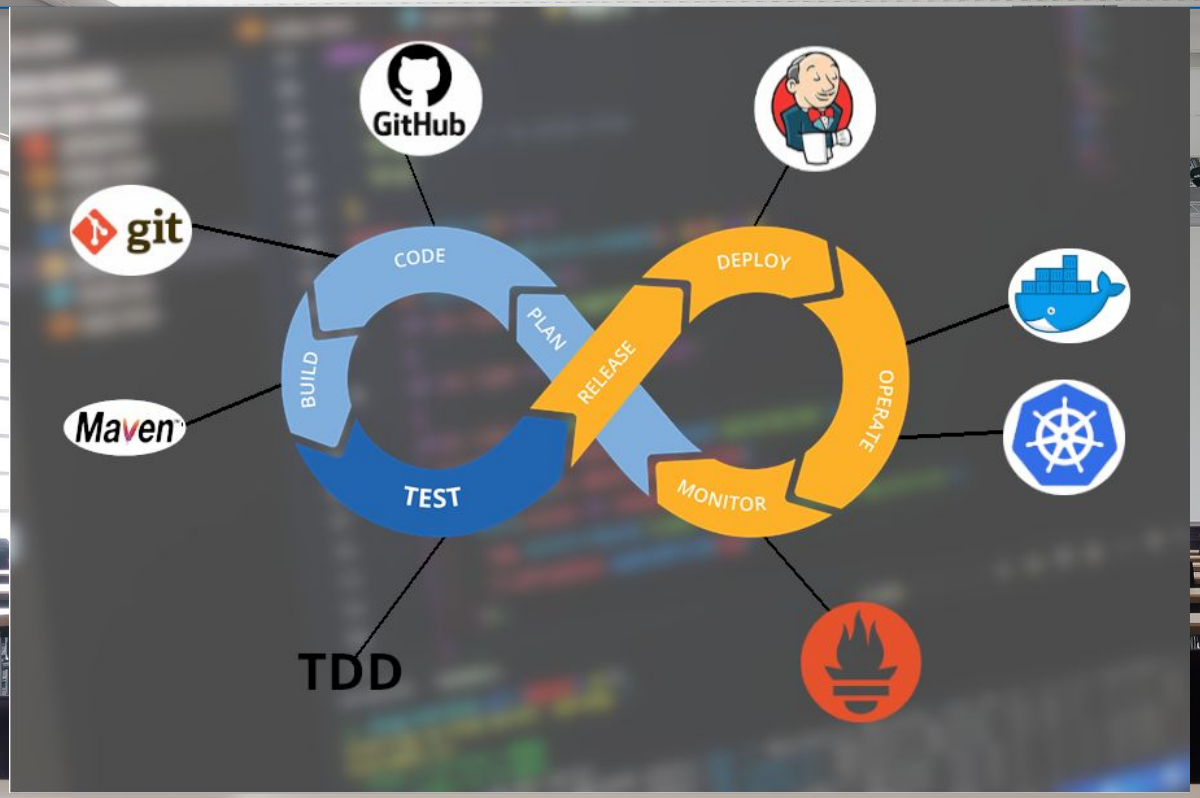
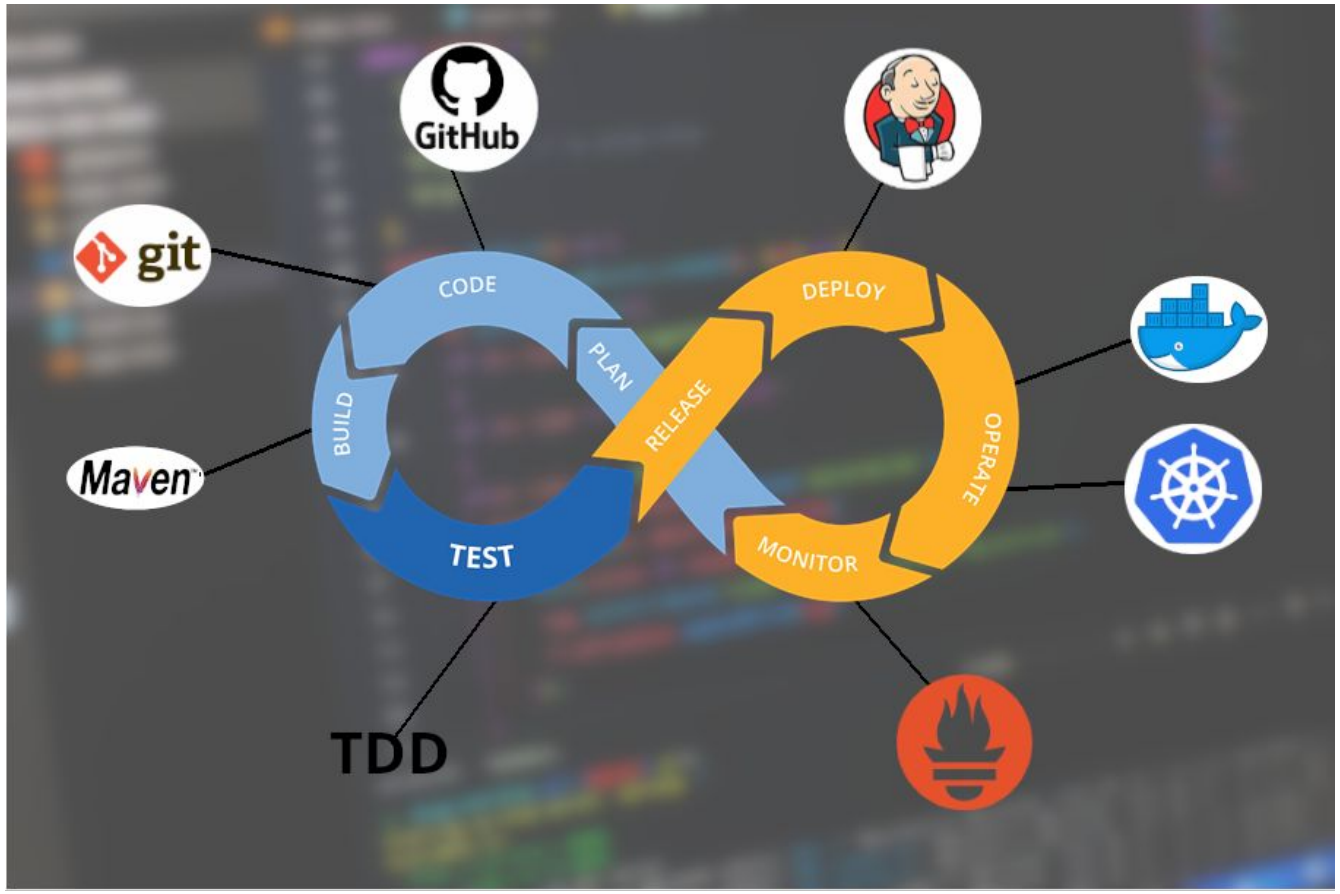
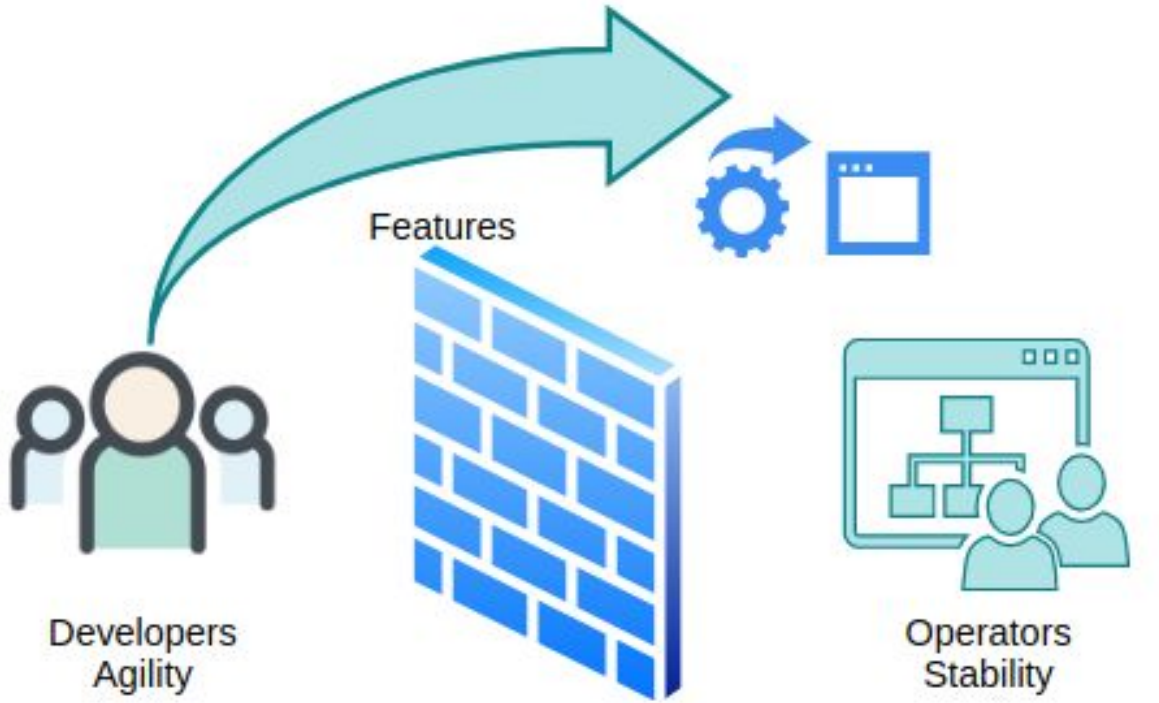
# docker,kubernetes, jenkins y gitflow CI/CD



- Conceptualización



- Developer vs SysAdmin



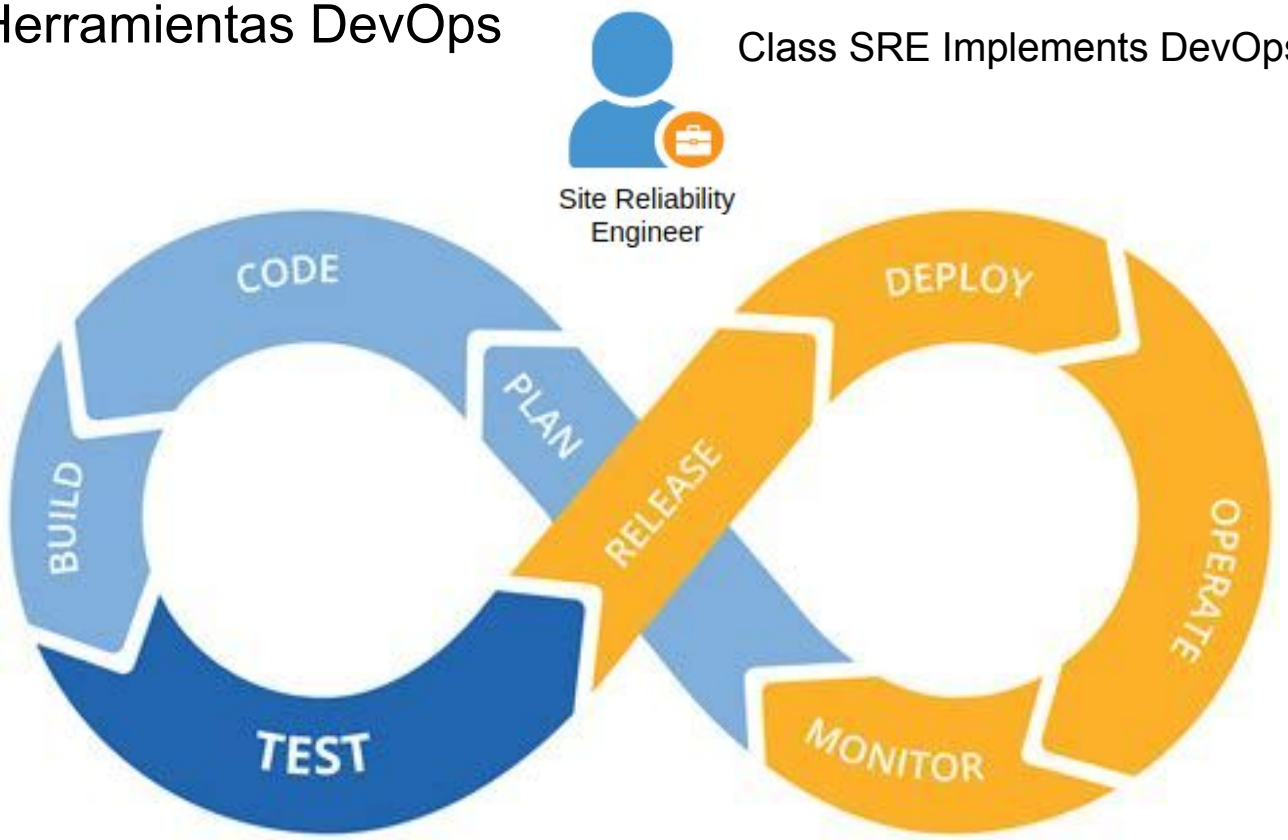

- Aislamiento de equipos
- Calendarios de releases fijos
- Puesta en producción más lenta
- Fricción entre los equipos
- Baja automatización
- Scripts rudimentarios

# ¿Que es DevOps?

@sotobotero

Es la unión de personas, procesos y tecnología para proporcionar valor continuamente a los clientes. Un compuesto de Desarrollo (Dev) y Operaciones (Ops).

- Cultura que involucra a Desarrolladores, Operaciones, Calidad y Seguridad.
- Conjunto de practicas
- Herramientas DevOps




Class SRE Implements DevOps

- Alto Rendimiento
- Mejores productos
- Entrega más rapida
- Satisfacción del cliente
- Mejora objetivos comerciales
- Métricas
- Automatización

A partir de 2011 Amazon.com implementó un cambio en producción cada 11.6 segundos sin impactar al usuario final . Ref Microservices Patterns / Christ Richarsond

## Aspectos fundamentales de DevOps

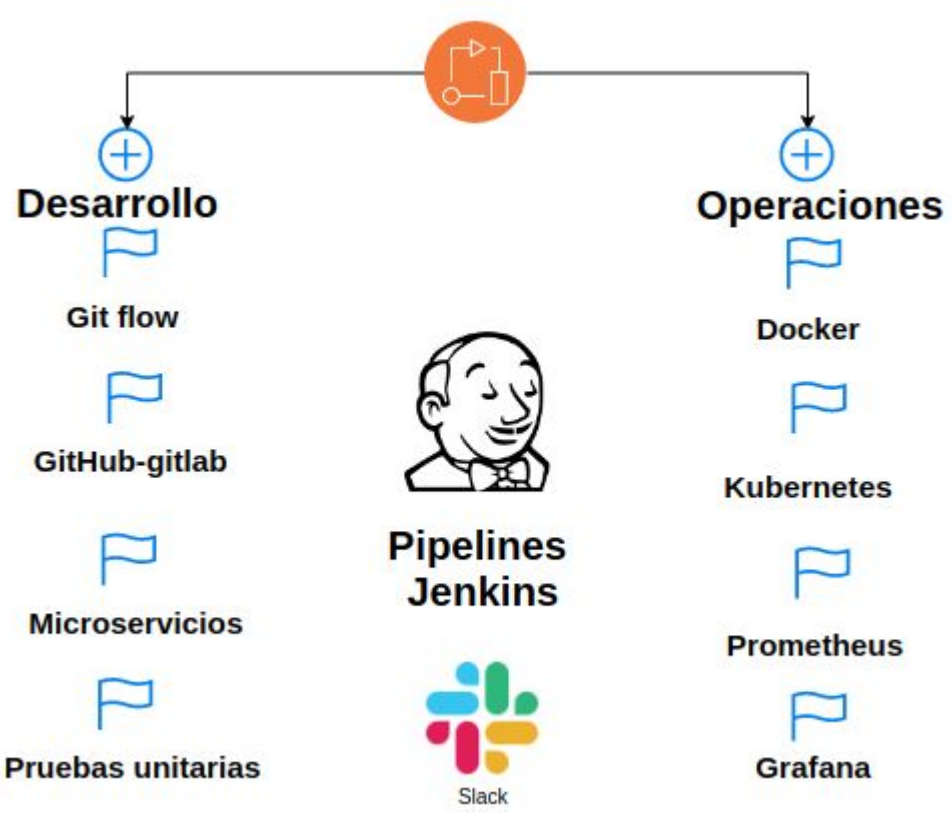

- **Control de versiones:** git, svn, -> Github, gitlab, bitbucket, mercurial
- **Integración continua:** Pipelines jenkins -> Automatizar compilaciones y pruebas cuando se hace commit
- **Entrega continua:** Suministro de software rápido y confiable en cualquier momento.
- I**nfraestructura como código:** Terraform, kubernetes -> Definición declarativa de la infraestructura mediante archivos de definición basados en texto. para revertir, desmontar y recrear entorno complejos.
- **Supervisión y registro:** Prometheus, grafana -> Monitorización, recopilación de métricas y vinculación de datos de performance.
- **Aprendizaje validado:** Análisis de datos para mejorar los procesos en cada nuevo ciclo.

# Estructura del curso

Site Reliability Engineer
(SRE)

50% Operaciones

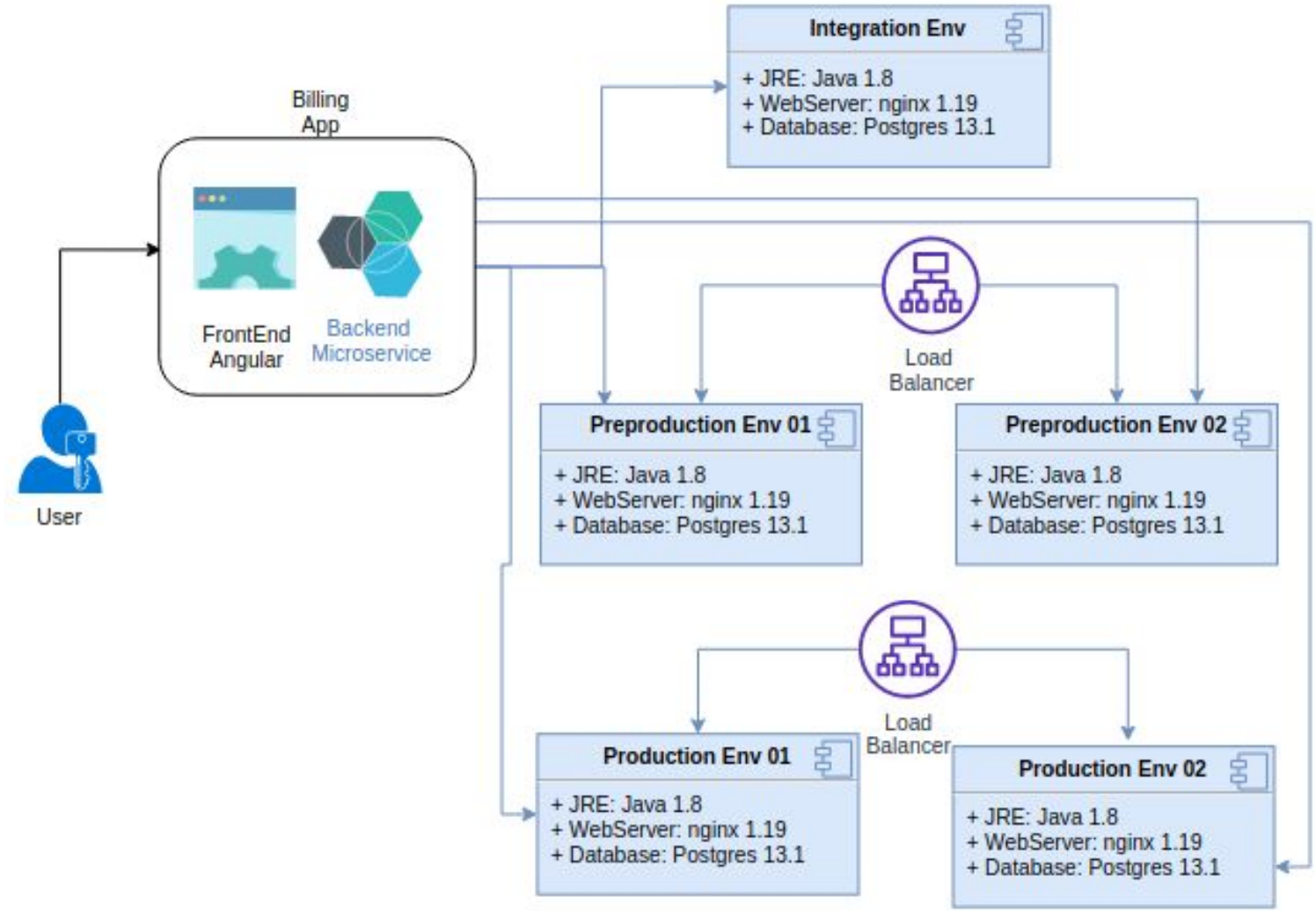
50% Desarrollo



# Caso de estudio



Se requiere desplegar la aplicación de facturación de la compañía, en los entornos de integración, preproducción y producción, las instalación debe contar alta disponibilidad, excepto en integración




**Requisitos técnicos:**

- Java 1.8
- Servidor Web Nginx
- Base de datos Postgres

# Docker Volúmenes y puertos

@sotobotero

**Local filesystem path : container mount path**

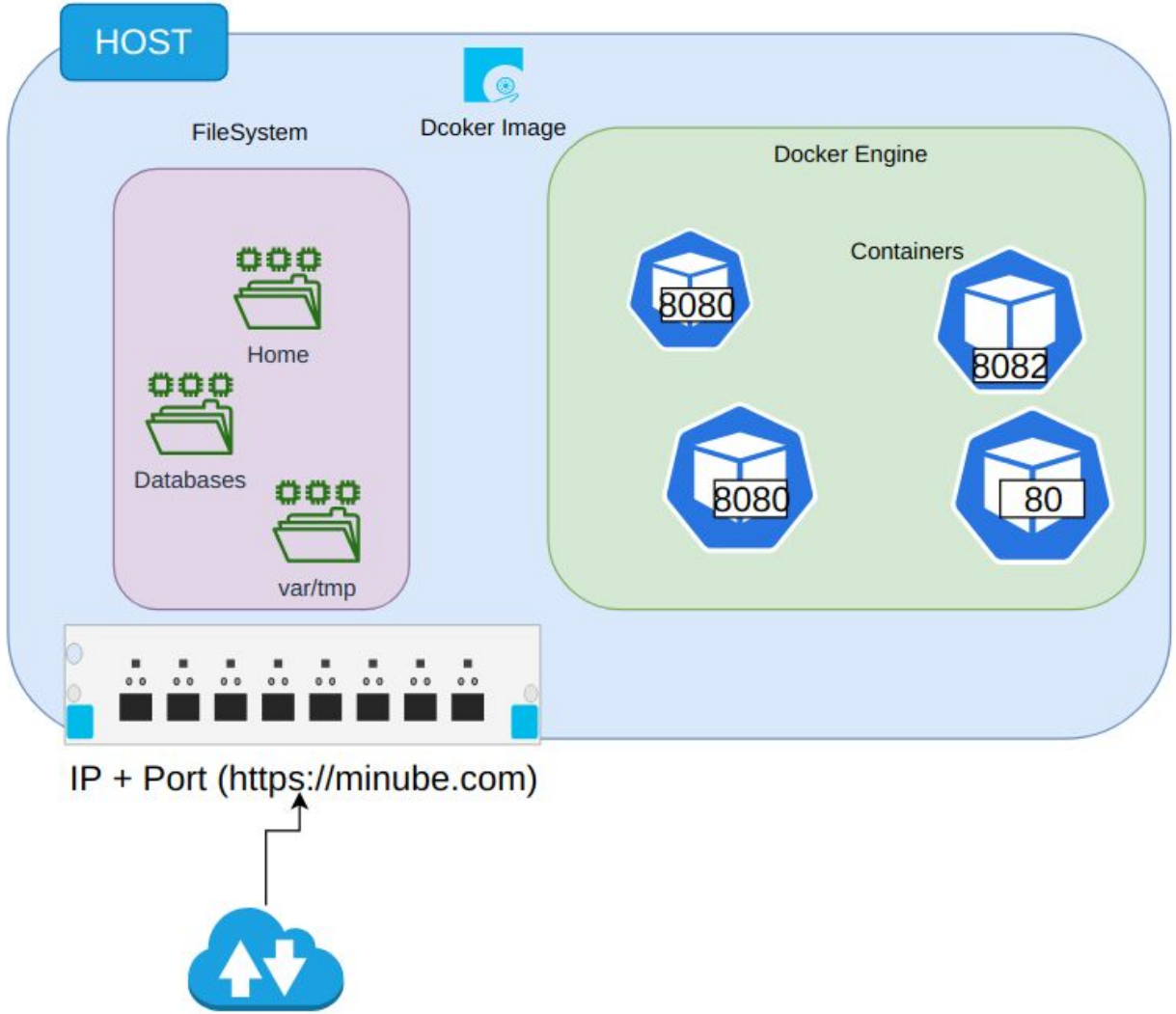
**Ejemplo 1:**
**- /var/lib/postgres_data:/var/lib/postgresql/data**

**Ejemplo 2:**
**- /home/bds/postgres:/var/lib/postgresql/data**

**Local port : container port**

**Ejemplo 1:**
**8080:8080**

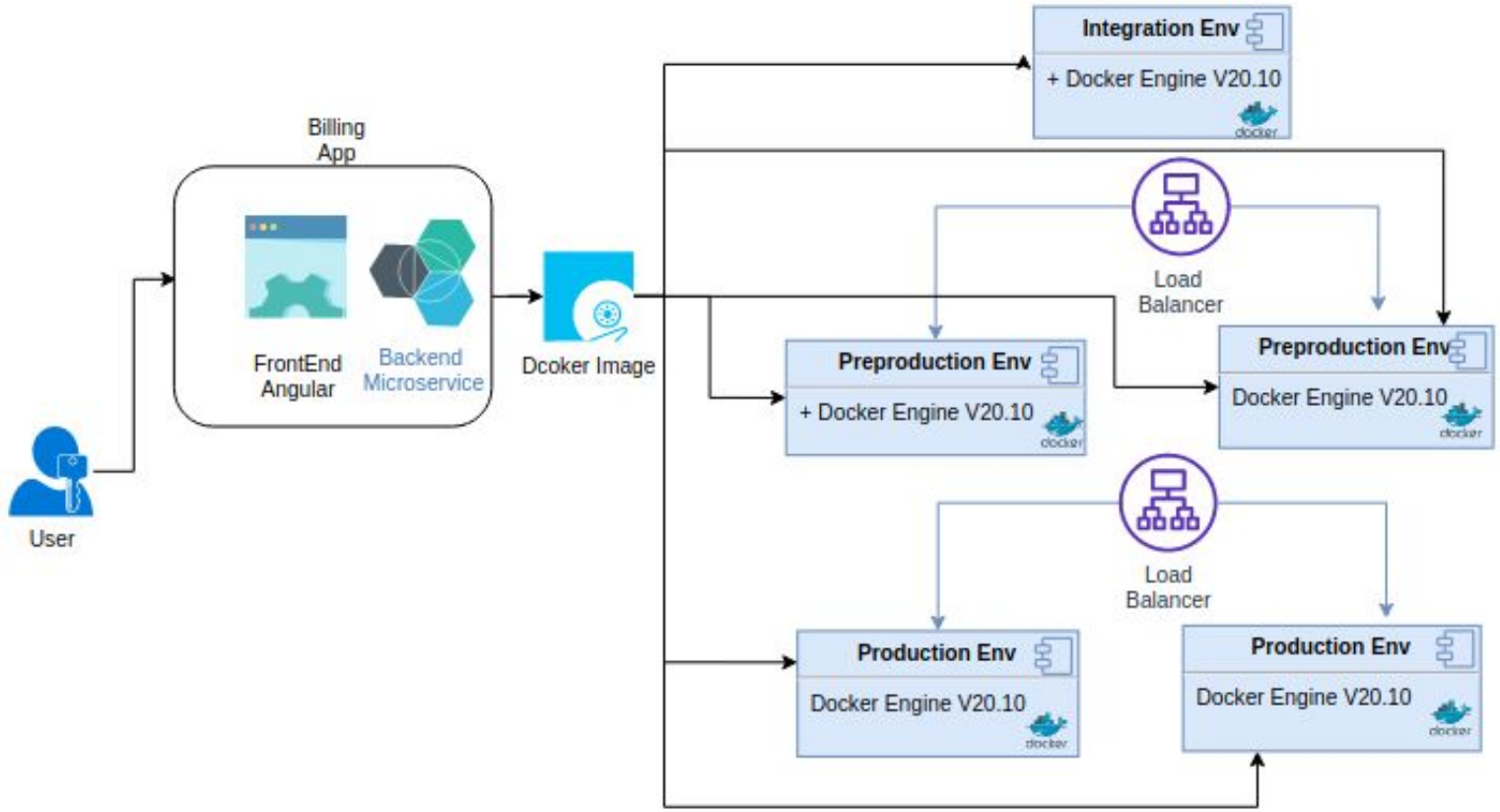
**Ejemplo 2:**
**8082:8080**



# Caso de estudio



Se requiere desplegar en producción la a aplicación facturación de la compañía, en los entornos de integración, preproducción y producción, las instalación debe contar alta disponibilidad, excepto en integración.

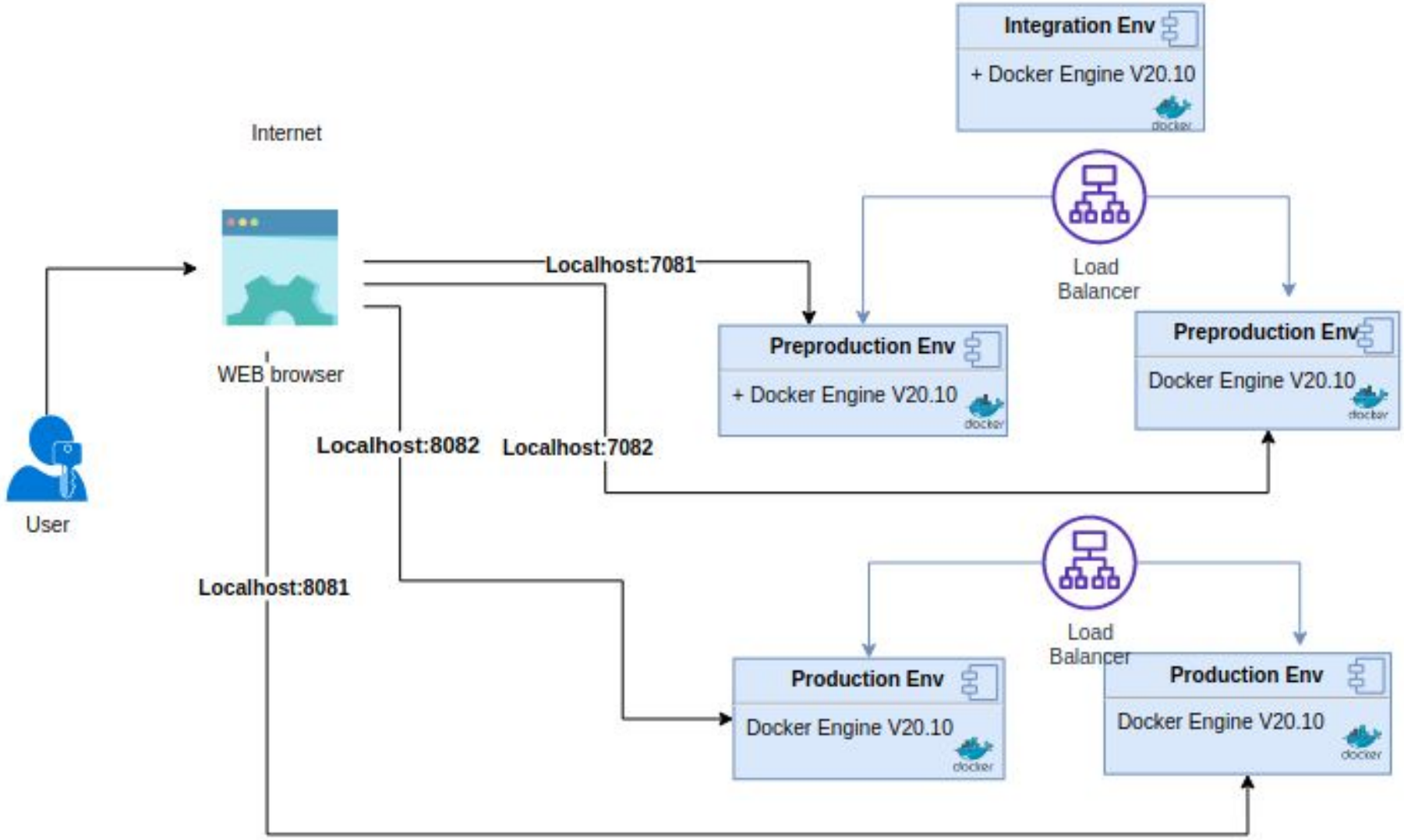



**Requisitos técnicos:**

- Java 1.8
- Servidor Web Nginx
- Base de datos Postgres

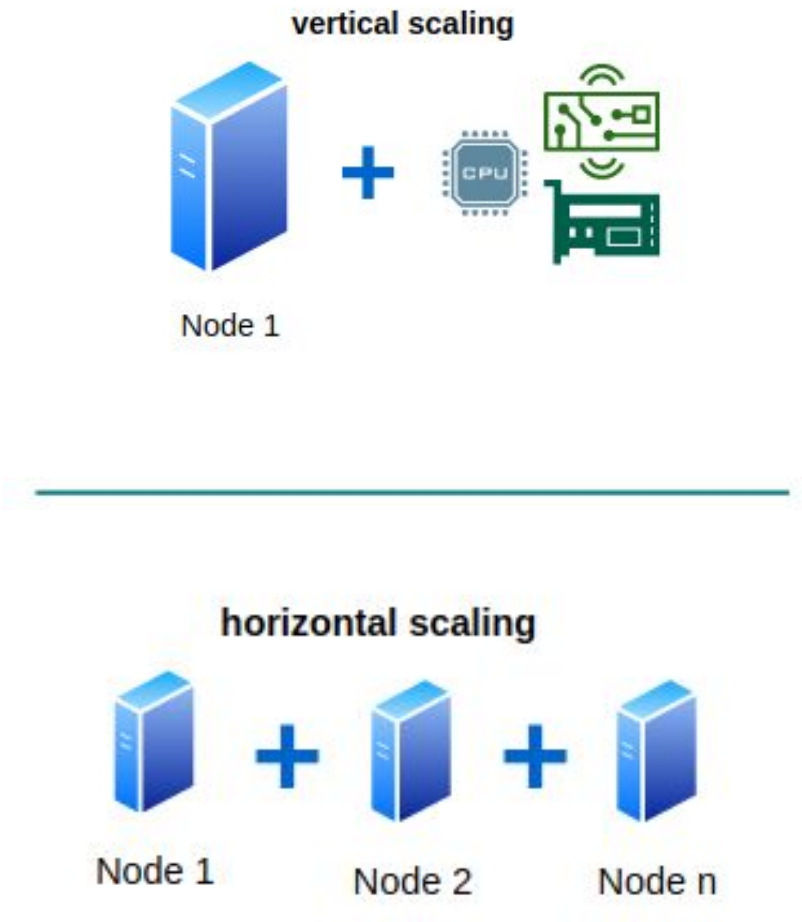
# Docker Network - virtual environments

@sotobotero



# Escalabilidad






Capacidad de los sistemas para adaptarse al crecimiento.
Por demanda y complejidad

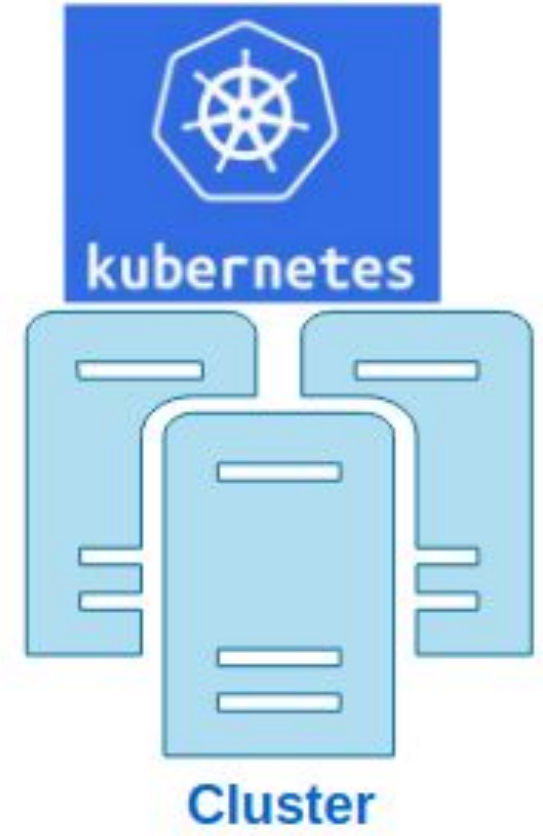
Sistema diseñado inicialmente para una carga de menos de 100 usuarios, y con el tiempo su demanda de usuario aumenta a 500, 8000, 60000 etc. A medida que aumenta los usuarios y/o la complejidad del sistema se requieren más recursos (RAM, procesamiento, Almacenamiento etc)

**Vertical:** Agregar más recursos al mismo nodo y aumentar el poder de cómputo.

**Horizontal:** Agregar más nodos que se adapten a la carga de trabajo.

# kubernetes

@sotobotero




Es un sistema opensource para automatizar el despliegue, escalado y administración de aplicaciones en contenedores. También conocido como k8s (pronounced Kate's).

- Opensource
- Auto-Sanado
- Escalado horizontal
- Balanceo de carga y discovery
- Licencia Apache v2
- Creado por google en 2015
- Escrito en Go
- Ultima version estable 1.20(2021)
- Mantenido por Cloud Native Computing Foundation (parte de linux foundation).

Inspirado en el sistema Borg de google, administrador de clusters capaz de operar cientos de miles de trabajos de miles de Aplicaciones diferentes en varios clusters cada uno con hasta decenas de máquinas.

# Principales Componentes

@sotobotero

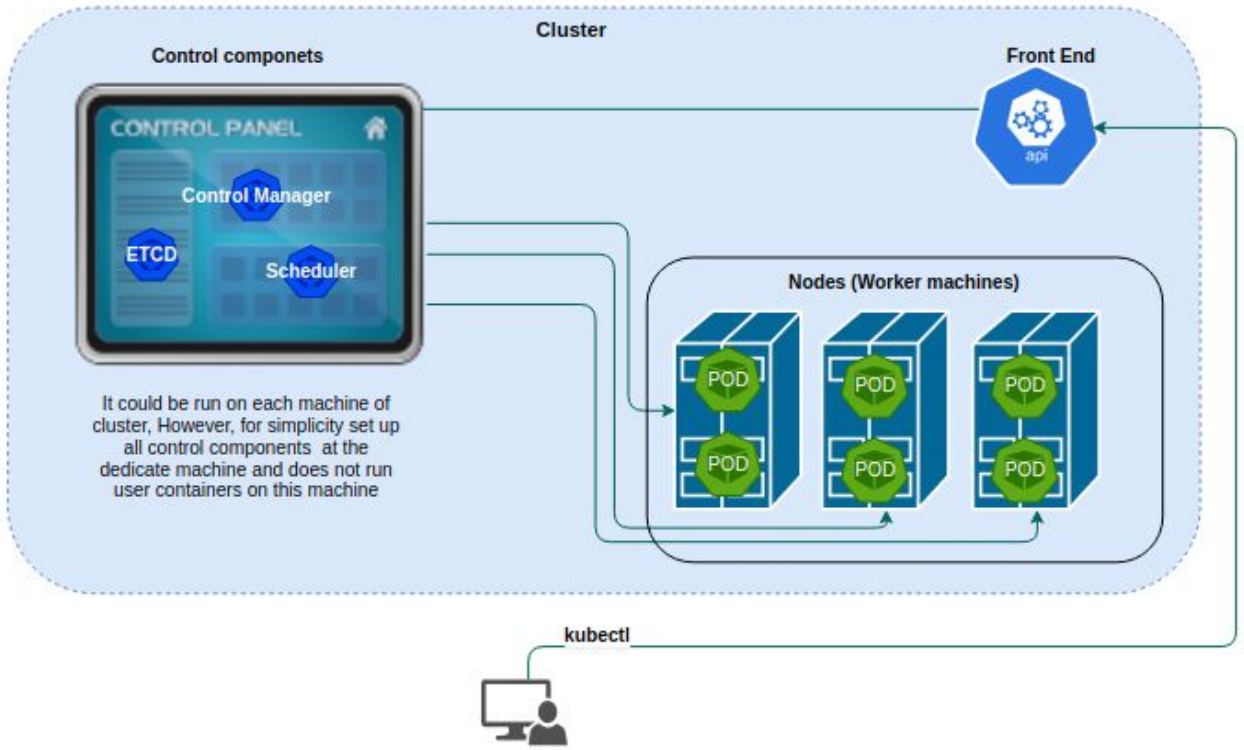
Todo se conoce como Objetos.
Peticiones de definición se hacen mediante la API y se almacenan en la base de datos etcd en formato yaml.

- **PODS:** Unidad más pequeña que se puede desplegar y gestionar en kubernetes. Es un grupo de uno o más contenedores que comparten almacenamiento y red y especificaciones de cómo ejecutarse. Son efimeros.
- **Deployments:** Describe el estado deseado de una implementación, ejecuta múltiples réplicas de la aplicación , reemplaza las que estan defectusas o las que no responden.
- **Services:**Definición de cómo exponer una aplicación que se ejecuta en un conjunto de pods como un servicio de red (por defecto se usa roud-robin para balanceo de carga).
- **Config Map:** Permite desacoplar la configuración para hacer las imagenes mas portables, almacenan variables de entorno, argumentos para línea de comandos, o configuración de volúmenes que pueden consumir los pods (no encriptación).
- **Labels:** Pares de clave valor (`"environment" : "qa"`) para organizar, seleccionar, consultar y monitorear objetos de forma más eficiente, ideales para UI y CLIs.
- Selectores: mecanismo para hacer consultas a las etiquetas.

```
kubectl get pods -l 'environment in (production),tier in (frontend)'
```

# kubernetes Arquitectura

@sotobotero




Cuando implementa Kubernetes, obtiene un clúster.
Un clúster de Kubernetes consta de un conjunto de máquinas trabajadoras, llamadas nodos, que ejecutan aplicaciones en contenedores. Cada clúster tiene al menos un nodo trabajador.
Los nodos trabajadores alojan los pods que son los componentes de la carga de trabajo de la aplicación.

**kube-apiserver:** Provee la interacción para las herramientas de administración kubectl or the Kubernetes dashboard.

**etcd:** Almacenamiento mantiene la configuración y el estado del cluster.

**kube-scheduler:** All crear o escalar la aplicaciones selecciona el nodo para los pods y los ejecuta.

**kube-controller-manager:** Supervisa controladores más pequeños que ejecutan tareas de replicar pods y manejar operaciones de los nodos.

❖ **All in one:** Se instala todo en un único nodo usando mini kube por propósitos educativos y

# Kubernetes Managed / Onpremise y Tipos de instalación



❖ **Single master and multiworker:** Un nodo para el control panel y uno o más nodos controlados por el master.

❖ **Single master, single etcd and multiworker:** Un nodo para el control panel, un nodo para almacenar la configuración y el estado y uno o más nodos controlados por el master.

❖ **Multi master, and multiworker:** Múltiples nodos para el control panel en alta disponibilidad y uno o más nodos controlados por el master en HA.

❖ **Multi master,multi etcd and multiworker:** Múltiples nodos para el control panel y múltiples nodos para el almacenamiento etcd en alta disponibilidad y uno o más nodos controlados por el master en HA.

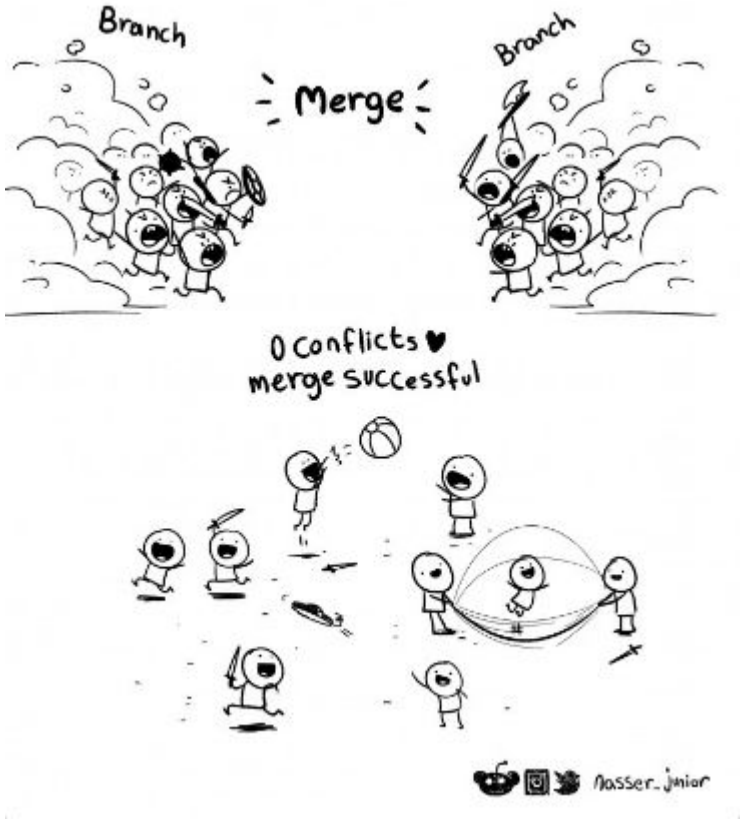
## Managed Kubernetes

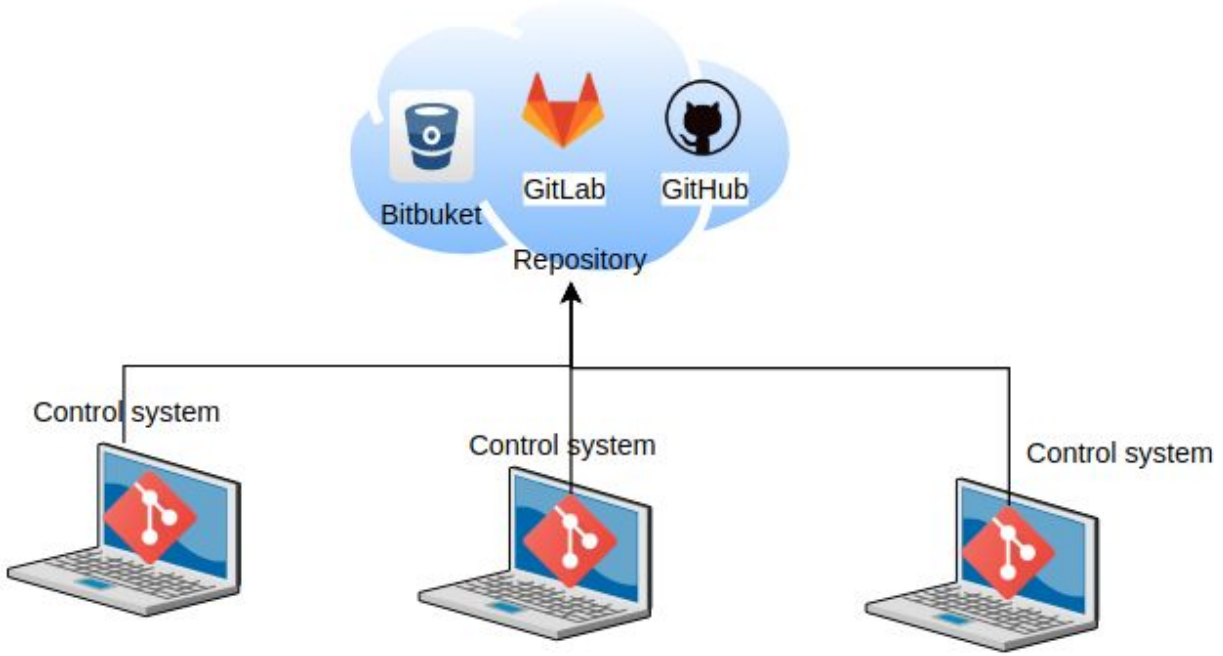
- AKS: Azure kubernetes service
- EKS: Amazon Elastic Kubernetes Service
- GKE: Google Kubernetes Engine
- IBM Cloud

# Git and repositories

@sotobotero

**Git:** Git es un sistema de gestión de versiones distribuido, opensource y se puede integrar con diferentes repositorios (puede usarse para controlar versiones de código, instalables, documentos etc).

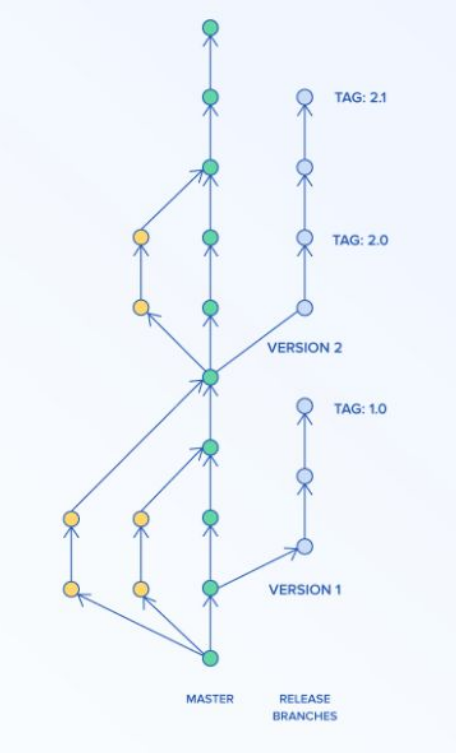


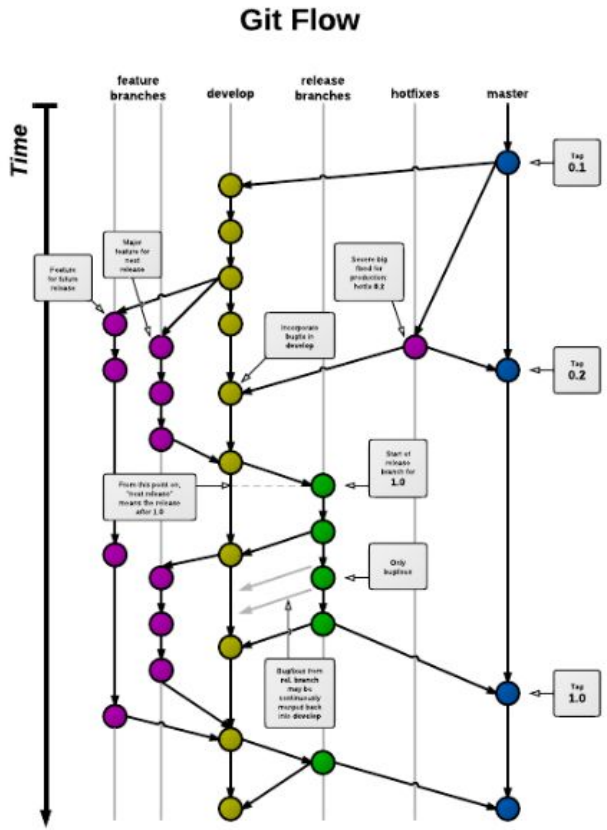



# Main concepts and Techniques/ Styles

@sotobotero

## Git Flow Vs Git trunk Based




**Trunk:** Es  la línea base del repositorio.

**Branch:** Nueva bifurcación del estado actual del código que crea una nueva ruta para evolucionarlo

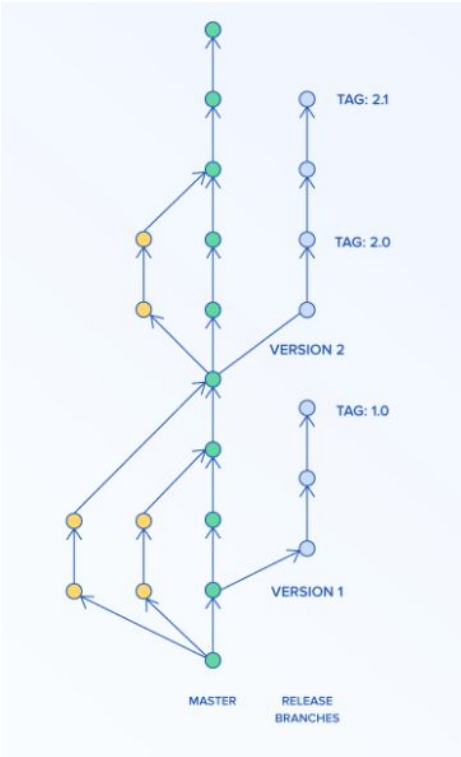
**Fork:** Duplica el repositorio y su historial.

**Tag:** Etiquetas para identificar versiones del código en un momento dado (snapshot) es como moverse a una rama, pero sin poder modificar ni hacer commit, a menos que se cree una rama.

# Git Trunk Based

@sotobotero

**Una única rama master de la cual parten los desarrolladores y crean ramas para sus características, cuando su rama está completa y ha sido probada se envía un MR a la rama principal (trunk/master)**

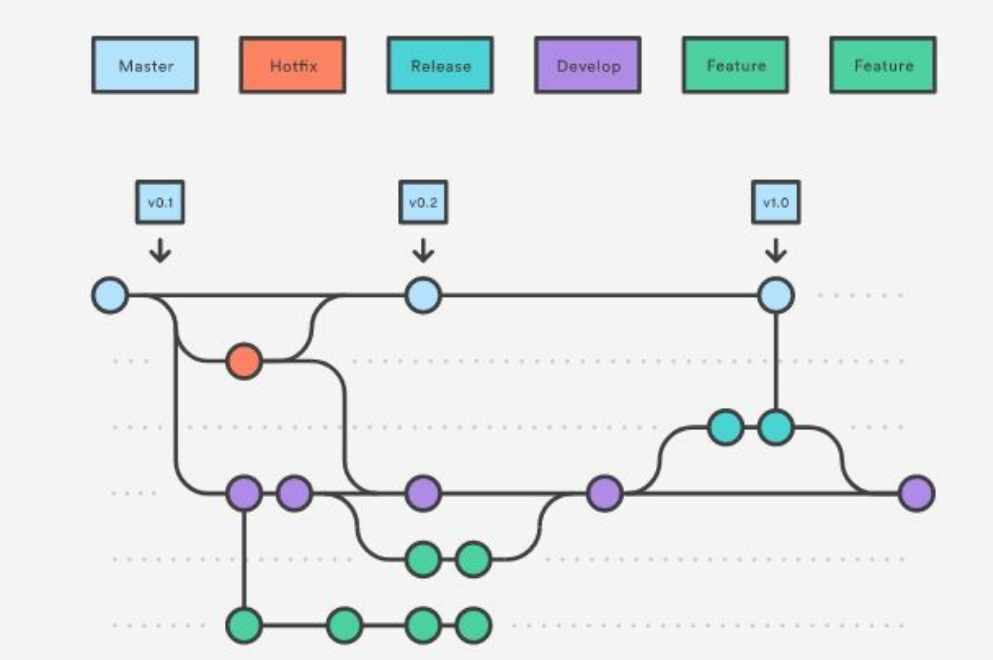



1. Crear una rama de ciclo de vida corto partiendo del trunk/master.
2. Realice los cambios aquí y haga los commits a esta rama. Se debe probar de manera exhaustiva locamente y el servidor de CI.
3. Cuando esté listo para hacer un merge, actualice su rama con los cambios de la rama trunk/master para evitar conflictos y agilizar la integración.
4. Haga el Mr de su rama al tronco principal.

# Git Flow

@sotobotero

**GitFlow:** Es una metodología de flujo de trabajo aplicado a un repositorio git.




1. **master:** Almacena el historial de versiones oficiales, nunca recibe código directo.
2. **develop:** Funciona como rama de integración de las features, nunca recibe código directo.
3. **features:** parten de develop, son temporales, es aquí donde los desarrolladores agregan el codigo de nuevas características(`feature/myfetaure`).
4. **release:** Parten de develop, preparación de la release, se despliega en entorno de pruebas, se hacen ajustes y se integra en master y develop. (release-x.y.z)
5. **hotfix:**Parten de máster, para arreglar errores urgentes, se integran en master y dev y se marca la versión con un tag. (hotfix-x.y.z)

# Gitflow vs trunk based

@sotobotero

**Trunk based** Estándar usado para equipos de ingeniería de alto rendimiento.

## Trunk-Based

1. Es una práctica requerida para la integración continua.
2. Cuando se está iniciando un proyecto
3. Cuando se requiere iterar rápido
4. Cuando el equipo de desarrollo principalmente está integrado por seniors.
5. Cuando se trabaja con un enfoque de TDD.

## Gitflow

1. Ideal para proyectos open source
2. Cuando el número de desarrolladores junior es alto
3. Si el índice de rotación del equipo es alto.
4. Cuando ya se cuenta con un producto establecido (en producción)
5. Calendario de releases fijo

# Tools

@sotobotero

## Instalar y configurar Git y crear una cuenta en un repositorio gratuito GitHub

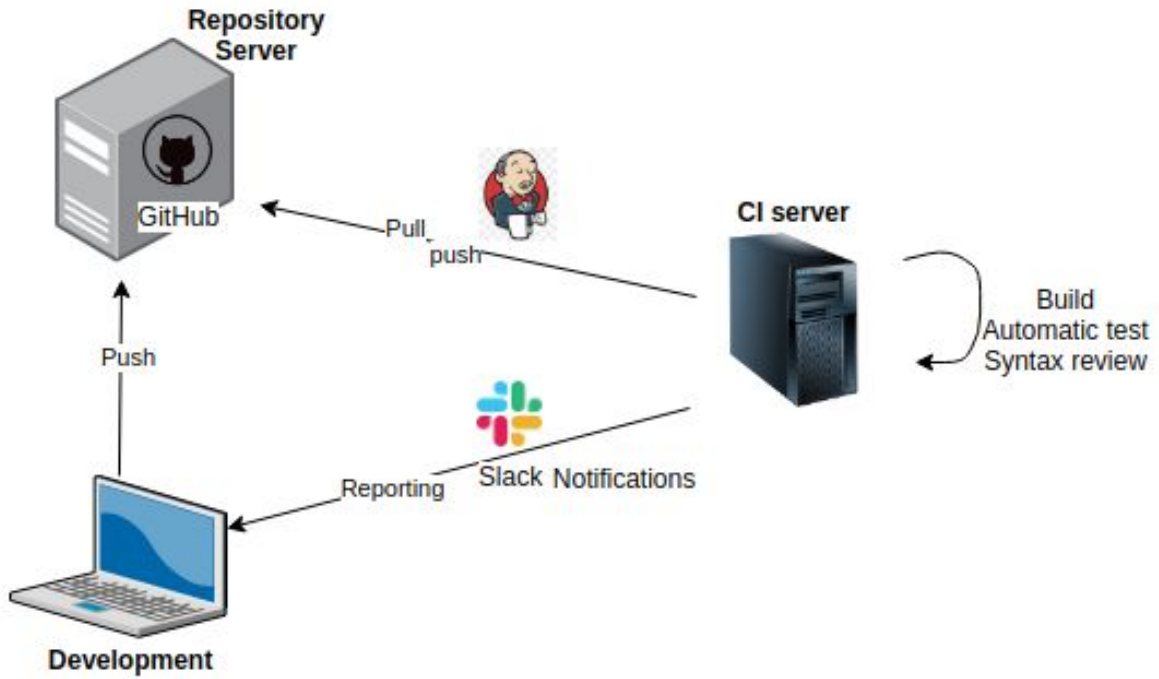
1. **Check versión**: git --version
2. **Install**: sudo apt install git
3. **Ver la configuración**: git config --list
4. **Agregar nombre de usuario:** git config --global user.name "Your Name"
5. **Agregar email:** git config --global user.email "youremail@domain.com"
6. **Editar manualmente la configuración:** nano ~/.gitconfig

# CI/CD / Continus Integration-Continus Delivery

@sotobotero

## Integración continua

1. Es una de las principales prácticas DevOps y consiste en automatizar la gestión de los cambios del código de múltiples contribuidores en un único proyecto de software.
2. Permite a los desarrolladores realizar merge frecuentemente en un repositorio central, luego permite que la compilación y pruebas automáticas sean ejecutadas.
3. El sistema de control de versiones es el core de todo el proceso de integración continua y se puede complementar con pruebas de código automático, revisión de sintaxis etc.




Git

# CD / Continus Delivery - Continus deployment

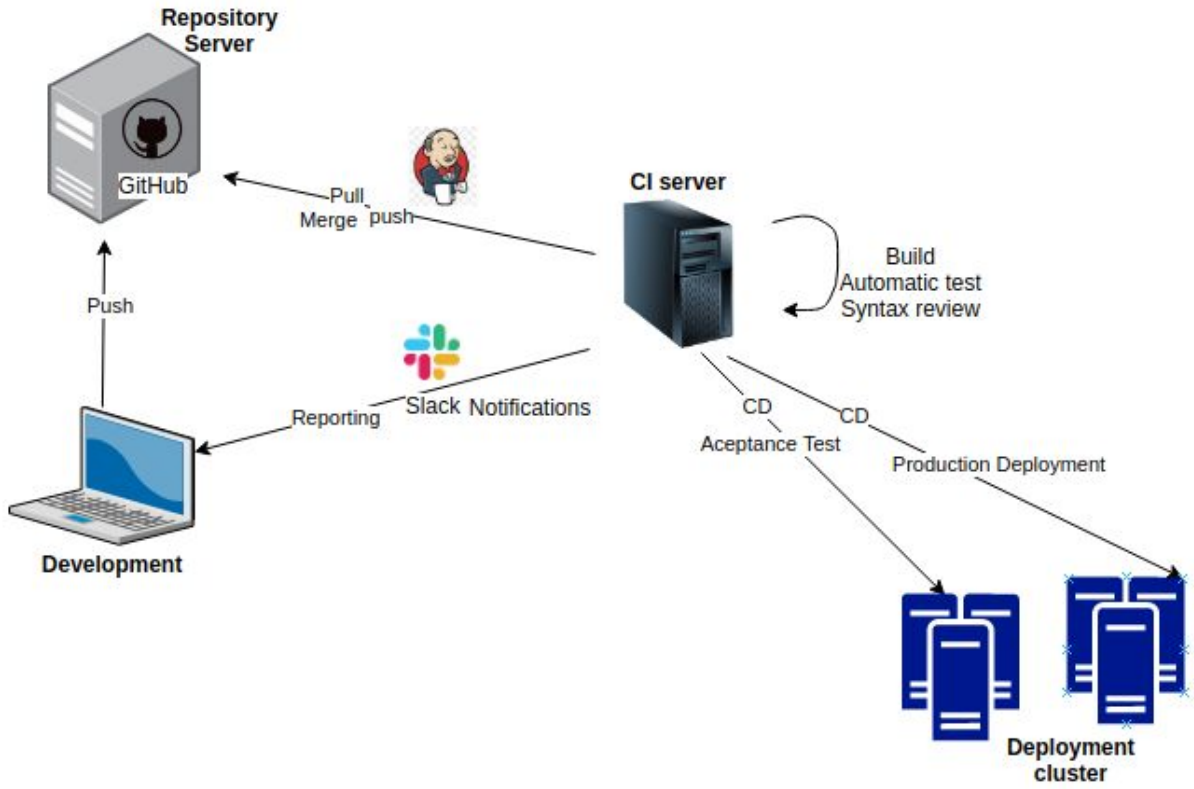
@sotobotero

## Entrega continua

1. Es una extensión del proceso de integración continua, dado que despliega todos los cambios de manera automática en el entorno de pruebas y/o producción.
2. Se cuenta con un proceso de despliegue automatizado que se ejecuta después de la fase de construcción o de forma manual con frecuencia predeterminada..

## Liberación continua

1. Va un paso más allá de la entrega continua en el nivel de automatización.
2. Cuando la aplicación pasa todas las fases anteriores el software es desplegado en producción y entregado al cliente final, no hay intervenciones humanas, solo si las pruebas fallan se detendrá el proceso.




Git

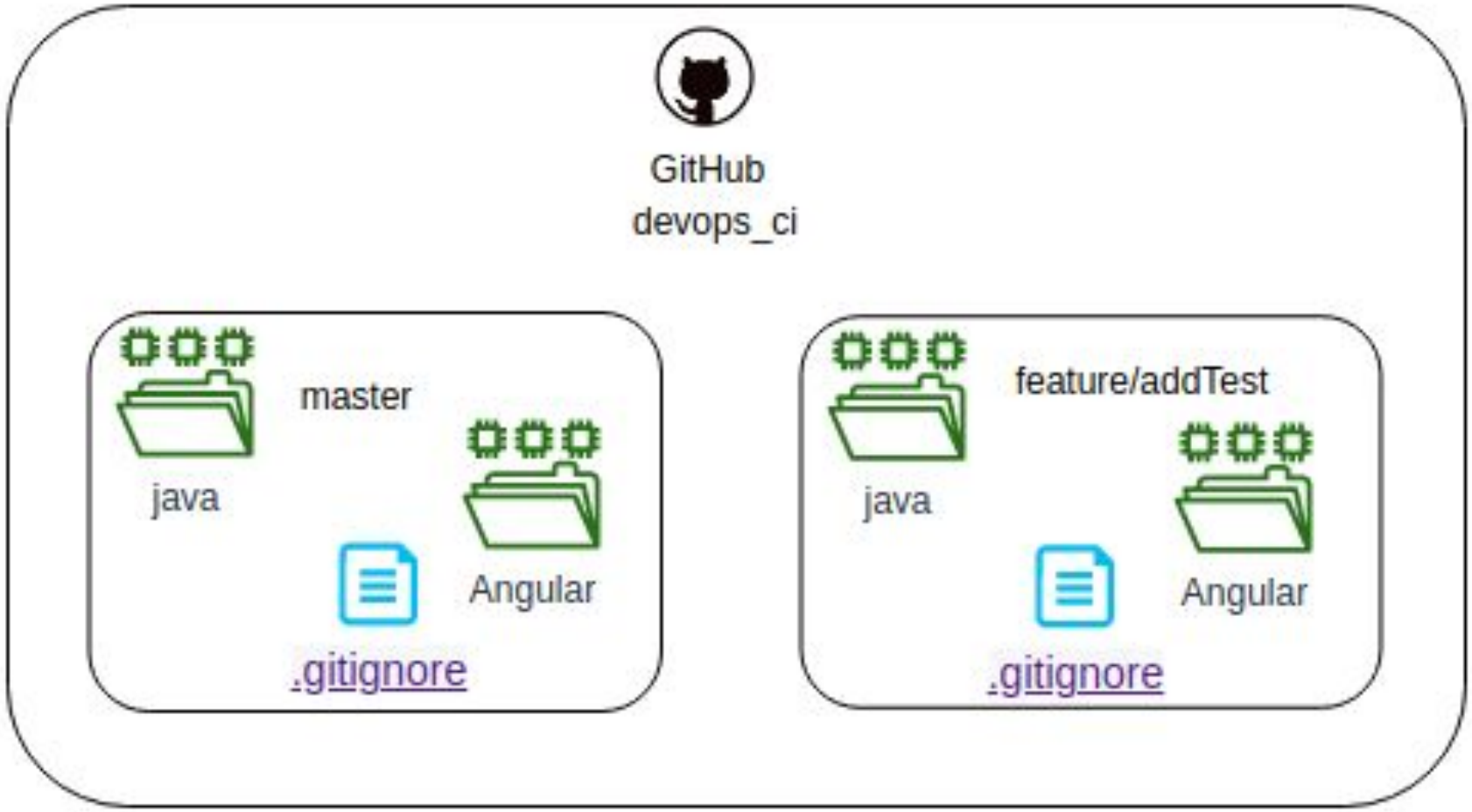
# Kubernetes Managed / Onpremise y Tipos de instalación

@sotobotero


- **Pipeline:** Grupo lógico de actividades que de manera conjunta realizan una tarea.
- **Jenkins:** Servidor de automatización opensource
- **Slack:** Plataforma propietaria de comunicación empresarial (Caracteristica mas relevante channel ).
- **SonarQube:** Plataforma opensource para la inspección continua de la calidad del código, puede ejecutar revisiones en automático.
- **Selenium:** Framework para probar aplicaciones web, permite escribir test funcionales de manera sencilla.

# Kubernetes Managed / Onpremise y Tipos de instalación

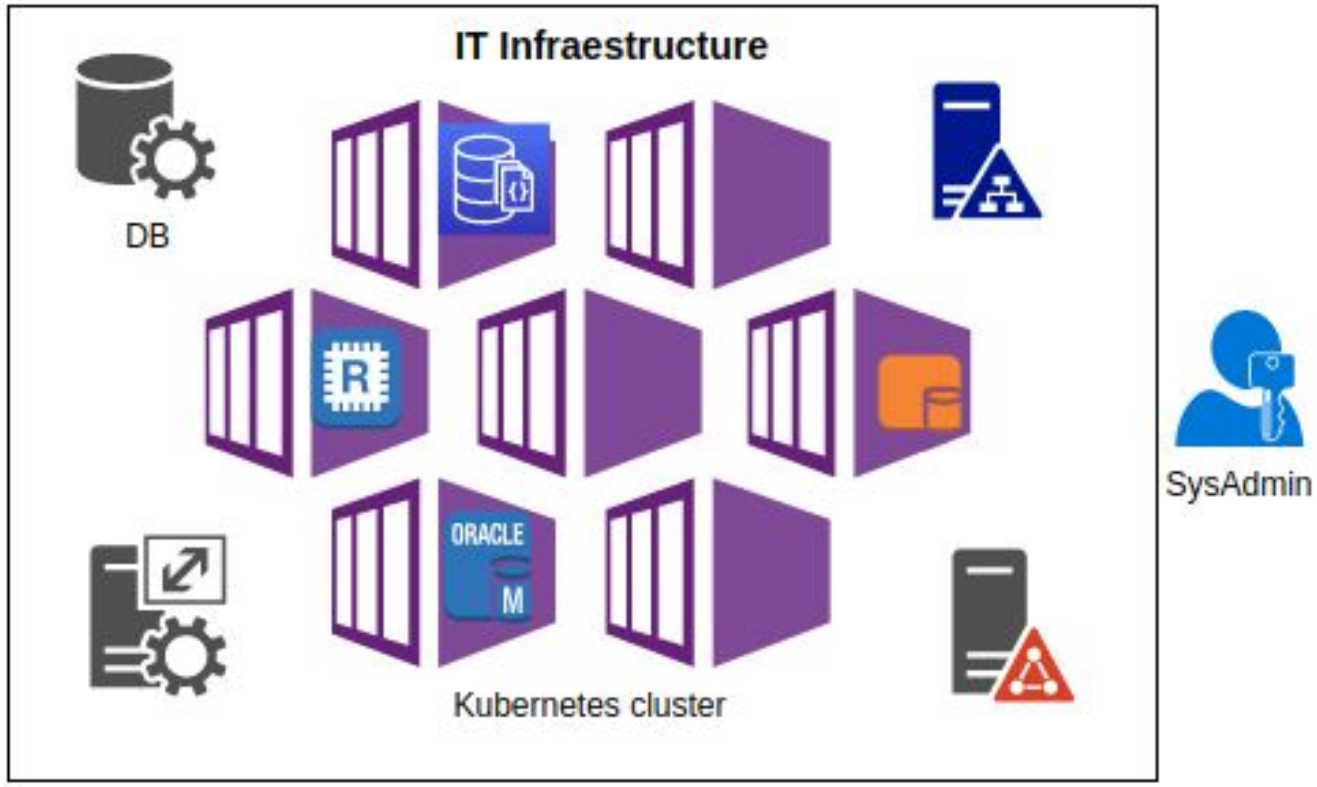
@sotobotero



# Monitorización / Prometheus

@sotobotero

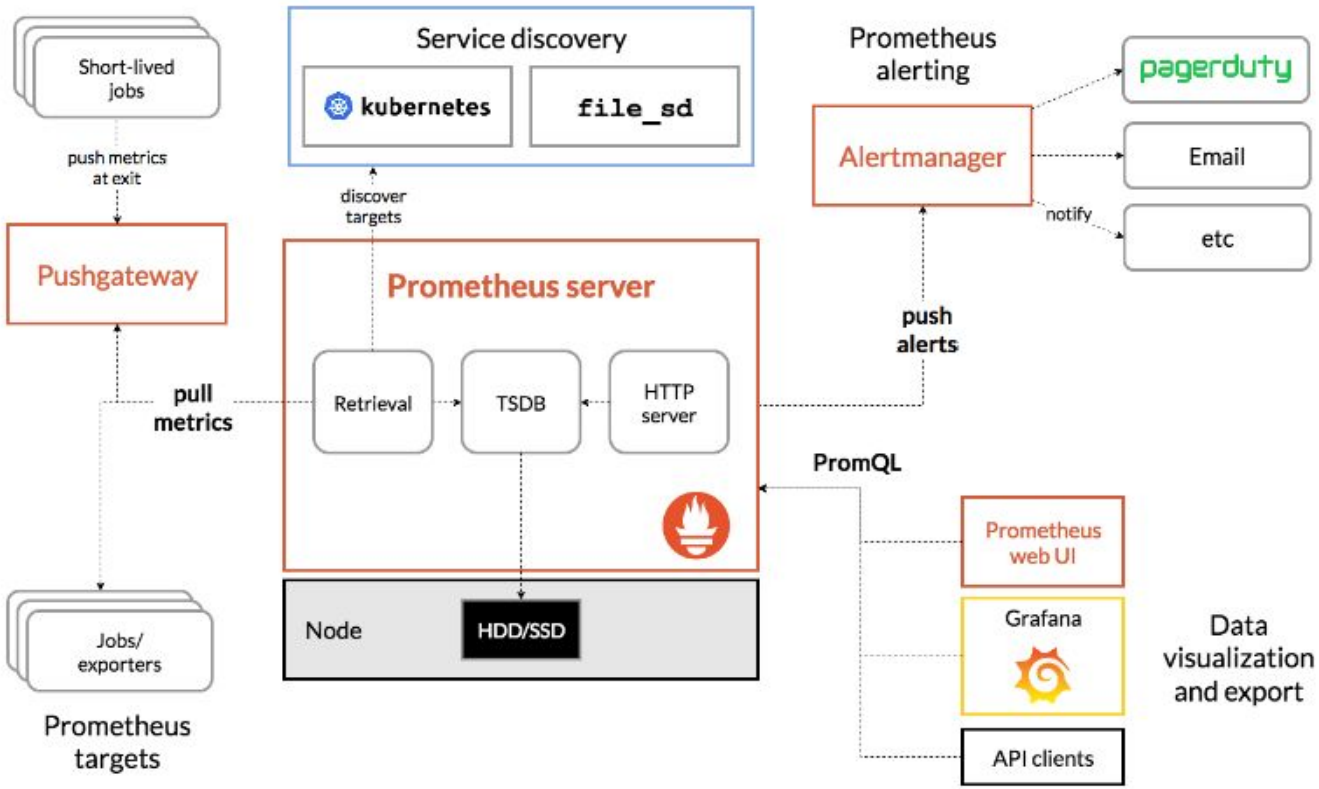
Prometheus es un sistema open source utilizado para la  monitorización de eventos y alertas, creado por SoundCloud en 2012 y mantenido por cloud native computing foundation (Linux foundation). Aporta ventaja en la monitorización de infraestructuras complejas.




- Monitorización de infraestructura física
- Monitorización de componentes funcionales (servicios)
- Detectar la raíz del problema.
- Alarmas en tiempo real.
- Notificacion via Email, WebHooks etc.
- Dashboards

# Monitorización / Prometheus

@sotobotero




- **Prometheus server:** Core de la aplicación, extracción y almacenamiento de métricas tiempo real.
- **Client Library:** Instrumentar el código de aplicaciones
- **Push gateway:** Soporte para jobs de aplicaciones de vida corta
- **Exporters:** Datasources que se ejecutan con cierta frecuencia y permiten la recolección de datos mediante el modelo pull.
- **Alert Manager:** Administra las alertas, envía notificaciones.
- PromQL: Prometheus query language utilizado para realizar consultas y construir dashboard usando grafana por ejemplo.

# Monitorización / Grafana

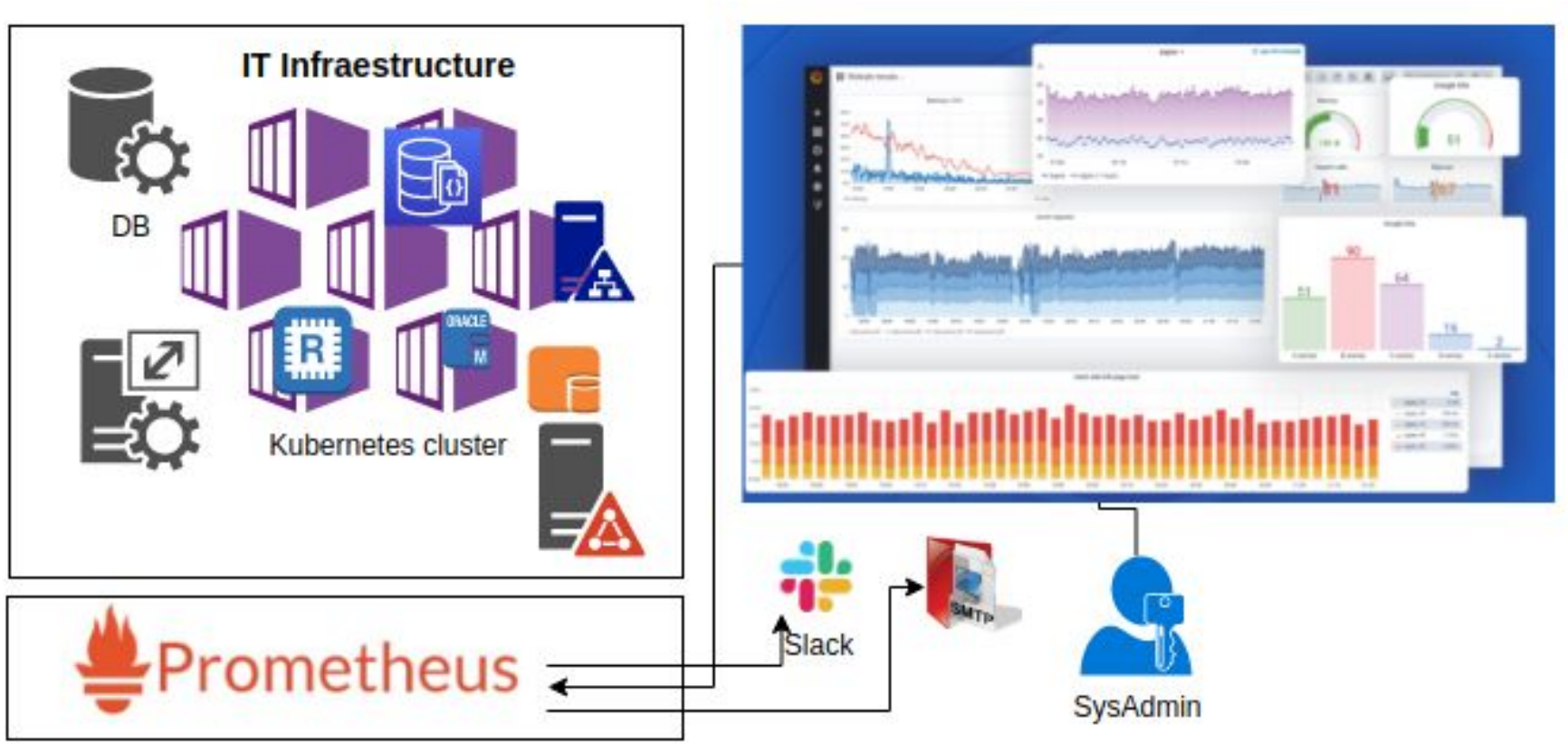
@sotobotero

Es una plataforma web open source de análisis y visualización interactiva, proporciona, gráficos, tableros de mando y alertas web al conectarse a una fuente de datos.
Soporta multiples datasource (AWS, Prometheus, Mysql, postgres)


Kubernetes

# Monitorización / Prometheus

@sotobotero



# GRACIAS