Modul Praktikum 8

Pipeline Sederhana pada Jenskin Berbasis Docker

1. Maksud

Maksud dari praktikum ini adalah untuk memberikan pengalaman langsung dalam membangun dan menjalankan *Continuous Integration/Continuous Delivery (CI/CD)* pipeline menggunakan Jenkins dengan pendekatan berbasis container melalui Docker. Melalui praktikum ini, peserta akan memahami bagaimana Docker dapat digunakan sebagai lingkungan eksekusi yang konsisten dalam alur kerja otomatisasi.

2. Tujuan

Setelah menyelesaikan praktikum ini, mahasiswa diharapkan mampu:

a) Memahami konsep dasar CI/CD dan peran Jenkins dalam proses tersebut.

b) Menginstal dan mengkonfigurasi Jenkins serta Docker pada sistem operasi Linux.

c) Membuat dan menjalankan pipeline sederhana menggunakan Jenkinsfile.

d) Menjalankan pipeline Jenkins di dalam container Docker.

e) Memahami manfaat penggunaan Docker dalam pipeline CI/CD.

3. Dasar Teori

3.1 Pengenalan Jenkins

Jenkins adalah alat open-source berbasis Java yang digunakan untuk otomatisasi tugas-tugas Continuous Integration dan Continuous Delivery (CI/CD). Jenkins membantu developer untuk secara otomatis membangun, menguji, dan mendeploy aplikasi setiap kali ada perubahan kode.

3.2 Pengenalan Docker

Docker adalah platform berbasis container yang memungkinkan aplikasi dan dependensinya dikemas dalam satu unit yang disebut container. Container menyediakan lingkungan yang ringan, konsisten, dan mudah dipindahkan antar mesin.

3.3 Pipeline pada Jenkins

Pipeline di Jenkins adalah rangkaian langkah-langkah yang didefinisikan dalam sebuah file bernama **Jenkinsfile**. File ini biasanya ditulis dalam sintaks Groovy dan digunakan untuk mendefinisikan alur kerja otomatis mulai dari build, test hingga deploy aplikasi.

3.4 Integrasi Jenkins dengan Docker

Integrasi Jenkins dengan Docker memungkinkan eksekusi pipeline dalam lingkungan container yang terisolasi. Ini meningkatkan keandalan, konsistensi lingkungan, dan kemudahan dalam manajemen dependensi.

4. Prosedur Praktik

4.1 Persiapan Awal

Langkah-langkah:

1. Pastikan sistem operasi Linux (misalnya Ubuntu) sudah siap.
2. Instal Docker Engine:

   sudo apt update && sudo apt install docker.io -y
3. Tambahkan user ke grup docker agar bisa menjalankan perintah tanpa sudo:

   sudo usermod -aG docker $USER

   Log out dan log in ulang untuk efek grup.
4. Instal Docker Compose (jika dibutuhkan):

   sudo curl -L "https://github.com/docker/compose/releases/download/v2.23.0/docker-compose- $(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

   sudo chmod +x /usr/local/bin/docker-compose

4.2 Instalasi Jenkins dalam Container Docker

Langkah-langkah:

1. Jalankan container Jenkins dengan Docker:

   docker run -d -p 8080:8080 -p 50000:50000 --name jenkins \

   -v jenkins_home:/var/jenkins_home \

   -v /var/run/docker.sock:/var/run/docker.sock \

   jenkins/jenkins
2. Buka browser dan akses: **http://<IP_SERVER>:8080**, contoh: http://localhost:8080.
3. Ikuti langkah instalasi awal Jenkins (masukkan password admin, pilih plugin, buat user).

4.3 Konfigurasi Plugin dan Tool

1. Di dashboard Jenkins → Manage Jenkins → Plugins → Install “Docker Pipeline”.
2. Pastikan Git juga sudah tersedia (umumnya sudah terinstal).
3. Set up tool paths jika diperlukan (Java, Maven, Node.js, dll.).

4.4 Membuat Projek Pipeline Baru

1. Klik "New Item" → Pilih "Pipeline" → Beri nama.
2. Di bagian Pipeline, pilih "Pipeline script from SCM".
3. Masukkan URL repositori GitHub Anda.
4. Tentukan path ke **Jenkinsfile** (biasanya di root direktori).
5. Simpan dan jalankan build pertama.

5. Studi Kasus dan Penyelesaian

Studi Kasus:

Sebuah tim pengembangan ingin membuat pipeline otomatis untuk aplikasi web sederhana berbasis Node.js. Aplikasi tersebut harus dibangun, diuji, dan di-deploy ke lingkungan staging hanya dengan satu klik atau commit ke repositori.

Penyelesaian:

Langkah-langkah:

1. Struktur Proyek yang ada di repository Github. Pada contoh ini repository github berada pada *node-app* dengan akun bernama *cravengithub*.

```
node-app/
├── app.js
├── package.json
└── Jenkinsfile
```

2. Isi file *Jenkinsfile*:

```
pipeline {
    agent any
    stages {
        stage('Clone Repository') {
            steps {
                git 'https://github.com/cravengithub/node-app.git '
            }
        }

        stage('Build') {
            steps {
                sh 'npm install'
            }
        }

        stage('Test') {
            steps {
                sh 'npm test' // asumsi telah dibuat unit test
            }
            post {
                success {
                    echo 'Tes berhasil!'
                }
                failure {
                    echo 'Tes gagal!'
                }
            }
        }

        stage('Deploy') {
            steps {
                sh 'echo "Menjalankan aplikasi..."'
                sh 'node app.js &'
            }
        }
    }
}
```

3. Hasil:
   Pipeline berhasil clone repo, lakukan build dan tes, lalu simulasi deploy aplikasi. Dengan Jenkinsfile, seluruh proses menjadi otomatis dan dapat dilacak.

6. Tugas

1. Buatlah repositori GitHub baru dengan proyek PHP sederhana.
2. Tuliskan Jenkinsfile yang dapat:
   a) Clone repo.
   b) Install dependencies.
   c) Jalankan unit test.
   d) Deploy aplikasi menggunakan Docker image lokal.
3. Jalankan pipeline tersebut di dalam container Jenkins.
4. Dokumentasikan hasilnya dalam format PDF atau Word beserta screenshot output Jenkins.