Modul: Integrasi Jenkins dengan Docker

Ringkasan singkat

Modul ini mengajarkan langkah praktis integrasi Jenkins dengan Docker: menyiapkan Jenkins (dijalankan di container atau host), membuat aplikasi contoh, membuat Dockerfile, menulis Jenkinsfile yang membangun image Docker, memberi tag, dan mendorong (push) ke Docker Hub. Modul berisi petunjuk langkah demi langkah, perintah yang siap-pakai, latihan singkat, kuis, dan bagian troubleshooting yang sering muncul di lab.

Tujuan Pembelajaran

Setelah mengikuti modul ini, peserta **mampu**:

1. Menjalankan Jenkins yang terhubung ke Docker Engine.
2. Membuat dan menyimpan credential GitHub & Docker Hub di Jenkins dengan aman.
3. Menulis Jenkinsfile yang membangun image Docker dan mendorong ke registry.
4. Mengidentifikasi & menyelesaikan masalah umum pada integrasi Jenkins–Docker.

Prasyarat

- Mesin host (Linux/Windows) dengan Docker Engine terpasang.
- Akses administrator ke mesin Jenkins (atau kemampuan menjalankan container Jenkins).
- Akun GitHub dan Docker Hub.
- Pengetahuan dasar: Git, Docker (image/container), dan CLI dasar.

Perlengkapan Lab

- Docker (Linux host atau Docker Desktop di Windows)
- Jenkins (direkomendasikan: run in container untuk lab)
- Browser untuk mengakses UI Jenkins
- Repository contoh (GitHub) yang berisi aplikasi sederhana (contoh: Flask/Node)

1. Menjalankan Jenkins (Quick-start)

Opsi A — Jenkins di Docker (Linux host, direkomendasikan)

```
# buat volume untuk data Jenkins
docker volume create jenkins_home
# jalankan Jenkins dan mount socket Docker
docker run -d --name jenkins-docker \
  -p 8080:8080 -p 50000:50000 \
  -v jenkins_home:/var/jenkins_home \
  -v /var/run/docker.sock:/var/run/docker.sock \
  jenkins/jenkins:lts
```

Kelebihan: Jenkins berjalan di lingkungan Linux, mudah akses Docker socket (/var/run/docker.sock) sehingga pipeline dapat membangun image.

Opsi B — Jenkins di Windows (Docker Desktop)

- Pastikan Docker Desktop berjalan.
- Jika Jenkins dijalankan sebagai Windows Service, ubah *Log On* service Jenkins ke akun user yang menjalankan Docker Desktop agar punya akses ke named pipe Docker (npipe:////./pipe/docker_engine).

---

2. Contoh Proyek (Struktur repo)

```
simple-app/
├── app.py        # contoh Flask app
├── requirements.txt
├── Dockerfile
└── Jenkinsfile
```

Contoh app.py (Flask minimal):

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello():
    return "Halo dari Flask + Docker + Jenkins!"

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000
```

requirements.txt:

```
Flask==2.2.5
```

Dockerfile:

```
FROM python:3.10-slim
WORKDIR /app
COPY requirements.txt ./
RUN pip install --no-cache-dir -r
requirements.txt
COPY . .
EXPOSE 5000
```

---

3. Jenkinsfile — Contoh (Linux agent)

```
pipeline {
  agent any
  environment {
    IMAGE_NAME = 'youruser/simple-app'
    REGISTRY = 'https://index.docker.io/v1/'
    REGISTRY_CREDENTIALS = 'dockerhub-credentials'
  }
  stages {
    stage('Checkout') { steps { checkout scm } }
    stage('Build') {
      steps { sh 'echo "Mulai build aplikasi"' }
    }
    stage('Build Docker Image') {
      steps { script { docker.build("${IMAGE_NAME}:${env.BUILD_NUMBER}") } }
    }
    stage('Push Docker Image') {
      steps {
        script {
          docker.withRegistry(REGISTRY, REGISTRY_CREDENTIALS) {
            def tag = "${IMAGE_NAME}:${env.BUILD_NUMBER}"
            docker.image(tag).push()
            docker.image(tag).push('latest')
          }
        }
      }
    }
  }
  post { always { echo 'Selesai build' } }
}
```

Catatan: pada agent Windows, gunakan bat dan docker login manual (withCredentials). Contoh Windows ada di lampiran.

Jenkinsfile — Contoh (Windows agent)

```
pipeline {
  agent any
  environment {
    IMAGE_NAME = 'youruser/simple-app'
    REGISTRY_CREDENTIALS = 'dockerhub-credentials'
  }
  stages {
    stage('Checkout') { steps { checkout scm } }
    stage('Build') { steps { bat 'echo "Build di Windows"' } }
    stage('Build Docker Image') { steps { bat "docker build -t
%IMAGE_NAME%:%BUILD_NUMBER% ." } }
    stage('Push Docker Image') {
      steps {
        withCredentials([usernamePassword(credentialsId: REGISTRY_CREDENTIALS,
usernameVariable: 'USER', passwordVariable: 'PASS')]) {
          bat 'docker login -u %USER% -p %PASS%'
          bat "docker push %IMAGE_NAME%:%BUILD_NUMBER%"
          bat "docker tag %IMAGE_NAME%:%BUILD_NUMBER% %IMAGE_NAME%:latest"
          bat "docker push %IMAGE_NAME%:latest"
        }
      }
    }
  }
}
```

4. Membuat Personal Access Token (PAT)

GitHub PAT (jika repo private): - Settings → Developer settings → Personal access tokens → Fine-grained / Classic - Scope minimal: repo (atau public_repo untuk repo public)

Docker Hub PAT: - Docker Hub → Account settings → Security → New Access Token - Beri permission: Read, Write, Delete (untuk push)

5. Menyimpan Credential di Jenkins

1. Buka *Manage Jenkins* → *Manage Credentials* → (Global) → *Add Credentials*
2. Tambah credential: **Kind:** Username with password
    - Username: Docker Hub username
    - Password: Docker PAT
    - ID: dockerhub-credentials (catat; akan dipakai di Jenkinsfile)
3. Untuk GitHub (jika private): bisa simpan sebagai **Secret text** (token) atau **Username with password**.

6. Menyimpan Credential di Jenkins

4. New Item → Pipeline → beri nama → OK
5. Di bagian *Pipeline*, pilih *Pipeline script from SCM* jika repo di Git
    - SCM: Git
    - Repository URL: https://github.com/xxx/simple-app.git
    - Credentials (jika private)
    - Script Path: Jenkinsfile
6. Simpan, lalu *Build Now*.

---

7. Checklist Verifikasi (setelah pipeline berjalan)

- ☐ Jenkins berhasil checkout dari repo (cek console log)
- ☐ docker build terlihat berhasil di log
- ☐ docker login & docker push menunjukkan Login Succeeded dan pushed
- ☐ Tag muncul di Docker Hub UI

---

8. Troubleshooting Cepat (kasus umum)

- **sh: command not found** → Agent Windows menjalankan sh. Gunakan bat di Windows pipeline atau jalankan agent Linux.
- **docker daemon is not running** → Jalankan Docker Desktop / service Docker.
- **named pipe error (Windows)** → Pastikan Jenkins service dan Docker Desktop dijalankan oleh user yang sama.
- **credential tidak ditemukan** → Pastikan credentialsId sama dan disimpan di scope Global.
- **token scope insufficient** → Buat token baru dengan permission Read/Write/Delete.
- **push lambat / rate limiting** → Cek koneksi, gunakan registry lokal (Harbor) di lab.

---

9. Latihan & Tugas (Praktek)

1. **Latihan cepat (15 menit):** Clone repo contoh, tambahkan credential Docker Hub di Jenkins, jalankan pipeline, amati proses build & push.
2. **Tugas mandiri (30 menit):** Tambahkan stage Unit Test sebelum Build Docker Image. Gunakan pytest (atau framework test sesuai bahasa). Pipeline hanya boleh push jika test lulus.

---

11. Referensi & Bahan Bacaan

- Official Jenkins docs (Pipeline, Credentials)
- Docker docs (images, login, registries)
- Artikel best-practices: jangan mount /var/run/docker.sock di server publik tanpa proteksi

---