# Descripción de la arquitectura del sistema

## Introducción:

La arquitectura del sistema se basa en una implementación de Kubernetes en AWS. La aplicación Flask se ejecuta en pods dentro de un clúster de Kubernetes, lo que permite una mayor escalabilidad, disponibilidad y facilidad de administración.

## Tecnologías:

* **Lenguaje de programación:** Python 3.9
* **Framework web:** Flask
* **ORM**: SQLAlchemy
* **Base de datos:** PostgreSQL
* **Contenedor:** Docker
* **Herramienta de orquestación:** Kubernetes
* **Servicio de nube:** AWS

## Arquitectura Cloud

## Componentes:

* **Amazon Elastic Kubernetes Service (EKS):** Servicio de Kubernetes gestionado que facilita la ejecución y la administración de aplicaciones en contenedores en AWS.
* **Nodos de trabajadores:** Instancias de EC2 que ejecutan pods de Kubernetes.
* **Pods:** Contenedores que se ejecutan en un nodo de trabajador.
* **Base de datos:** Base de datos PostgreSQL alojada en Amazon Relational Database Service (RDS).
* **AWS Secrets Manager:** Servicio que almacena y administra las credenciales de la base de datos y otras variables de entorno.
* **Amazon CloudWatch**: Servicio de monitoreo que recopila y analiza datos de métricas, logs y eventos de la aplicación.
* **VPC:** Red privada virtual en la que se ejecuta la aplicación.

## Diagrama:

Imagen que contiene Diagrama

Descripción generada automáticamente

## Descripción del ciclo de vida

**Modelo de desarrollo:**

* **Evolución del producto:** Se seguirá un modelo de desarrollo iterativo incremental, donde se implementarán nuevas funcionalidades y correcciones de errores en entregas frecuentes.

**Desarrollo local:**

* Se utilizará Docker para crear un entorno de desarrollo local consistente y aislado.
* Se implementarán tests unitarios y de integración para asegurar la calidad del código.

**Tipos de pruebas obligatorias:**

* **Pruebas unitarias:** Verifican el comportamiento individual de cada módulo del código.
* **Pruebas de integración:** Aseguran la correcta interacción entre los diferentes módulos del sistema.
* **Pruebas de aceptación:** Verifican que el sistema cumple con los requisitos funcionales y no funcionales.

**Entornos:**

* **Desarrollo:** Entorno local para la creación e integración de nuevas funcionalidades.
* **Producción:** Entorno donde se ejecuta el software para los usuarios finales.

**Modelo de ramas:**

* Se utilizará una estrategia de dos ramas:
  + **main:** Rama principal donde se encuentra la versión actual del software en producción.
  + **dev:** Rama de desarrollo donde se realizan las nuevas funcionalidades y correcciones de errores.

**Decisiones de despliegue:**

* **Rama main:**
  + Se utilizará un pipeline en Jenkins para realizar el despliegue a producción.
  + El pipeline se ejecutará automáticamente al realizar un push o un pull request a la rama main.
  + El pipeline incluirá las siguientes etapas:
    - Compilación del código.
    - Ejecución de pruebas automatizadas.
    - Despliegue a Kubernetes.
* **Rama dev:**
  + Se utilizará un action de Github para verificar si el código pasa las pruebas al realizar un push.
  + Si las pruebas son exitosas, el código se puede integrar a la rama dev.
  + El código de la rama dev se puede integrar a la rama main para su posterior despliegue a producción.