enfócate primero en dominar una plataforma.

Una vez que tengas una base sólida, será más fácil aprender otras, ya que muchos conceptos se comparten.

## Etapa 2

# Habilidades específicas de plataforma: Profundizando en tu nube elegida

Una vez que elegiste una plataforma cloud, es momento de aprender en detalle sus servicios principales.

Dominar estas capacidades te permitirá trabajar con confianza en proyectos reales.

### Servicios clave que debes conocer en cualquier plataforma cloud

#### Servicios de almacenamiento

Pasa al almacenamiento con servicios como S3 (AWS) o Blob Storage (Azure).

Debes comprender cómo funciona el almacenamiento de objetos, cómo crear buckets o contenedores, subir archivos y gestionar permisos.

#### Servicios de cómputo

Comienza con servicios como EC2 (AWS) o Máquinas Virtuales (Azure). Aprende a lanzar instancias, conectarte a ellas y entiende los tipos y precios

#### Redes en la nube

Explora conceptos como VPCs, subredes, tablas de rutas, gateways e infraestructura de seguridad como los grupos de seguridad.

Aquí tus conocimientos de redes (Fase 1) cobran verdadera importancia.

#### Ejemplo práctico

Si una aplicación responde lento, puede ser que la base de datos esté en otra región que los servidores web.

Si no entiendes cómo funciona la red en la nube, este tipo de problemas serían mucho más difíciles de identificar y resolver.

## Etapa 3

# Infraestructura como Código (IaC)

En esta etapa, aprenderás a gestionar y aprovisionar tu infraestructura cloud mediante código en lugar de hacerlo manualmente desde la consola.

La Infraestructura como Código (IaC) permite:

- Automatizar la creación y configuración de recursos en la nube.
- Aumentar la velocidad y consistencia en entornos de desarrollo, pruebas y producción.
- Versionar y reutilizar configuraciones de forma segura y colaborativa.

### IaC - Automatización de los despliegues en la nube

Define y despliega recursos automáticamente en la nube con código reutilizable, seguro y versionado fácilmente.

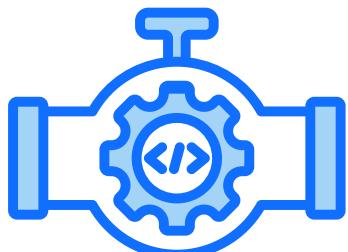
### Infraestructura como código en profundidad

Aprende herramientas como Terraform, Pulumi o CloudFormation para gestionar infraestructura de forma programática y eficiente.

# IaC – Automatización del despliegue en la nube

Cuando ya sabes crear recursos manualmente desde la consola cloud, pronto notarás que esto no es escalable.

Imagínate tener que crear manualmente decenas de recursos para producción: es lento, propenso a errores y difícil de repetir.



Aquí es donde entra la Infraestructura como Código (IaC). Defines la infraestructura con código, usando herramientas como Terraform y todo se despliega de forma automática y controlada.

## Herramientas populares de IaC:



### Terraform

La más utilizada, compatible con casi todos los proveedores cloud. Aprende su sintaxis básica, variables, módulos y manejo del estado.



### AWS CloudFormation

Herramienta nativa de AWS para definir infraestructura como código. Solo funciona en AWS.



### Ansible

Ideal para configurar servidores y aplicaciones luego de que la infraestructura ha sido desplegada.

## Beneficios clave de IaC



### Control de versiones

Puedes guardar tu infraestructura como código en Git, con historial, cambios y colaboración en equipo.



### Repetibilidad

Despliega la misma infraestructura tantas veces como quieras, en desarrollo, pruebas o producción.



### Automatización

Integra IaC en pipelines CI/CD para reducir errores humanos y acelerar despliegues.



### Documentación automática

El código actúa como documentación viva de tu infraestructura, facilitando la comprensión y el mantenimiento.

## Ejemplo:

Desplegar la misma aplicación en ambientes distintos (desarrollo, pruebas y producción).

Con Terraform y Ansible puedes asegurarte de que cada entorno sea casi idéntico, lo que permite validar antes de pasar a producción.

## Profundizando en Infraestructura como Código (IaC) con Terraform

La Infraestructura como Código (IaC) es una práctica fundamental en ingeniería cloud.

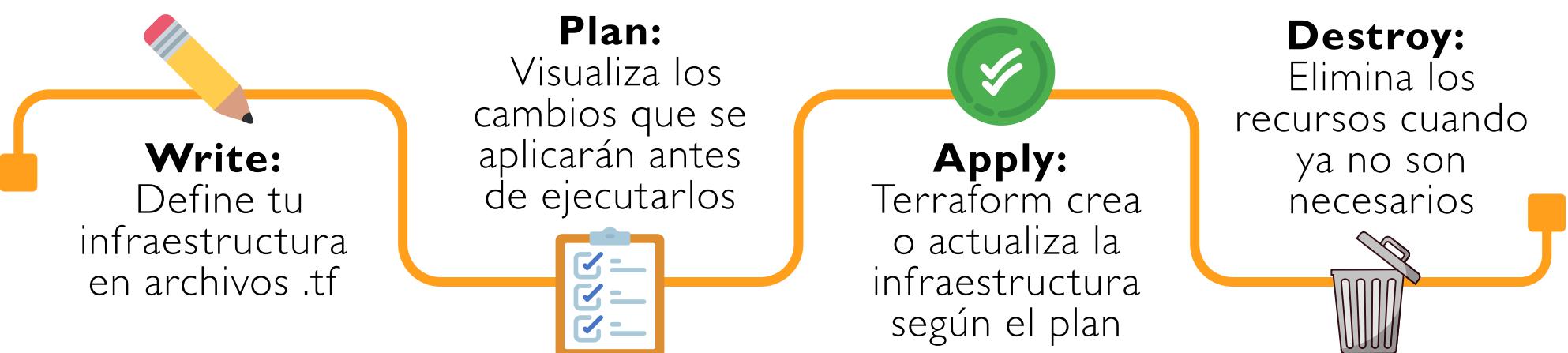
Permite gestionar y aprovisionar infraestructura usando código, sin tener que realizar acciones manuales desde la consola.

Terraform es la herramienta más popular de IaC es compatible con casi todos los proveedores de nube y se basa en un enfoque declarativo.



### Conceptos esenciales que debes aprender:

- Providers (Proveedores):** Conectan Terraform con servicios de nube como AWS, Azure o GCP.
- Resources (Recursos):** Son los componentes que deseas crear, como máquinas virtuales, redes o bases de datos.
- Variables:** Parámetros reutilizables que permiten personalizar la configuración del entorno.
- Modules (Módulos):** Bloques reutilizables para organizar y encapsular recursos de forma más ordenada.
- State (Estado):** Registro que mantiene Terraform sobre qué infraestructura está gestionando.



### Dato útil:

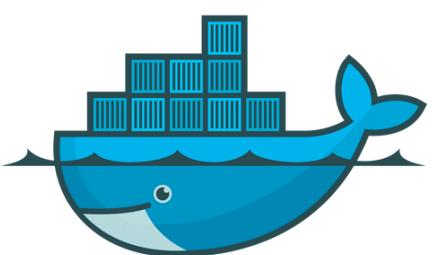
Este flujo te ayuda a trabajar de forma segura, repetible y con control de versiones sobre la infraestructura cloud.

## Fase 4:

### Contenedores y orquestación de contenedores

Los contenedores han revolucionado la forma en que las aplicaciones se construyen, despliegan y ejecutan en la nube.

Permiten empaquetar una aplicación junto con todas sus dependencias, asegurando que funcione igual en cualquier entorno.



**Docker**  
Plataforma para construir, empaquetar y ejecutar contenedores



**Kubernetes (K8s)**  
Sistema de orquestación que automatiza el despliegue, escalado y gestión de contenedores.

## Fase 4:

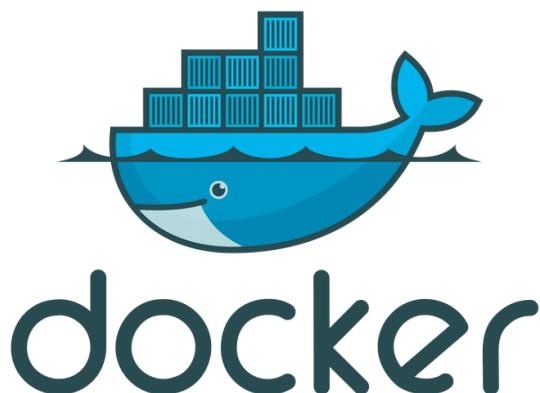
# Contenedores

A medida que avanzas como ingeniero cloud, conocerás la tecnología de contenedores, que ha revolucionado la forma en que se empaquetan y despliegan las aplicaciones.

### **Piensa en este escenario:**

Tu equipo desarrolla una app que funciona perfecto en sus laptops. Pero cuando la suben a la nube, falla por dependencias, configuraciones del sistema operativo o versiones de Python que no coinciden.

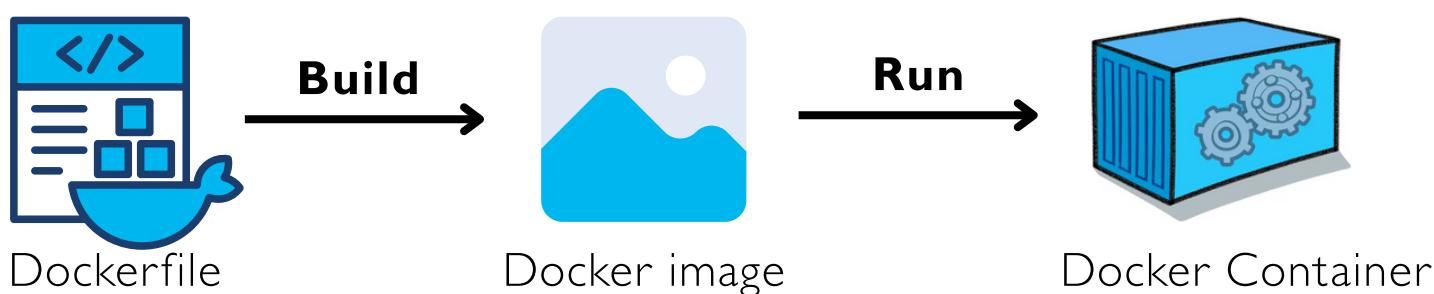
El clásico problema de “en mi máquina sí funciona” que los contenedores ayudan a solucionar.



La herramienta estándar para contenedores Docker agrupa el código de tu aplicación con todas sus dependencias y configuraciones, para que se ejecute igual en cualquier entorno. No importa si es tu laptop, un servidor o la nube.

### **¿Qué deberías aprender de Docker?**

- Sintaxis de Dockerfile: Aprende a definir imágenes paso a paso
- Construcción, etiquetas y versiones de imágenes
- Ciclo de vida de contenedores
- Multi-stage builds: Para generar imágenes más ligeras y eficientes
- Docker Compose: Ejecutar varias apps en conjunto
- Redes y volúmenes: Comunicar contenedores entre sí y datos persistentes
- Buenas prácticas de seguridad para tus imágenes



## Fase 4: Orquestación de Contenedores

Ejecutar contenedores individuales está bien para pruebas, pero no es suficiente en producción, donde puedes tener cientos de contenedores para microservicios, bases de datos y otras aplicaciones.

Aunque Docker resuelve cómo empaquetar y ejecutar contenedores, Kubernetes resuelve cómo desplegarlos, escalarlos y gestionarlos de forma automatizada y eficiente.

Kubernetes (también llamado K8s) se ha convertido en el estándar abierto más usado para orquestar contenedores a gran escala.

Kubernetes es como un administrador automático de operaciones.

Se encarga de:

- Desplegar aplicaciones contenidas
- Escalarlas cuando es necesario
- Actualizarlas sin interrupciones
- Revertir cambios si algo falla



# kubernetes

### Conceptos esenciales de Kubernetes:

- Pods, Deployments y Services
- ConfigMaps y Secrets (variables de configuración y credenciales)
- Namespaces y cuotas de recursos
- Autoscaling (escalado automático horizontal y vertical)
- StatefulSets para apps con estado
- Volúmenes persistentes para almacenamiento
- Ingress para enrutar tráfico desde fuera del clúster
- RBAC (control de acceso basado en roles)

### Consejo práctico:

No comiences creando un clúster desde cero. Usa servicios gestionados:

- EKS (AWS)
- AKS (Azure)
- GKE (Google Cloud)

Estos servicios gestionan por ti la creación y mantenimiento del clúster, lo que facilita el aprendizaje y la implementación.

## Fase 4:

# Arquitectura nativa en la nube

Para los ingenieros cloud, dominar los contenedores y las herramientas de orquestación es fundamental para crear aplicaciones modernas, escalables y resilientes.

Al comprender cómo funciona la contenerización, se abre paso naturalmente hacia los principios de la arquitectura cloud-native, que se enfoca en:

### Principios clave de la arquitectura nativa en la nube



#### Microservicios

Divide las aplicaciones en servicios pequeños e independientes que se pueden desarrollar, escalar y mantener por separado.



#### Infraestructura Inmutable

En lugar de modificar los componentes, se reemplazan por completo.



#### Resiliencia

Diseña sistemas capaces de resistir fallas mediante redundancia, balanceo de carga y recuperación progresiva.



#### Observabilidad

Implementa monitoreo, trazabilidad y registros centralizados para entender el comportamiento de las apps distribuidas y detectar errores.

### Experiencia práctica recomendada

1

Despliega una aplicación en Kubernetes, ya sea en un entorno local con Minikube o usando servicios gestionados como EKS (AWS), AKS (Azure) o GKE (Google).

2

Crea una app de varios niveles, por ejemplo: frontend + API + base de datos. Esto te ayudará a comprender cómo Kubernetes gestiona despliegues, redes, escalado y persistencia en un escenario real.

3

Explora conceptos avanzados de Kubernetes, como:

- CRDs (definiciones de recursos personalizados)
- Operadores para automatizar tareas
- Service Mesh (como Istio)
- Flujos GitOps para despliegues declarativos

## Fase 5: **CI/CD y prácticas DevOps**

Una vez que sabes construir, desplegar y escalar aplicaciones en la nube, el siguiente paso es automatizar esos procesos.

Aquí es donde entran en juego las prácticas de CI/CD (Integración y Entrega Continua), pilares del enfoque DevOps.

Estas prácticas permiten que los equipos desarrollen, prueben y desplieguen código de manera rápida, segura y continua.



## Fase 5: CI/CD y Prácticas DevOps

Ahora que ya dominas contenedores y Kubernetes, es momento de automatizar la forma en que tu código llega a producción.

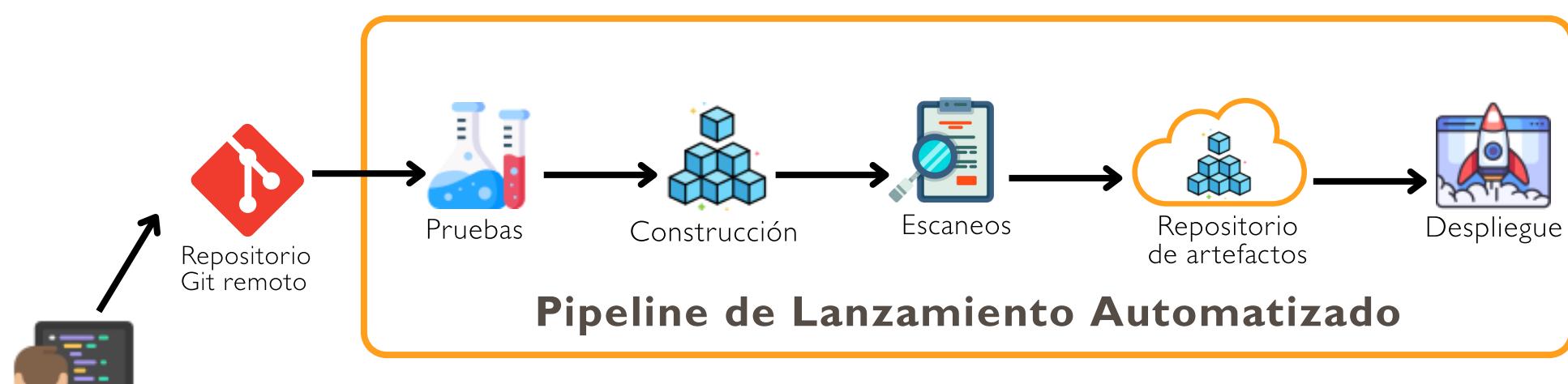
### Imagina este escenario:

Tu equipo tiene 10 microservicios desplegados en Kubernetes, cada uno desarrollado por un equipo diferente.

¿Cómo puedes llevar todos esos cambios del repositorio al clúster sin hacer despliegues manuales, sin provocar caídas y sin dar acceso directo al clúster a todos?

### La respuesta está en los pipelines de CI/CD

Un pipeline CI/CD actúa como puente entre tu código y la infraestructura.



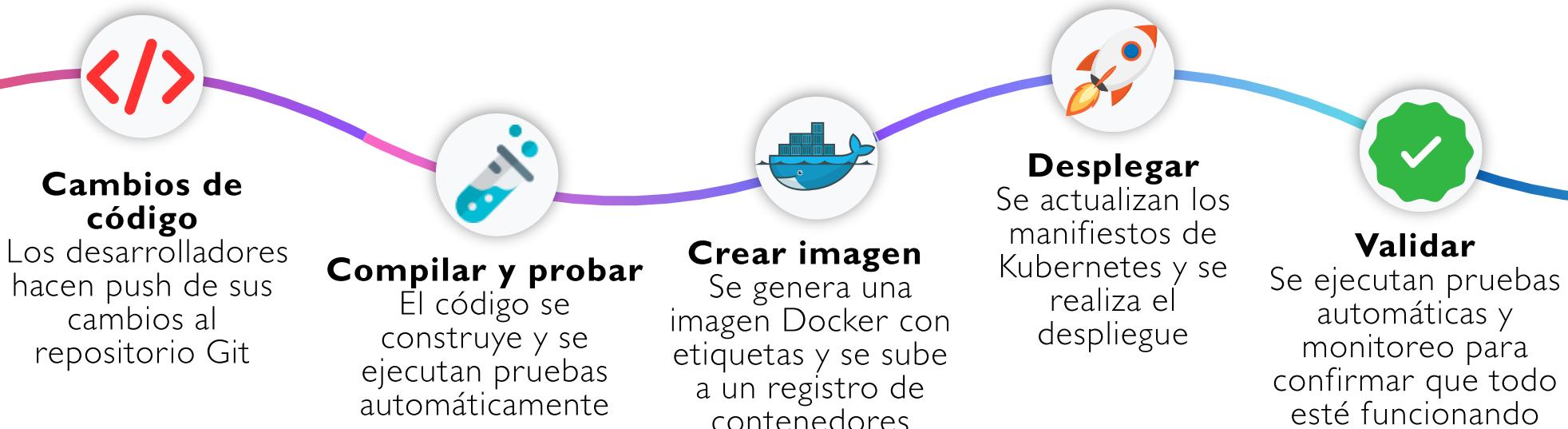
### CI = Integración Continua

Automatiza la validación de cambios al integrar el trabajo de múltiples desarrolladores.

### CD = Entrega Continua

Automatiza la entrega del código a entornos de pruebas o producción tras validar que todo funciona bien.

## ¿Cómo funciona?



## Fase 5: Prácticas de CI/CD y DevOps (Parte final)

Sin automatización, estarías construyendo imágenes, actualizando manifiestos y desplegando manualmente cada vez que hay un cambio de código.

Este proceso no solo consume tiempo, sino que es propenso a errores humanos.

Cloud Engineering y DevOps: dos prácticas que van de la mano. Ambas se enfocan en:

- **Automatización:** para minimizar tareas repetitivas y errores
- **Colaboración:** entre equipos de desarrollo, operaciones y QA
- **Mejora continua:** entrega rápida, confiable y constante de nuevas funcionalidades

Dominar DevOps te permite construir sistemas cloud más eficientes, seguros y listos para escalar

### ¿Cómo empezar con CI/CD?

- 
- 1 **Comprende los conceptos clave:**  
Aprende qué es CI/CD y cómo encaja dentro del ciclo de vida del software.
  - 2 **Elige una herramienta CI/CD:**  
Puedes comenzar con GitHub Actions, GitLab CI, Jenkins, CircleCI, entre otras.
  - 3 **Crea tu primer pipeline:**  
Automatiza el flujo desde que haces push al repositorio hasta que tu app es probada y desplegada.



“Un mundo sin CI/CD” puede ser caótico: despliegues manuales, errores frecuentes y mucho estrés. Con CI/CD, todo es más confiable, repetible y rápido.

## Fase 6: Monitoreo, Registro y Observabilidad

Una vez que tus aplicaciones están desplegadas y automatizadas, necesitas asegurarte de que funcionan correctamente, escalan bien y responden ante errores.

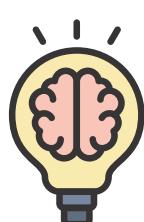
Aquí es donde entra en juego esta fase: la visibilidad total de tu sistema en producción.



## Fase 6: Monitoreo, Logging y Observabilidad

A medida que tus aplicaciones se vuelven más complejas, es vital asegurarte de que estén funcionando correctamente y que puedas detectar y resolver problemas rápido y con información precisa.

### Imagina esto:



1. Te llaman a las 2 a. m. porque la aplicación está caída.
2. No tienes monitoreo ni registros configurados.
3. Tendrías que revisar manualmente cada parte del sistema para encontrar el error... bajo presión... mientras los usuarios esperan.
4. Cada minuto sin visibilidad cuenta.



Te avisa cuando algo se sale de los parámetros normales definidos.

Puedes usar herramientas como Prometheus, Grafana o CloudWatch (AWS).



Es como tener cámaras que registran todo lo que ocurre dentro del sistema.

Por ejemplo: ELK (Elasticsearch, Logstash o Fluentd, Kibana), o CloudTrace en entornos cloud.



Es el conjunto completo que integra monitoreo, registros y trazas.

Es clave para entender el estado interno del sistema a partir de su comportamiento externo.

### Un sistema bien instrumentado debe incluir:

- Métricas: rendimiento, uso de recursos, tasas de error
- Alertas: cuando ocurre una falla o anomalía
- Dashboards: visión visual del estado del sistema
- Logs y trazas: eventos detallados y rutas que siguió cada solicitud

## Fase 7: Seguridad en la Nube

Después de aprender a desplegar, automatizar y monitorear sistemas en la nube, es momento de enfocarse en algo crítico: proteger tu infraestructura, tus aplicaciones y tus datos.

La seguridad en la nube no es opcional, es parte fundamental de cualquier arquitectura profesional.



## Fase 7: Seguridad en la Nube

La seguridad no debe ser una idea de último minuto. Debe estar presente en cada paso de tu carrera como ingeniero cloud. A medida que tus sistemas se vuelven más complejos y aumenta tu superficie de ataque, la seguridad se vuelve más crítica que nunca.

### **Escenario realista:**

Tu empresa sufre una filtración de datos porque un bucket S3 fue configurado como público por error. No había controles de seguridad ni auditorías automatizadas. Ahora la empresa enfrenta:

- Multas regulatorias
- Pérdida de confianza de los clientes
- Muchas horas perdidas solucionando el problema

Este es un ejemplo claro de por qué la seguridad en la nube es esencial.

### **Medidas clave de seguridad que debes aplicar**

#### **● Políticas IAM robustas**

- Usa el principio de mínimos privilegios
- Aplica control de acceso basado en roles (RBAC)
- Audita accesos de forma regular

#### **● Cifrado de datos**

- Cifra datos sensibles en tránsito y en reposo
- Usa protocolos estándar y servicios de gestión de claves

#### **● Seguridad en redes**

- Configura grupos de seguridad, firewalls, VPCs
- Separa recursos sensibles en subredes privadas

#### **● Monitoreo de seguridad**

- Habilita logs, alertas automáticas y trazabilidad
- Detecta y responde a comportamientos sospechosos

#### **● Gestión de vulnerabilidades**

- Realiza escaneos periódicos
- Aplica parches de seguridad y lleva registro de actualizaciones

#### **● CI/CD seguro**

- Integra pruebas de seguridad en el pipeline
- Incluye escaneo de código, dependencias e imágenes de contenedor

## Fase 7: Seguridad en la Nube

Un entorno cloud bien protegido debe considerar permisos mínimos, seguridad de red, cifrado, auditorías regulares y automatización de verificaciones.

La siguiente pirámide representa los niveles clave de seguridad en la nube:

### Modelo en capas de seguridad Cloud

#### AUTOMATIZACIÓN DE SEGURIDAD

Configura análisis automáticos de cumplimiento, escaneo de vulnerabilidades y alertas ante comportamientos anómalos.

#### PROTECCIÓN DE DATOS

Implementa cifrado en reposo y en tránsito. Usa herramientas de gestión de claves (como AWS KMS) y almacenamiento seguro.

#### SEGURIDAD DE RED

Incluye grupos de seguridad, subredes privadas, VPNs y listas de control (ACLs) para restringir el tráfico entrante/saliente.

#### GESTIÓN DE IDENTIDAD Y ACCESO (IAM)

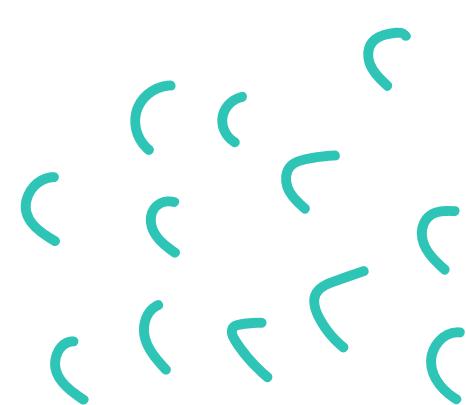
Define usuarios, roles y políticas siguiendo el principio de menor privilegio

#### MODELO DE RESPONSABILIDAD COMPARTIDA

Entiende qué partes protege el proveedor (infraestructura) y cuáles son tu responsabilidad (configuración, datos, accesos, etc.).



"Tú necesitas proteger todos los puntos de acceso... el atacante solo necesita encontrar uno abierto." Un error de configuración puede ser suficiente para comprometer todo el sistema.



# NUESTRO BOOTCAMP EN CLOUD Y DEVOPS

Aprenderás de las herramientas más demandadas en el sector y convertirte en un Cloud Engineer.

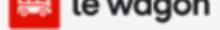


**EMPEZAR AHORA**

# #1 MEJOR BOOTCAMP EN CLOUD DE ESPAÑA

Premiado por Financial Magazine como el mejor Bootcamp en Cloud Computing.

## TOP 10 Mejores Bootcamps en Cloud Computing

Posición	Centro	Enlace
1	 <b>Blockstellart</b>	<a href="#">Blockstellart</a>
2	 <b>IMMUNE</b> TECHNOLOGY INSTITUTE	<a href="#">Immune Technology Institute</a>
3	 <b>IRONHACK</b>	<a href="#">Ironhack</a>
4	 <b>KSchool</b>	<a href="#">KSchool</a>
5	 <b>Upgrade hub</b>	<a href="#">Upgrade Hub</a>
6	 <b>CampusDual TIC</b>	<a href="#">Campus Dual Tic</a>
7	 <b>The Bridge</b>	<a href="#">The Bridge</a>
8	 <b>le wagon</b>	<a href="#">Le Wagon</a>
9	 <b>educación IT</b>	<a href="#">Educacion IT</a>

**EMPEZAR AHORA**