**🔧 Common Stack for All Projects**

* **Frontend**: Thymeleaf (Spring MVC)
* **Microservices**: 3 per project
* **Service Discovery**: Eureka Server
* **Monitoring/Tracing**:
  + Prometheus & Grafana
  + ELK Stack (Elasticsearch, Logstash, Kibana)
  + Zipkin (distributed tracing)
* **Databases**: H2 (for quick testing), MySQL, MongoDB
* **API Tools**:
  + Swagger (combined documentation using SpringDoc OpenAPI Gateway Aggregation)
* **Security & Quality**:
  + Snyk (dependency vulnerability scan)
  + SonarQube (code quality)
* **Performance Testing**:
  + JMeter
  + Gatling

**📦 Project Structure (Per Project)**

**🔁 Microservices**

* **Service 1**: Core Business Logic (e.g., Users, Products, etc.)
* **Service 2**: Related Service (e.g., Orders, Reviews)
* **Service 3**: Supporting Service (e.g., Notifications, Inventory)

**🌐 Gateway**

* API Gateway with Spring Cloud Gateway + Swagger aggregation

**🔍 Discovery**

* Eureka Server for service registration

**🧪 DevOps & Observability Setup**

Each service should have:

* Actuator endpoints (/actuator/prometheus, /actuator/health, etc.)
* logback-spring.xml setup for Logstash
* Zipkin tracing enabled via Spring Cloud Sleuth
* Dockerfile for containerization
* docker-compose for local infra (Prometheus, Grafana, ELK, Zipkin, MySQL, MongoDB)

**📁 Project Ideas**

**1. Smart Library Management**

* User Service (registration, login, roles)
* Book Service (add/view/search books)
* Notification Service (late return alerts)

**2. Mini e-Commerce**

* Product Service (catalog)
* Order Service (create/view orders)
* Review Service (product reviews)

**3. Event Management Platform**

* Event Service (event creation, schedule)
* Ticket Service (booking, cancellation)
* Email Service (confirmation emails)

**4. Online Food Ordering**

* Restaurant Service (menus, locations)
* Cart/Order Service
* Delivery Tracking Service

**5. Healthcare Appointment System**

* Patient Service
* Appointment Scheduling Service
* Doctor Service

**6. Job Recruitment Portal**

* Job Posting Service
* Application Service
* Resume Parser Service (can use Mongo for unstructured resumes)

**Create private Github Repo for each projects. Add all team members and me as a collaborator. Maintain all the codes in the private repo only.**

**Add a detailed README.md file in each service and also to the application root folder.**

**🛠️ Tools & Setup Script**

* Project scaffolding templates
* docker-compose.yml with Prometheus, Grafana, Zipkin, ELK, MySQL, MongoDB
* Spring Boot base config for Prometheus, ELK, Sleuth, Eureka, Swagger
* pom.xml templates with plugins for Snyk, SonarQube
* JMeter + Gatling test cases
* Thymeleaf templates for basic UI (login, dashboard, etc.)
* API Gateway with OpenAPI aggregation for combined Swagger

Sample Folder Structure



bus-ticket-booking-app/

├── eureka-server/

├── api-gateway/

├── frontend-thymeleaf/

├── services/

│ ├── user-service/

│ ├── booking-service/

│ └── bus-service/

├── monitoring/

│ ├── prometheus/

│ ├── grafana/

│ ├── zipkin/

│ └── elk/

│ ├── logstash/

│ ├── kibana/

│ └── elasticsearch/

├── infrastructure/

│ ├── mysql/

│ ├── mongodb/

│ └── h2/

├── performance/

│ ├── gatling/

│ └── jmeter/

├── quality/

│ └── sonarqube/

├── security/

│ └── snyk/

└── swagger-docs/

The detailed **step-by-step setup** for your **Bus Ticket Booking Microservice App**:

**✅ Step 1: pom.xml for Each Service**

Let’s use **Maven** for this setup.

**🎯 Common Dependencies (for all services)**

Place these in each pom.xml:

<dependencies>

<!-- Spring Boot Starters -->

<dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-web</artifactId> </dependency>

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

<dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-data-jpa</artifactId> </dependency>

<dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-security</artifactId> </dependency>

<dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-validation</artifactId> </dependency>

<!-- Discovery Client -->

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

<!-- Zipkin Tracing -->

<dependency> <groupId>org.springframework.cloud</groupId> <artifactId>spring-cloud-starter-zipkin</artifactId> </dependency>

<dependency> <groupId>org.springframework.cloud</groupId> <artifactId>spring-cloud-starter-sleuth</artifactId> </dependency>

<!-- Swagger/OpenAPI -->

<dependency> <groupId>org.springdoc</groupId> <artifactId>springdoc-openapi-ui</artifactId> <version>1.6.15</version> </dependency>

<!-- Database (MySQL/Mongo depending on service) -->

<dependency> <groupId>mysql</groupId> <artifactId>mysql-connector-java</artifactId> </dependency>

<dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-data-mongodb</artifactId> </dependency>

<!-- Dev Tools -->

<dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-devtools</artifactId> <scope>runtime</scope> </dependency>

</dependencies>

Add this to properties:

<properties>

<java.version>17/21</java.version>

<spring-cloud.version>2025.0.0-RC1</spring-cloud.version>

</properties>

And for dependencyManagement:

<dependencyManagement>

<dependencies>

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-dependencies</artifactId>

<version>${spring-cloud.version}</version>

<type>pom</type>

<scope>import</scope>

</dependency>

</dependencies>

</dependencyManagement>

**✅ Step 2: docker-compose.yml (Observability + DBs)**

Create a docker-compose.yml in the docker/ directory:

version: '3.8'

services:

mysql:

image: mysql:8

environment:

MYSQL\_ROOT\_PASSWORD: root

MYSQL\_DATABASE: ticketdb

ports:

- "3307:3306"

mongodb:

image: mongo:5

ports:

- "27018:27017"

zipkin:

image: openzipkin/zipkin

ports:

- "9411:9411"

prometheus:

image: prom/prometheus

volumes:

- ./monitoring/prometheus/prometheus.yml:/etc/prometheus/prometheus.yml

ports:

- "9090:9090"

grafana:

image: grafana/grafana

ports:

- "3000:3000"

elasticsearch:

image: docker.elastic.co/elasticsearch/elasticsearch:8.8.0

environment:

discovery.type: single-node

ports:

- "9200:9200"

logstash:

image: docker.elastic.co/logstash/logstash:8.8.0

volumes:

- ./logging/elk/logstash/logstash.conf:/usr/share/logstash/pipeline/logstash.conf

ports:

- "5000:5000"

kibana:

image: docker.elastic.co/kibana/kibana:8.8.0

ports:

- "5601:5601"

Also create a basic prometheus.yml config:

global:

scrape\_interval: 10s

scrape\_configs:

- job\_name: 'spring-boot-apps'

metrics\_path: '/actuator/prometheus'

static\_configs:

- targets: ['host.docker.internal:8081', 'host.docker.internal:8082']

**✅ Step 3: Spring Boot Main Class + application.yml**

**Main Class (BookingServiceApplication.java)**

@SpringBootApplication

@EnableDiscoveryClient

public class BookingServiceApplication {

public static void main(String[] args) {

SpringApplication.run(BookingServiceApplication.class, args);

}

}

**application.yml (sample for booking service)**

server:

port: 8081

spring:

application:

name: booking-service

datasource:

url: jdbc:mysql://localhost:3306/ticketdb

username: root

password: root

jpa:

hibernate:

ddl-auto: update

show-sql: true

eureka:

client:

service-url:

defaultZone: http://localhost:8761/eureka

management:

endpoints:

web:

exposure:

include: "\*"

tracing:

sampling:

probability: 1.0

Apply similar application.yml and port changes for other services (bus-service, notification-service, etc.).

**✅ Step 4: API Gateway + Swagger Aggregation**

**api-gateway Main Dependencies**

Add to pom.xml:

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-gateway</artifactId>

</dependency>

<dependency>

<groupId>org.springdoc</groupId>

<artifactId>springdoc-openapi-webflux-ui</artifactId>

<version>1.6.15</version>

</dependency>

**application.yml for Gateway**

server:

port: 8080

spring:

application:

name: api-gateway

cloud:

gateway:

routes:

- id: booking-service

uri: http://localhost:8081

predicates:

- Path=/booking/\*\*

- id: bus-service

uri: http://localhost:8082

predicates:

- Path=/bus/\*\*

- id: notification-service

uri: http://localhost:8083

predicates:

- Path=/notify/\*\*

eureka:

client:

service-url:

defaultZone: http://localhost:8761/eureka

**Combined Swagger Aggregation (OpenAPI WebFlux)**

Each service should expose its OpenAPI spec:

/v3/api-docs

API Gateway can combine them using SpringDoc Swagger UI + manually build index.