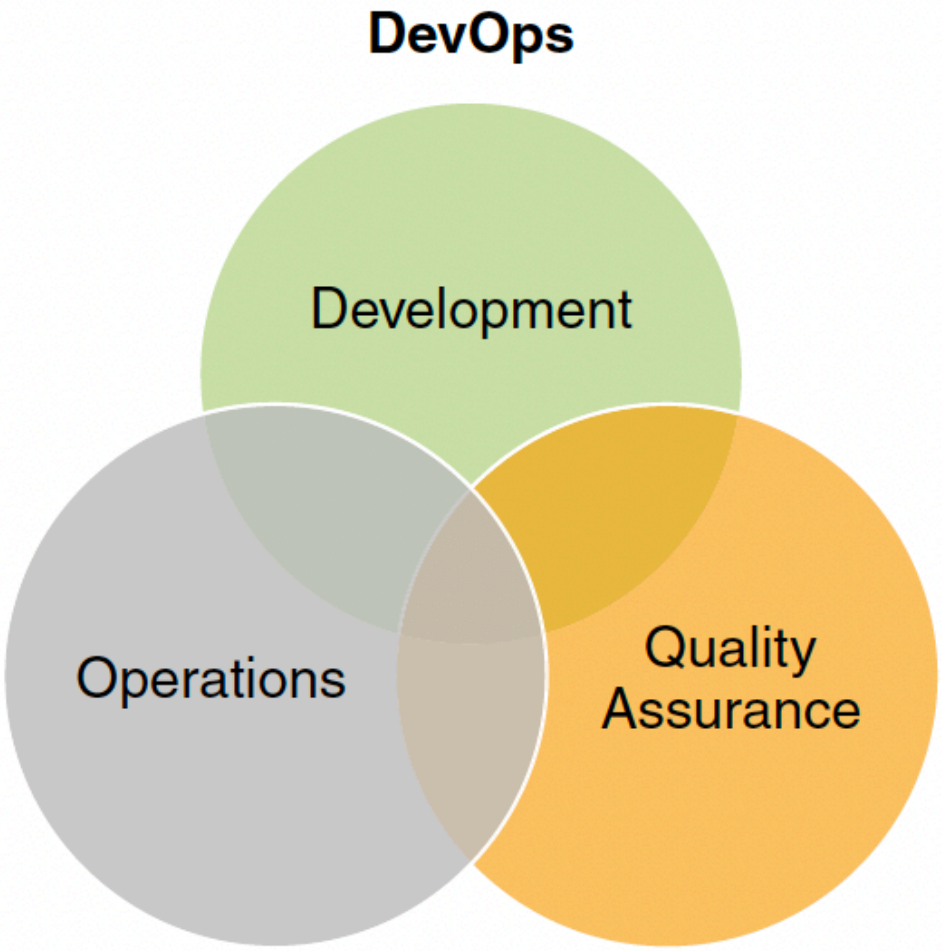
CULTURA DEVOPS

¿Qué es DevOps?

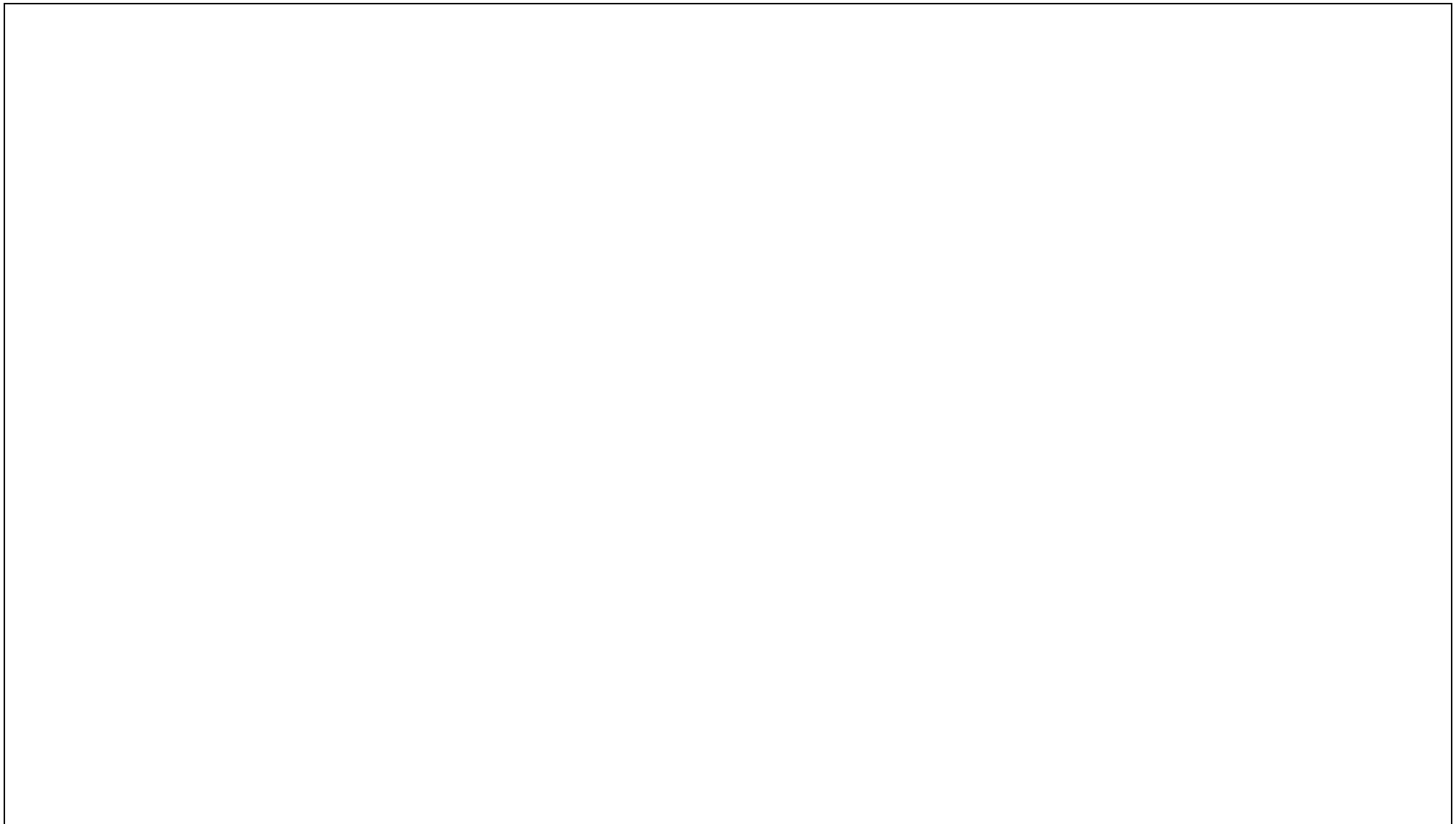
¿Qué es DevOps?

- Development + Operations
  - Concepción → Desarrollo → Entrega
  - Desarrolladores: innovación y agilidad
  - Sysadmin: garantías y estabilidad
- Agilidad: *Lean manufacturing*
  - La agilidad llega a los procesos de negocio
  - Falta incluir a los sysadmin



	Definiciones
Desarrollo	Se refiere al proceso de crear software, donde los desarrolladores escriben y actualizan el código fuente de las aplicaciones.
Operaciones	Se centran en la gestión y mantenimiento de los sistemas y la infraestructura en los que se ejecuta el software. Incluye tareas como la configuración, el monitoreo y la resolución de problemas.

¿Qué es DevOps?



Elementos clave para la comunicación y colaboración

- Deployment (despliegues) frecuentes
- Pruebas automáticas
  - TDD: *Test-driven design*
  - BDD: *Behavior-driven design*
- CI/CD: *Continuous Integration* + *Continuous Delivery*
- Feedback de usuarios
- Monitorización de apps/infraestructura

¿Qué significan **integración**, **entrega** (*delivery*) y **despliegue** (*deployment*)?

On your marks, get set,... go!

	Definiciones
Integración continua	Llevar automáticamente los cambios de varios desarrolladores en el código de una aplicación a un repositorio compartido para cada nueva versión.
Entrega continua	Trasladar la aplicación de software desde el entorno de desarrollo y dejarla disponible para su despliegue en un entorno de producción. Incluye pruebas, empaquetado y preparación de cada *release*.
Despliegue	Instalación de una aplicación en su entorno de producción, ya sea en un servidor, un conjunto de servidores, un contenedor, la nube, etc.


¿DevOps es un nuevo rol?

¿Qué no es DevOps?

- Es una cultura, no un rol
  - Si fuese un rol $\Rightarrow$ nuevo núcleo aislado (silo)
  - No son superhumanos
- Responsabilidades: desarrollo (código), calidad (pruebas) y operaciones (sysadmin)
  - No exclusivas
  - Proceso colaborativo

¿Por qué DevOps?

Motivación

- Sistemas frágiles

  $\Leftarrow$ Falta de comunicación y herramientas

  $\Rightarrow$ Despliegues complejos y propensos a errores

- Deuda técnica
- Arquitectura poco sólida
- Requisitos no funcionales poco solventes

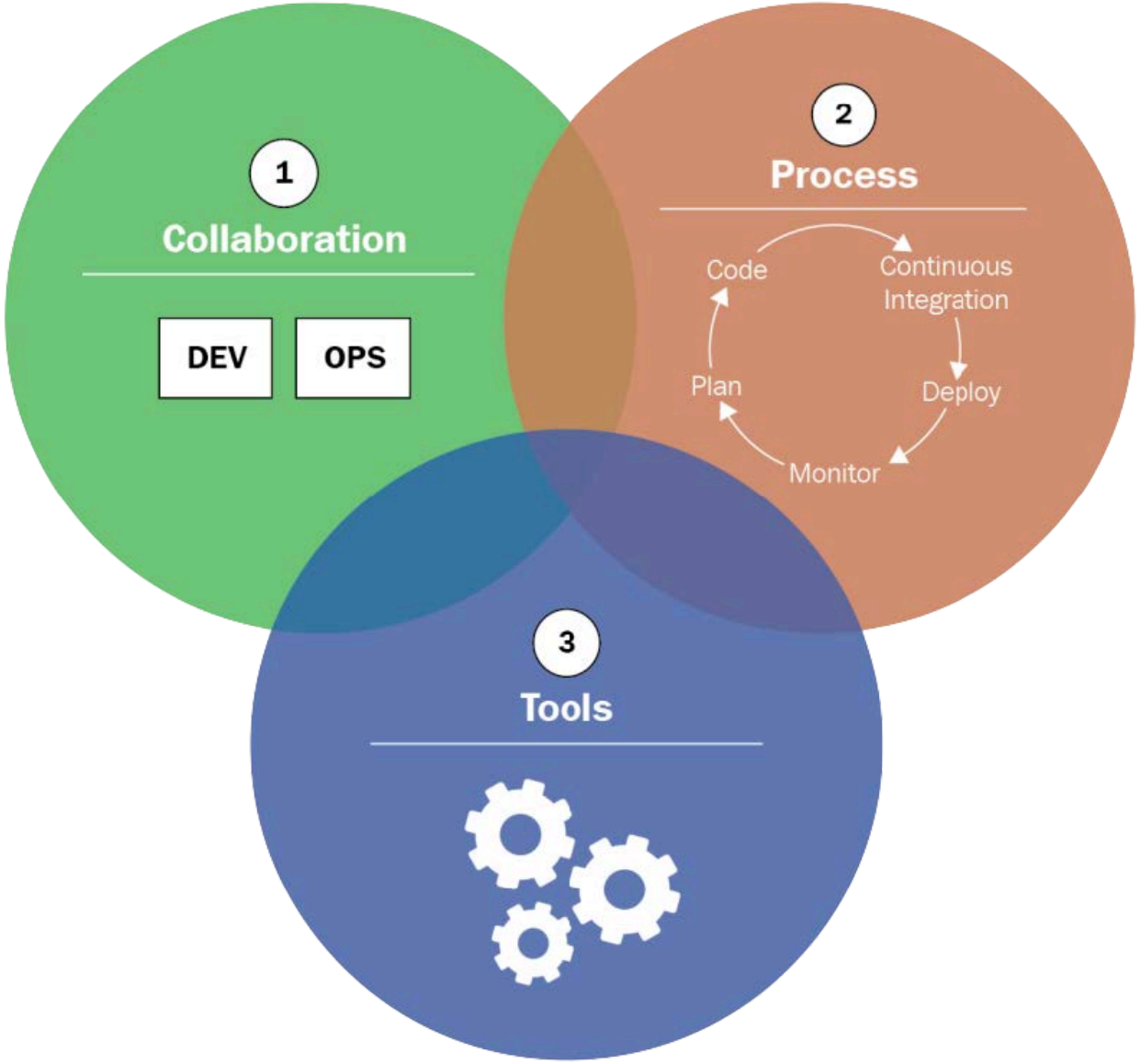
	Definiciones
Deuda técnica	Decisiones tomadas durante el desarrollo de un software que, en el corto plazo, permiten un desarrollo más rápido o una solución temporal, pero que crean problemas de NFR a largo plazo
Requisitos No Funcionales (NFR)	Aspectos que no están relacionados directamente con la funcionalidad de un sistema software, sino con características no directamente vinculados a sus funciones específicas (rendimiento, usabilidad, confiabilidad, seguridad, eficiencia, etc.)
Arquitectura software	Estructura y diseño organizativo de un sistema de software, sobre cómo sus componentes interactúan entre sí y cómo se organizan para lograr sus objetivos de manera efectiva. Proporciona un marco conceptual para abordar aspectos de los NFR.

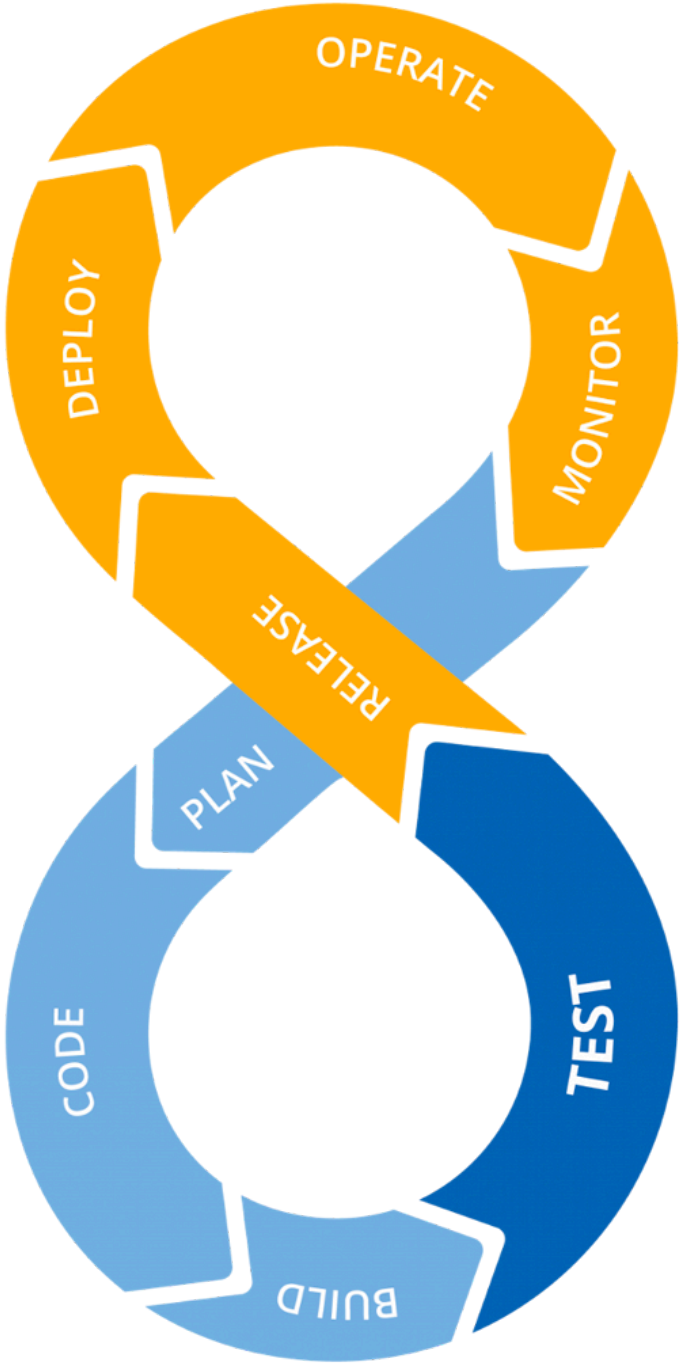
Cultura DevOps

Definición de Donovan Brown

> DevOps es la unión de personas, procesos y productos para una entrega continua de valor a los usuarios finales

- Procesos con Agilidad
- Personas en Colaboración
- Productos con Herramientas



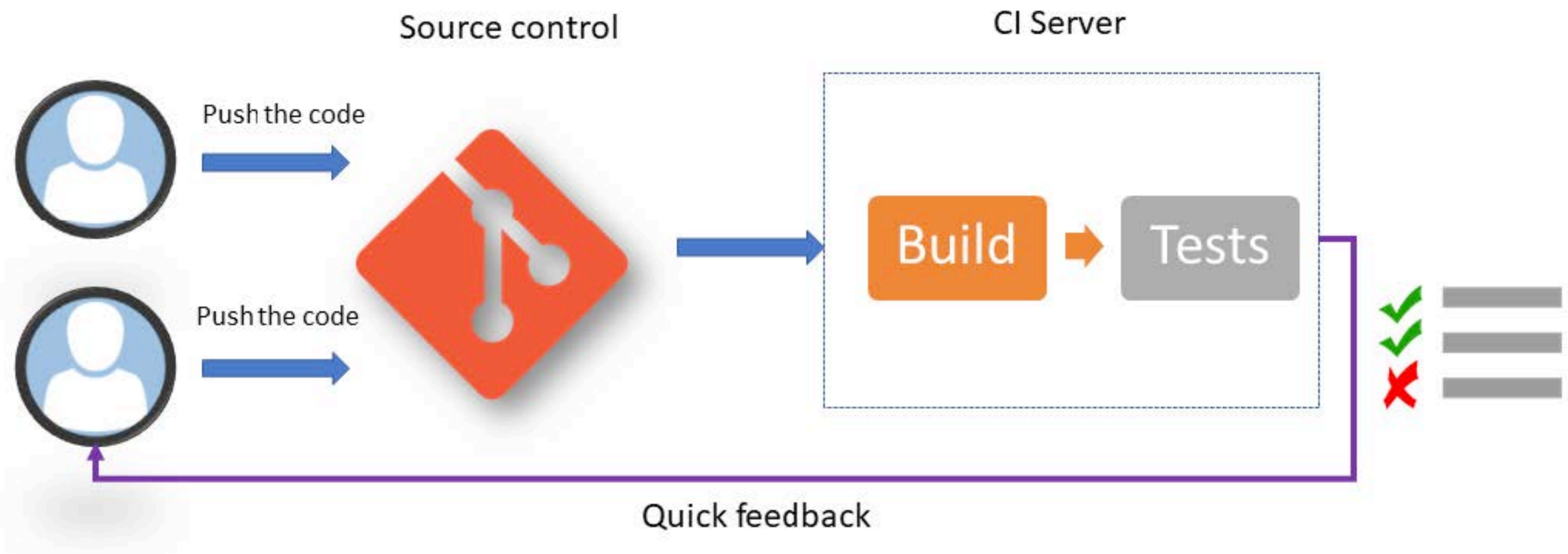


CI/CD: Continous Integration / Continuous Delivery

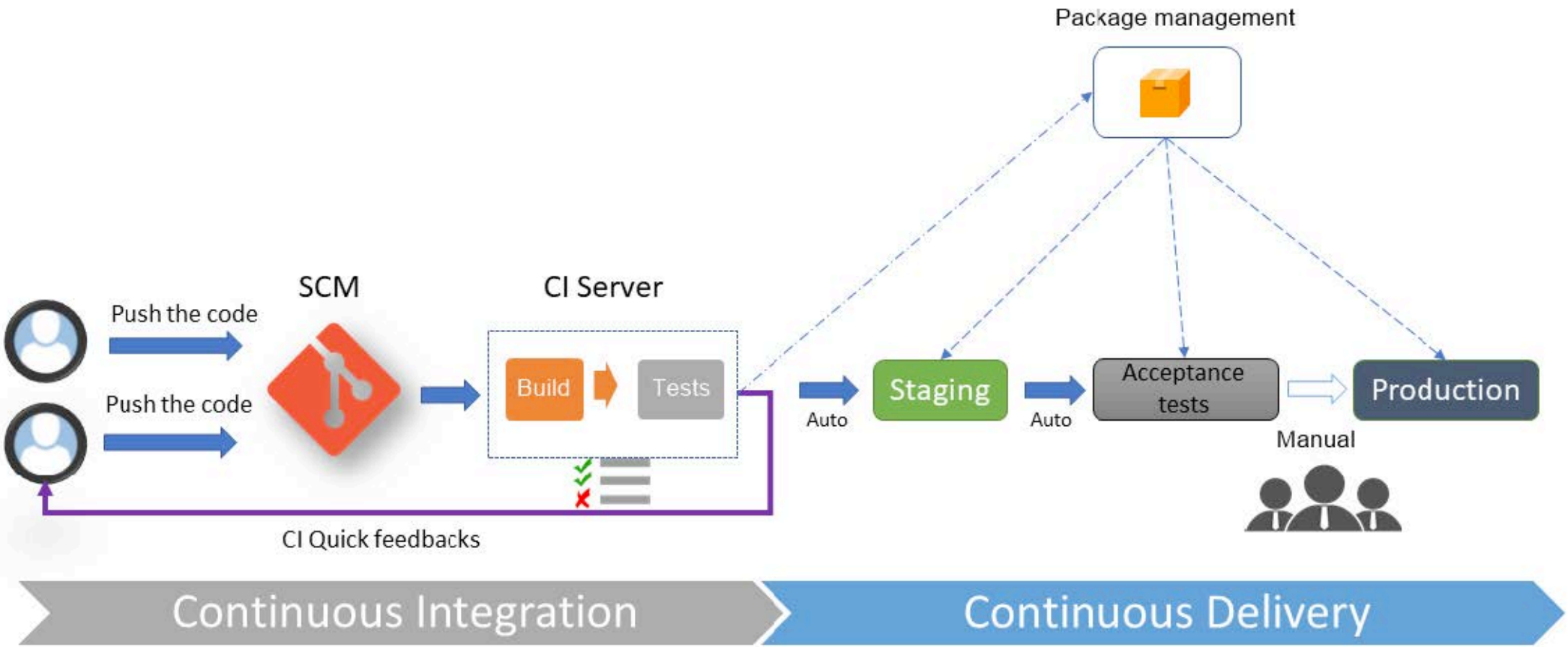
- Continuous integration (CI)
- Continuous delivery (CD)
- Continuous deployment

Cada proceso tiene su propio pipeline

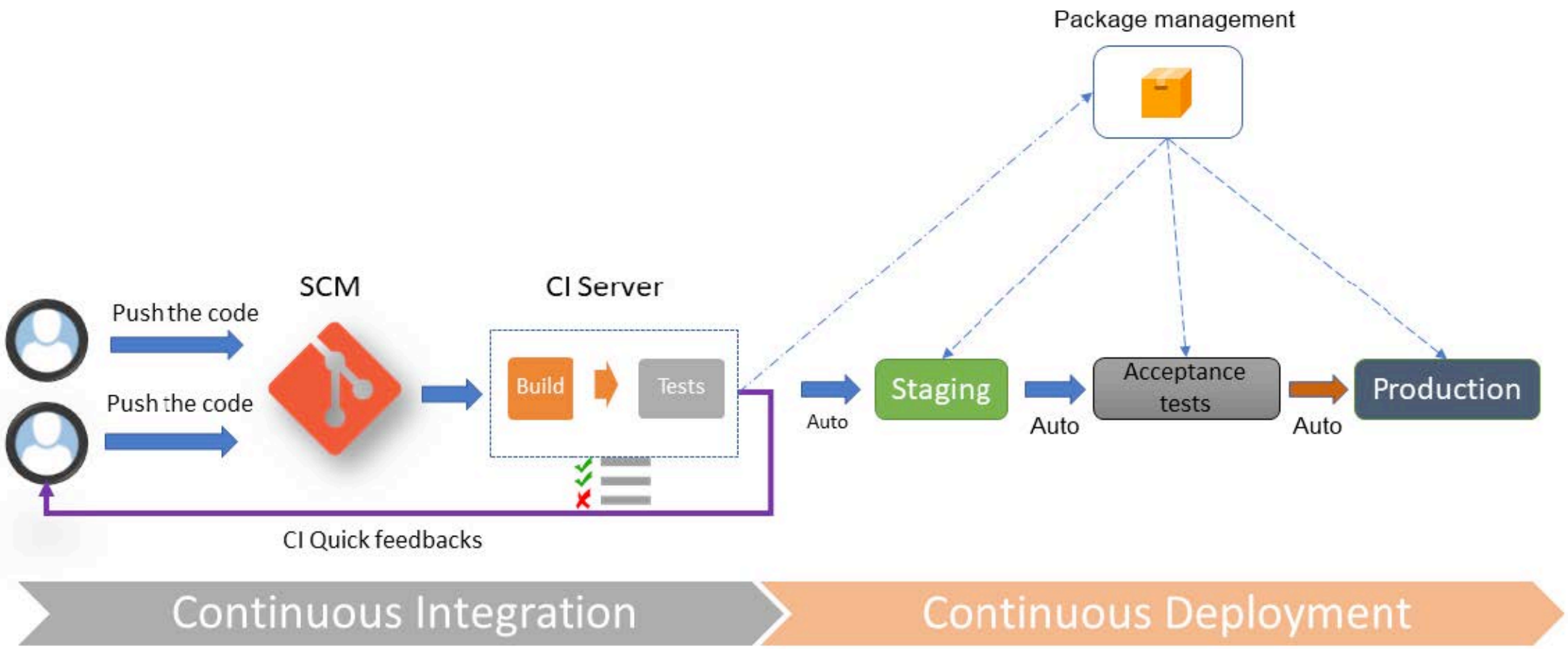
Pipeline de CI



Pipeline de CD



Continuous Deployment



📙	**Definiciones**
Build	acción de compilar y ensamblar el código fuente de una aplicación en un formato ejecutable o en un conjunto de artefactos que se pueden utilizar en un entorno de ejecución específico
Pipeline	un conjunto automatizado y secuencial de procesos que permiten la ejecución de tareas específicas. Analogía de una línea de montaje de la industria de fabricación
Staging	entorno de prueba que replica el entorno de producción para realizar pruebas finales (con usuarios) antes del despliegue

	Definiciones
Artefacto	resultado del *build*. Pueden ser binarios ejecutables, bibliotecas, paquetes de instalación, etc., necesarios para ejecutar la aplicación
Release	una versión específica y completa de una aplicación o software que se considera lista para ser distribuida y utilizada por los usuarios finales
Release Candidate (RC)	*release* con el potencial de convertirse en la versión final o lanzamiento si no se encuentran problemas significativos durante las pruebas

Prácticas DevOps

1. Automatizar la infrastructura: IaC
2. Automatizar los despliegues: Provisioning
3. Medir, monitorizar y experimentar: Feature flags

1. Automatizar la infrastructura

- IaC: *Infrastructure as Code*
  - Automatizar: MV/contenedores
  - Imágenes *estándar* (v.g. NGINX + MariaDB + Ruby)
  - Entorno de destino
- Provisioning
- Feature flags

IaC por configuración

Contenerización e inmutabilidad

Máquinas virtuales (mutables) versus Contenedores (inmutables)

Ejemplo con Docker

Dockerfile:

```
FROM ubuntu
RUN apt-get update
RUN apt-get install -y nginx
ENTRYPOINT ["/usr/sbin/nginx","-g","daemon off;"]
EXPOSE 80
```

IaC con tipos declarativos

- Terraform / OpenTofu
- Vagrant
- Ansible
- Azure ARM template
- Azure Bicep
- PowerShell DSC
- Puppet
- Chef
- Etc.

Ejemplo usando terraform para definir un servicio de AWS con un contenedor de Docker que sirve una página web en un cluster de ECS (*Elastic Container Service*)

```
provider "aws" {
  region = "West Europe" # Cambia esto según tu región de AWS
}

resource "aws_ecs_cluster" "example_cluster" {
  name = "example-cluster"
}

resource "aws_ecs_task_definition" "example_task" {
  family                   = "example-task"
  network_mode             = "bridge"
  requires_compatibilities = ["EC2"]
  ...
```

```
  ...
  container_definitions = <<EOF
[
  {
    "name": "example-container",
    "image": "nginx:latest",
    "portMappings": [
      {
        "containerPort": 80,
        "hostPort": 80
      }
    ]
  }
]
EOF
}
...
```

```
...
resource "aws_ecs_service" "example_service" {
  name            = "example-service"
  cluster         = aws_ecs_cluster.example_cluster.id
  task_definition = aws_ecs_task_definition.example_task.arn
  launch_type     = "EC2"
  desired_count   = 1

  network_configuration {
    # Coloca las ID de tus subredes aquí
    subnets         = ["subnet-xxxxxxxxxxxxxx", "subnet-yyyyyyyyyyyyyy"]
    # Coloca la ID de tu grupo de seguridad aquí
    security_groups = ["sg-xxxxxxxxxxxxxxxx"]
  }
}
```

2. Automatizar los despliegues

- IaC: *Infrastructure as Code*
- Provisioning
    - Despliegues manuales engorrosos
    - No repetibles. Hay que automatizar
- Feature flags

Provisioning (aprovisionamiento)

Opciones

- PaaS
- Recursos serverless
- Red

Herramientas

- terraform
- Azure ARM template, Azure CLI, Azure PowerShell
- AWS Cloud training
- Google Cloud Deployment Manager
- Etc.

Buenas prácticas de IaC & provisioning

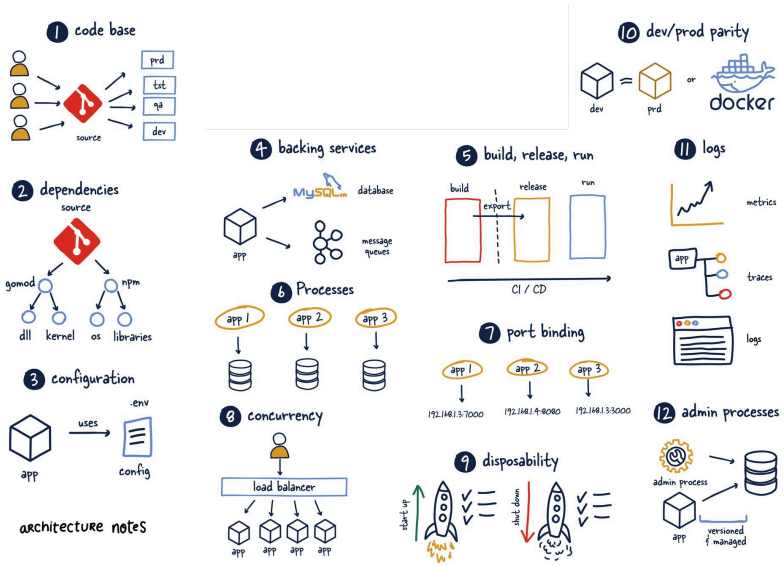
Análogas al desarrollo de software:

- Automatizar todo en el código, nada manual
- Someter a SCM (*Source Control Manager*) para versionar, rastrear, fusionar y restaurar
- Guardar el código de IaC junto al de la aplicación (mismo repo)
- Código de la IaC debe ser idempotente
- Integrar con CI/CD

3. Medir, monitorizar y experimentar

- IaC: *Infrastructure as Code*
- Provisioning
- Feature flags
    - A/B testing
    - Distintas versiones, geografías, periodos de tiempo, navegadores, dispositivos, etc.
    - Experimentos en producción

	Definiciones
IaC	práctica en la que la infraestructura de sistemas, redes y otros recursos tecnológicos se gestiona y aprovisiona utilizando código y archivos de configuración en lugar de realizar configuraciones manuales o a través de interfaces gráficas
Provisioning	proceso de preparar y configurar de manera automática los recursos de infraestructura necesarios para ejecutar una aplicación o servicio
Feature flags	técnica de desarrollo de software que permite habilitar o deshabilitar características específicas de una aplicación durante o después del despliegue



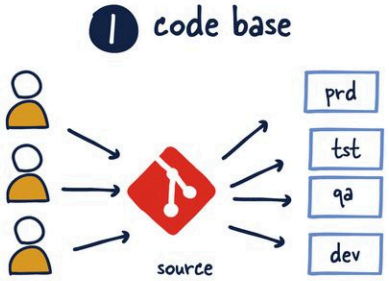
Metodología 12-Factor App

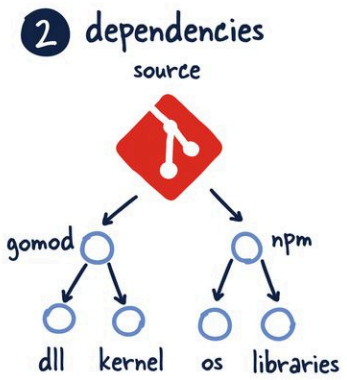
- Guia para apps SaaS modernas: portables, escalables, mantenibles.
- Define principios para codebase, configuración, dependencias y procesos.
- https://12factor.net/

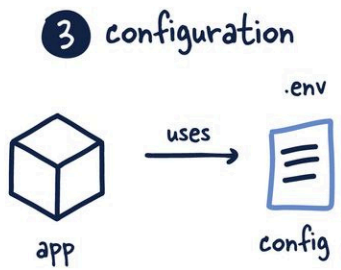
12 factores para apps SaaS modernas

1. Codebase: una base de codigo por app, versionada.
2. Dependencies: declarar y aislar dependencias explícitamente.
3. Config: configuración fuera del codigo, via variables de entorno.
4. Backing services: tratar servicios externos como recursos adjuntos.
5. Build, release, run: separar etapas de build y ejecución.
6. Processes: ejecutar como procesos stateless, sin compartir estado.
7. Port binding: exponer servicios por un puerto propio.
8. Concurrency: escalar por procesos, no por hilos internos.
9. Disposability: inicio rápido y apagado limpio.
10. Dev/prod parity: entornos dev y prod lo más parecidos posible.
11. Logs: logs como flujo de eventos, no archivos locales.
12. Admin processes: tareas admin como procesos puntuales.

Metodología 12-Factor App







architecture notes