UNIVERSIDAD TECNOLOGICA DE SANTIAGO
(UTESA)

PRESENTADO POR:

ERIK CRUZ
1-18-0759

PRESENTADO A:

IVAN MENDOZA

ASIGNATURA:

ALGORITMOS PARALELOS

ASIGNACION:

TAREA SEMANA 4

Santiago de los Caballeros
República Dominicana
Febrero, 2024

Introducción

En el desarrollo de software contemporáneo, la combinación de DevOps y CI/CD ha surgido como una metodología fundamental para optimizar la eficiencia y la calidad en el ciclo de vida del software. DevOps promueve la integración estrecha entre los equipos de desarrollo y operaciones, mientras que CI/CD automatiza la integración, prueba y entrega de código de manera continua. Juntos, estos enfoques aceleran el desarrollo, reducen los tiempos de entrega y mejoran la calidad del software, al tiempo que fomentan una cultura de colaboración y mejora continua. Este informe explora en detalle los principios, beneficios, herramientas y casos de estudio relacionados con DevOps y CI/CD, destacando su relevancia en el desarrollo de software moderno.

DevOps

DevOps es más que solo un conjunto de herramientas y procesos; es una cultura que promueve la colaboración y la integración entre los equipos de desarrollo de software y operaciones de TI. Este enfoque busca eliminar las barreras entre estos equipos para lograr una entrega de software más rápida y confiable. La automatización es un pilar fundamental en DevOps, permitiendo la ejecución eficiente y repetible de tareas como la construcción de software, las pruebas y la implementación. Además, DevOps fomenta la adopción de prácticas como la infraestructura como código (IaC), que permite gestionar la infraestructura de manera programática, facilitando la escalabilidad y la consistencia.

La arquitectura de microservicios y el uso de contenedores, como Docker, son comunes en los entornos DevOps debido a su capacidad para mejorar la flexibilidad y la escalabilidad de las aplicaciones. Estas tecnologías permiten desplegar componentes de software de forma independiente, lo que facilita la iteración y la entrega continua. Además, el monitoreo y la observabilidad son aspectos clave en DevOps, ya que permiten a los equipos detectar y solucionar problemas de rendimiento o errores en tiempo real. Herramientas como Prometheus, Grafana y ELK Stack son populares para este propósito.

CI/CD

La Integración Continua (CI) es una práctica que implica la integración frecuente de cambios de código en un repositorio compartido, seguida de la ejecución automatizada de pruebas para garantizar la integridad del código. Esta práctica proporciona una retroalimentación rápida a los desarrolladores, lo que permite detectar errores temprano en el ciclo de desarrollo. Por otro lado, la Entrega Continua (CD) implica la automatización del proceso de entrega de software, desde la construcción hasta el despliegue en entornos de prueba o producción. Esto incluye la gestión automatizada de versiones y la orquestación de pipelines de CI/CD utilizando herramientas como Jenkins, GitLab CI/CD, CircleCI o Azure DevOps.

La adopción de CI/CD permite a los equipos de desarrollo acelerar el ciclo de entrega de software, mejorar la calidad del código y aumentar la confiabilidad de las implementaciones. Con CI/CD, los equipos pueden implementar cambios de manera más segura y frecuente, lo que les permite responder de manera más ágil a las demandas del mercado y a los comentarios de los usuarios. Además, CI/CD facilita la identificación temprana de problemas en el código, lo que reduce el tiempo y el esfuerzo requeridos para corregirlos. En resumen, CI/CD es fundamental en la adopción exitosa de DevOps, ya que proporciona las herramientas y prácticas necesarias para automatizar y optimizar el proceso de entrega de software.

Mencionar cuales herramientas se utilizan y para qué.

Jenkins

Función: Herramienta de automatización de código abierto para la integración y el despliegue continuos.
Uso: Se utiliza para automatizar el proceso de construcción, prueba y despliegue de aplicaciones, así como para orquestar flujos de trabajo complejos.

GitLab CI/CD

Función: Plataforma integral de desarrollo de software que incluye funcionalidades de integración y entrega continuas.
Uso: Permite automatizar la construcción, prueba y despliegue de aplicaciones directamente desde el repositorio GitLab, facilitando la colaboración y la entrega continua.

CircleCI:

Función: Plataforma de integración y entrega continuas en la nube.
Uso: Proporciona herramientas para automatizar el proceso de construcción, prueba y despliegue de aplicaciones en entornos de desarrollo y producción.

Travis CI:

Función: Plataforma de integración continua en la nube.
Uso: Se utiliza para automatizar la construcción y prueba de aplicaciones, así como para desplegar aplicaciones en entornos de desarrollo y producción de manera automática.

GitHub Actions:

Función: Sistema de automatización incorporado en GitHub para integración y entrega continuas.
Uso: Permite crear flujos de trabajo automatizados directamente desde los repositorios de GitHub para construir, probar y desplegar aplicaciones de manera eficiente.

Investigar cuales herramientas permiten desarrollar pipelines y un ejemplo de pipeline de cada una.

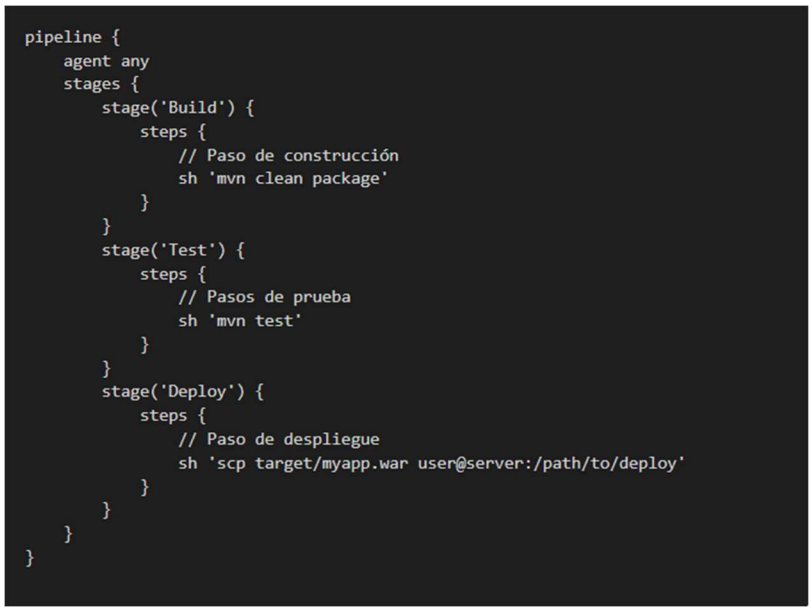
Un pipeline son una serie de pasos automatizados que permiten llevar a cabo diferentes tareas relacionadas con la construcción, prueba y despliegue de un software de manera eficiente y repetible. En un sentido más amplio, un pipeline puede abarcar todo el proceso de desarrollo de software, desde la integración del código hasta su entrega en producción.

Está compuesto por etapas individuales, llamadas "stages", y cada etapa puede contener una o más "jobs" que realizan tareas específicas dentro de esa etapa.

Los pipelines permiten automatizar el proceso de desarrollo y despliegue de software, lo que resulta en una mayor eficiencia, consistencia y calidad en la entrega del producto final. Además, los pipelines proporcionan visibilidad y trazabilidad a lo largo de todo el proceso, lo que facilita la detección y corrección de problemas de manera temprana en el ciclo de desarrollo.

Aquí algunas herramientas populares que permiten desarrollar pipelines y ejemplos básicos de pipelines utilizando cada una:

Jenkins:



```
pipeline {
    agent any
    stages {
        stage('Build') {
            steps {
                // Paso de construcción
                sh 'mvn clean package'
            }
        }
        stage('Test') {
            steps {
                // Pasos de prueba
                sh 'mvn test'
            }
        }
        stage('Deploy') {
            steps {
                // Paso de despliegue
                sh 'scp target/myapp.war user@server:/path/to/deploy'
            }
        }
    }
}
```

GitLab CI/CD:

```
stages:
  - build
  - test
  - deploy

build:
  stage: build
  script:
    - mvn clean package

test:
  stage: test
  script:
    - mvn test

deploy:
  stage: deploy
  script:
    - scp target/myapp.war user@server:/path/to/deploy
```

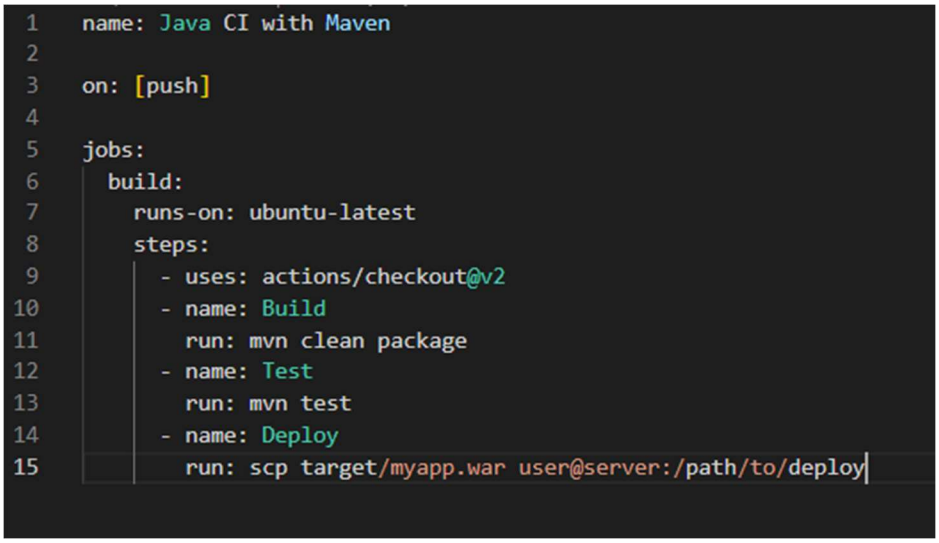
CircleCI:

```
version: 2.1
jobs:
  build:
    docker:
      - image: maven:3.6.3-jdk-11
    steps:
      - checkout
      - run:
          name: Build
          command: mvn clean package
      - run:
          name: Test
          command: mvn test
      - run:
          name: Deploy
          command: scp target/myapp.war user@server:/path/to/deploy
```

Travis CI:

```
language: java
script:
  - mvn clean package
  - mvn test
  - scp target/myapp.war user@server:/path/to/deploy
```

Github Actions:



```yaml
name: Java CI with Maven

on: [push]

jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v2
      - name: Build
        run: mvn clean package
      - name: Test
        run: mvn test
      - name: Deploy
        run: scp target/myapp.war user@server:/path/to/deploy
```

Conclusión

En resumen, la combinación de DevOps y CI/CD ha revolucionado la forma en que se desarrolla y entrega el software en la actualidad. Estas metodologías no solo se centran en la automatización de procesos, sino que también promueven una cultura de colaboración, mejora continua y entrega rápida y confiable de software. DevOps, como cultura, busca eliminar las barreras entre los equipos de desarrollo y operaciones, fomentando la automatización, la colaboración y la integración continua. La arquitectura de microservicios y el uso de contenedores son comunes en los entornos DevOps, permitiendo una mayor flexibilidad y escalabilidad.

Por otro lado, CI/CD se centra en automatizar la integración, prueba y entrega de código de manera continua. Herramientas como Jenkins, GitLab CI/CD, CircleCI, Travis CI y GitHub Actions permiten a los equipos desarrollar pipelines para automatizar todo el proceso de desarrollo y despliegue de software. La adopción exitosa de DevOps y CI/CD proporciona beneficios significativos, incluida una mayor velocidad de entrega, una mayor calidad del software, una menor incidencia de errores y una mayor capacidad de adaptación a los cambios del mercado. En última instancia, DevOps y CI/CD están transformando la forma en que se desarrolla, entrega y opera el software en la era digital, impulsando la innovación y mejorando la competitividad de las empresas en un entorno empresarial cada vez más dinámico y exigente.