

Del código a la producción

Pedro Antonio Vargas Alfaro

Amir Abants Canto Gamboa

Aldo Luna Bueno

Infraestructura como Código (IaC)

Introducción a IaC

IaC (Infrastructure as Code) es el manejo y provisión de la infraestructura configurada y determinada a través de código-

Este procedimiento reemplaza el tradicional proceso de implementación de hardware y software, a través de una configuración manual.



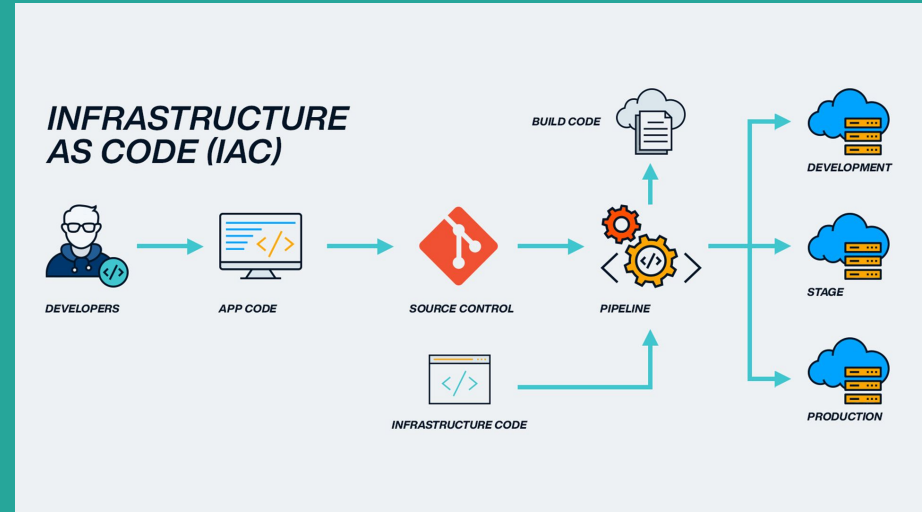
Introducción a IaC

De esta manera, se eliminan problemas de compatibilidad posiblemente generados por hardware y/o software, pues es especificado de esta manera, garantizando las versiones detalladas, minimizando posibles errores humanos, a través de código.



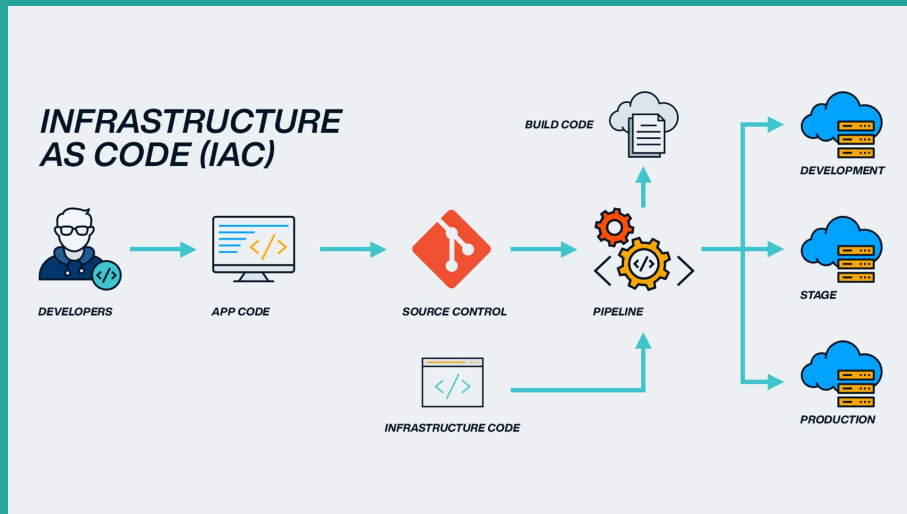
Escritura de IaC

- Terraform (HashiCorp)
- Ansible (M. DeHaan)
- Pulumi (J. Duffy & E. Rudder)
- AWS CloudFormation (Amazon Web Services)
- Jenkins (Kohsuke Kawaguchi)
- Puppet (Luke Kanies)



Escritura de IaC

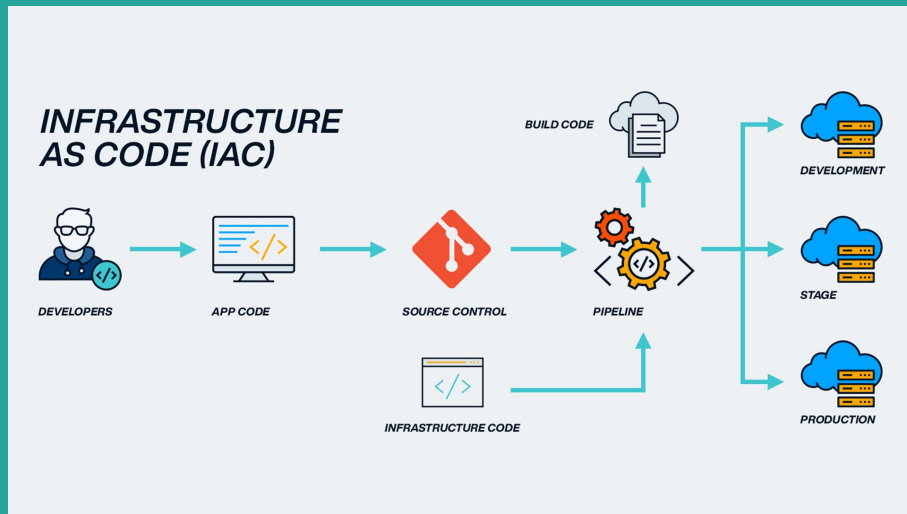
La IaC junto al uso a través de entregas y cambios cortos, juntos a prácticas como nomenclatura apropiada, pretende facilitar el control de errores, pues un código *modularizado* es más fácil de revisar y corregir, y se puede observar qué cambio exacto ha causado errores y corregir con prontitud.



Escritura de IaC

Beneficios:

- Rapidez
- Consistencia
- Mejor control de versiones
- Automatización
- Feedback integrado
- Seguridad a lo largo
- Sólida interoperabilidad de herramientas



Patrones para módulos

¿Qué es un módulo?

- Un pequeño programa que realiza acciones una máquina local, API, o host remoto
- Los módulos son expresados como código, usualmente en Python, y contienen metadata que define cuando y donde una tarea especifica de automatizacion es ejecutada y qué usuarios pueden ejecutarla

Tarea teórica

- Investigar una herramienta de IaC (p. ej. Terraform) y describir cómo organiza sus módulos.
- Proponer la estructura de archivos y directorios para un proyecto hipotético que incluya tres módulos: *network*, *database* y *application*. Justificar la jerarquía elegida.



Terraform

Investigar una herramienta de IaC (p. ej. Terraform) y describir cómo organiza sus módulos.

En terraform, se sigue la siguiente estructura estándar de módulos:

- `Módulo Root`
 - `README.`
 - `LICENSE.`
 - `main.tf, variables.tf, outputs.tf.`
 - `Variables y outputs deben tener descripciones`
 - `Módulos anidados.`
 - `Examples.`
-

Terraform

Root module

Investigar una herramienta de IaC (p. ej. Terraform) y describir cómo organiza sus módulos.

Módulo Root

- Único elemento requerido para la estructura estándar
 - Los archivos de Terraform deben existir en el directorio root del repositorio.
 - Debe ser el entrypoint (main.tf) y se espera que siga prácticas óptimas.
-

Terraform

ReadMe

Investigar una herramienta de IaC (p. ej. Terraform) y describir cómo organiza sus módulos.

README

- El módulo root y cualquier módulo anidado deberá tener un archivo README
 - Debería haber una descripción de el módulo y para lo que debería ser utilizado.
 - Considerar incluir un diagrama visual retratando los recursos de infraestructura que el módulo podría crear y su relación
-

Terraform

License

Investigar una herramienta de IaC (p. ej. Terraform) y describir cómo organiza sus módulos.

En terraform, se sigue la siguiente estructura estándar de módulos:

- La licencia bajo la que el módulo está disponible
 - Si se publica el módulo abiertamente, las organizaciones no adoptarán un módulo a menos que esté presente una licencia clara
 - Incluso si se trabaja en un código no-open-source se debería incluir una licencia.
-

Terraform

`main.tf`, `variables.tf`, `outputs.tf`

Investigar una herramienta de IaC (p. ej. Terraform) y describir cómo organiza sus módulos.

- Estos son los nombres recomendados para un módulo mínimo, incluso si están vacíos.
 - **`main.tf`** debería ser el principal entrypoint
 - Para un módulo simple, debería ser el lugar para todos los recursos creados
 - Para un módulo complejo, la creación de recursos podría estar fraccionada en múltiples archivos pero cualquier llamada a un módulo anidado debería ser dentro de **`main.tf`**
 - `variables.tf` & `outputs.tf` deberían contener declaraciones para variables y salidas, respectivamente
-

Terraform

Nested modules

Investigar una herramienta de IaC (p. ej. Terraform) y describir cómo organiza sus módulos.

- Los módulos anidados deben existir bajo el subdirectorio `modules/`
- Respecto a módulos y README
 - Cualquier módulo con un `README.md` se considera utilizable por externos
 - Si no existe README, se considera de uso interno únicamente
- Utilizados para dividir comportamiento complejo en módulos más pequeños para ser elegidos y usados por usuarios avanzados
- Si el módulo root incluye llamadas a módulos anidados, ellos deberían usar rutas relativas como `./modules/consul-cluster` tal que Terraform lo considere parte del mismo repositorio o paquete
- If a repository or package contains multiple nested modules, they should ideally be composable by the caller, rather than calling directly to each other and creating a deeply-nested tree of modules.

Terraform

Investigar una herramienta de IaC (p. ej. Terraform) y describir cómo organiza sus módulos.

```
$ tree minimal-module/
```

```
.  
├── README.md  
├── main.tf  
├── variables.tf  
└── outputs.tf
```


Terraform

Investigar una herramienta de IaC (p. ej. Terraform) y describir cómo organiza sus módulos.

```
$ tree minimal-module/
```

```
.  
├── README.md  
├── main.tf  
├── variables.tf  
└── outputs.tf
```

Terraform

Proponer la estructura de archivos y directorios para un proyecto hipotético que incluya tres módulos: `network`, `database` y `application`. Justificar la jerarquía elegida.

```
$ tree complete-module/
```

```
.
├── README.md
├── main.tf
├── variables.tf
├── outputs.tf
├── ...
├── modules/
│   ├── network/
│   │   ├── README.md
│   │   ├── ...
│   ├── database/
│   │   ├── README.md
│   │   ├── ...
│   ├── application/
│   │   ├── README.md
│   │   ├── ...
│   └── .../
└── examples/
    ├── exampleA/
    │   ├── main.tf
    ├── exampleB/
    └── .../
```

Tarea teórica

- Investigar una herramienta de IaC (p. ej. Terraform) y describir cómo organiza sus módulos.
- Proponer la estructura de archivos y directorios para un proyecto hipotético que incluya tres módulos: network, database y application. Justificar la jerarquía elegida.



Ansible

Investigar una herramienta de IaC (p. ej. Terraform) y describir cómo organiza sus módulos.

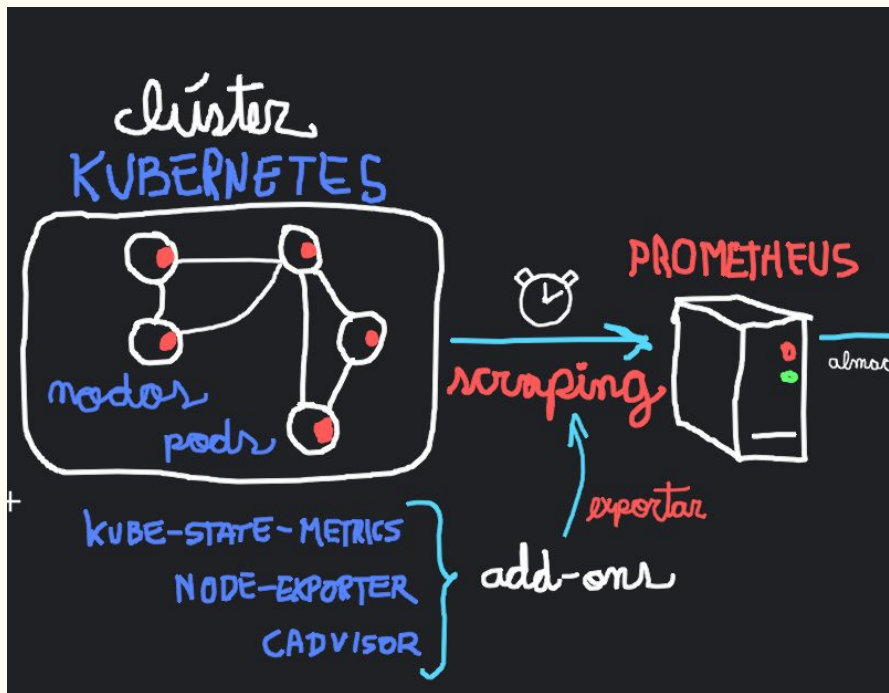
¿Cómo funcionan los módulos?

- **Task:** Defines the action to be applied to a managed host, but does not specify the host it will be applied to.
 - **Play:** The core unit of Ansible execution
 - **Ansible Playbook:** Includes 1 or more plays. Playbooks are written in YAML, are human-readable, and easy to share.
 - **Ansible Role:** Packages Ansible content—
 - **Collections:** Bundles of Ansible content designed to help automation developers **Plugins:** Pieces of code that build on Ansible's core functionality
-

Monitorizar Kubernetes

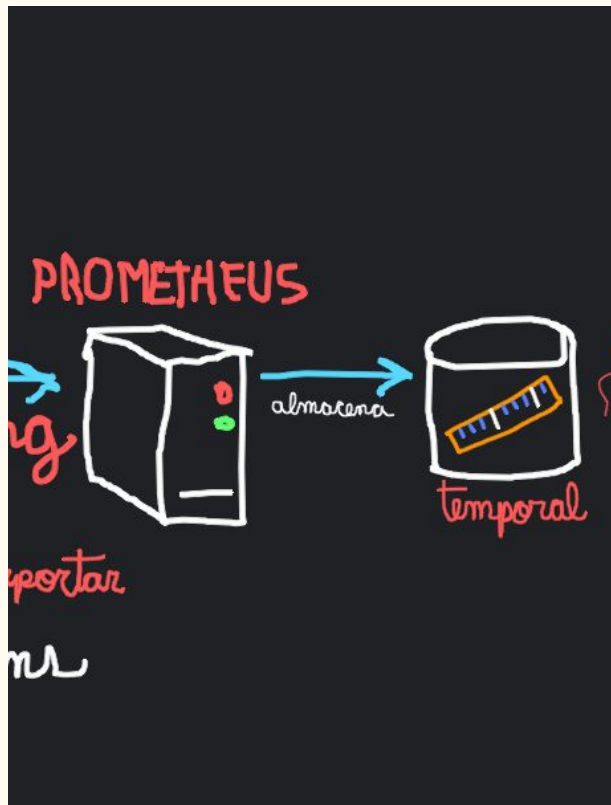


Prometheus y Kubernetes



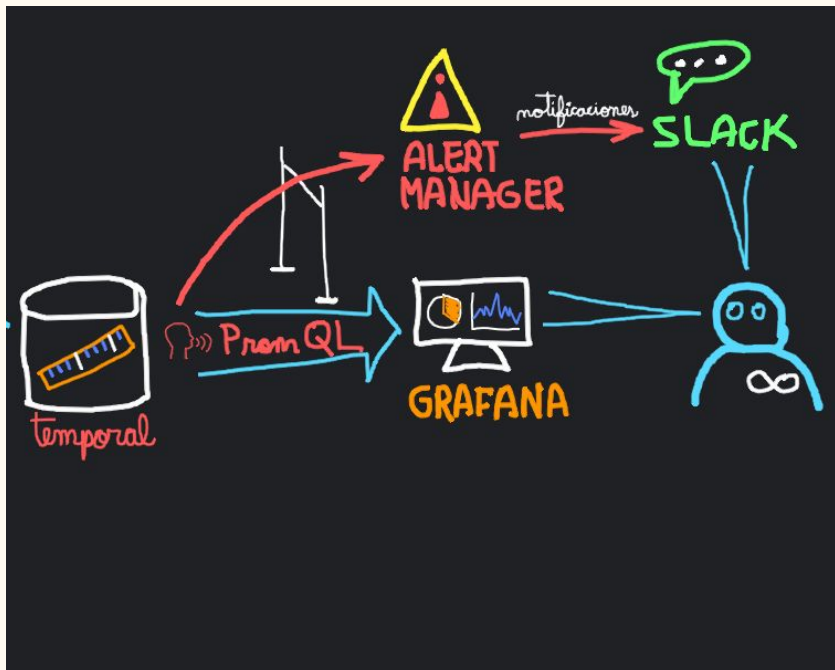
1. Prometheus se conecta a los endpoints del clúster de Kubernetes y recoge métricas de rendimiento.
 2. Las métricas son extraídas mediante exportadores en un formato que Prometheus puede leer.
 3. Las métricas se extraen en intervalos de tiempo configurables.
-

Prometheus

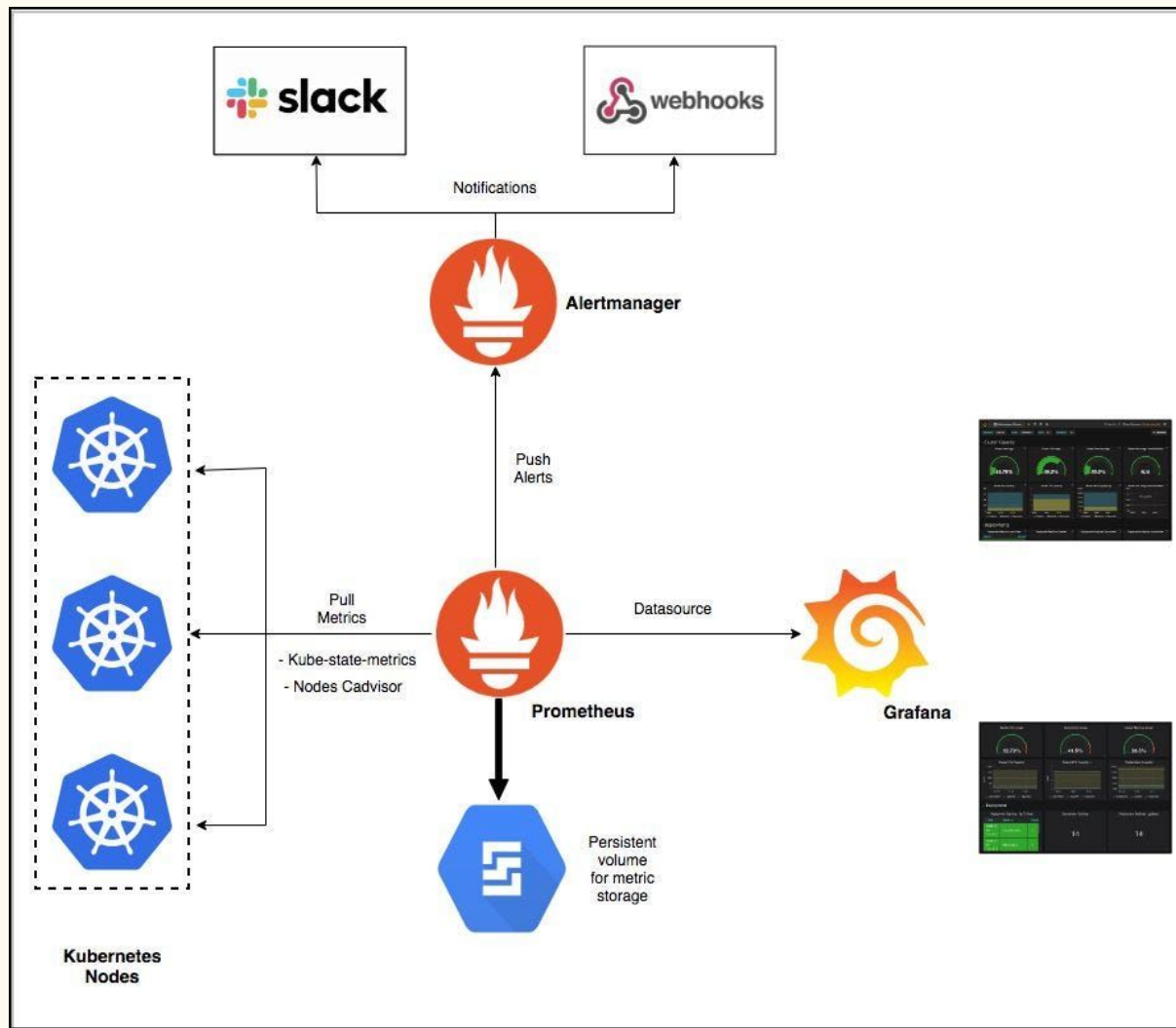


1. Prometheus almacena temporalmente las métricas organizadas por etiquetas.
-

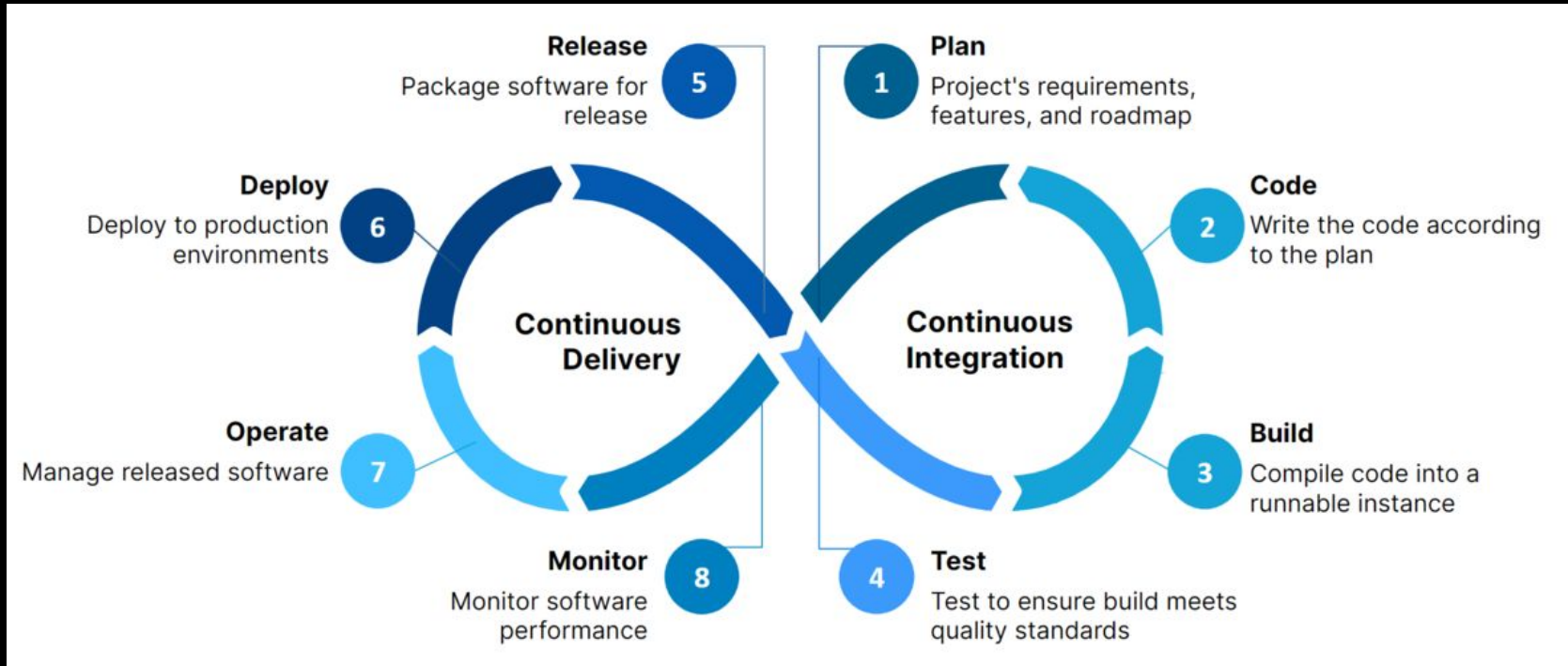
Grafana y Alertmanager



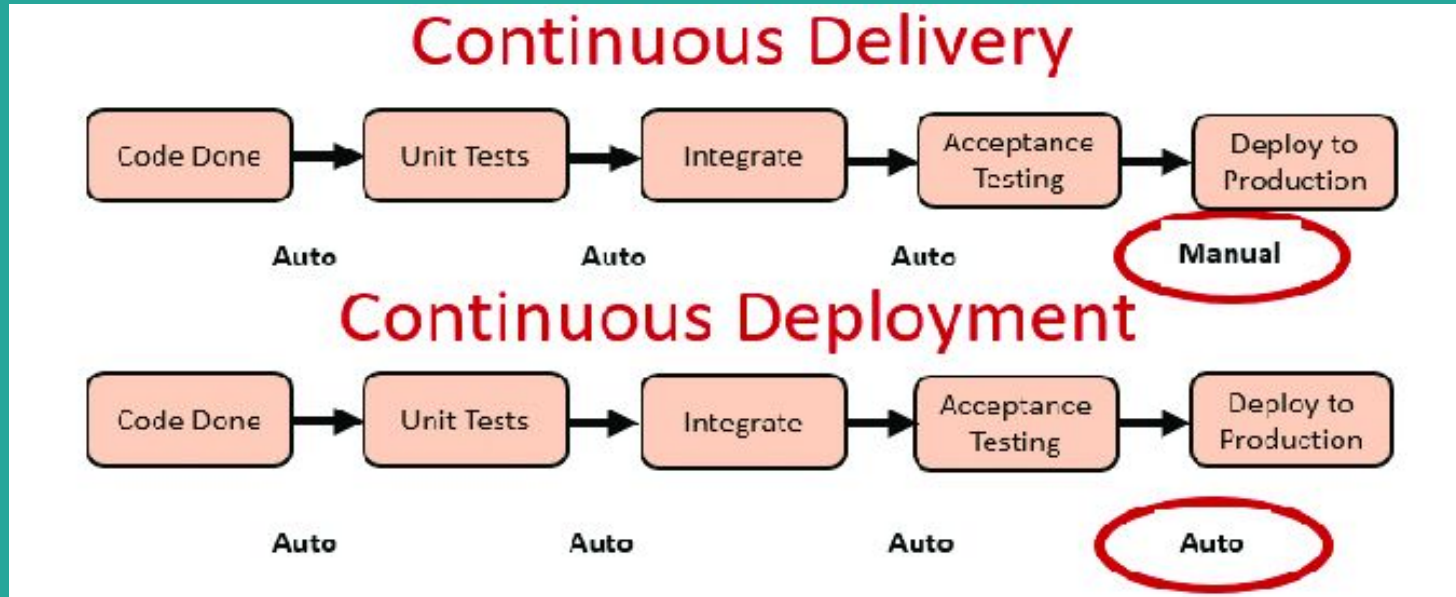
1. Grafana pide las métricas a la base de datos de Prometheus mediante el lenguaje PromQL para visualizarlas.
 2. Prometheus dispara una alerta que envía a Alertmanager cada vez que el umbral de una métrica es superado.
 3. Alertmanager envía la alerta a Slack o Webhooks para notificarla.
-



CI/CD (Integración continua / Despliegue continuo)



Continuous Delivery vs Continuous Deployment



La importancia de las pruebas automáticas

1. Pruebas Unitarias (Unit Tests)

Verifican el correcto funcionamiento de componentes individuales (funciones, clases o métodos).

2. Pruebas de Integración (Integration Tests)

Aseguran que los módulos o servicios interactúen correctamente entre sí (APIs, bases de datos, microservicios).

3. Pruebas de seguridad (Security Tests)

Identifican vulnerabilidades (inyección SQL, XSS, exposiciones de datos, configuraciones inseguras).