Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ciencias y Sistemas
Manejo e Implementación de Archivos
Primer Semestre 2024
Catedráticos:
Ing. Everest Darwin Medinilla Rodriguez
Ing. Marco Tulio Aldana Prilwitz
Tutores académicos:
Diego Molina
Ariana Perez

Práctica 3

Automatizar pipeline de CI/CD para una aplicación

Objetivos

Objetivos Generales

- Aplicar los conocimientos adquiridos a lo largo de la carrera de Ingeniería en Ciencias y Sistemas para generar software de alta calidad y escalable, a través de diferentes técnicas de desarrollo y utilizando tecnologías de última generación.

Objetivos Específicos

- Comprender y explorar los usos de docker
- Creacion y gestion de entornos usando Kubernetes
- Implementar de manera práctica la arquitectura de microservicios

Descripción

Automatizar un pipeline de CI/CD para la aplicación de microservicios elaborada en la práctica 2.
Se debe hacer la implementación de varios pasos, desde la integración y prueba hasta el despliegue.

STAGES DEL CI:

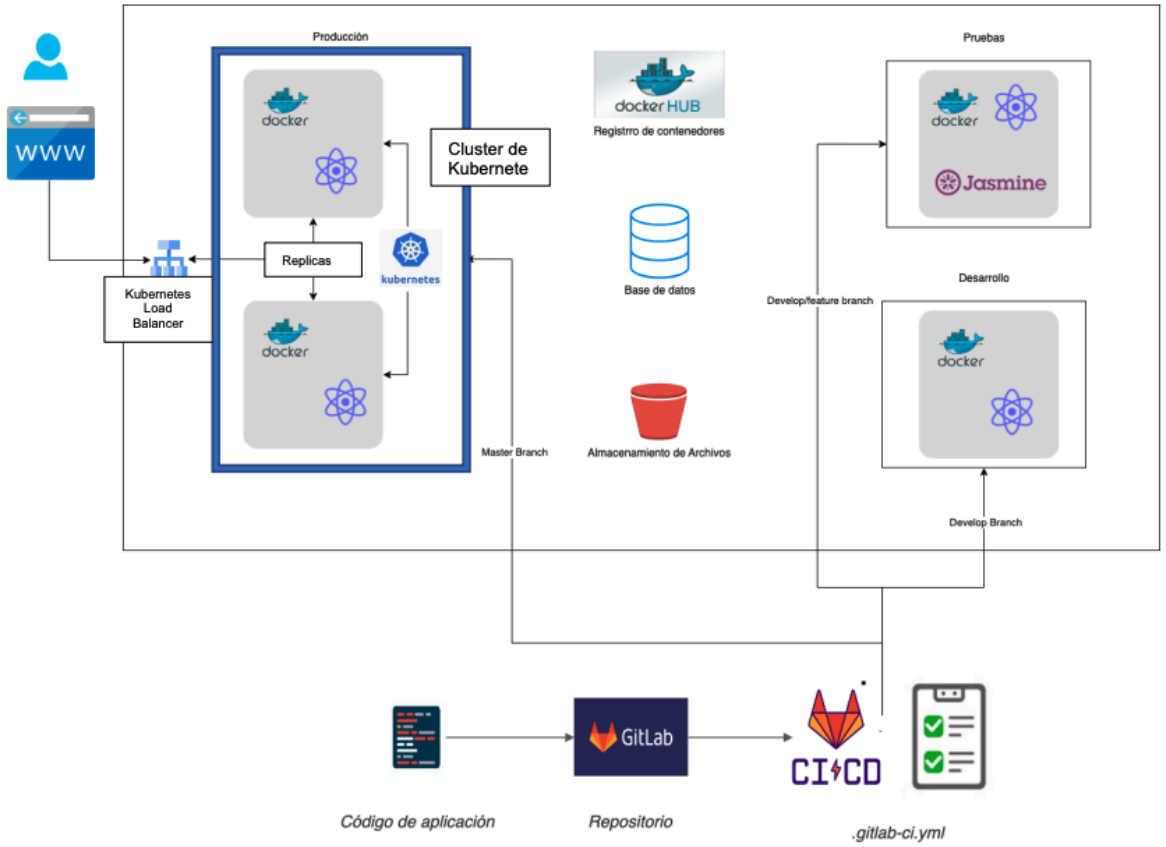
- Build
- Test
- Post-Build

STAGES DEL CD:

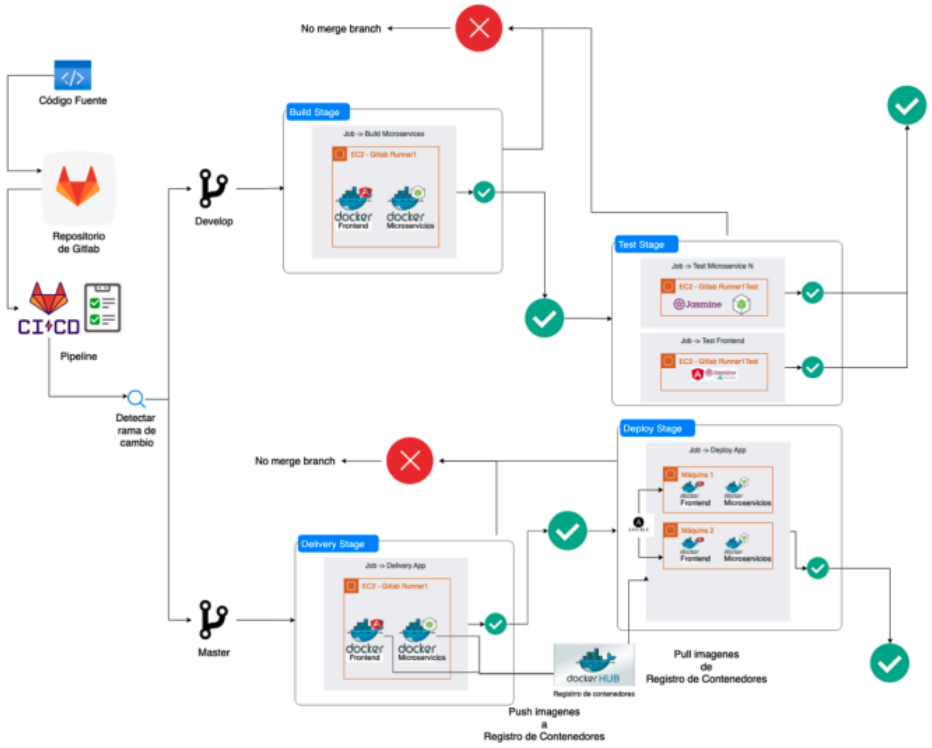
- Delivery
- Deploy

Ejemplo de arquitectura:

Recuerda que las peticiones se harán desde postman, no es necesario un frontend para la práctica.



Descripción de Flujo:



Primero, se creará una nueva funcionalidad en una rama **feature**. Posteriormente, se llevará a cabo un commit y un push al repositorio, lo que desencadenará la ejecución de las fases de construcción (build) y prueba (test).

A continuación, se iniciará una solicitud de fusión (merge request) de la **feature** hacia la rama de desarrollo (**develop**). Durante este proceso, se ejecutarán las fases de construcción y prueba, y se incorporará un post-build que activará el archivo docker-compose de desarrollo, utilizando sus respectivos Dockerfiles. En este punto, los cambios deberían ser visibles en el entorno de desarrollo, sin embargo, no se reflejarán en el entorno de producción.

Si todo transcurre correctamente hasta este momento, se acumularán puntos en estas etapas del proceso.

El siguiente paso implica ejecutar la fase de entrega (delivery), que se realiza en las ramas de versión (**release**). Aquí, se deben configurar al menos dos trabajos (jobs): uno para generar el objeto de versión y otro para cargar las imágenes de Docker en el registro de contenedores. Se recomienda ejecutar primero la generación del release, seguida de la entrega de las imágenes de Docker. Es importante destacar que para generar un objeto de versión, deben crear una etiqueta (tag), y esta condición debe especificarse en cada job de la fase de entrega. Aunque no sea estrictamente necesario tener una rama de versión para ejecutar los trabajos, su presencia es valiosa como un registro histórico de versiones.

Finalmente, se realizará una solicitud de fusión hacia la rama principal (**main/master**), desencadenando la ejecución de Ansible o kubernetes para implementar la nueva versión de la aplicación. En este punto, el archivo docker-compose deberá utilizar las imágenes de Docker almacenadas en el registro de contenedores (DockerHub).

- ➔ Como recordatorio, es esencial mantener todas las máquinas activas para verificar los cambios en el sistema.
- ➔ Queda a discreción del estudiante que herramientas usar para el despliegue de la aplicación (Ansible o kubernetes)

Tecnologías permitidas

Herramientas	Tipo
React con vite	Frontend
Node.js, Flask, Java Sprint Boot, python, go	Backend
Jasmine, Junit, Jest, Testing Library	Test
AWS, Azure, Google Cloud, Oracle, Huawei y Alibaba	Nube
Ansible, Terraform	Infraestructura
Jenkins, TravisCI, GitLab CI, Github actions	CI/CD
Git, GitHub, GitLab, Bitbucket	Control de Versiones
Prometheus, Grafana, ELK Stack	Monitoreo
Docker	Contenedores
Kubernetes	Orquestación

Documentación

- Contratos de microservicios
- Definir tecnología a utilizar para el despliegue y el porqué fue la elección
- Gestión de aplicaciones:
  - Cómo hacer un despliegue de aplicaciones en la herramienta seleccionada.
  - Uso de YAML para definir recursos de la herramienta seleccionada.
- Definir CICD
  - Definir Stages
  - Definir Jobs
- Manual de instalación de runner

Entregables

- ➔ **Subir a la UEDI el link del Repositorio, debe contener:**
  - ◆ Documentación en archivo PDF o MARKDOWN.
  - ◆ Código fuente de la aplicación
  - ◆ **Pipeline automatizado**: Se incluye el pipeline con las etapas de Build, Test, Delivery y Deploy

Requerimientos mínimos

Para tener derecho a calificación, el estudiante deberá cumplir con los siguientes requerimientos mínimos:

- Documentación completa
- Último commit subido antes de la hora y fecha de entrega.
- Pipeline completo.
- Nombre del repositorio: **PracticasSA**
- Agregar al auxiliar al repositorio, con el rol Developer:
  - Sección A: **di3gini**
  - Sección B: **pqmad**

Restricciones

- Se debe hacer uso de un repositorio en la nube para realizar la entrega de su proyecto (Gitlab, Github, Bitbucket, etc.)
- Se trabajará en **parejas o individual,** según lo que indique el auxiliar.
- No se calificará nada de manera local.
- Las copias completas/parciales serán merecedoras de una nota de 0 puntos, los responsables serán reportados al catedrático de la sección y a la Escuela de Ciencias y Sistemas.

Fecha de Entrega

Día 26 de marzo de 2024 a las 23:59 hrs la entrega se realizará por medio de UEDI, en caso exista algún problema, se estará habilitando un medio alterno por medio del auxiliar del laboratorio.