Nama: Latifa Keysha
NIM: 2311083019
Kelas: TRPL 3C

Langkah-Langkah dan Konfigurasi Perpustakaan

> **UPDATED:** Dokumentasi ini sudah disesuaikan dengan konfigurasi terbaru termasuk:
>
> - ☑ Docker Compose dengan RabbitMQ
> - ☑ Kubernetes deployment files
> - ☑ Port mapping yang benar
> - ☑ Semua 10+ microservices

TAHAP 1: Persiapan Lingkungan (Prerequisites)

Sebelum masuk ke deployment, pastikan environment berikut sudah terinstall:

1. **Docker Desktop** (untuk container)
2. **Kubectl** (opsional, untuk Kubernetes deployment)
3. **Java JDK 17+** & **Maven** (untuk development)

Verifikasi instalasi:

```
docker --version
docker-compose --version
java -version
```

Catatan Storage:

- Docker membutuhkan minimal **2-5 GB** free space
- Jika C: drive penuh, pindahkan Docker data ke D: drive via Docker Desktop Settings

---

TAHAP 2: Quick Start dengan Docker Compose (RECOMMENDED)

Cara tercepat untuk menjalankan semua services:

Opsi A: Build & Run Otomatis

```
# Build semua images
.\build-all.ps1

# Jalankan semua services
docker-compose up -d
```

```
# Cek status
docker-compose ps

# Lihat logs
docker-compose logs -f
```

Opsi B: Manual Step-by-Step

Untuk Eureka Server

cd eureka .\mvnw.cmd clean package -DskipTests docker build -t localhost:5000/eurekaservice:1.0 . docker push localhost:5000/eurekaservice:1.0 cd ..

Untuk API Gateway

cd api-get-away .\mvnw.cmd clean package -DskipTests docker build -t localhost:5000/apigatewayservice:1.0 . docker push localhost:5000/apigatewayservice:1.0 cd ..

````
---

## TAHAP 3: Konfigurasi Dasar Kubernetes (Namespace & Secrets)

Berdasarkan struktur project, kita memiliki file `namespace.yaml` dan beberapa
secret.

### 1. namespace.yaml

Ini untuk mengisolasi resource project Anda.

```yaml
apiVersion: v1
kind: Namespace
metadata:
  name: perpustakaan-ns
---
apiVersion: v1
kind: Namespace
metadata:
  name: jenkins-ns
---
apiVersion: v1
kind: Namespace
metadata:
  name: logging-ns
### **Opsi B: Manual Step-by-Step**

```powershell
````

```
# 1. Build semua Docker images
docker-compose build

# 2. Start services
docker-compose up -d

# 3. Verifikasi semua running
docker-compose ps
```

Port Mapping (Docker Compose)

Service	Internal Port	External Port	URL
RabbitMQ AMQP	5672	5672	-
RabbitMQ Management	15672	15672	http://localhost:15672
Eureka	8761	10761	http://localhost:10761
API Gateway	8080	10080	http://localhost:10080
Anggota	8081	10081	http://localhost:10081
Buku	8082	10082	http://localhost:10082
Peminjaman	8084	10083	http://localhost:10083
Pengembalian	8085	10084	http://localhost:10084
Command	8088	10085	http://localhost:10085
Query	8087	10086	http://localhost:10086
RabbitMQ Peminjaman	8086	10087	http://localhost:10087
Rabbit Pengembalian	8089	10088	http://localhost:10088

Testing

```
# Cek Eureka Dashboard
Start-Process "http://localhost:10761"

# Cek RabbitMQ Management
Start-Process "http://localhost:15672"  # guest/guest

# Test API via Gateway
curl http://localhost:10080/anggota-service/anggota
curl http://localhost:10080/buku-service/buku
```

Stop Services

```
docker-compose down
```

TAHAP 3: Kubernetes Deployment (Alternative - Advanced)

Jika ingin deploy ke Kubernetes (lebih complex, untuk production):

Quick Deploy

```
# Deploy semua services sekaligus
kubectl apply -f kubernetes/deploy-all.yaml

# Atau deploy bertahap
kubectl apply -f kubernetes/namespace.yaml
kubectl apply -f kubernetes/rabbitmq-deployment.yaml
kubectl apply -f kubernetes/eureka-deployment.yaml
kubectl apply -f kubernetes/api-gateway-deployment.yaml
kubectl apply -f kubernetes/anggota-service-deployment.yaml
kubectl apply -f kubernetes/buku-service-deployment.yaml
# ... dst untuk service lainnya
```

Monitoring Kubernetes

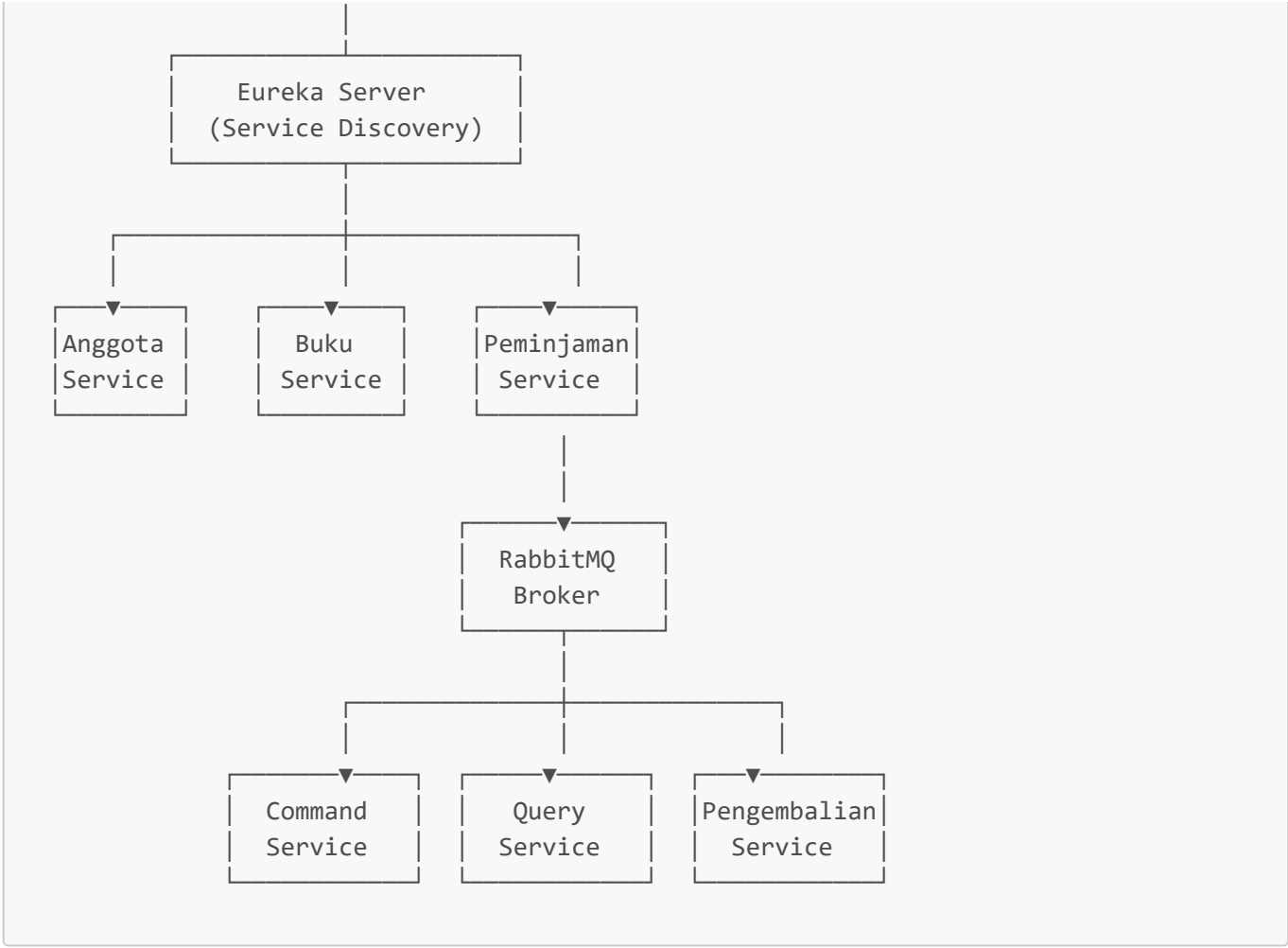
```
# Lihat semua pods
kubectl get pods -n perpustakaan-ns

# Lihat services
kubectl get svc -n perpustakaan-ns

# Lihat logs
kubectl logs -f deployment/anggota-deployment -n perpustakaan-ns

# Port forward untuk testing
kubectl port-forward svc/eureka-server 8761:8761 -n perpustakaan-ns
```

Dokumentasi lengkap Kubernetes: Lihat kubernetes/README.md

TAHAP 4: Arsitektur & Konfigurasi Services

Microservices Architecture

```
┌─────────────────────────────────────────────────┐
│                  API Gateway                    │
│                  (Port 8080)                    │
└──────────────────┬──────────────────────────────┘
```



Database Configuration

Semua services menggunakan **H2 in-memory database**:

- **Keuntungan:** Ringan, tidak perlu setup database eksternal
- **Kekurangan:** Data hilang saat restart

Contoh konfigurasi (application.properties):

```
spring.datasource.url=jdbc:h2:mem:anggotadb
spring.datasource.driverClassName=org.h2.Driver
spring.datasource.username=sa
spring.datasource.password=password
spring.h2.console.enabled=true
spring.jpa.hibernate.ddl-auto=create-drop
```

RabbitMQ Configuration

Services yang menggunakan RabbitMQ:

- Peminjaman Service (producer)
- Pengembalian Service (producer)
- Command Service (consumer)
- Query Service (consumer)

- RabbitMQ Peminjaman Service (consumer)
- Rabbit Pengembalian Service (consumer)

Environment variables (otomatis dari docker-compose.yml):

```
SPRING_RABBITMQ_HOST=rabbitmq
SPRING_RABBITMQ_PORT=5672
SPRING_RABBITMQ_USERNAME=guest
SPRING_RABBITMQ_PASSWORD=guest
```

TAHAP 5: Troubleshooting

Service tidak start

```
# Cek logs
docker-compose logs <service-name>

# Restart specific service
docker-compose restart <service-name>
```

Port sudah dipakai

```
# Cek port yang dipakai
netstat -ano | findstr :<port>

# Atau stop semua dan restart
docker-compose down
docker-compose up -d
```

Image error / build gagal

```
# Rebuild tanpa cache
docker-compose build --no-cache

# Atau rebuild specific service
docker-compose build --no-cache <service-name>
```

C: Drive penuh

```
# Cleanup Docker
docker system prune -a -f
```

```
# Atau pindahkan Docker ke D: drive
# Settings > Resources > Advanced > Disk image location
```

TAHAP LAMA (Reference): Build Manual per Service (TIDAK DIGUNAKAN LAGI)

Untuk keperluan development atau troubleshooting, berikut cara build manual:

```
# Build per service
cd <service-folder>
.\mvnw.cmd clean package -DskipTests
docker build -t perpustakaan/<service-name>:latest .
cd ..
```

Catatan: Docker Compose sudah melakukan ini secara otomatis via `build-all.ps1`

TAHAP 5: Deployment Microservices (Kubernetes Reference)

Ini adalah bagian inti (anggota-service, buku-service, peminjaman-service, dll). Berikut template umum yang bisa Anda copy-paste dan ubah namanya.

Contoh: anggota-service-deployment.yaml

```
# Anggota Service Deployment & Service
# Author: Latifa Keysha (2311083019) - TRPL 3C

apiVersion: apps/v1
kind: Deployment
metadata:
  name: anggota-deployment
  namespace: perpustakaan-ns
  labels:
    app: anggota
    environment: production
spec:
  replicas: 2  # Jumlah instance
  selector:
    matchLabels:
      app: anggota
  template:
    metadata:
      labels:
        app: anggota
        environment: production
    spec:
      containers:
      - name: anggota-container
```

```
        image: perpustakaan/anggota-service:latest  # Sesuaikan image
        imagePullPolicy: Always
        ports:
        - containerPort: 8081  # Sesuaikan port aplikasi Spring Boot
        env:
        - name: EUREKA_CLIENT_SERVICEURL_DEFAULTZONE
          value: "http://eureka-service:8761/eureka"
        - name: SPRING_RABBITMQ_HOST
          value: "rabbitmq"
        - name: SPRING_RABBITMQ_PORT
          value: "5672"
        - name: SPRING_RABBITMQ_USERNAME
          valueFrom:
            secretKeyRef:
              name: rabbitmq-secret
              key: rabbitmq-username
        - name: SPRING_RABBITMQ_PASSWORD
          valueFrom:
            secretKeyRef:
              name: rabbitmq-secret
              key: rabbitmq-password
        resources:
          requests:
            memory: "256Mi"
            cpu: "100m"
          limits:
            memory: "512Mi"
            cpu: "250m"
        livenessProbe:
          httpGet:
            path: /actuator/health
            port: 8081
          initialDelaySeconds: 60
          periodSeconds: 10
        readinessProbe:
          httpGet:
            path: /actuator/health
            port: 8081
          initialDelaySeconds: 30
          periodSeconds: 5
---
apiVersion: v1
kind: Service
metadata:
  name: anggota-service
  namespace: perpustakaan-ns
spec:
  selector:
    app: anggota
  ports:
  - port: 8081
    targetPort: 8081
  type: ClusterIP
```

Contoh: buku-service-deployment.yaml

```yaml
# Buku Service Deployment & Service
# Author: Latifa Keysha (2311083019) - TRPL 3C

apiVersion: apps/v1
kind: Deployment
metadata:
  name: buku-deployment
  namespace: perpustakaan-ns
  labels:
    app: buku
    environment: production
spec:
  replicas: 2
  selector:
    matchLabels:
      app: buku
  template:
    metadata:
      labels:
        app: buku
        environment: production
    spec:
      containers:
      - name: buku-container
        image: perpustakaan/buku-service:latest
        imagePullPolicy: Always
        ports:
        - containerPort: 8082
        env:
        - name: EUREKA_CLIENT_SERVICEURL_DEFAULTZONE
          value: "http://eureka-service:8761/eureka"
        - name: SPRING_RABBITMQ_HOST
          value: "rabbitmq"
        - name: SPRING_RABBITMQ_PORT
          value: "5672"
        resources:
          requests:
            memory: "256Mi"
            cpu: "100m"
          limits:
            memory: "512Mi"
            cpu: "250m"
        livenessProbe:
          httpGet:
            path: /actuator/health
            port: 8082
          initialDelaySeconds: 60
          periodSeconds: 10
        readinessProbe:
```

```
          httpGet:
            path: /actuator/health
            port: 8082
          initialDelaySeconds: 30
          periodSeconds: 5
---
apiVersion: v1
kind: Service
metadata:
  name: buku-service
  namespace: perpustakaan-ns
spec:
  selector:
    app: buku
  ports:
  - port: 8082
    targetPort: 8082
  type: ClusterIP
```

Catatan: Ulangi pattern yang sama untuk **peminjaman-service**, **pengembalian-service**, **command-service**, dan **query-service** dengan menyesuaikan:

- Nama deployment
- Nama container
- Image name
- Port number
- Service name

TAHAP 6: Monitoring (Prometheus & Grafana)

1. Konfigurasi Prometheus di Application

Setiap service sudah dikonfigurasi dengan **Actuator** dan **Prometheus**:

application.properties (untuk semua service):

```
# Actuator & Prometheus untuk Monitoring
management.endpoint.health.show-details=always
management.endpoints.web.exposure.include=health,info,metrics,prometheus
management.endpoint.health.probes.enabled=true
management.metrics.export.prometheus.enabled=true
```

2. Akses Prometheus Metrics

Setiap service expose metrics di:

```
http://localhost:8081/actuator/prometheus  (Anggota)
http://localhost:8082/actuator/prometheus  (Buku)
```

```
http://localhost:8084/actuator/prometheus  (Peminjaman)
http://localhost:8085/actuator/prometheus  (Pengembalian)
```

3. Konfigurasi Grafana

Jika menggunakan Grafana untuk visualisasi:

1. Akses Grafana: `http://localhost:<Grafana_NodePort>`
2. Login default: `admin/admin`
3. Add Data Source → Pilih **Prometheus**
4. URL: `http://prometheus-service:9090`
5. Save & Test
6. Import Dashboard ID **4701** (JVM Micrometer) untuk melihat statistik Java otomatis

---

TAHAP 7: Jenkins (CI/CD)

Untuk otomatisasi deploy.

jenkins-deployment.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: jenkins
  namespace: jenkins-ns
spec:
  replicas: 1
  selector:
    matchLabels:
      app: jenkins
  template:
    metadata:
      labels:
        app: jenkins
    spec:
      containers:
      - name: jenkins
        image: jenkins/jenkins:lts
        ports:
        - containerPort: 8080
        - containerPort: 50000
        volumeMounts:
        - name: jenkins-home
          mountPath: /var/jenkins_home
      volumes:
      - name: jenkins-home
        persistentVolumeClaim:
          claimName: jenkins-pvc
---
```

```
apiVersion: v1
kind: Service
metadata:
  name: jenkins-service
  namespace: jenkins-ns
spec:
  selector:
    app: jenkins
  ports:
  - name: http
    port: 8080
    targetPort: 8080
    nodePort: 30080
  - name: jnlp
    port: 50000
    targetPort: 50000
  type: NodePort
```

jenkins-pvc.yaml

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: jenkins-pvc
  namespace: jenkins-ns
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 5Gi
```

Konfigurasi Jenkins:

1. Akses Jenkins: `http://localhost:30080`
2. Get password awal:

   ```
   kubectl exec -n jenkins-ns -it deployment/jenkins -- cat
   /var/jenkins_home/secrets/initialAdminPassword
   ```

3. Install plugins: **Docker Pipeline**, **Kubernetes**, **Git**, **Maven Integration**
4. Klik **New Item**
5. Pilih **Pipeline**, beri nama `anggota-service-pipeline`
6. Scroll ke bawah bagian **Pipeline**
7. **Definition**: Pipeline script from SCM
8. **SCM**: Git
9. **Repository URL**: Masukkan link GitHub/GitLab project Anda

10. **Script Path**: `anggota/Jenkinsfile` (Sesuaikan dengan lokasi Jenkinsfile di repo)
11. **Save** & **Build Now**

Contoh Jenkinsfile (anggota/Jenkinsfile):

```
pipeline {
    agent any

    environment {
        // --- KONFIGURASI KHUSUS ANGGOTA SERVICE ---
        // Author: Latifa Keysha (2311083019) - TRPL 3C
        SERVICE_NAME = 'anggotaservice'
        DOCKER_DIR = 'anggota'
        K8S_DEPLOYMENT = 'anggota-deployment'
        K8S_NS = 'perpustakaan-ns'
        CONTAINER_NAME = 'anggota-container'

        // Gunakan Tag Statis 1.0 agar hemat penyimpanan
        IMAGE_TAG = "localhost:5000/${SERVICE_NAME}:1.0"
    }

    tools {
        maven 'maven-3'
    }

    stages {
        stage('Initialize') {
            steps {
                echo '========== Stage 1: Initialize =========='
                sh 'docker --version'
                sh 'kubectl version --client'
                sh 'mvn --version'
            }
        }

        stage('Build Maven') {
            steps {
                echo '========== Stage 2: Build Maven =========='
                dir("${DOCKER_DIR}") {
                    sh 'mvn clean package -DskipTests'
                }
            }
        }

        stage('Build Docker Image') {
            steps {
                echo '========== Stage 3: Build Docker Image =========='
                dir("${DOCKER_DIR}") {
                    sh "docker build -t ${IMAGE_TAG} ."
                }
            }
        }
```

```
        stage('Push Image') {
            steps {
                echo '========== Stage 4: Push Image =========='
                sh "docker push ${IMAGE_TAG}"
            }
        }

        stage('Deploy to Kubernetes') {
            steps {
                echo '========== Stage 5: Deploy to Kubernetes =========='
                sh """
                kubectl set image deployment/${K8S_DEPLOYMENT} \
                ${CONTAINER_NAME}=${IMAGE_TAG} -n ${K8S_NS}

                kubectl rollout restart deployment/${K8S_DEPLOYMENT} -n ${K8S_NS}

                kubectl rollout status deployment/${K8S_DEPLOYMENT} -n ${K8S_NS} -
--timeout=300s
                """
            }
        }
    }

    post {
        success {
            echo 'Pipeline berhasil!'
        }
        failure {
            echo 'Pipeline gagal!'
        }
        always {
            sh 'docker image prune -f'
        }
    }
}
```

Ulangi konfigurasi pipeline untuk semua service (buku, peminjaman, pengembalian, dll).

TAHAP 8: Eksekusi (Urutan Deploy)

Jalankan perintah berikut di terminal (pastikan berada di folder root `perpustakaan`):

Setup Namespace & Config:

```
kubectl apply -f kubernetes/01-namespace/namespace.yaml
kubectl apply -f kubernetes/02-secrets/rabbitmq-secret.yaml
```

Infrastructure (Message Broker):

```
kubectl apply -f kubernetes/03-infrastructure/rabbitmq-deployment.yaml

# Tunggu hingga ready
kubectl wait --for=condition=ready pod -l app=rabbitmq -n perpustakaan-ns --
timeout=180s
```

Core Services (Urutan Penting):

Deploy Eureka dulu (agar service lain bisa register):

```
kubectl apply -f kubernetes/04-services/eureka-deployment.yaml

# Tunggu sebentar sampai eureka up
kubectl wait --for=condition=ready pod -l app=eureka -n perpustakaan-ns --
timeout=180s
```

Deploy Services (Anggota, Buku, Peminjaman, Pengembalian, Command, Query):

```
kubectl apply -f kubernetes/04-services/anggota-deployment.yaml
kubectl apply -f kubernetes/04-services/buku-deployment.yaml
kubectl apply -f kubernetes/04-services/peminjaman-deployment.yaml
kubectl apply -f kubernetes/04-services/pengembalian-deployment.yaml
kubectl apply -f kubernetes/04-services/command-deployment.yaml
kubectl apply -f kubernetes/04-services/query-deployment.yaml
```

Catatan: Saat ini di folder `04-services` hanya ada `anggota-deployment.yaml` dan `eureka-deployment.yaml`. Anda perlu membuat file deployment untuk service lainnya mengikuti template yang sama.

Deploy API Gateway terakhir:

```
kubectl apply -f kubernetes/04-services/api-gateway-deployment.yaml
```

DevOps Tools:

```
kubectl apply -f kubernetes/07-jenkins/jenkins-pvc.yaml
kubectl apply -f kubernetes/07-jenkins/jenkins-deployment.yaml
```

Verifikasi:

```

```

```
# Cek semua pods
kubectl get pods -n perpustakaan-ns

# Cek services
kubectl get svc -n perpustakaan-ns

# Cek logs jika ada error
kubectl logs -n perpustakaan-ns deployment/anggota-deployment
```

File Konfigurasi Penting

1. application.properties (Template untuk semua service)

```
# Server
server.port=8081  # Sesuaikan per service

# Application Name
spring.application.name=anggota  # Sesuaikan per service

# H2 Database
spring.datasource.url=jdbc:h2:mem:anggotadb  # Sesuaikan per service
spring.datasource.driverClassName=org.h2.Driver
spring.datasource.username=sa
spring.datasource.password=password
spring.h2.console.enabled=true
spring.jpa.database-platform=org.hibernate.dialect.H2Dialect
spring.jpa.hibernate.ddl-auto=create-drop

# Eureka Client
eureka.client.register-with-eureka=true
eureka.client.fetch-registry=true
eureka.client.service-url.defaultZone=http://localhost:8761/eureka

# Actuator & Prometheus
management.endpoint.health.show-details=always
management.endpoints.web.exposure.include=health,info,metrics,prometheus
management.endpoint.health.probes.enabled=true
management.metrics.export.prometheus.enabled=true
```

2. pom.xml (Dependencies Utama)

```
<dependencies>
    <!-- Spring Boot Web -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
```

```
        <!-- Spring Data JPA -->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>

        <!-- H2 Database -->
        <dependency>
            <groupId>com.h2database</groupId>
            <artifactId>h2</artifactId>
            <scope>runtime</scope>
        </dependency>

        <!-- Eureka Client -->
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
        </dependency>

        <!-- Actuator -->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-actuator</artifactId>
        </dependency>

        <!-- Prometheus -->
        <dependency>
            <groupId>io.micrometer</groupId>
            <artifactId>micrometer-registry-prometheus</artifactId>
        </dependency>

        <!-- Lombok -->
        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
            <optional>true</optional>
        </dependency>
    </dependencies>
```

Ringkasan Port Services

Service	Port	Endpoint	Image Name	Status
Anggota	8081	/api/anggota	localhost:5000/anggotaservice:1.0	☑ Ada
Buku	8082	/api/buku	localhost:5000/bukuservice:1.0	☑ Ada
Peminjaman	8084	/api/peminjaman	localhost:5000/peminjamanservice:1.0	☑ Ada
Pengembalian	8085	/api/pengembalian	localhost:5000/pengembalianservice:1.0	☑ Ada

Service	Port	Endpoint	Image Name	Status
Command Service	8088	/api/command	localhost:5000/commandservice:1.0	☑ Ada
Query Service	8087	/api/query	localhost:5000/queryservice:1.0	☑ Ada
Eureka Server	8761	/eureka	localhost:5000/eurekaservice:1.0	☑ Ada
API Gateway	8080	/*	localhost:5000/apigatewayservice:1.0	☑ Ada
Jenkins	30080	-	jenkins/jenkins:lts	☑ Ada
RabbitMQ	5672	-	rabbitmq:3-management	☑ Ada
RabbitMQ UI	15672	-	-	☑ Ada

Dokumentasi ini dibuat oleh:
Latifa Keysha - 2311083019 - TRPL 3C

Terakhir diperbarui: 18 Januari 2026