Nombre: Walter Ariel Baya
DNI: 39749587

Ejercicio Clase 11 ADS

Elegir una de las herramientas utilizadas en DevOps **para desarrollo de software**. En base a la misma, explicarla detalladamente:

- Por qué la eligieron
- Su uso
- Su aplicación
- En qué etapa se utiliza
- Dar ejemplos concretos de su uso

Elegir también una de las herramientas utilizadas en DevOps **para tecnología de la información**. En base a la misma, explicarla detalladamente:

- Por qué la eligieron
- Su uso
- Su aplicación
- En qué etapa se utiliza
- Dar ejemplos concretos de su uso.

Pueden elegir una herramienta vista en clase, u otra que no hayamos mencionado durante las clases 10 y 11.

Herramienta para Desarrollo de Software elegida: Docker

¿Por qué elegí Docker?

Elegí Docker por su capacidad para estandarizar entornos de desarrollo mediante contenedores, eliminando problemas de compatibilidad, el típico caso que solía ocurrir antes era “che esto funciona en mi máquina pero cuando lo tiro en producción explota por todos lados", esto contribuye enormemente acelerando el despliegue de aplicaciones, las vuelve portables también asegura que haya más eficiencia en recursos y por eso hoy en día es tan común y solicitada en la industria porque es a día de hoy fundamental para modernizar flujos de trabajo DevOps.

Uso y Aplicación

Docker empaqueta aplicaciones y sus dependencias en contenedores, que se ejecutan de manera aislada sobre el sistema operativo anfitrión que es en el que estamos desplegando la aplicación de docker justamente, garantizando consistencia desde el desarrollo hasta la producción

- **Imágenes**: Plantas de solo lectura que definen el contenedor (ej: `Dockerfile` con capas de Ubuntu, Apache, código PHP).
- **Contenedores**: Instancias en ejecución de las imágenes, gestionadas mediante la CLI de Docker, los comandos que se usan para detener e iniciar un contenedor son: (`docker run`, `docker stop`).
- **Docker Compose**: Orquesta múltiples contenedores (ej: app + base de datos) con un archivo `docker-compose.yml`.

Etapa en DevOps

Se utiliza en:

1. **Desarrollo**: Crear entornos locales idénticos a producción.
2. **Pruebas**: Ejecutar tests en contenedores aislados.
3. **Despliegue**: Empaquetar aplicaciones para implementación en nube (AWS ECS, Kubernetes).

Ejemplos Concretos

1. **Lo que más hice you fueron aplicaciones web con microservicios**:
   - Un `Dockerfile` define la imagen de un servicio Node.js, por ejemplo un microservicio de cuentas de un determinado banco, se usa solo para eso.
   - `docker-compose.yml` integra el servicio con las base de datos del backend PostgreSQL.
   - Despues lo desplegamos en un servidor en AWS ECS sin modificar el código.

2. **Pruebas de Integración**:
   - Testcontainers levanta una base de datos PostgreSQL temporal para pruebas, eliminándola al finalizar, a esto lo llamamos en Java en particular a Mockito como Mockear, crear un Mock que es algo asi como una base de datos o un repositorio "de mentira" que sirva para hacer pruebas, cosa de no afectar directamente al repositorio que tenemos desplegado y que es de negocio.

Herramienta para Tecnología de la Información: Jenkins

Seleccioné Jenkins por su liderazgo en automatización de CI/CD, aparte de que hace un tiempo tuve un líder de desarrollo en el trabajo que me dio un breve vistazo a los Jobs y demás y me quedo un grato recuerdo de la misma, lo que tiene de bueno principalmente es que es fácil de integrar con herramientas como Docker, Git y Kubernetes. Es esencial para equipos que necesitan usar pipelines personalizables y retroalimentación inmediata.

Uso y Aplicación

Jenkins automatiza:

- **Compilación**: Cada vez que alguno manda los cambios al repo en GIT, ejecuta `mvn clean install`.
- **Pruebas**: Ejecuta suites con Selenium o JUnit, esto sirve cuando tenemos tests cargados que son como lo mínimo a que pasen antes de que lo pueda revisar alguien de Quality Assurance.
- **Despliegue**: Implementa en servidores o contenedores (ej: Kubernetes) mediante pipelines codificados en **Groovy**.

  Su arquitectura distribuida permite escalar trabajos en múltiples nodos.

Etapa en DevOps

Actúa en:

1. **Integración Continua (CI)**: Fusiona código diariamente, ejecuta tests.

2. **Entrega Continua (CD)**: Despliega en entornos de preproducción.
3. **Monitoreo**: Notifica fallos via Slack/email y genera dashboards, a mi en general siempre me mandaron mails nunca utilicé Slack, pero se que es popular.

Ejemplos Concretos

1. **Pipeline para Aplicación Java**:

```
pipeline {
  agent any
  stages {
    stage('Build') { steps { sh 'mvn package' } }
    stage('Test') { steps { sh 'mvn test' } }
    stage('Deploy') { steps { sh 'kubectl apply -f deployment.yaml' } }
  }
}
```

   Este pipeline se activa automáticamente al hacer `push` en Git, o sea al mandar los cambios se hace un Build para construir la aplicación o microservicio, despues lo probamos, ¿Qúe onda? ¿Salio todo bien?... Si es asi se hace deploy, sino se envía un mensaje de que hay error en algún test.

2. **Escalado de Infraestructura**:

- Un plugin de AWS EC2 permite agregar nodos de Jenkins bajo demanda para cargas pesadas, cosa que si estamos con nuestro microservicio que provee información sobre competencias de Formula 1 y de repente de un día para otro pasamos de tener 1000 usuarios a 1M debido a la llegada de Franco Colapinto podamos escalar la aplicación agregando más nodos para satisfacer con buen rendimiento las necesidades de los clientes y no se caiga o justamente se vuelva muy extremadamente lento el servicio que se provee.

Conclusión y Extra

Docker y Jenkins son **complementarias**: Docker asegura consistencia en entornos, mientras Jenkins automatiza la entrega de software. Juntas, permiten implementar prácticas DevOps como:

- **Microservicios escalables** (Docker + Kubernetes).
- **Pipelines CI/CD confiables** (Jenkins + Docker para pruebas).
  Su adopción reduce errores humanos, acelera despliegues y mejora la colaboración entre equipos de desarrollo y operaciones

  Si bien en general todo el mundo usa Amazon Web Services y es lo más frecuente, yo utilicé la Cloud de IBM siempre porque realicé unos cursos y nunca tuve la oportunidad de tocar directamente la cloud de AWS, pero si la de IBM, no dejo ejemplos de la misma porque se que no es muy frecuente, pero me gustaría dejar una foto masomenos de como se ve el entorno que provee.