

Actividad 2

Que es IaC

La infraestructura como código consiste en proveer la infraestructura de computación usando código en vez de procesos y configuraciones manuales.

Uno de los principales beneficios es la duplicación de estructuras, lo que permite experimentar (agregar nuevos servicios para futuras características) en esta sin alterar la otra, evitar posibles errores de configuración por la instalación manual de la infraestructura.



Terraform

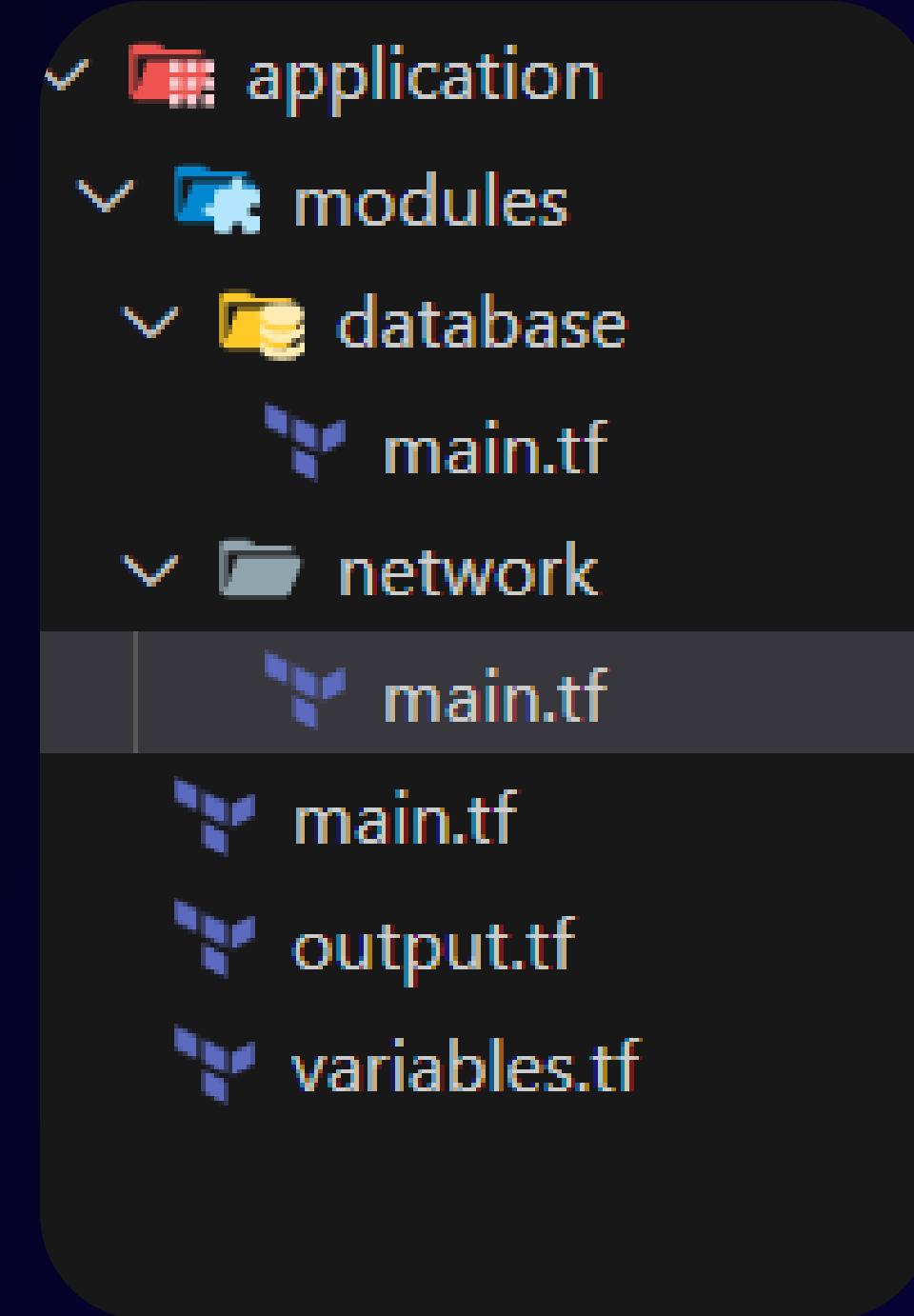
Terraform es una herramienta de infraestructura como código que permite crear, modificar y versionar infraestructuras de forma segura y eficaz. Esto incluye componentes de bajo nivel como instancias de computación, almacenamiento, redes y conexión a proveedores.



Módulos

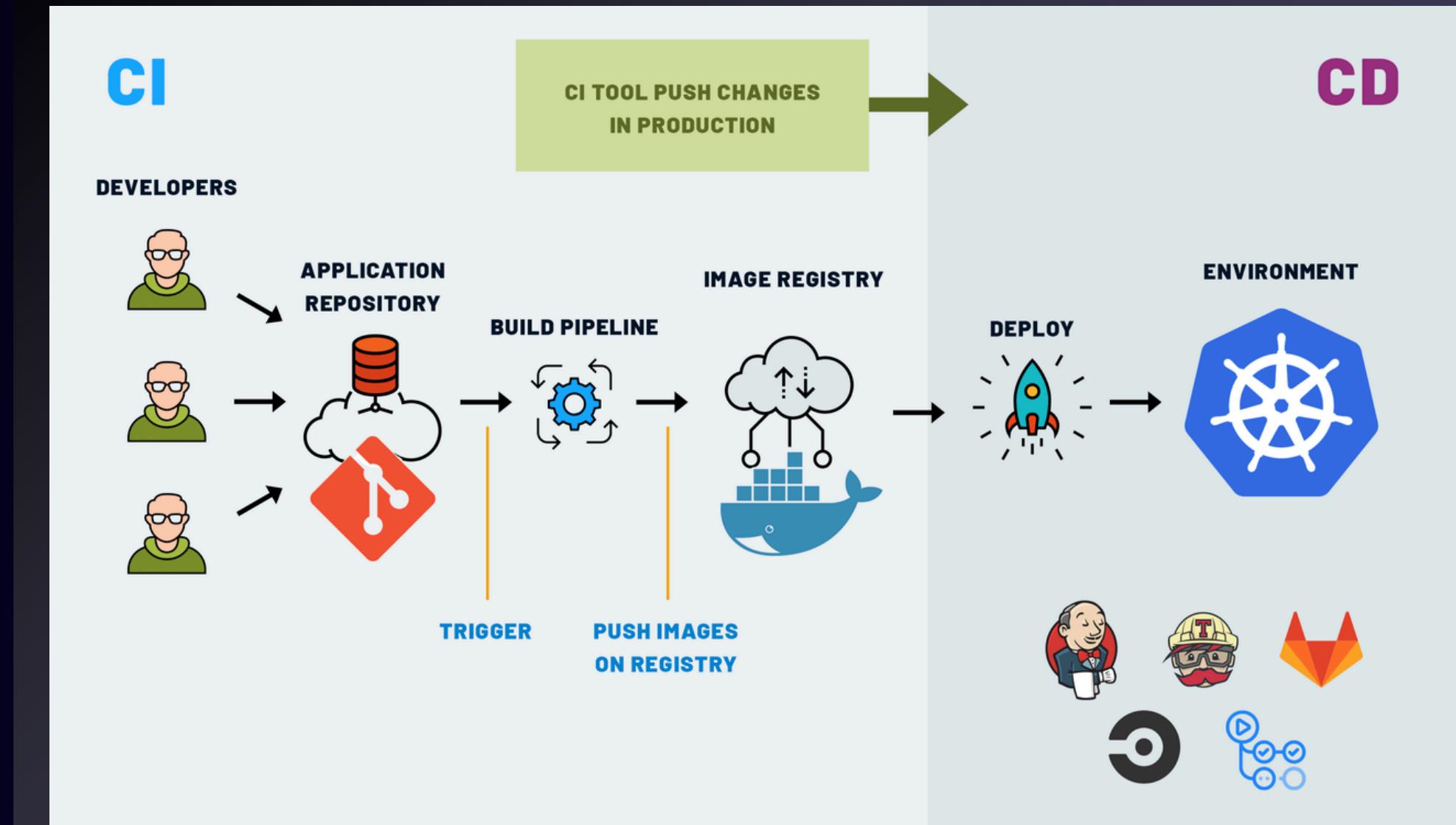
En los módulos de Terraform, se declaran los proveedores, recursos y variables. Para mantener un código limpio y modular, se suele dividir en varios archivos:

- main.tf : Define los recursos e incluye la configuración principal del módulo.
- variables.tf : Declara las variables de entrada que permiten personalizar el módulo.
- Output.tf : Define los valores de salida que se pueden usar en otros módulos o entornos.



Flujo de despliegue de un desarrollador

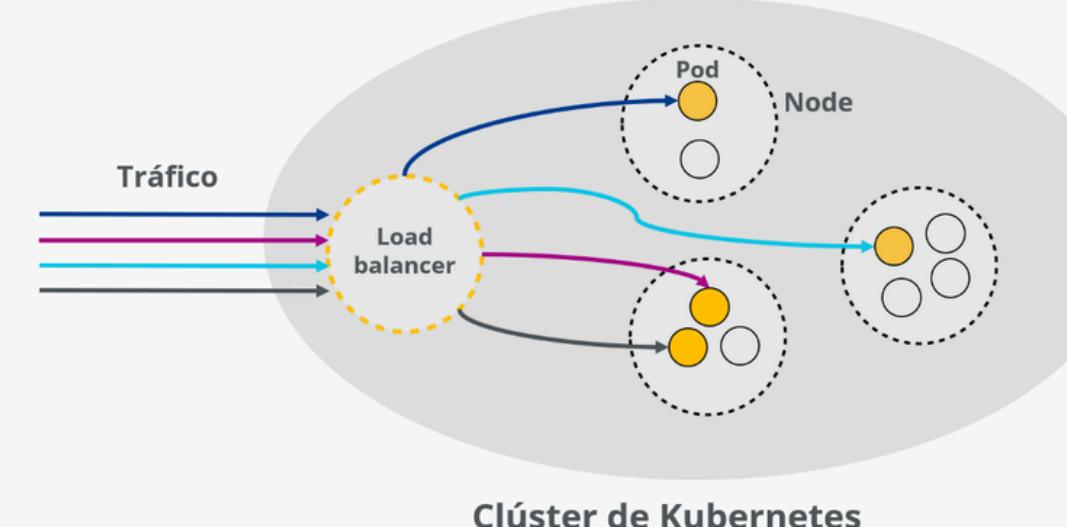
- El desarrollador sube los cambios al repositorio
- Se dispara una serie de comandos para construir una nueva imagen docker
- Se realiza el deploy



Ventajas de usar Kubernetes en un evento de alto tráfico

- Autoescalamiento: Kubernetes aumenta o disminuye el número de replicas de los pods en función del tráfico
- Balanceo de carga: Kubernetes distribuye el tráfico entre los pods evitando sobrecargas
- Disponibilidad: Si algún pod falla, Kubernetes reprograma los contenedores en algún otro pod disponible

Load balancer en Kubernetes

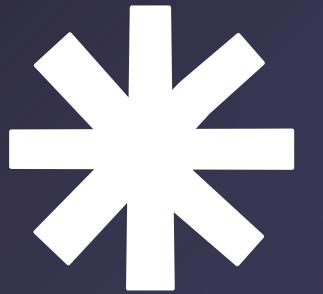


ONOS



Observabilidad

- Logs: Son registros de eventos.
- Métricas: Son datos cuantitativos que se usan para determinar el estado de salud ~~de la~~ aplicación.
- Trazas: Son registros que contienen la información necesaria para saber el recorrido de una solicitud a través de distintos componentes.



Ventajas

- Detección de errores.
- Optimización de recursos.
- Diagnósticos precisos.



Prometheus *

Recolecta las métricas, usa
PromQL



Grafana

Grafana
Permite visualizar las
métricas obtenidas con
Prometheus.



Integrando Prometheus y Grafana con Kubernetes¹

- Se inicializa el clúster.
- Se crea un namespace para el monitoreo (buenas prácticas)
- Se añade un repositorio de charts para Prometheus

```
helm repo add prometheus-community https://prometheus-community.github.io/helm-charts
```



Integrando Prometheus y Grafana con Kubernetes²

- Se instala Prometheus

```
helm install prometheus prometheus-community/prometheus --namespace monitoring
```

- Se añade un repositorio de charts para Grafana y se instala

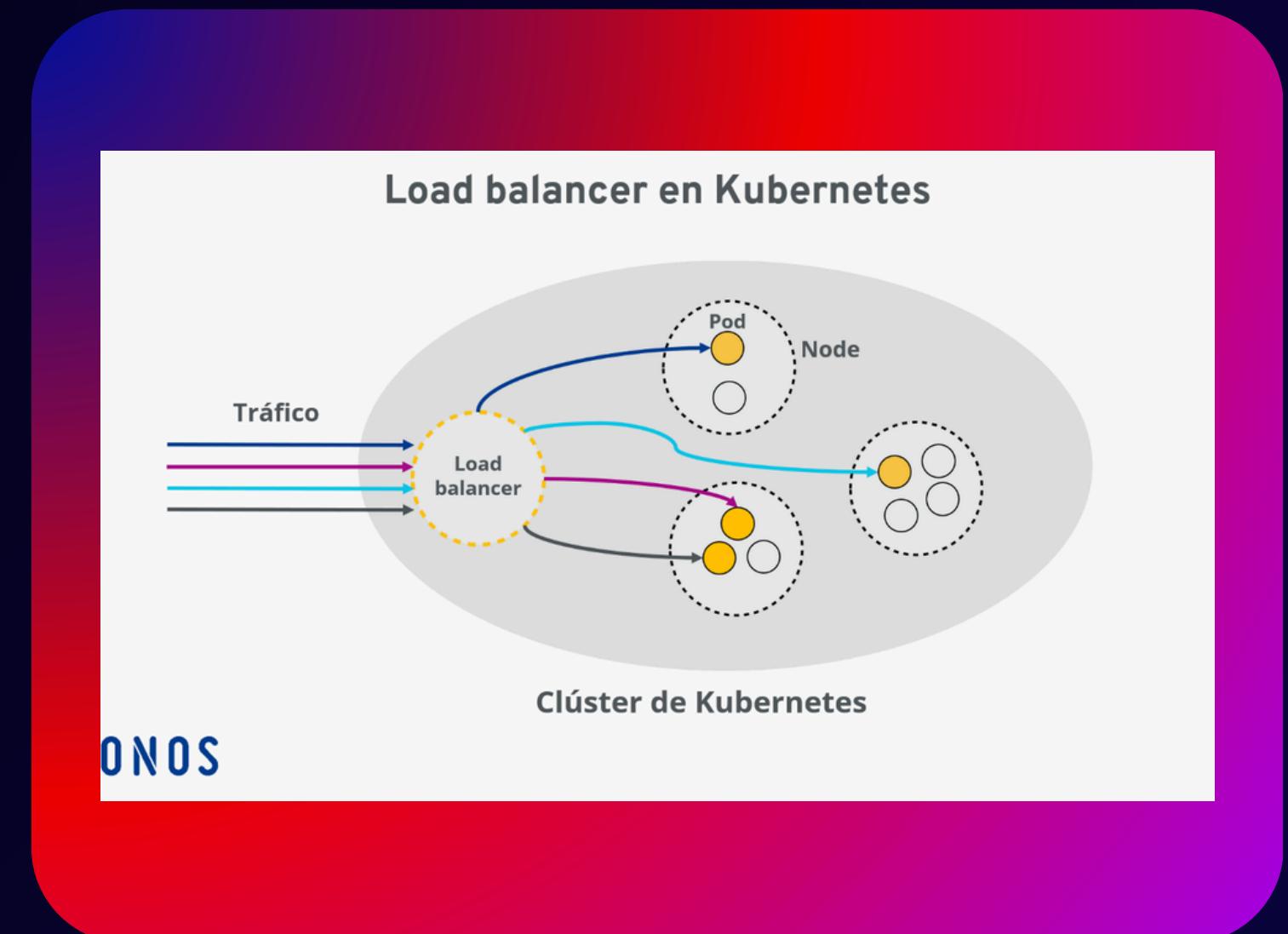
```
helm repo add grafana https://grafana.github.io/helm-charts
```

```
helm install grafana grafana/grafana --namespace monitoring
```

Métricas y alarmas para una página web

Métricas

- Uso de CPU/memoria
- Tasa de errores
- Latencia de peticiones
- Latencia de red
- Tasa de paquetes perdidos

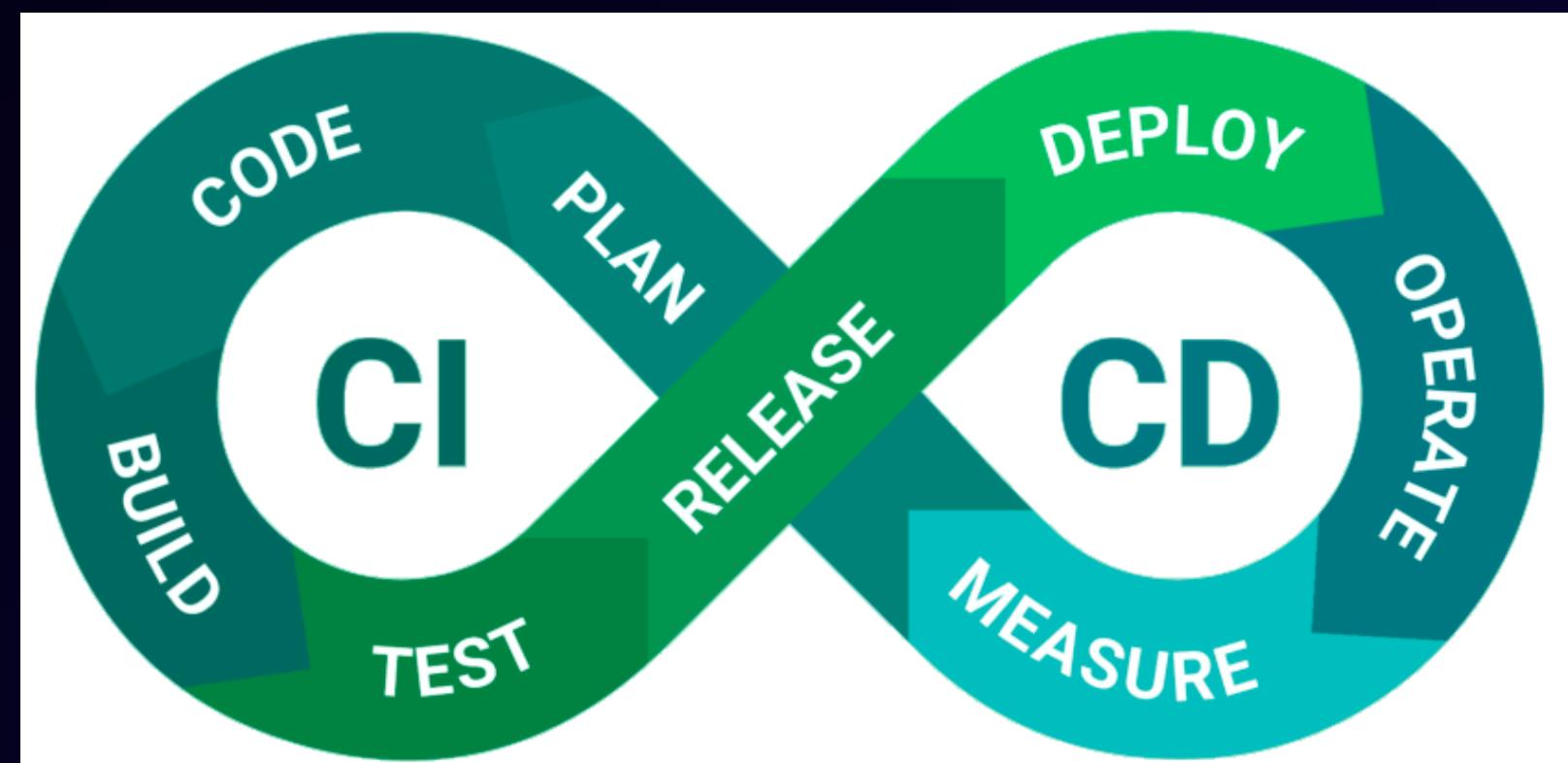


Entrega continua

- Se requiere aprobación antes de desplegar los cambios en producción
- Los cambios en el código son probados automáticamente

Despliegue continuo

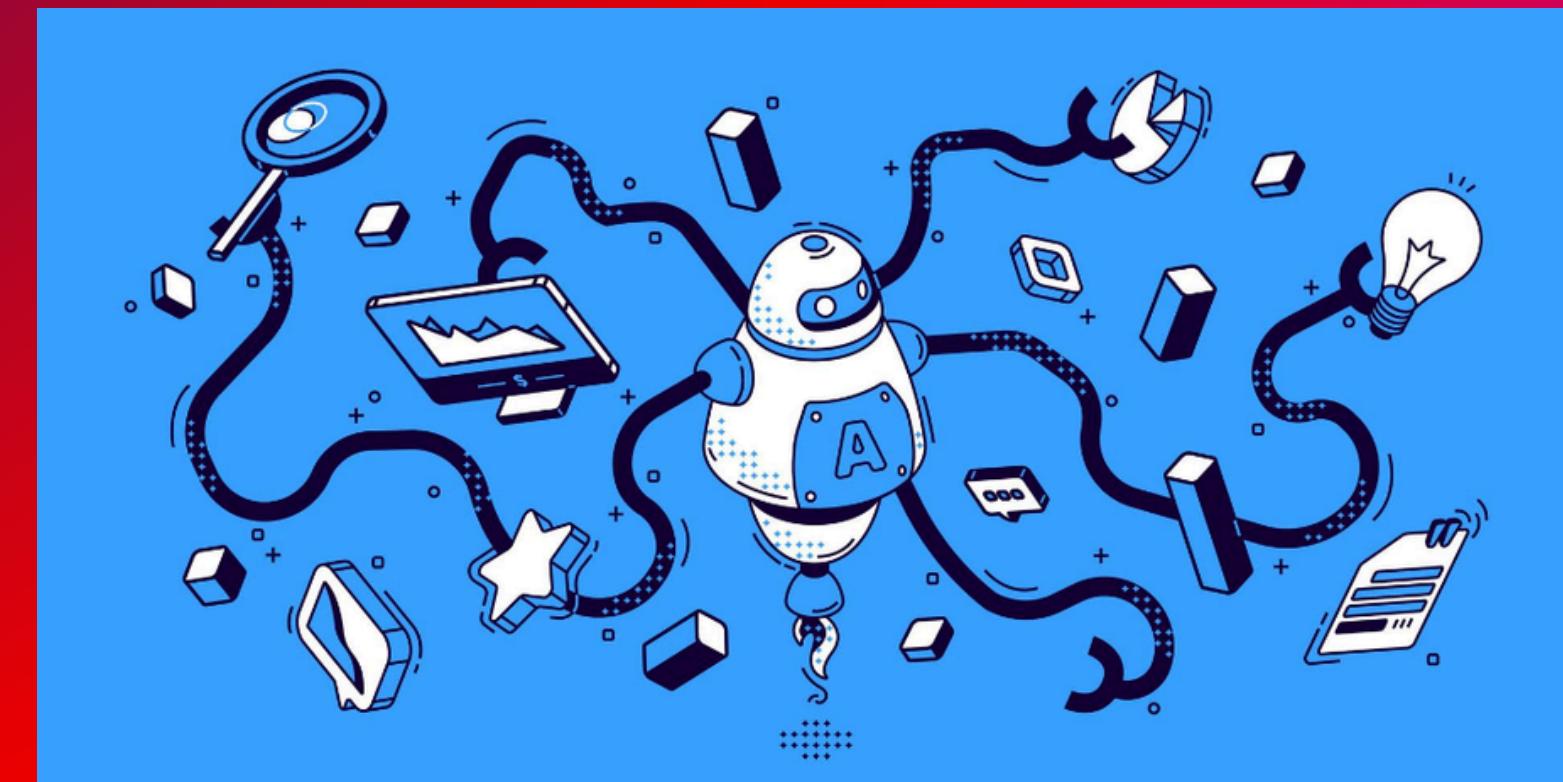
- Cada cambio que pasa por las pruebas es desplegado a producción automáticamente
- No hay intervención humana y presentan mayor riesgo si las pruebas no son muy robustas



Importancia de las pruebas automáticas

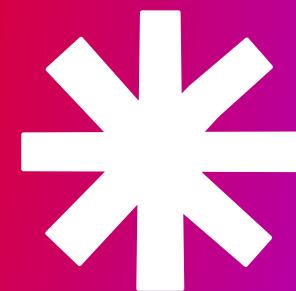
Las pruebas automáticas son fundamentales para garantizar la calidad del software y la velocidad de desarrollo en CI/CD

- Pruebas unitarias: Garantizan la modularidad del software
- Pruebas de integración: Garantizan que las dependencias interactúen correctamente
- Pruebas de seguridad: Garantizan que los datos sensibles estén protegidos



Collaboration in Programming

Working
Together in Code



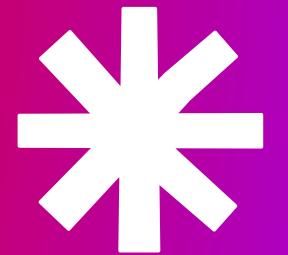
Programming often requires collaboration, especially in large projects. Developers use version control tools like Git to manage code, allowing multiple programmers to work on the same project seamlessly.



Testing and debugging are crucial steps in programming. Testing verifies that code functions as expected, while debugging identifies and resolves errors. By carefully testing and refining code, programmers ensure their applications are reliable, user-friendly, and free of bugs, enhancing the end-user experience.

Testing and Debugging

Working
Together in Code



AI, Automation, and Beyond

The Future of Programming

The future of programming is evolving with advancements in artificial intelligence, machine learning, and automation. As programming languages and tools become more sophisticated, programmers can tackle complex tasks more efficiently. These developments open up new opportunities for innovation, shaping the way we interact with technology.

Thynk Unlimited

Thank You