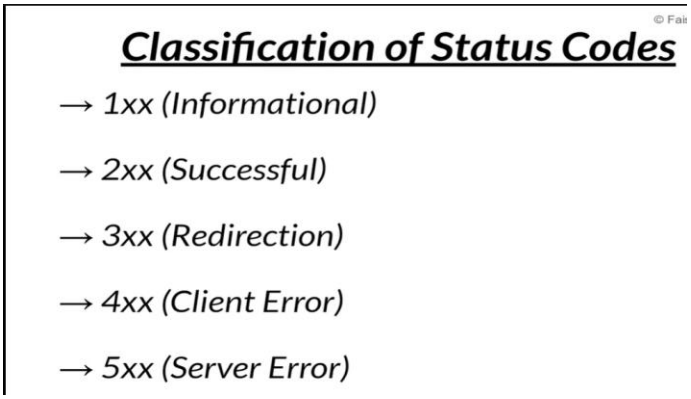
Java, Microservices, Kubernetes

Status Codes



→ 200 OK

→ 201 Created

→ 204 No Content

→ 301 Moved Permanently

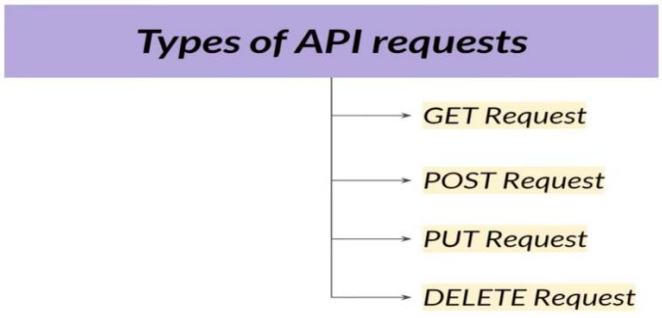
→ 400 Bad Request

→ 401 Unauthorized

→ 403 Forbidden

→ 404 Not Found

→ 500 Internal Server Error



GET Request

→ Retrieve or GET resources from server

→ Used only to read data

POST Request

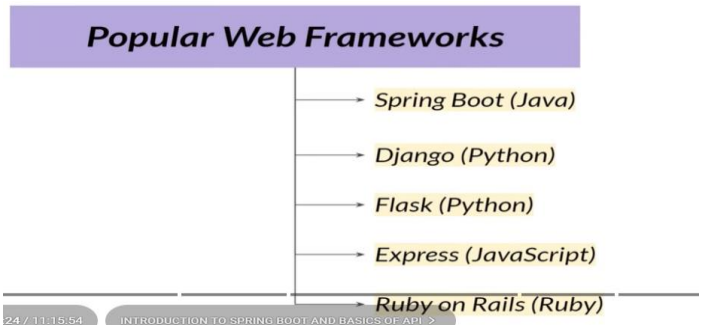
→ Create resources from server

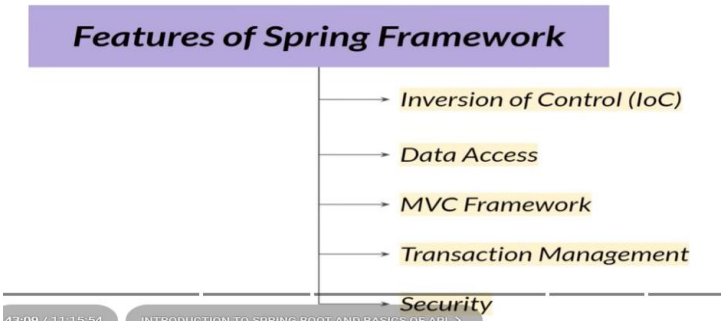
PUT Request

→ Update existing resources on Server

DELETE Request

→ Used to DELETE resources from Server





Spring boot = Spring Framework + Prebuilt Configuration + Embedded Servers

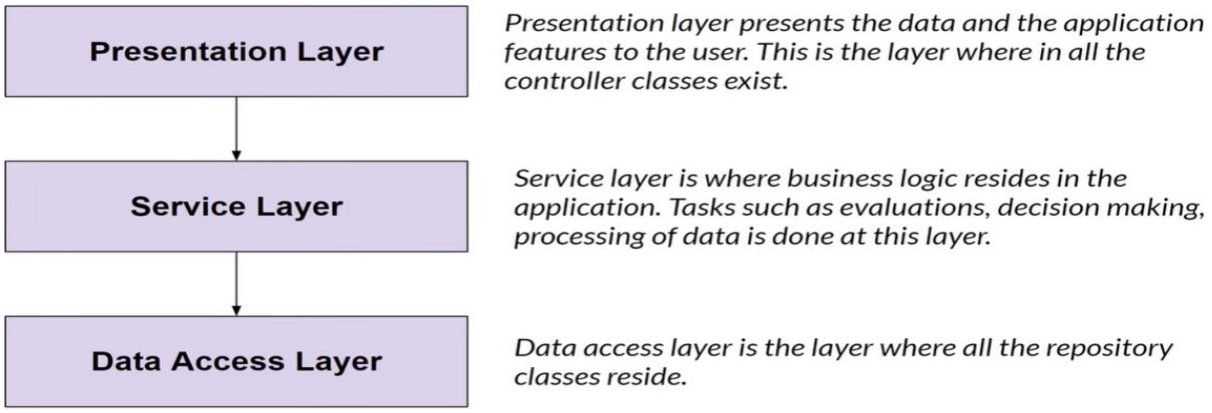
Components of Spring Boot

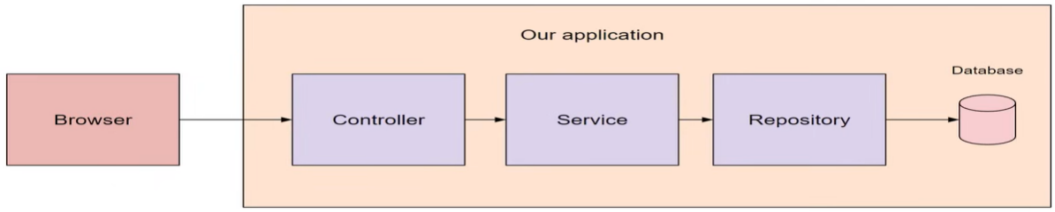
- → *Spring Boot Starters*
- → *Auto Configuration*
- → *Spring Boot Actuator*
- → *Embedded Server*
- → *Spring Boot DevTools*

Advantages of Spring Boot

- → *Stand alone and Quick Start*
- → *Starter code*
- → *Less configuration*
- → *Reduced cost and application development time*

Spring Boot Architecture

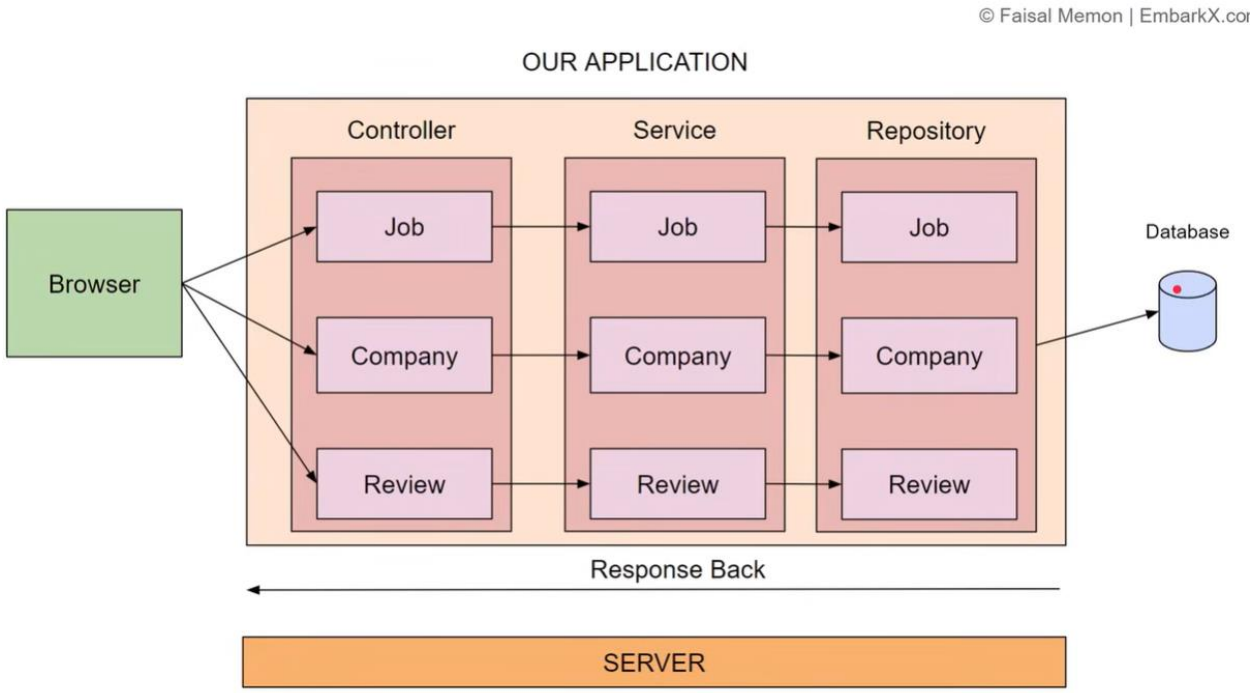




Dispatcher servlet forwarded request to appropriate controller

```
 .FirstSpringApplication   : Started FirstSpringApplication in 1.362 seconds (process
  running for 1.665)
2023-07-06T14:51:16.906+05:30  INFO 11884 --- [nio-8080-exec-1] o.a.c.c.C.[Tomcat]
 .[localhost].[/]       : Initializing Spring DispatcherServlet 'dispatcherServlet'
2023-07-06T14:51:16.906+05:30  INFO 11884 --- [nio-8080-exec-1] o.s.web.servlet
.DispatcherServlet        : Initializing Servlet 'dispatcherServlet'
2023-07-06T14:51:16.907+05:30  INFO 11884 --- [nio-8080-exec-1] o.s.web.servlet
 .DispatcherServlet        : Completed initialization in 1 ms
```

Creating a sample application on Job posting



What is JPA?

```
public class Job {
    private  Long id;
    private  String title;
    private  String description;
    private  String minSalary;
    private  String maxSalary;
    private  String location;
}
```

id	title	description	minSalary	maxSalary	location
1	Senior Software Engineer	Senior Software Engineer	30000	40000	40000

Transfer objects into relational databases

Advantages of using JPA

→ *Easy and Simple*

→ *Makes querying easier*

→ *Allows to save and update objects*

→ *Easy integration with Spring Boot*

Add dependencies

Spring Data JPA SQL

Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.

H2 Database SQL

Provides a fast in-memory database that supports JDBC API and R2DBC access, with a small (2mb) footprint. Supports embedded and server modes as well as a browser based console application.

Companies

```
GET /companies
PUT /companies/{id}
POST /companies
DELETE /companies/{id}
GET /companies/{id}
```

Reviews

```
GET /companies/{companyId}/reviews
POST /companies/{companyId}/reviews
GET /companies/{companyId}/reviews/{reviewId}
PUT /companies/{companyId}/reviews/{reviewId}
DELETE /companies/{companyId}/reviews/{reviewId}
```

Introduction to Spring Boot Actuator

What is it?

Provides built-in production-ready features to monitor and manage your application

Gives you the ability to monitor and manage your applications

Features

→ *Built in endpoints*

→ *Ability to view real time metrics*

→ *Customizable*

Spring Boot Actuator OPS

Supports built in (or custom) endpoints that let you monitor and manage your application - such as application health, metrics, sessions, etc.

Expose all endpoints in actuators

application.properties

```
spring.application.name=job
server.port=7071
spring.h2.console.enabled=true
spring.datasource.url=jdbc:h2:mem:test
spring.jpa.generate-ddl=true
spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=update

#Actuators
management.endpoints.web.exposure.include=*
```

localhost:7071/actuator

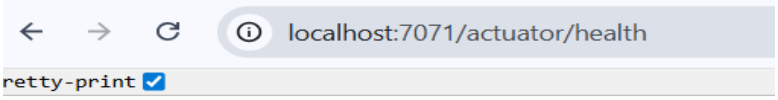
tty-print


```
_links": {
  "self": {
    "href": "http://localhost:7071/actuator",
    "templated": false
  },
  "beans": {
    "href": "http://localhost:7071/actuator/beans",
    "templated": false
  },
  "conditions": {
    "href": "http://localhost:7071/actuator/conditions",
    "templated": false
  },
  "configprops": {
    "href": "http://localhost:7071/actuator/configprops",
    "templated": false
  },
  "configprops-prefix": {
    "href": "http://localhost:7071/actuator/configprops/{prefix}",
    "templated": true
  },
  "env": {
    "href": "http://localhost:7071/actuator/env",
    "templated": false
  },
  "env-toMatch": {
    "href": "http://localhost:7071/actuator/env/{toMatch}",
    "templated": true
  },
  "info": {
    "href": "http://localhost:7071/actuator/info",
    "templated": false
  },
  "loggers": {
    "href": "http://localhost:7071/actuator/loggers",
    "templated": false
  },
  "loggers-name": {
    "href": "http://localhost:7071/actuator/loggers/{name}",
    "templated": true
  },
  "threaddump": {
    "href": "http://localhost:7071/actuator/threaddump",
    "templated": false
  },
  "sbom": {
    "href": "http://localhost:7071/actuator/sbom",
    "templated": false
  },
```


Understanding Actuator Endpoints

Endpoint	Purpose
/health	Shows application health information, useful for checking the status of the application, such as database connectivity, disk space, and custom health checks.
/info	Displays arbitrary application information, commonly used to display application version, git commit information, etc.
/metrics	Shows 'metrics' information that allows you to understand the performance and behavior of your running application.
/loggers	Allows you to query and modify the logging level of your application's loggers.
/beans	Provides a complete list of all the Spring beans in your application
/shutdown	Allows your application to be gracefully shut down

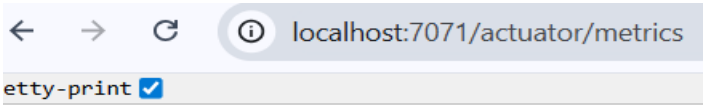
```
management.endpoint.health.show-details=always
```




```
"components": {
  "db": {
    "details": {
      "database": "H2",
      "validationQuery": "isValid()"
    },
    "status": "UP"
  },
  "diskSpace": {
    "details": {
      "total": 107373129728,
      "free": 57136533504,
      "threshold": 10485760,
      "path": "D:\\Learning\\Projects\\microservices\\job\\.",
      "exists": true
    },
    "status": "UP"
  },
  "livenessState": {
    "status": "UP"
  },
  "ping": {
    "status": "UP"
  },
  "readinessState": {
    "status": "UP"
  },
  "ssl": {
    "details": {
      "expiringChains": [],
      "invalidChains": [],
      "validChains": []
    },
    "status": "UP"
  }
},
"groups": [
  "liveness",
  "readiness"
],
"status": "UP"
```


We can check metrics

```
},
"metrics": {
  "href": "http://localhost:7071/actuator/metrics",
  "templated": false
},
"metrics-requiredMetricName": {
  "href": "http://localhost:7071/actuator/metrics/{requiredMetricName}",
  "templated": true
}
```


```
"names": [
  "application.ready.time",
  "application.started.time",
  "disk.free",
  "disk.total",
  "executor.active",
  "executor.completed",
  "executor.pool.core",
  "executor.pool.max",
  "executor.pool.size",
  "executor.queue.remaining",
  "executor.queued",
  "hikaricp.connections",
  "hikaricp.connections.acquire",
  "hikaricp.connections.active",
  "hikaricp.connections.creation",
  "hikaricp.connections.idle",
  "hikaricp.connections.max",
  "hikaricp.connections.min",
  "hikaricp.connections.pending",
  "hikaricp.connections.timeout",
  "hikaricp.connections.usage",
  "http.server.requests",
```



```
{
  "availableTags": [
    {
      "tag": "main.application.class",
      "values": [
        "com.app.job.JobApplication"
      ]
    }
  ],
  "baseUnit": "seconds",
  "description": "Time taken to start the application",
  "measurements": [
    {
      "statistic": "VALUE",
      "value": 4.739
    }
  ],
  "name": "application.started.time"
}

  },
  "beans": {
    "href": "http://localhost:7071/actuator/beans",
    "templated": false
  },
```

localhost:7071/actuator/beans

tty-print

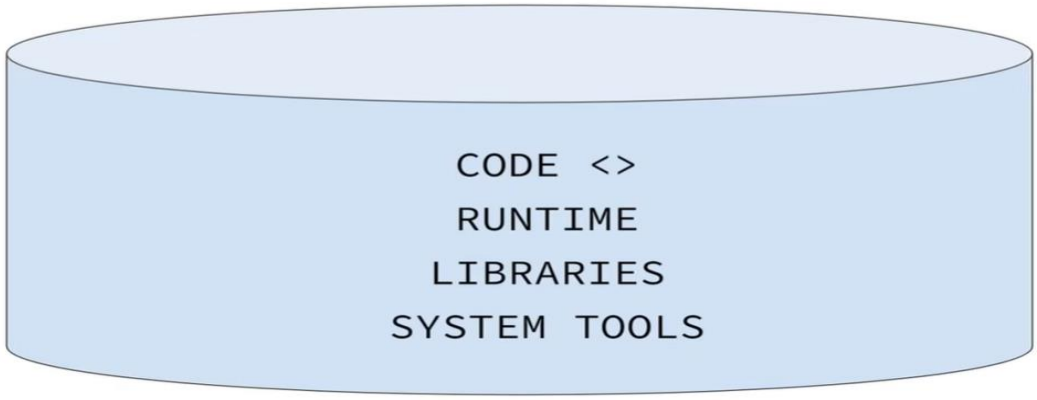

```
contexts": {
 "job": {
   "beans": {
     "org.springframework.boot.servlet.autoconfigure.actuate.web.ServletEndpointManagementContextConfiguration": {
       "aliases": [],
       "dependencies": [],
       "scope": "singleton",
       "type": "org.springframework.boot.servlet.autoconfigure.actuate.web.ServletEndpointManagementContextConfiguration"
     },
     "applicationTaskExecutor": {
       "aliases": [
         "bootstrapExecutor"
       ],
       "dependencies": [
         "org.springframework.boot.autoconfigure.task.TaskExecutorConfigurations$TaskExecutorConfiguration",
         "threadPoolTaskExecutorBuilder"
       ],
       "resource": "class path resource [org/springframework/boot/autoconfigure/task/TaskExecutorConfigurations$TaskExecutorConfiguration.class]",
       "scope": "singleton",
       "type": "org.springframework.scheduling.concurrent.ThreadPoolTaskExecutor"
     },
```


Introduction to Docker

Let's Define Docker

Docker is an open-source platform that allows you to automate the deployment, scaling, and management of applications using containerization

Docker Container



Virtual Machines

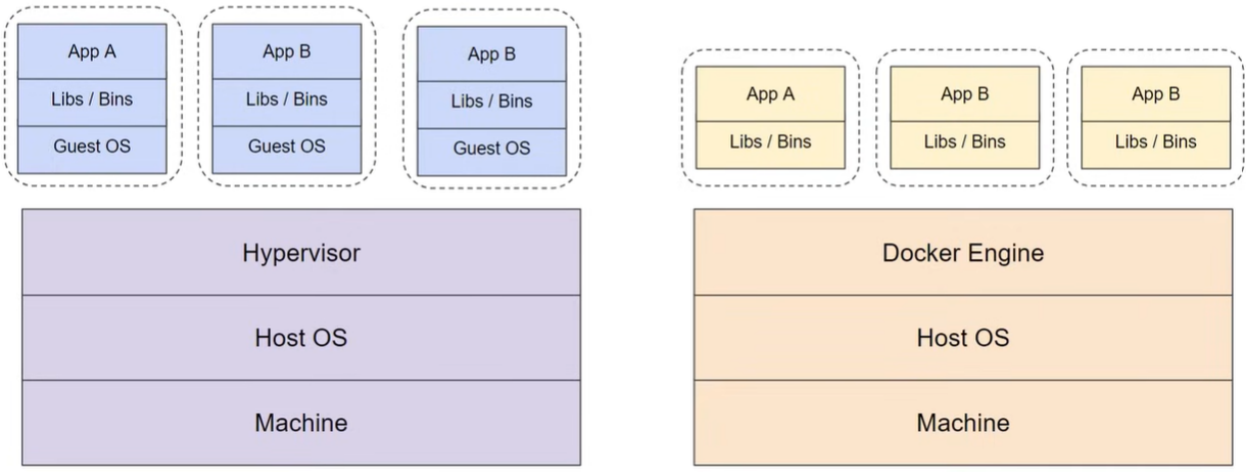
→ *VMs act like separate computers inside your computer*

→ *Each virtual machine behaves like a separate computer*

→ *Virtual machines are created and managed by virtualization software*

→ *They provide a flexible and scalable way to utilize hardware resources*

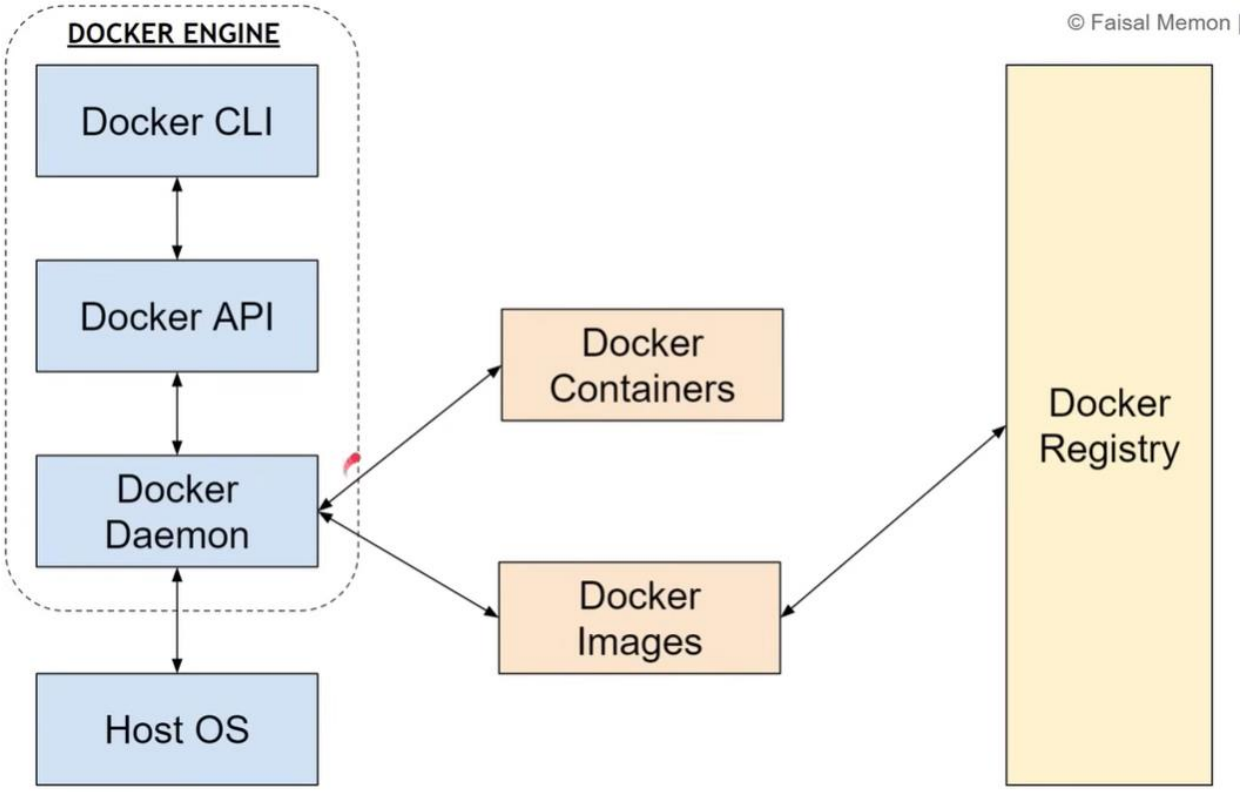
Docker over VM's



Parameters	Virtual Machines (VMs)	Docker Containers
Size	Relatively large and resource-intensive	Lightweight and resource-efficient
Startup Time	Longer boot time as full OS needs to start	Almost instant startup as no OS boot required
Resource Utilization	Utilizes more system resources (CPU, memory)	Utilizes fewer system resources
Isolation	Strong isolation between VMs	Isolated, but shares host OS kernel
Portability	Portable, but requires OS compatibility	Highly portable, independent of host OS

Parameters	Virtual Machines (VMs)	Docker Containers
Scalability	Scaling requires provisioning of new VMs	Easy to scale by creating more containers
Ecosystem	VM-specific tools and management frameworks	Docker ecosystem with extensive tooling
Development Workflow	Slower setup and provisioning process	Faster setup and dependency management
Deployment Efficiency	More overhead due to larger VM size	Efficient deployment with smaller container

Docker Architecture



Concepts in Docker

→ ***Images***: *Docker images are templates that define the container and its dependencies.*

→ ***Containers***: *Containers are runtime environments created from Docker images.*

→ ***Docker Engine***: *The Docker Engine is the runtime that runs and manages containers*

→ ***Dockerfile***: *A Dockerfile is a file that contains instructions to build a Docker image.*

→ ***Docker Hub***: *Docker Hub is a cloud-based registry that hosts a vast collection of Docker images*

Docker Registry

What is it?

Docker Registry is a storage and distribution system for named Docker images

Importance of Docker Registry

- *Centralized Resource*
- *Easy Versioning*
- *Share your Docker images*

Containerizing Our Spring Boot Application

Dockerfile

```
FROM openjdk:11
VOLUME /tmp
ADD target/my-app.jar my-app.jar
EXPOSE 8080
ENTRYPOINT ["java","-jar","/my-app.jar"]
```

How does the process work?

→ *Cloud Native Buildpacks*

→ *Spring Boot Maven Plugin*

→ *Layering*

→ *Paketo Buildpacks*

Advantages

→ *No Dockerfile Needed*

→ *Sensible Defaults*

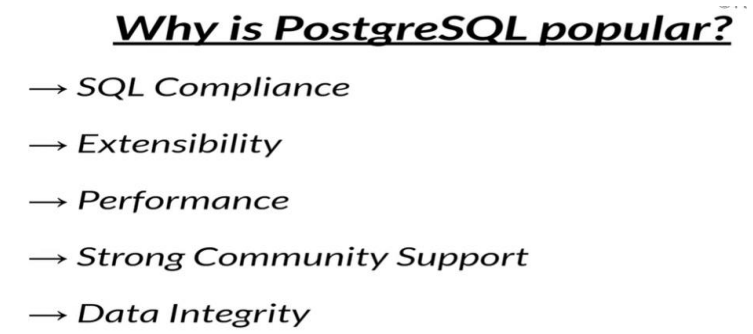
→ *Consistent Environment*

→ *Security*

→ *Layering & Efficiency*

Docker image command in spring-boot

./mvnw spring-boot:build-image "-Dspring-boot.build-image.imageName=krajesh978/<image-name>"




```
PS D:\Learning\Projects\microservices\job> ./mvnw spring-boot:build-image "-Dspring-boot.build-image.imageName=krajesh978/jobappname"
[INFO] Scanning for projects...
[INFO]
[INFO] ----------------------------< com.app:job >----------------------------
[INFO] Building job 0.0.1-SNAPSHOT
[INFO]   from pom.xml
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] >>> spring-boot:4.0.0:build-image (default-cli) > package @ job >>>
[INFO]
[INFO] --- resources:3.3.1:resources (default-resources) @ job ---
[INFO] Copying 1 resource from src\main\resources to target\classes
[INFO] Copying 0 resource from src\main\resources to target\classes
[INFO]
[INFO] --- compiler:3.14.1:compile (default-compile) @ job ---
[INFO] Recompiling the module because of added or removed source files.
[INFO] Compiling 16 source files with javac [debug parameters release 21] to target\classes
[INFO]
```

Push Docker image to docker registry

```
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  02:06 min
[INFO] Finished at: 2025-12-15T12:36:47-05:00
[INFO] ------------------------------------------------------------------------
PS D:\Learning\Projects\microservices\job> docker login
Authenticating with existing credentials... [Username: krajesh978]

i Info → To login with a different account, run 'docker logout' followed by 'docker login'


Login Succeeded
PS D:\Learning\Projects\microservices\job> docker push krajesh978/jobappname
Using default tag: latest
The push refers to repository [docker.io/krajesh978/jobappname]
181b18e8a948: Waiting
995d0e308d84: Waiting
39732bc75041: Waiting
a45214c3647e: Waiting
a45214c3647e: Pushed
47edbc81cc29: Pushed
c8c3983f781e: Mounted from paketobuildpacks/ubuntu-noble-run-tiny
054ae2035a47: Pushed
f1ac0198231e: Mounted from paketobuildpacks/ubuntu-noble-run-tiny
a4e9ceebc8f3: Pushed
```

Docker Commands

```
→ docker pull <image>

→ docker push <username/image>

→ docker run -it -d -p <host-port>:<container-port>
--name <name> <image>

→ docker stop <container-id/container-name>

→ docker start <container-id/container-name>

→ docker rm <container-id/container-name>

→ docker rmi <image-id/image-name>

→ docker ps

→ docker ps -a

→ docker images

→ docker exec -it <container-name/container-id> bash

→ docker build -t <username/image> .

→ docker logs <container-name/container-id>

→ docker inspect <container-name/container-id>
```

Running application in docker

```
PS D:\Learning\Projects\microservices\job> docker run -p 8080:7071 krajesh978/jobappname
Calculating JVM memory based on 7236796K available memory
For more information on this calculation, see https://paketo.io/docs/reference/java-reference/#memory-calculator
Calculated JVM Memory Configuration: -XX:MaxDirectMemorySize=10M -Xmx6602776K -XX:MaxMetaspaceSize=122019K -XX:Res
250, Loaded Class Count: 19129, Headroom: 0%)
Enabling Java Native Memory Tracking
Adding 146 container CA certificates to JVM truststore
Spring Cloud Bindings Enabled
Picked up JAVA_TOOL_OPTIONS: -Djava.security.properties=/layers/paketo-buildpacks_bellsoft-liberica/java-security-
xDirectMemorySize=10M -Xmx6602776K -XX:MaxMetaspaceSize=122019K -XX:ReservedCodeCacheSize=240M -Xss1M -XX:+UnlockD
stics -Dorg.springframework.cloud.bindings.boot.enable=true

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
```

PostgreSQL Driver SQL

A JDBC and R2DBC driver that allows Java programs to connect to a PostgreSQL database using standard, database independent Java code.

Set database configurations in application properties

```
#Postgresql
spring.datasource.url=jdbc:postgresql://localhost:5432/jobapp
spring.datasource.username=postgres
spring.datasource.password=BennSys
spring.jpa.database=POSTGRESQL
spring.jpa.database-platform=org.hibernate.dialect.PostgreSQLDialect
```

Run Postgres docker image

```
See 'docker run --help' for more information
PS D:\Learning\Projects\microservices\job> docker run -d --name postgres_db -e POSTGRES_PASSWORD=postgres postgres
Unable to find image 'postgres:latest' locally
latest: Pulling from library/postgres
6dd6fcdc9495: Pull complete
c3ff19dd627c: Pull complete
1733a4cd5954: Pull complete
```

Run PGADMIN on docker

```
Run 'docker run --help' for more information
PS D:\Learning\Projects\microservices\job> docker run -d --name pgadmin -e PGADMIN_DEFAULT_EMAIL=admin@admin.com -e PGADMIN_DEFAULT_PASSWORD=postgres dpage/pgadmin4
Unable to find image 'dpage/pgadmin4:latest' locally
latest: Pulling from dpage/pgadmin4
014e56e61396: Pull complete
6a37ad29194b: Pull complete
2e0b10cc2685: Pull complete
eb13c0937646: Pull complete
72fb190ed6ea: Pull complete
94937133806c: Pull complete
```

Docker network helps to communicate one container to another container

```
docker run -d --name db -e POSTGRES_PASSWORD=mysecretpassword postgres

docker run -d --name pgadmin -e PGADMIN_DEFAULT_EMAIL=user@domain.com -e
PGADMIN_DEFAULT_PASSWORD=SuperSecret dpage/pgadmin4

docker exec -it pgadmin ping db

docker rm -f db pgadmin

USING NETWORKS:
docker run -d --name db --network my-network -e POSTGRES_PASSWORD=mysecretpassword
postgres

docker run -d --name pgadmin --network my-network -e
PGADMIN_DEFAULT_EMAIL=user@domain.com -e PGADMIN_DEFAULT_PASSWORD=SuperSecret
dpage/pgadmin4

docker exec -it pgadmin ping db
```

🐳 Docker Compose — Definition

Docker Compose is a tool used to **define, configure, and run multi-container Docker applications** using a single YAML file (`docker-compose.yml`).

In simple words:
👉 *Docker Compose lets you start multiple Docker containers together with one command.*

📌 One-line definition (interview-ready)

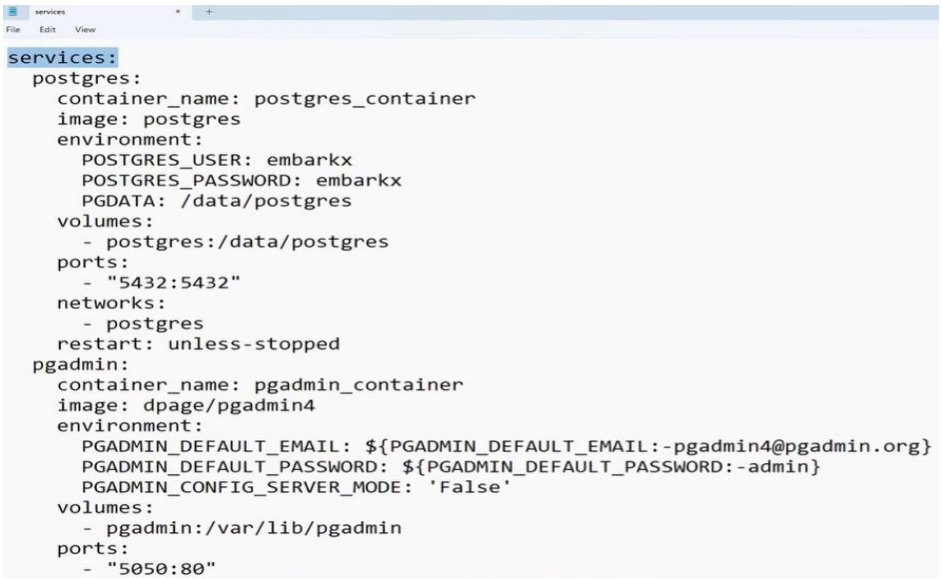
> **Docker Compose is a container orchestration tool that uses a YAML file to define and manage multiple Docker containers as a single application.**

🧩 Why Docker Compose is used

- Run **multiple services together** (e.g., Spring Boot + DB + Kafka)
- Define **networking, volumes, and environment variables**
- Start and stop everything with **one command**
- Ideal for **local development, testing, and CI/CD**

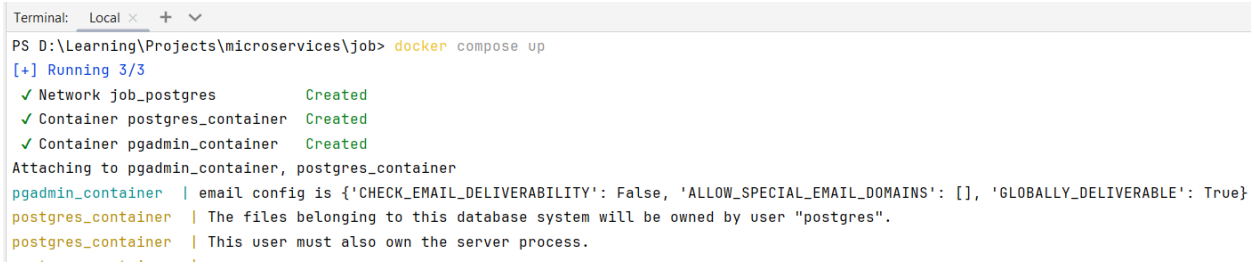
```
Start the PostgreSQL service:
docker run -d \
  --name postgres_container \
  -e POSTGRES_USER=embarkx \
  -e POSTGRES_PASSWORD=embarkx \
  -e PGDATA=/data/postgres \
  -v postgres:/data/postgres \
  -p 5432:5432 \
  --network postgres \
  --restart unless-stopped \
  postgres

Start the pgAdmin service:
docker run -d \
  --name pgadmin_container \
  -e PGADMIN_DEFAULT_EMAIL=pgadmin4@pgadmin.org \
  -e PGADMIN_DEFAULT_PASSWORD=admin \
  -e PGADMIN_CONFIG_SERVER_MODE=False \
  -v pgadmin:/var/lib/pgadmin \
  -p 5050:80 \
  --network postgres \
  --restart unless-stopped \
  dpage/pgadmin4
```

Create a docker compose file format



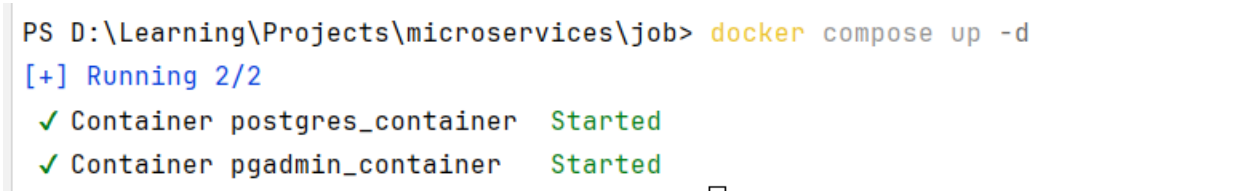
```yaml
services:
  postgres:
    container_name: postgres_container
    image: postgres
    environment:
      POSTGRES_USER: embarkx
      POSTGRES_PASSWORD: embarkx
      PGDATA: /data/postgres
    volumes:
      - postgres:/data/postgres
    ports:
      - "5432:5432"
    networks:
      - postgres
    restart: unless-stopped
  pgadmin:
    container_name: pgadmin_container
    image: dpage/pgadmin4
    environment:
      PGADMIN_DEFAULT_EMAIL: ${PGADMIN_DEFAULT_EMAIL:-pgadmin4@pgadmin.org}
      PGADMIN_DEFAULT_PASSWORD: ${PGADMIN_DEFAULT_PASSWORD:-admin}
      PGADMIN_CONFIG_SERVER_MODE: 'False'
    volumes:
      - pgadmin:/var/lib/pgadmin
    ports:
      - "5050:80"
```

To run docker compose command



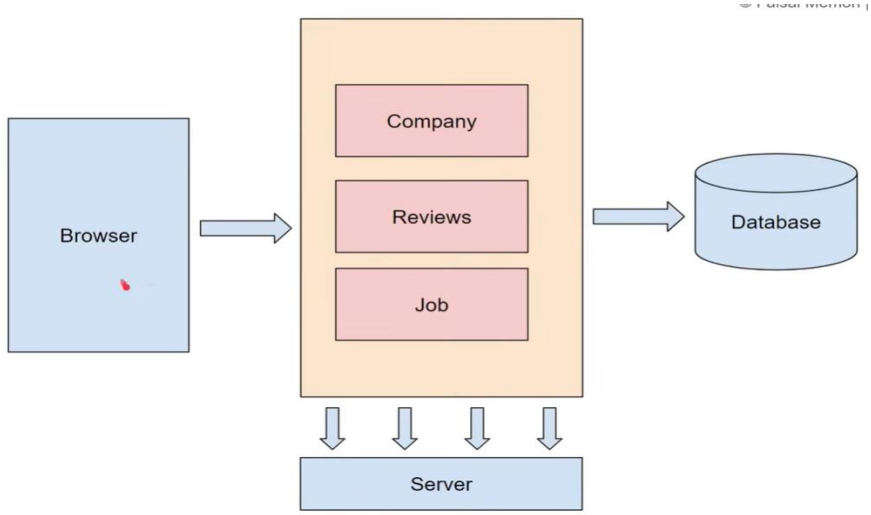
```
Terminal:  Local
PS D:\Learning\Projects\microservices\job> docker compose up
[+] Running 3/3
 ✓ Network job_postgres          Created
 ✓ Container postgres_container  Created
 ✓ Container pgadmin_container   Created
Attaching to pgadmin_container, postgres_container
pgadmin_container  | email config is {'CHECK_EMAIL_DELIVERABILITY': False, 'ALLOW_SPECIAL_EMAIL_DOMAINS': [], 'GLOBALLY_DELIVERABLE': True}
postgres_container  | The files belonging to this database system will be owned by user "postgres".
postgres_container  | This user must also own the server process.
```

In detach mode



```
PS D:\Learning\Projects\microservices\job> docker compose up -d
[+] Running 2/2
 ✓ Container postgres_container  Started
 ✓ Container pgadmin_container   Started
```

Monolithic architecture is a design where all the components of an application are **interconnected** and **interdependent**



Problems

→ *Difficult to Implement Changes*

→ *Lack of Scalability*

→ *Long-term Commitment to a Single Technology Stack*

→ *Application Complexity and Its Effect on Development and Deployment*

→ *Slowing Down of IDEs*

→ *Increased Application Start Time*

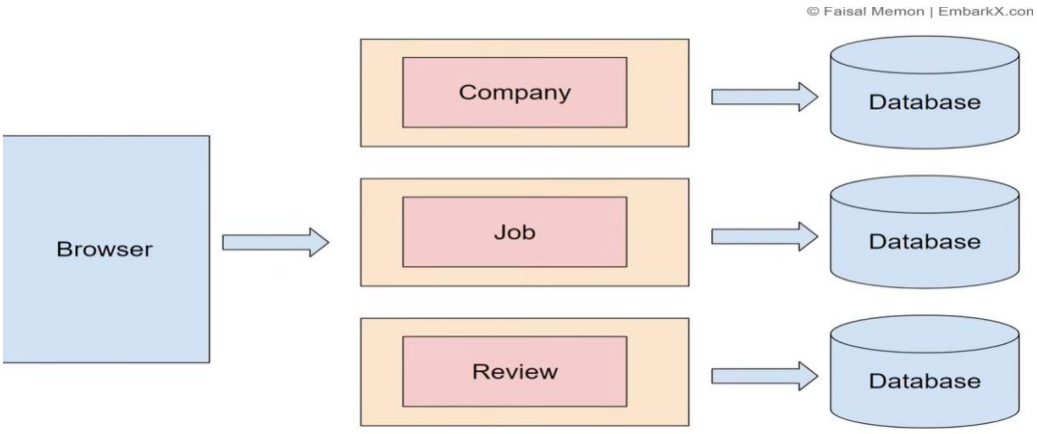
→ *Large Project Size*

→ *Deploying for Small Changes*

→ *Team Collaboration and Autonomy*

What are Microservices and Why do we need them?

Microservices structures an application as a collection of small autonomous services



Principles of Microservices©

→ *Single Responsibility*

→ *Independence*

→ *Decentralization*

→ *Failure Isolation*

→ *Continuous Delivery/Deployment*

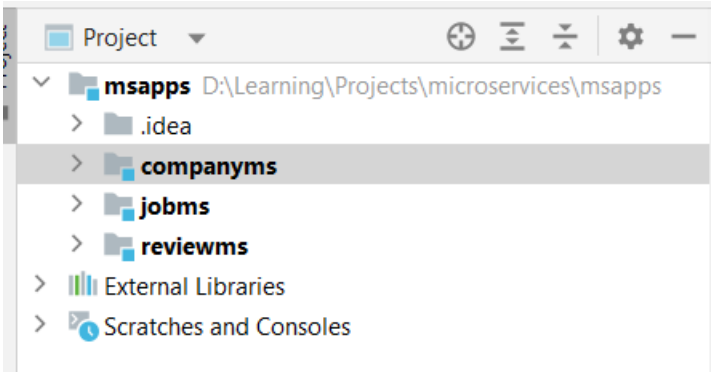
Overcoming Monolithic Architecture Challenges with Microservices

→ *Scalability*

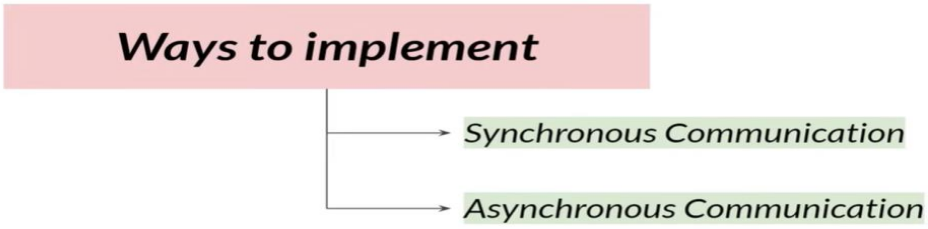
→ *Flexibility*

→ *Simplicity*

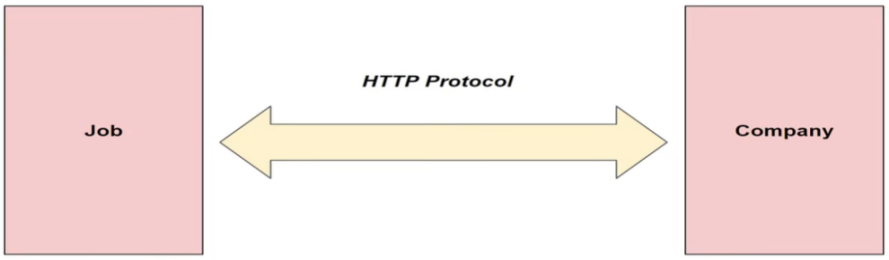
Created three different micro-services application

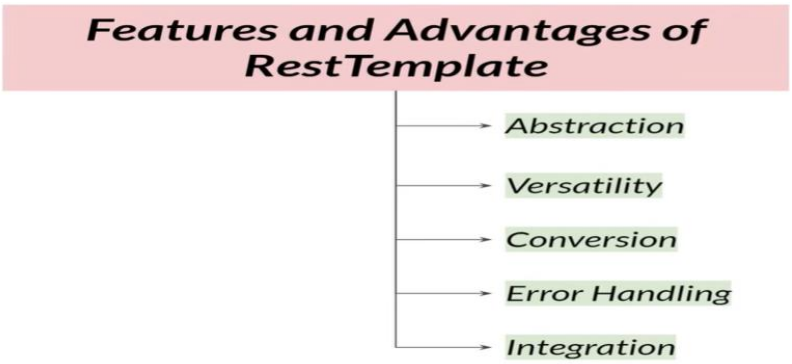


Introduction to InterService Communication



What Is REST Template and Why Do You Need It?

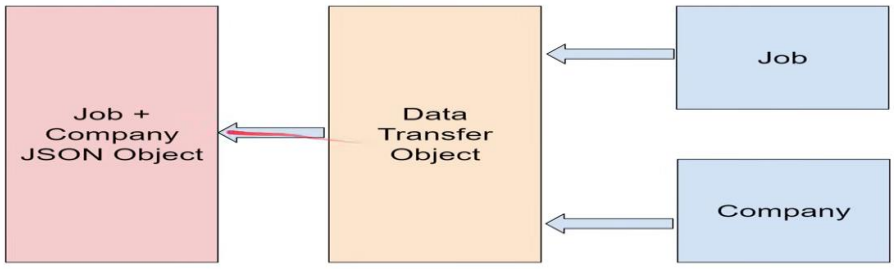




DTO Pattern

What is it?

Design pattern used to transfer data between software application subsystems



Before company data mapping in job we have companyId

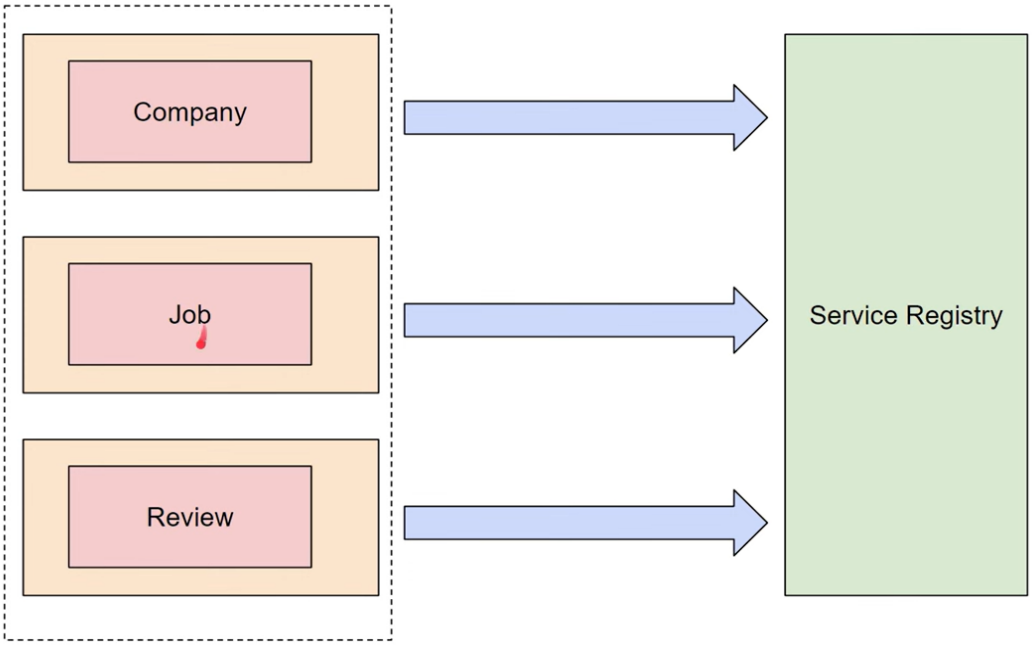
Jobs mapping with company Id with help of convertToDto

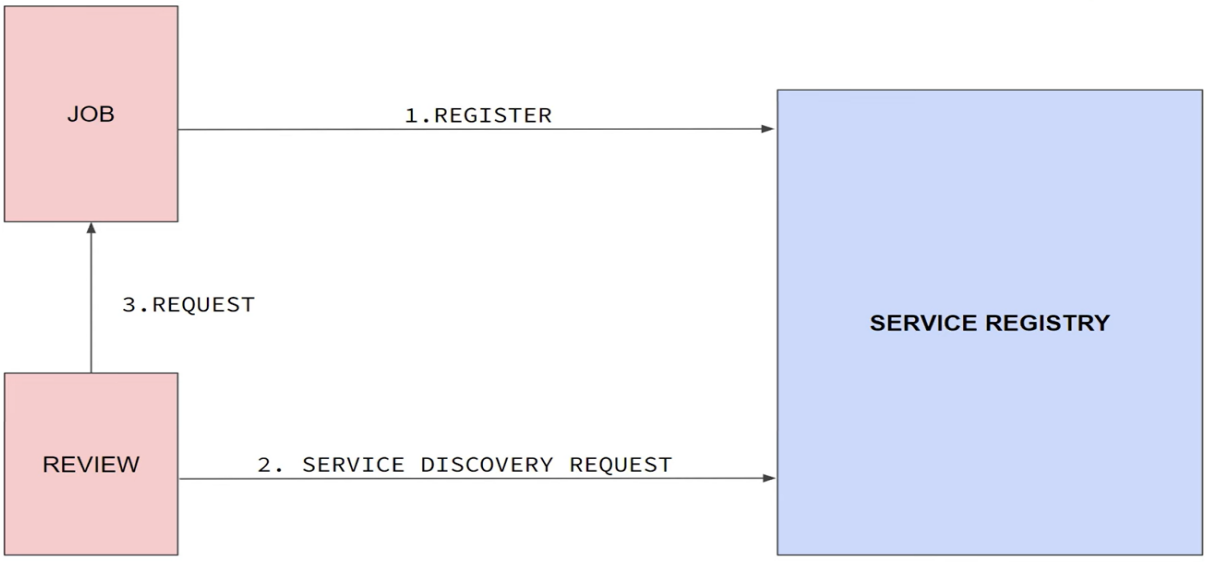
```
1 usage
private JobWithCompanyDTO convertToDto(Job job){
    RestTemplate restTemplate = new RestTemplate();
    JobWithCompanyDTO jobWithCompanyDTO = new JobWithCompanyDTO();
    jobWithCompanyDTO.setJob(job);
    Company company = restTemplate.getForObject(
            url: "http://localhost:8082/companies/" + job.getCompanyId(),
                Company.class);
    jobWithCompanyDTO.setCompany(company);
    return jobWithCompanyDTO;
}
```

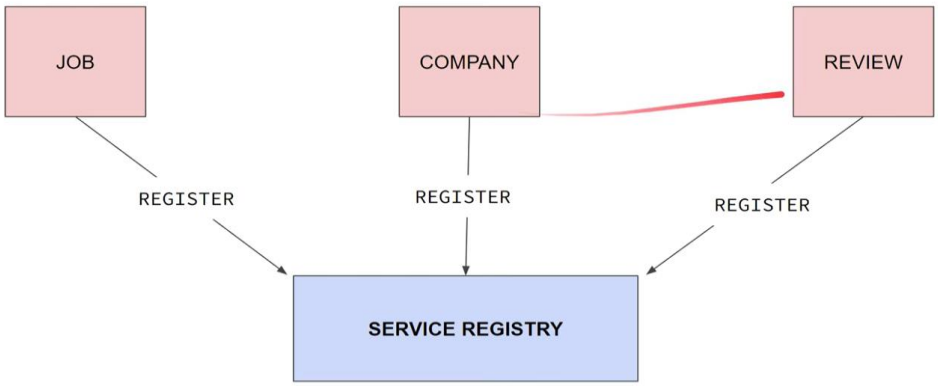
Spring Cloud Eureka

What is it?

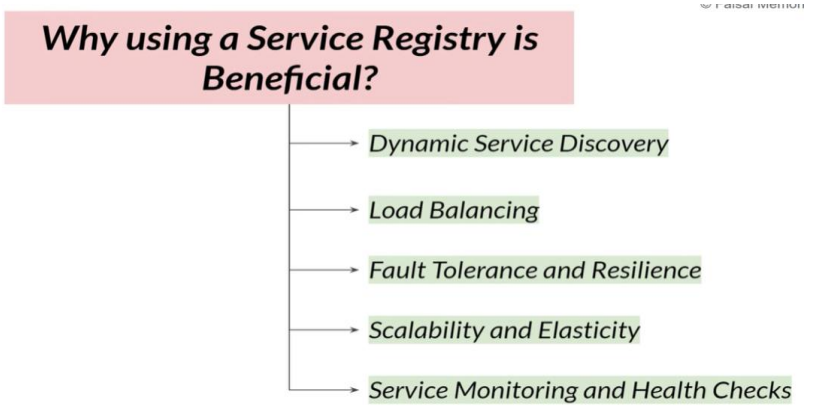
A service registry is used in microservices architectures to enable dynamic service discovery

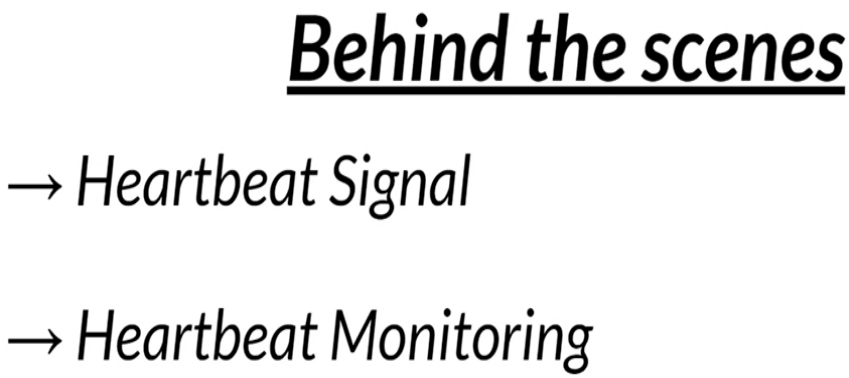

Company
Job
Review
Service Registry


JOB
1.REGISTER
3.REQUEST
SERVICE REGISTRY
REVIEW
2. SERVICE DISCOVERY REQUEST

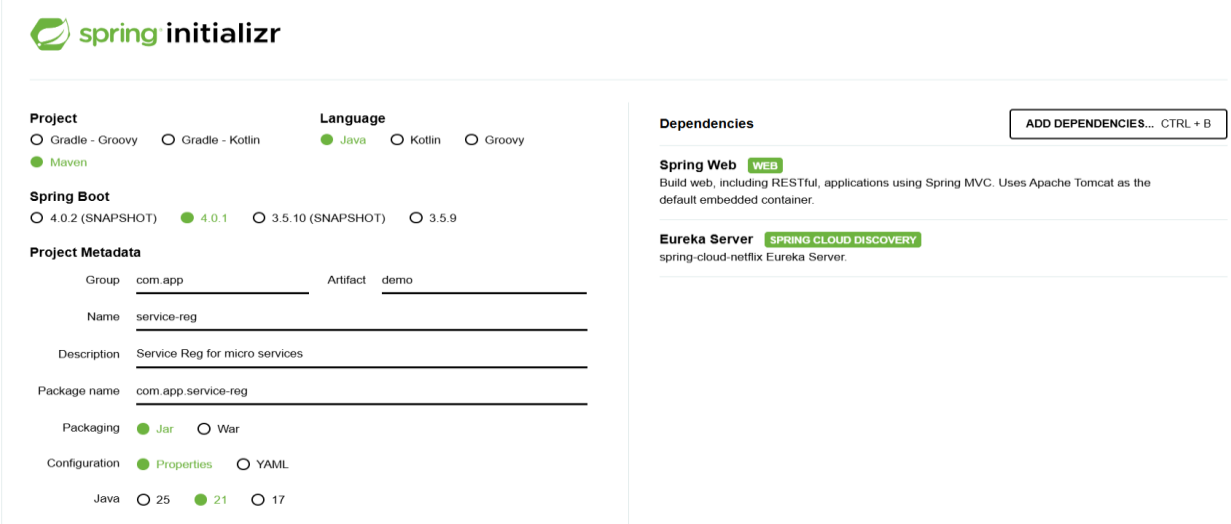

JOB
COMPANY
REVIEW
REGISTER
REGISTER
REGISTER
SERVICE REGISTRY

Service Registry





Register microservices application to eureka server for that create new server-reg project



After initializing the project add annotation to enable eureka server @EnableEurekaServer

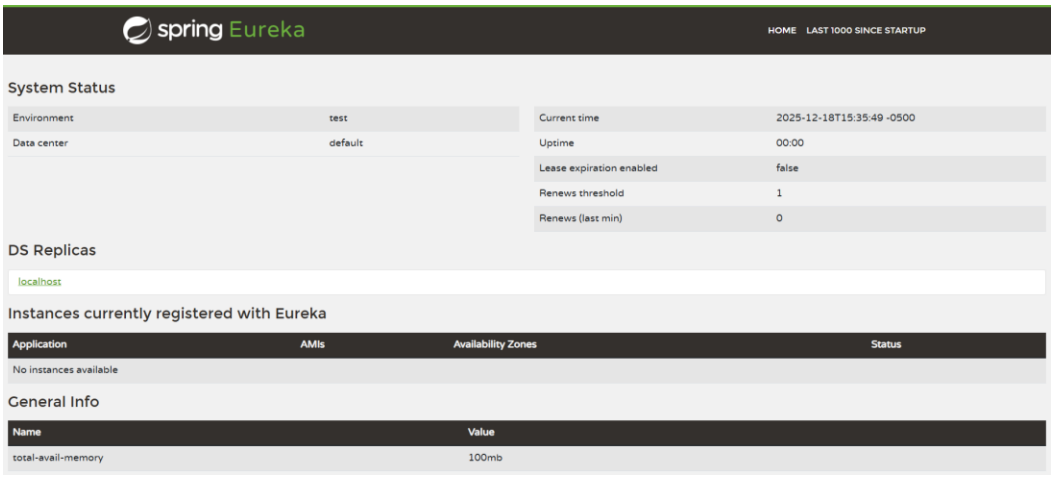
```java
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.netflix.eureka.server.EnableEurekaServer;

@SpringBootApplication
@EnableEurekaServer
public class ServiceRegApplication {

    public static void main(String[] args) { SpringApplication.run(ServiceRegApplication.class, args); }

}
```

Add properties for eureka server registry

application.properties

```
spring.application.name=service-registry
server.port:8761

eureka.client.register-with-eureka=false
eureka.client.fetch-registry=false


```

After running the application, we will get eureka server page



To register the microservice application in eureka server use this dependency in microservices applications

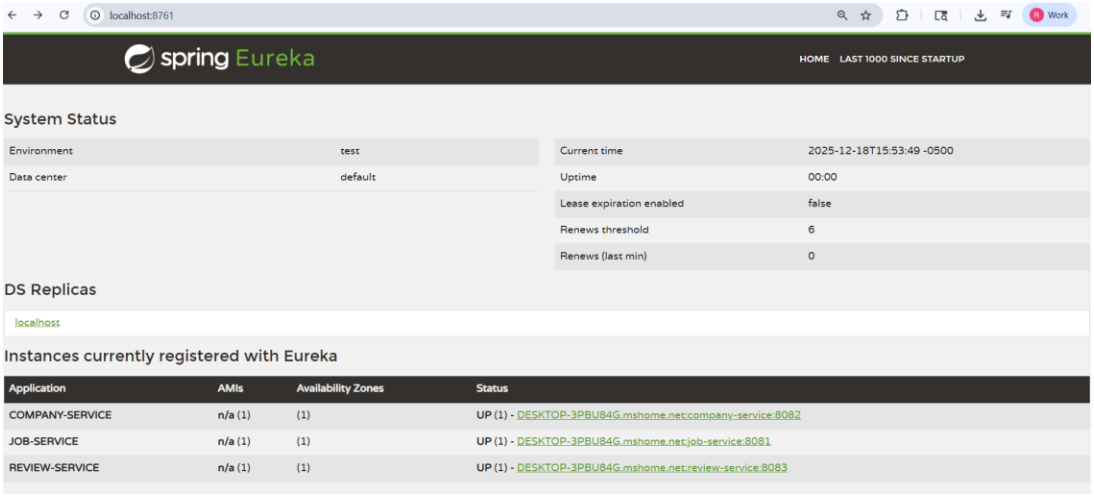
Eureka Discovery Client SPRING CLOUD DISCOVERY

A REST based service for locating services for the purpose of load balancing and failover of middle-tier servers.

To register the ms application using this application properties

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

All services registered in eureka server



Instead of hard coding Ip addresses in application we can use service to communicate one microservice to another microservice using eureka

```
1 usage   rajesh kakumani *
private JobWithCompanyDTO convertToDto(Job job){
    JobWithCompanyDTO jobWithCompanyDTO = new JobWithCompanyDTO();
    jobWithCompanyDTO.setJob(job);
    Company company = restTemplate.getForObject(
            url: "http://COMPANY-SERVICE:8082/companies/" + job.getCompanyId(),
                Company.class);
    jobWithCompanyDTO.setCompany(company);
    return jobWithCompanyDTO;
}
```

To change the JSON structure using mapper class

```
{
    "company": {
        "id": 1,
        "name": "Indo Bennsys",
        "description": "Software Company"
    },
    "description": "Test",
    "id": 1,
    "location": "United States",
    "maxSalary": "200000",
    "minSalary": "150000",
    "review": [
        {
            "description": "Good company",
            "id": 1,
            "rating": 5.5,
            "title": "Software Engineer"
        },
        {
            "description": "Good company",
            "id": 2,
            "rating": 5.5,
            "title": "Software Engineer"
        }
    ],
```


```
2 usages
public class JobMapper {

    1 usage
    public static JobDTO mapToJobWithCompanyDto(Job job, Company company, List<Review> reviewResponse){
        JobDTO jobDTO = new JobDTO();
        jobDTO.setId(job.getId());
        jobDTO.setTitle(job.getTitle());
        jobDTO.setDescription(job.getDescription());
        jobDTO.setLocation(job.getLocation());
        jobDTO.setMinSalary(job.getMinSalary());
        jobDTO.setMaxSalary(job.getMaxSalary());
        jobDTO.setCompany(company);
        jobDTO.setReview(reviewResponse);
        return jobDTO;
    }

}
```

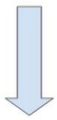
```
private JobDTO convertToDto(Job job){
    Company company = restTemplate.getForObject(
            url: "http://COMPANY-SERVICE:8082/companies/" + job.getCompanyId(),
            Company.class);
    ResponseEntity<List<Review>> reviewResponse= restTemplate.exchange( url: "http://REVIEW-SERVICE:8083/reviews?companyId=" + job.getCompanyId(),
            HttpMethod.GET, requestEntity: null,
            new *
            new ParameterizedTypeReference<List<Review>>() {
            });
    List<Review> reviews = reviewResponse.getBody();
    JobDTO jobDTO = JobMapper.mapToJobWithCompanyDto(job, company, reviews);
    jobDTO.setCompany(company);
    jobDTO.setReview(reviews);
    return jobDTO;
}
```

Open Feign: An Introduction

What is it?

Feign is a declarative web service client designed to make writing HTTP clients easier

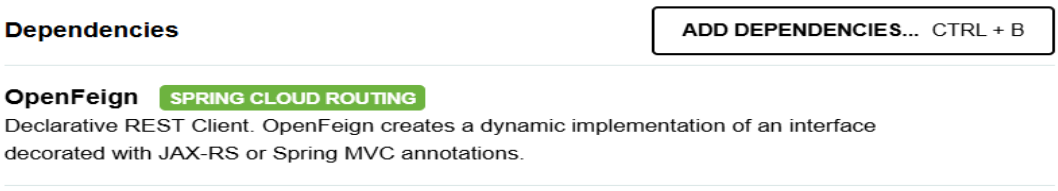
```
Company company =
restTemplate.getForObject("http://COMPANY-SERVICE/companies/" +
job.getCompanyId(), Company.class);
```



```
Company company = companyClient.getCompany(job.getCompanyId());
```

Why use OpenFeign

- Ease of use
- Integration with Eureka
- Built-in Load Balancing with Ribbon
- Support for FallBacks and Circuit Breakers



Enable feign clients in controller

```
@EnableFeignClients
@RestController
@RequestMapping("/job")
public class JobController {

    6 usages
    private JobService jobService;
```

Introduction to Distributed Tracing

How would you track a request from start to end?

Distributed Tracing *enables you to trace your request from start to end*

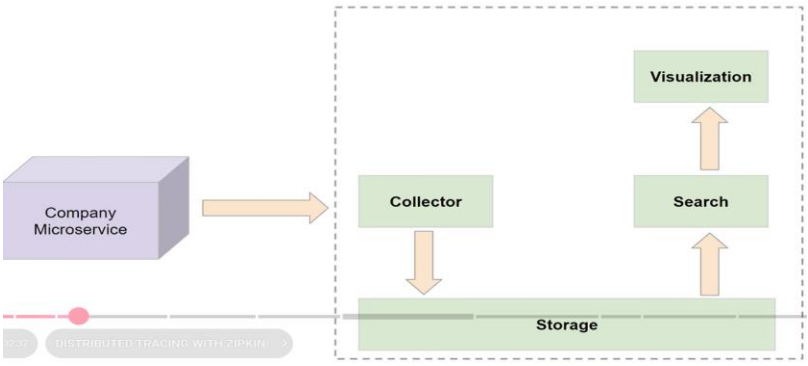
Problems that Distributed Tracing Solves

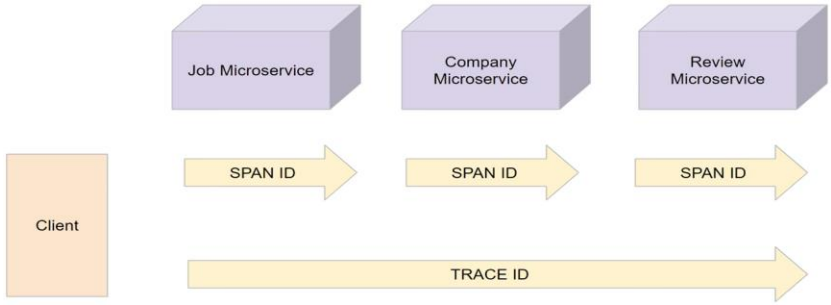
- *Request Visualization*
- *Identify Performance Bottleneck*
- *Error Analysis and Debugging*
- *Tracking Dependency*
- *Performance Optimization*

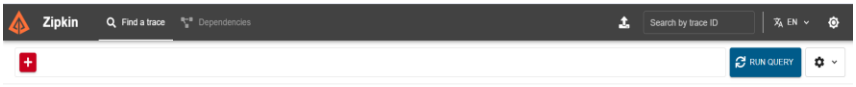
Introduction to Zipkin

What is Zipkin?

Zipkin is an open-source distributed tracing system







Introduction to Micrometer

What is Micrometer?

Micrometer provides insights that help you keep tabs on your application's performance

How Micrometer Helps

→ Helps you collect metrics from your application

→ Acts as a middleman or a bridge between your application and the metrics collection systems

→ It offers a vendor-neutral interface

→ You can abstract away the complexities of interacting with different metrics collection systems

→ Micrometer simplifies the process of collecting metrics from your application

Add zipkin dependency in spring boot projects

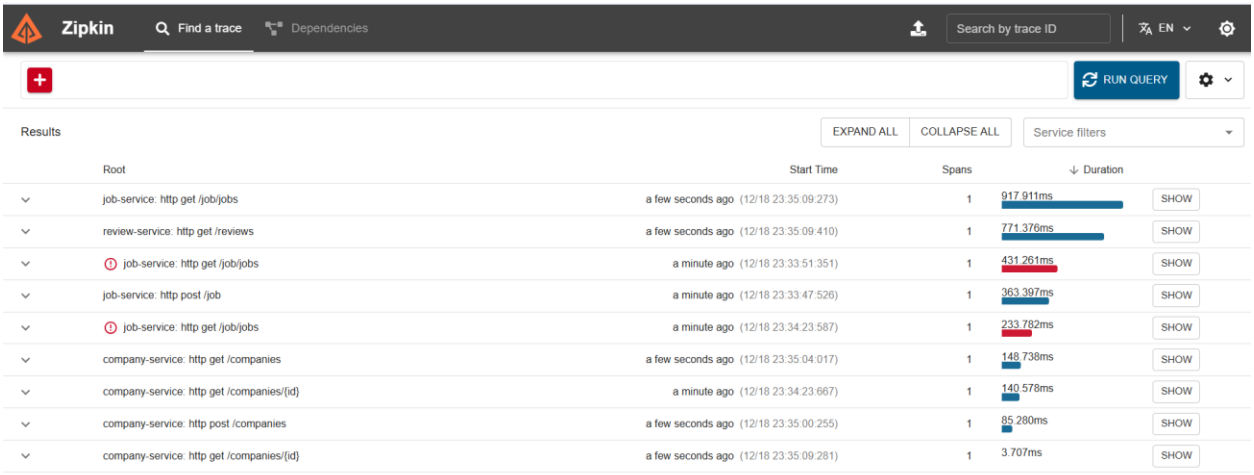
Zipkin OBSERVABILITY

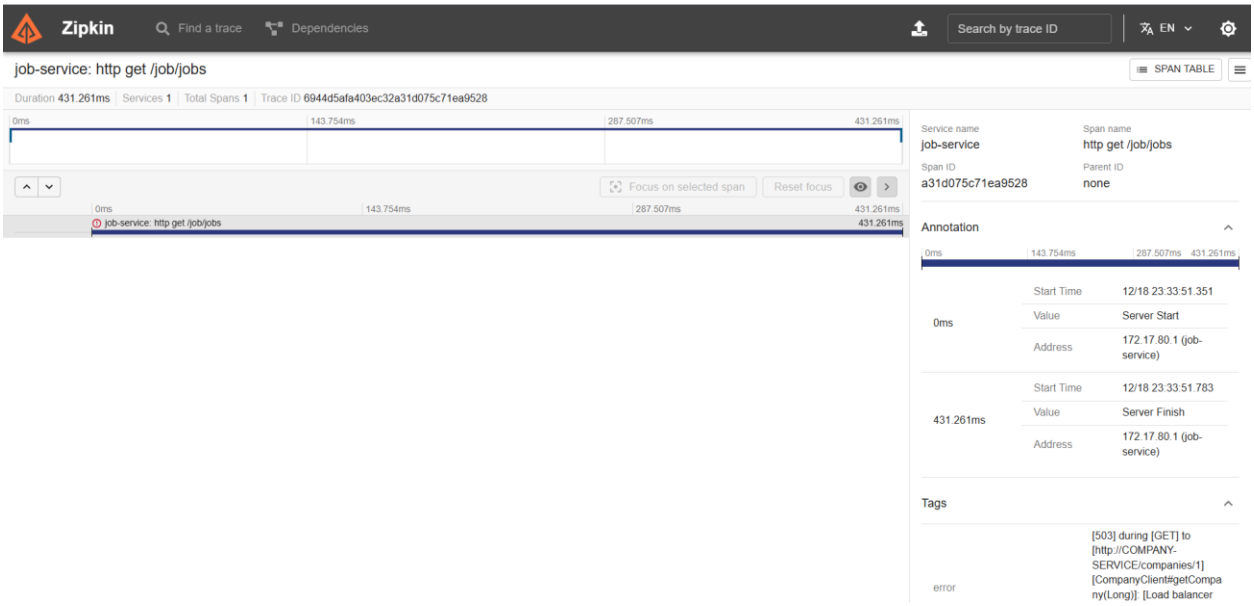
Enable and expose span and trace IDs to Zipkin.

Set zip properties in application.properties file

```
#Zipkin
management.tracing.sampling.probability=1.0
```

After sending the requests we can find the spanId and traceid in zipkin server





Introduction to Configuration Management

Configuration Management

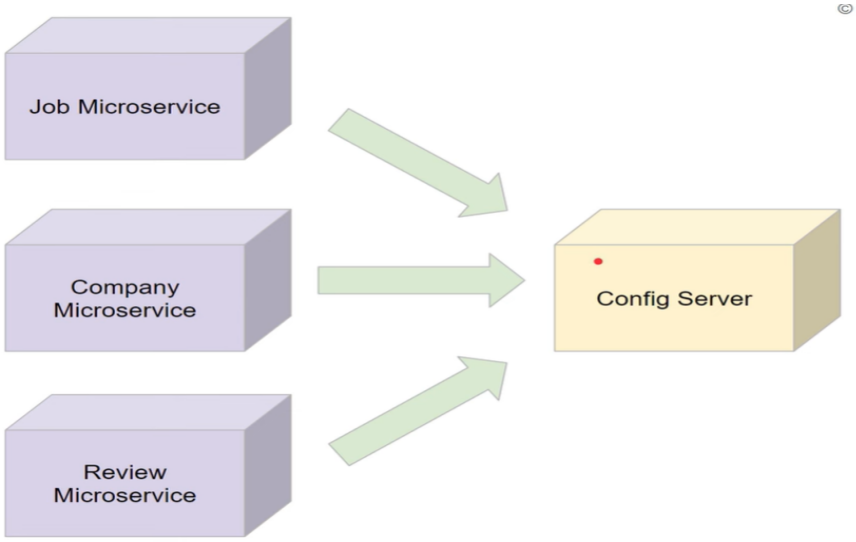
→ *Simply means managing and controlling the configurations of each microservice in the system*

→ *Configuration may include details such as database connections, external service URLs, caching settings, and more*

→ ***Challenge**: As the number of Microservices increases in your architecture, managing the individual configurations can become a complex task.*

→ *A centralized Config Server provides a central place for managing configurations across all microservices*

→ *It simplifies configuration management and increases operational efficiency.*



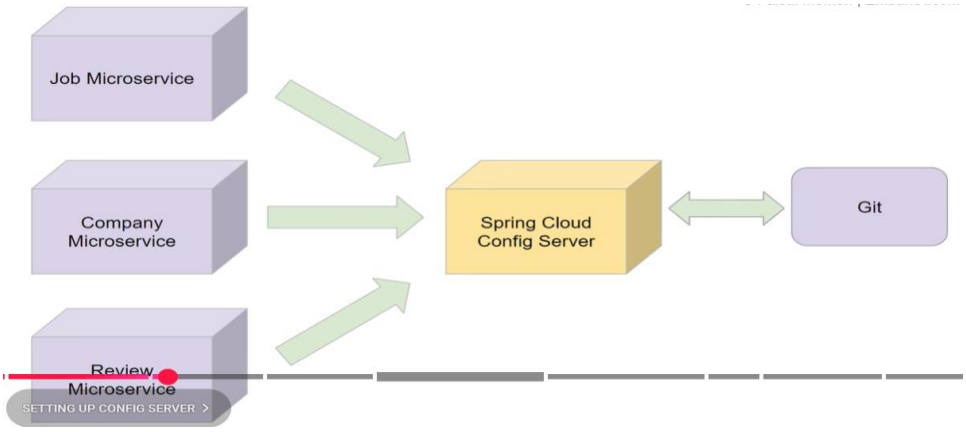
Features of a Config Server

- *Centralized and Versioned Configuration*
- *Dynamic Updates*
- *Security*
- *Application and Profile Specific Configuration*

Benefits of a Config Server

- Single source of truth
- Easier to manage and update configurations
- Enhances security and control
- Easy to deploy and scale microservices

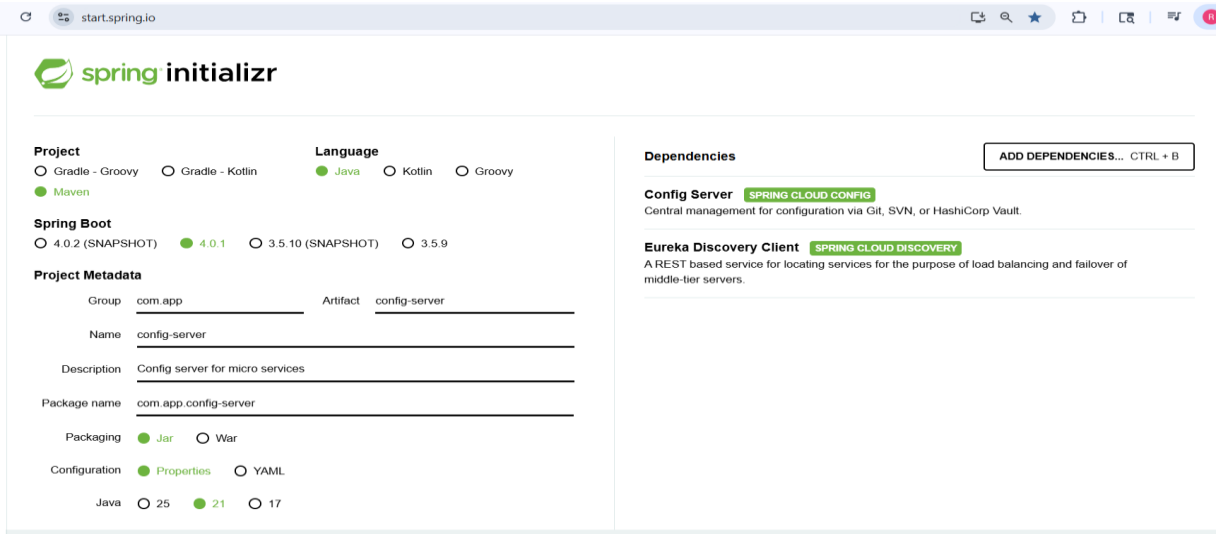
Spring Cloud Config Server is part of the Spring Cloud project, a suite of tools specifically designed for building and managing cloud-native applications.



Spring Cloud Config Server

- Storing configurations
- Serving configurations
- Refreshing configurations
- Easy integration with Spring Boot
- Support for different environments
- Encryption and Decryption

Create a project for config server with necessary dependencies



Define annotation for config server

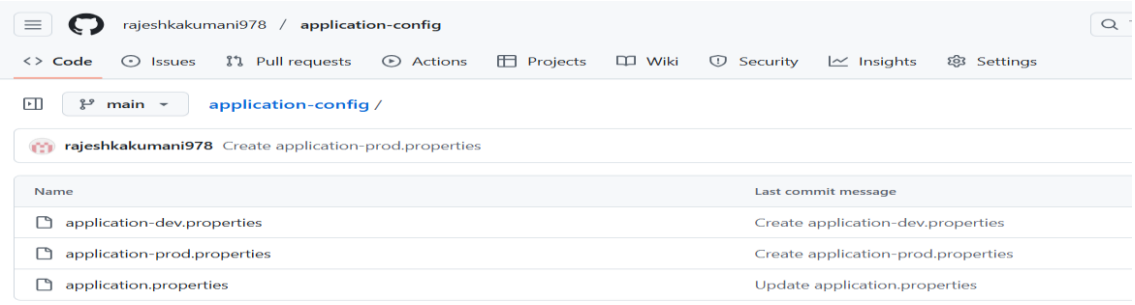
```java
@SpringBootApplication
@EnableConfigServer
public class ConfigServerApplication {

    public static void main(String[] args) { SpringApplication.run(ConfigServerApplication.class, args); }

}
```

Set application properties

config-server\...\application.properties | ConfigServerApplication.java

```
spring.application.name=config-server
server.port=8084

#Eureka
eureka.client.service.defaultZone=http://localhost:8761/eureka/
```

Create a git repository for config server



Configure git repository in config server

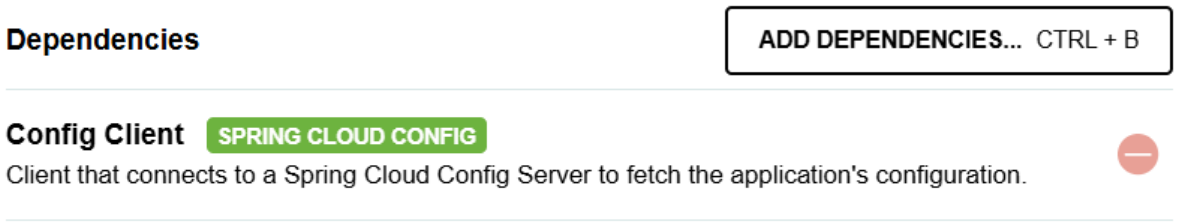
application.properties | ConfigServerApplication.java

```
spring.application.name=config-server
server.port=8084

#Eureka
eureka.client.service.defaultZone=http://localhost:8761/eureka/
spring.cloud.config.server.git.uri=https://github.com/rajeshkakumani978/application-config
```

Whenever we commit the changes config server automatically refresh the changes.


```
{
  "name": "application",
  "profiles": [
    "prod"
  ],
  "label": null,
  "version": "1f0a2d85d39bc2cc285a578b4e53897235626cef",
  "state": "",
  "propertySources": [
    {
      "name": "https://github.com/rajeshkakumani978/application-config/application-prod.properties",
      "source": {
        "spring.application.name": "job-service-prod"
      }
    },
    {
      "name": "https://github.com/rajeshkakumani978/application-config/application.properties",
      "source": {
        "spring.application.name": "job-service-demo"
      }
    }
  ]
}
```

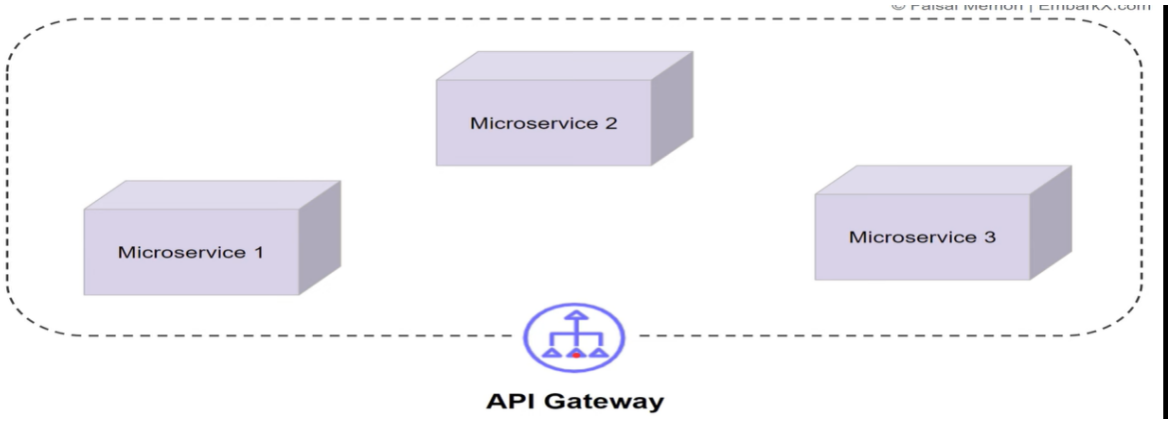

Add config client dependency to all microservices application to fetch the configuration from config server



To fetch the configuration from config server, use these properties

```
#Config server
spring.config.import=optional:configserver:http://localhost:8084
spring.profiles.active=dev
```

Introduction to API Gateways



Advantages

→ *It encapsulates the internal system architecture*

→ *Handle cross-cutting concerns like security, load balancing, rate limiting, and analytics*

→ *Can authenticate incoming requests and pass only valid requests to the services*

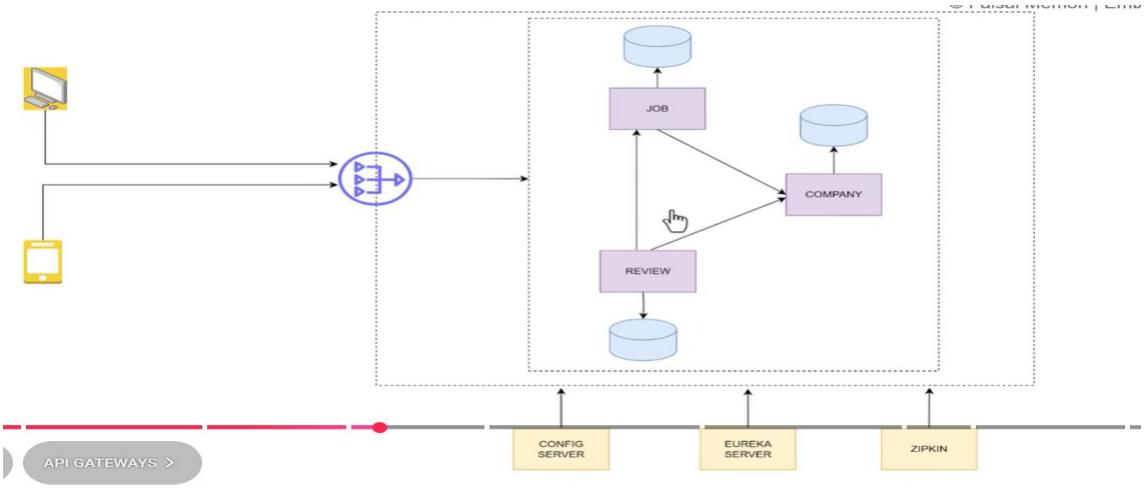
→ *Can aggregate responses from different microservices*

→ *Plays a crucial role in simplifying client interactions and managing cross-cutting concerns*

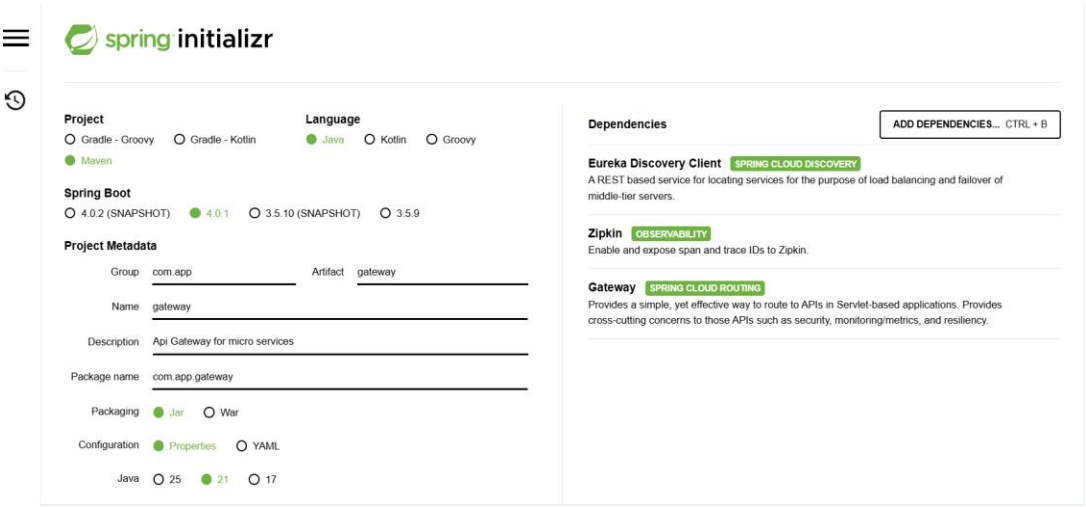
API Gateway Functions

Capabilities of API Gateway

- Request Routing
- Load Balancing
- Authentication and Authorization
- Rate Limiting
- Request and Response Transformation
- Aggregation of Data from Multiple Services



Create a project for Api gateway with following dependencies



To configure route url's in gateway

```
#Gateway
spring.cloud.gateway.discovery.locator.enabled: true
spring.cloud.gateway.discovery.locator.lower-case-service-id: true
spring.cloud.gateway.routes[0].id=company-service
spring.cloud.gateway.routes[0].uri=lb://COMPANY-SERVICE
spring.cloud.gateway.routes[0].predicates[0]=Path=/companies/**

spring.cloud.gateway.routes[1].id=job-service
spring.cloud.gateway.routes[1].uri=lg://JOB-SERVICE
spring.cloud.gateway.routes[1].predicates[0]=Path=/job/**

spring.cloud.gateway.routes[2].id=review-service
spring.cloud.gateway.routes[2].uri=lb://REVIEW-SERVICE
spring.cloud.gateway.routes[2].predicates[0]=Path=/reviews/**
```

Introduction to Fault Tolerance

Fault Tolerance

Ability to continue operating without interruption

Need for Fault Tolerance

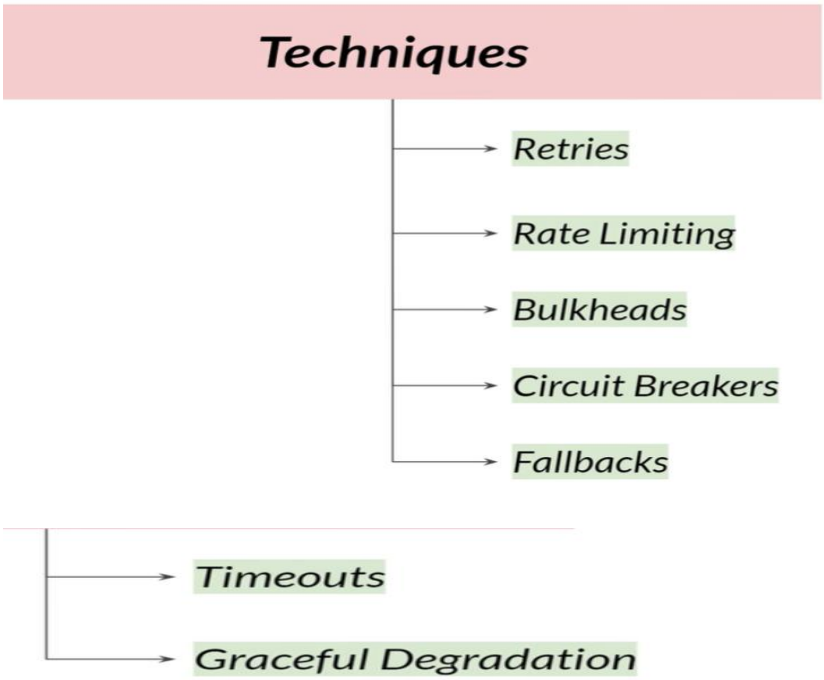
- *Fault Isolation*
- *Network Latency*
- *Deployment issues*
- *Increased Complexity*
- *Elasticity*

- *Tolerating External Failures*

Introduction to Resilience4J

Resilience

Ability or capacity to recover quickly from difficulties



***Resilience4J** is a lightweight, easy-to-use fault tolerance library*

Retry Module

→ *It's not uncommon for a network call or a method invocation to fail temporarily*

→ *We might want to retry the operation a few times before giving up*

→ *Retry module enables to easily implement retry logic in our applications*

RateLimiter

→ *We might have a service which can handle only a certain number of requests in given time*

→ *RateLimiter module allows us to enforce restrictions and protect our services from too many requests*

Bulkhead

⟶ *Isolates failures and prevents them from cascading through the system*

⟶ *Limit the amount of parallel executions or concurrent calls to prevent system resources from being exhausted*

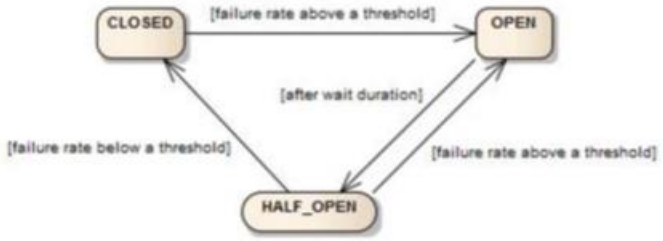
CircuitBreaker

⟶ *Used to prevent a network or service failure from cascading to other services*

⟶ *Circuit breaker 'trips' or opens and prevents further calls to the service*

Introduction

The CircuitBreaker is implemented via a finite state machine with three normal states: CLOSED, OPEN and HALF_OPEN and two special states DISABLED and FORCED_OPEN.



The CircuitBreaker uses a sliding window to store and aggregate the outcome of calls. You can choose between a count-based sliding window and a time-based sliding window. The count-based sliding window aggregrates the outcome of the last N calls. The time-based sliding window aggregrates the outcome of the calls of the last N seconds.

Add a ressiliance4j dependency in application

Resilience4J SPRING CLOUD CIRCUIT BREAKER

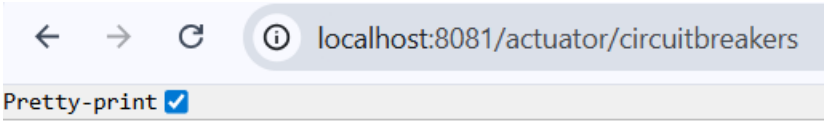
Spring Cloud Circuit breaker with Resilience4j as the underlying implementation.

set ressilience4j properties

```java
@Override
@CircuitBreaker(name = "companyBreaker")
public List<JobDTO> findAll() {
    List<Job> jobs = jobRepository.findAll();
    return jobs.stream().map(this::convertToDto).collect(Collectors.toList());
}
```

```properties
#Ressilience4j
resilience4j.circuitbreaker.instances.companyBreaker.register-health-indicator=true
resilience4j.circuitbreaker.instances.companyBreaker.sliding-window-type=COUNT_BASED
resilience4j.circuitbreaker.instances.companyBreaker.sliding-window-size=10
resilience4j.circuitbreaker.instances.companyBreaker.minimum-number-of-calls=5
resilience4j.circuitbreaker.instances.companyBreaker.failure-rate-threshold=50
resilience4j.circuitbreaker.instances.companyBreaker.wait-duration-in-open-state=10s
resilience4j.circuitbreaker.instances.companyBreaker.permitted-number-of-calls-in-half-open-state=3
resilience4j.circuitbreaker.instances.companyBreaker.automatic-transition-from-open-to-half-open-enabled=true
resilience4j.timelimiter.instances.companyBreaker.timeout-duration=3s
resilience4j.timelimiter.instances.companyBreaker.cancel-running-future=true
resilience4j.retry.instances.companyBreaker.max-attempts=3
resilience4j.retry.instances.companyBreaker.wait-duration=500ms
resilience4j.ratelimiter.instances.companyBreaker.limit-for-period=10
resilience4j.ratelimiter.instances.companyBreaker.limit-refresh-period=1s
resilience4j.ratelimiter.instances.companyBreaker.timeout-duration=0

#Actuators
management.endpoints.web.exposure.include=*
management.health.circuitbreakers.enabled=true
management.endpoint.health.show-details=always
```

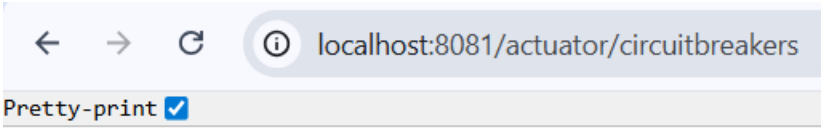
If service is down state will change to open



Pretty-print ☑


```
{
  "circuitBreakers": {
    "companyBreaker": {
      "bufferedCalls": 10,
      "failedCalls": 5,
      "failureRate": "50.0%",
      "failureRateThreshold": "50.0%",
      "notPermittedCalls": 0,
      "slowCallRate": "0.0%",
      "slowCallRateThreshold": "100.0%",
      "slowCalls": 0,
      "slowFailedCalls": 0,
      "state": "OPEN"
    }
  }
}
```

```
{
  "circuitBreakers": {
    "companyBreaker": {
      "bufferedCalls": 1,
      "failedCalls": 0,
      "failureRate": "-1.0%",
      "failureRateThreshold": "50.0%",
      "notPermittedCalls": 0,
      "slowCallRate": "-1.0%",
      "slowCallRateThreshold": "100.0%",
      "slowCalls": 0,
      "slowFailedCalls": 0,
      "state": "HALF_OPEN"
    }
  }
}
```


```
{
  "circuitBreakers": {
    "companyBreaker": {
      "bufferedCalls": 0,
      "failedCalls": 0,
      "failureRate": "-1.0%",
      "failureRateThreshold": "50.0%",
      "notPermittedCalls": 0,
      "slowCallRate": "-1.0%",
      "slowCallRateThreshold": "100.0%",
      "slowCalls": 0,
      "slowFailedCalls": 0,
      "state": "CLOSED"
    }
  }
}
```


Fallback Message

```
@Override
@CircuitBreaker(name = "companyBreaker", fallbackMethod = "companyFallBackMethod")
public List<JobDTO> findAll() {
    List<Job> jobs = jobRepository.findAll();
    return jobs.stream().map(this::convertToDto).collect(Collectors.toList());
}

no usages  new *
public List<String> companyFallBackMethod(Exception e){
    List<String> list = new ArrayList<>();
    list.add("Dummy");
    return list;
}
```

Retry annotation

```java
@Override
// @CircuitBreaker(name = "companyBreaker", fallbackMethod = "companyFallBackMethod")
@Retry(name = "companyBreaker", fallbackMethod = "companyFallBackMethod")
public List<JobDTO> findAll() {
    List<Job> jobs = jobRepository.findAll();
    return jobs.stream().map(this::convertToDto).collect(Collectors.toList());
}
```

```
resilience4j.retry.instances.companyBreaker.max-attempts=3
resilience4j.retry.instances.companyBreaker.wait-duration=500ms
```

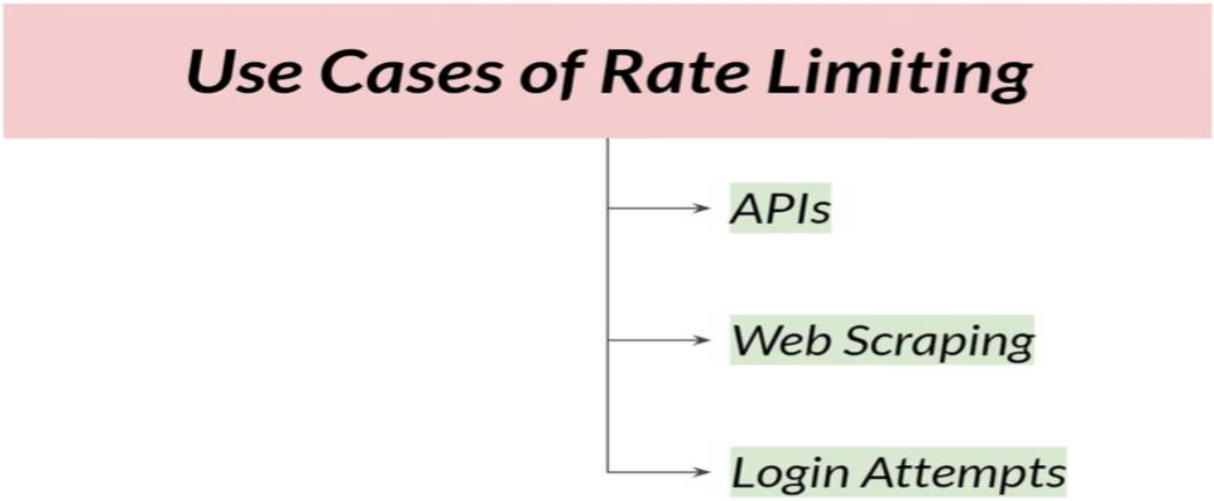
What Is Rate Limiting and Why Is It Needed?

Rate Limiting

Rate limiting is a technique for limiting network traffic.

Importance of Rate Limiting

- Preventing Abuse
- Resource Allocation
- Cost Management



Distributed Denial of Service (DDoS) attacks

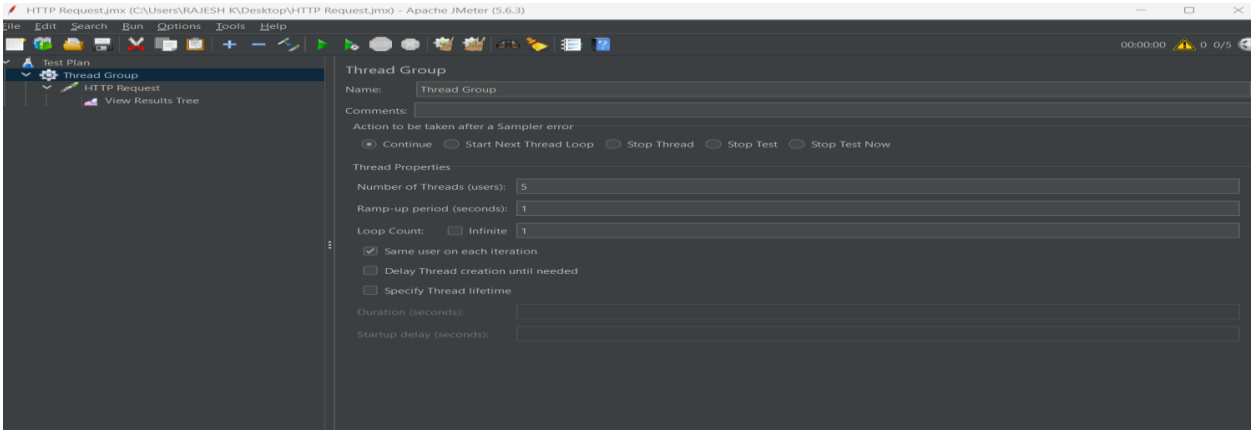
Set ratelimiter annotation

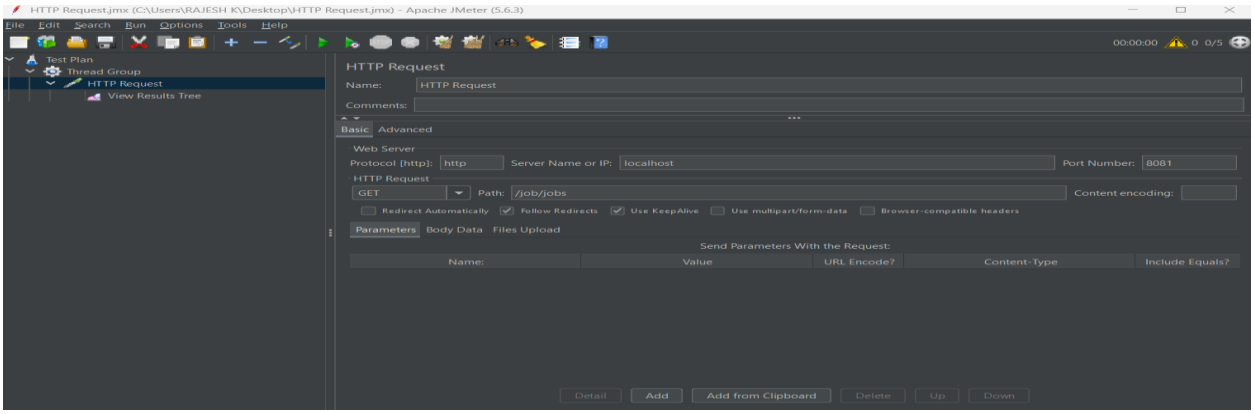
```
// @CircuitBreaker(name = "companyBreaker", fallbackMethod = "companyFallBackMethod")
// @Retry(name = "companyBreaker", fallbackMethod = "companyFallBackMethod")
@RateLimiter(name = "companyBreaker", fallbackMethod = "companyFallBackMethod")
public List<JobDTO> findAll() {
    List<Job> jobs = jobRepository.findAll();
    return jobs.stream().map(this::convertToDto).collect(Collectors.toList());
}
```

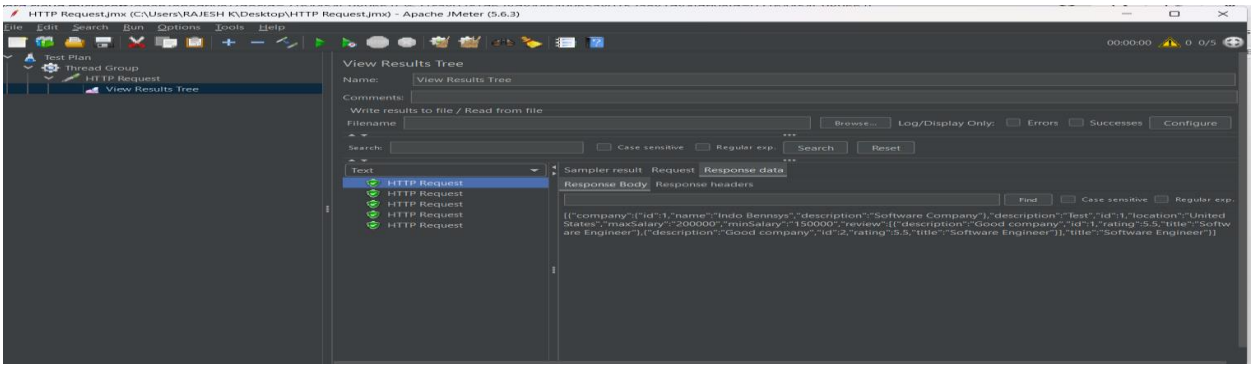
Set RateLimiter Properties

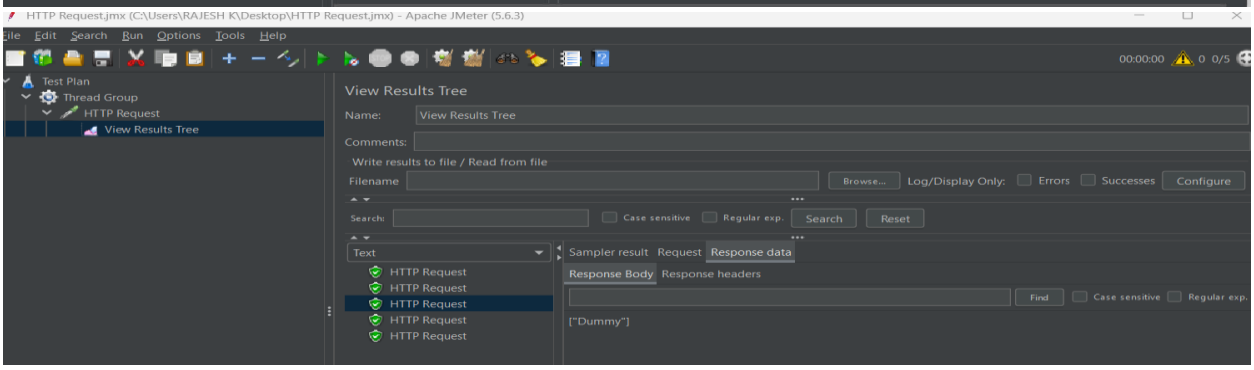
```
resilience4j.ratelimiter.instances.companyBreaker.limit-for-period=2
resilience4j.ratelimiter.instances.companyBreaker.limit-refresh-period=4s
resilience4j.ratelimiter.instances.companyBreaker.timeout-duration=0

#Actuators
```

Test with Jmeter

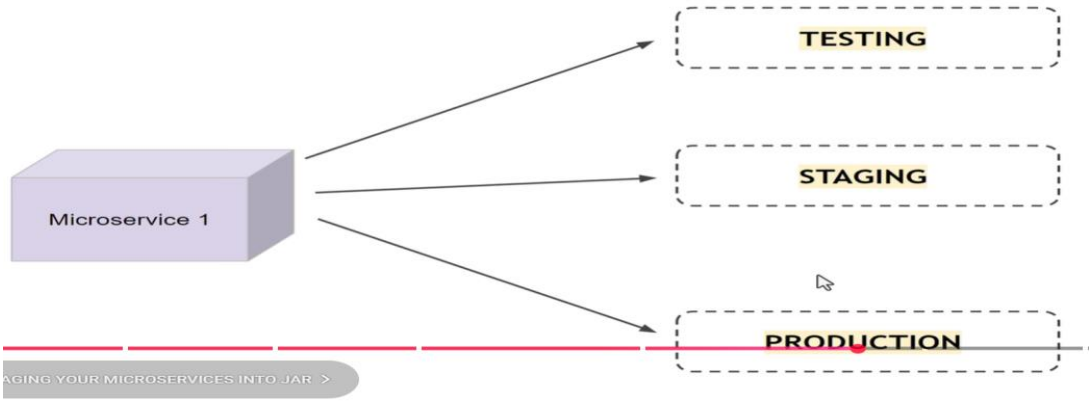








Introduction to Microservice Packaging



Execution of Spring Boot Application

→ *Code Compilation*

→ *Running the Main Class*

→ *Classpath and Dependencies*

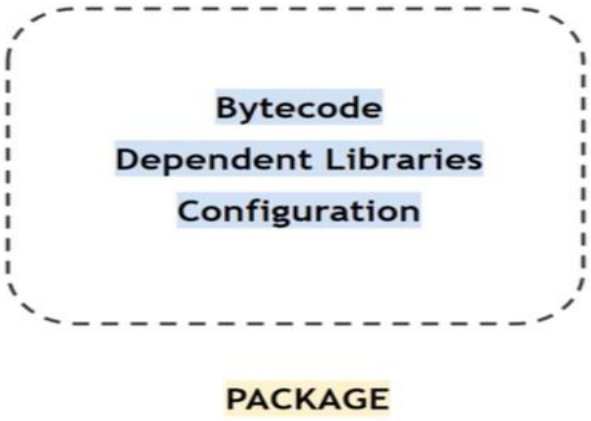
→ *Embedded Server*

→ *Source Code Changes*

→ *Development Mode*

What is Packaging

Packaging is involves compiling your source code into bytecode, bundling it with any dependent libraries, and creating a single, executable artifact that can be easily distributed and run.



Commands

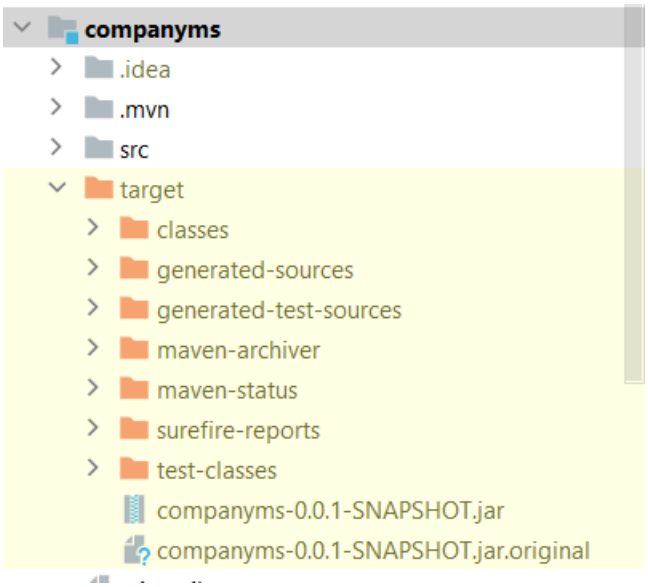
→ *mvn clean*

→ *mvn package*

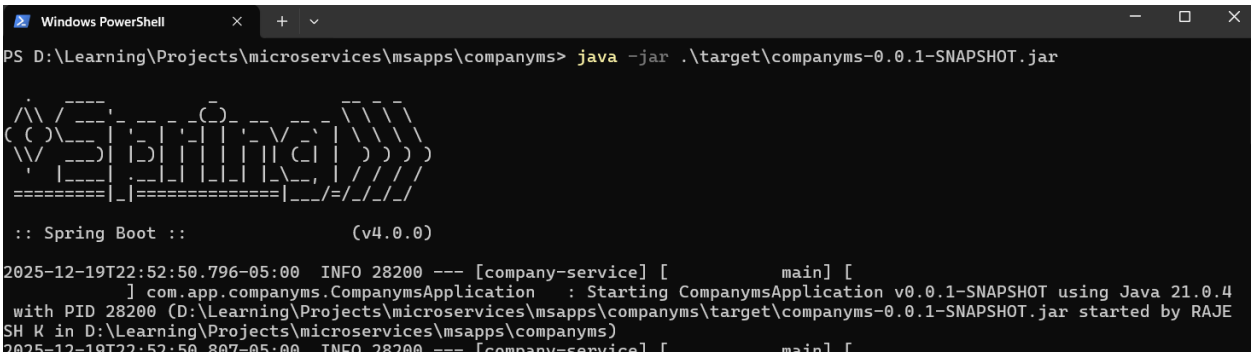
→ *mvn clean package*

To create package using this command

```
PS D:\Learning\Projects\microservices\msapps\companyms> ./mvnw package
[INFO] Scanning for projects...
```

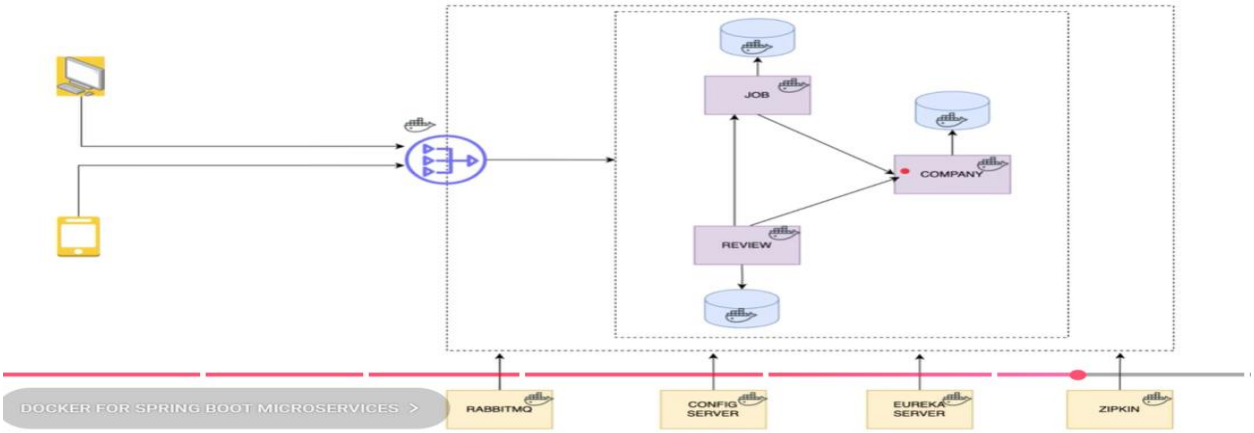


We can test the jar using this command



Containerizing Spring Boot Microservices

Running all services in docker



We are going to run all the services using single file using **docker compose**

Create application-docker.properties files in all microservices and set these properties

```
application-docker.properties
job-service.url=http://jobs:8081
company-service.url=http://company:8082
review-service.url=http://review:8083

spring.application.name=company-service
server.port=8082

#H2
```

Set Url in feign client

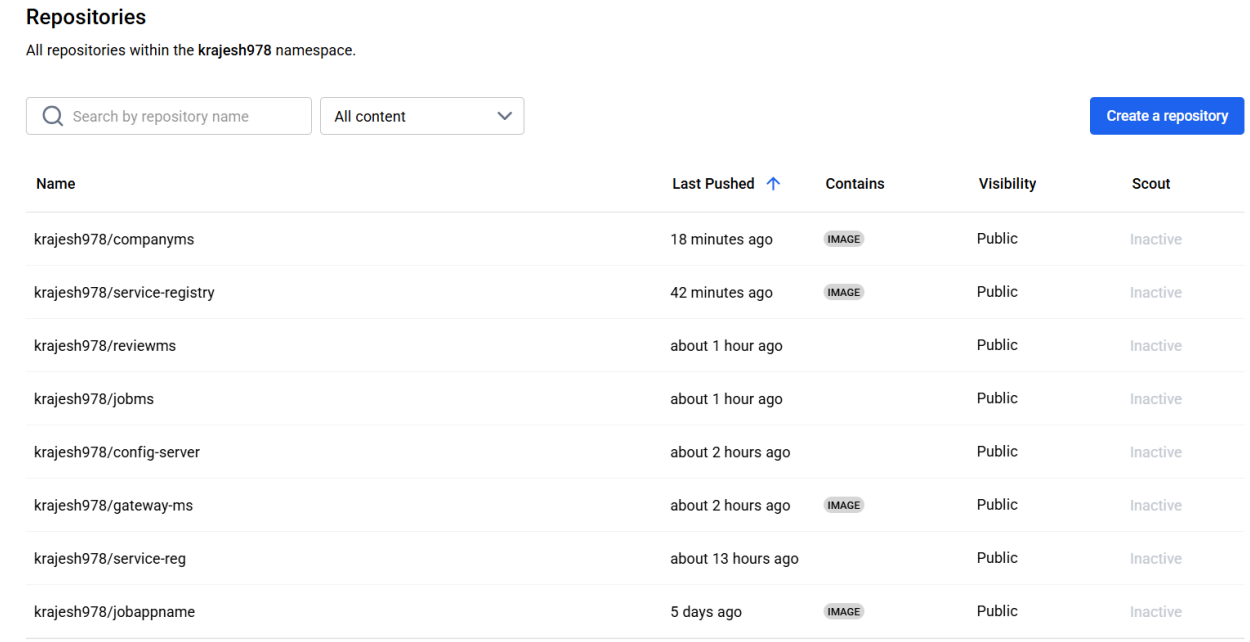
```
3 usages   rajesh kakumani
@FeignClient(name = "COMPANY-SERVICE", url="${company-service.url}")
public interface CompanyClient {

    rajesh kakumani
    @GetMapping("/companies/{companyId}")
    Company getCompany(@PathVariable("companyId") Long companyId);

}
```

Generate service-reg docker image

```
Loading personal and system profiles took 686ms.
PS D:\Learning\Projects\microservices\msapps\service-reg> ./mvnw spring-boot:build-image "-Dspring-boot.build-image.imag
eName=krajesh978/service-reg"
[INFO] Scanning for projects...
Downloading from central: https://repo.maven.apache.org/maven2/org/springframework/boot/maven-metadata.xml
```

Push docker image in docker registry

```
[INFO] ------------------------------------------------------------------------
PS D:\Learning\Projects\microservices\msapps\service-reg> docker push krajesh978/service-reg
Using default tag: latest
The push refers to repository [docker.io/krajesh978/service-reg]
```



Create all images and push to docker repositor

```
#Docker images

#Service-reg
./mvnw spring-boot:build-image "-Dspring-boot.build-image.imageName=krajesh978/service-reg"

#Gateway
./mvnw spring-boot:build-image "-Dspring-boot.build-image.imageName=krajesh978/gateway-ms"

#Config-server
./mvnw spring-boot:build-image "-Dspring-boot.build-image.imageName=krajesh978/config-server"

#Company Ms
./mvnw spring-boot:build-image "-Dspring-boot.build-image.imageName=krajesh978/companyms"

#Job Ms
./mvnw spring-boot:build-image "-Dspring-boot.build-image.imageName=krajesh978/jobms"

#Review Ms
./mvnw spring-boot:build-image "-Dspring-boot.build-image.imageName=krajesh978/reviewms"
```

To run all the images at a time using docker compose

docker-compose.yaml

```yaml
services:
  postgres:
    container_name: postgres_db
    image: postgres
    environment:
      POSTGRES_USER: postgres
      POSTGRES_PASSWORD: postgres
      PGDATA: data/postgres
    volumes:
      - postgres:/data/postgres
    ports:
      - "5432:5432"
    networks:
      - postgres
    restart: unless-stopped

  pgadmin:
    container_name: pgadmin_container
    image: dpage/pgadmin4
    environment:
      PGADMIN_DEFAULT_EMAIL: ${PGADMIN_DEFAULT_EMAIL:-pgadmin4@pgadmin.org}
      PGADMIN_DEFAULT_PASSWORD: ${PGADMIN_DEFAULT_PASSWORD:-admin}
      PGADMIN_CONFIG_SERVER_MODE: 'False'
    volumes:
      - pgadmin:/var/lib/pgadmin
```

Run the images using docker command

```
Windows PowerShell
PS D:\Learning\Projects\microservices\msapps\companyms> docker compose up -d
[+] Running 12/12
 ✔Network companyms_default                  Created
 ✔Network companyms_postgres                 Created
 ✔Network companyms_microservice-network     Created
 ✔Container pgadmin_container                Started
 ✔Container zipkin                           Started
 ✔Container postgres_db                      Started
 ✔Container service-registry                 Started
 ✔Container config-server                    Started
 ✔Container gateway                          Started
 ✔Container job                              Started
 ✔Container company                          Started
 ✔Container review                           Started
PS D:\Learning\Projects\microservices\msapps\companyms>
```

companyms	-	-	-	3.25%	1 m				
postgres_db	d8b876d9ffcc	postgres	5432:5432	0.04%	1 m				
pgadmin_container	a90e16e707fd	dpage/pgadmin4	5050:80	0.12%	1 m				
zipkin	95d6d079f6b3	openzipkin/zipkin-slim:latest	9411:9411	0.2%	1 m				
service-registry	4d7a0c60c74c	krajesh978/service-registry:latest	8761:8761	2.5%	1 m				
gateway	bf6697bb3661	krajesh978/gateway-ms:latest	8085:8085	0.39%	1 m				
config-server	c6dcf845110c	krajesh978/config-server:latest	8084:8084	0%	1 m				
company	5e8f2a37dad7	krajesh978/companyms:latest	8082:8082	0%	1 m				
job	62ec87c4476e	krajesh978/jobms:latest	8081:8081	0%	1 m				
review	a63c91120828	krajesh978/reviewms:latest	8083:8083	0%	1 m				

Stop the running images using docker command

```
PS D:\Learning\Projects\microservices\msapps\companyms> docker compose down
[+] Running 12/12
 ✔Container job                                Removed
 ✔Container company                            Removed
 ✔Container gateway                            Removed
 ✔Container review                             Removed
 ✔Container config-server                      Removed
 ✔Container service-registry                   Removed
 ✔Container pgadmin_container                  Removed
 ✔Container postgres_db                        Removed
 ✔Container zipkin                             Removed
 ✔Network companyms_microservice-network       Removed
 ✔Network companyms_postgres                   Removed
 ✔Network companyms_default                    Removed
```

To check what are all the images we created

```
PS D:\Learning\Projects\microservices\msapps\companyms> docker images
REPOSITORY                                TAG       IMAGE ID       CREATED         SIZE
postgres                                  latest    38d5c9d52203   11 days ago     643MB
dpage/pgadmin4                            latest    50700ac17936   12 days ago     823MB
paketobuildpacks/ubuntu-noble-run-tiny    0.0.47    9573df49a1c9   2 weeks ago     38.9MB
cassandra                                 latest    33524cfd117a   7 weeks ago     581MB
confluentinc/cp-kafka                     7.5.0     fbbb6fa11b25   2 years ago     1.33GB
confluentinc/cp-zookeeper                 7.5.0     02f6c042bb9a   2 years ago     1.33GB
krajesh978/companyms                      latest    513ae9645be6   46 years ago    670MB
krajesh978/jobms                          latest    f86e7a00f18a   46 years ago    674MB
<none>                                    <none>    5d0788799607   46 years ago    674MB
krajesh978/config-server-ms               latest    adc60fce8b93   46 years ago    601MB
krajesh978/config-server                  latest    adc60fce8b93   46 years ago    601MB
<none>                                    <none>    aeb03abc86fa   46 years ago    670MB
<none>                                    <none>    eee989db28f8   46 years ago    670MB
krajesh978/service-reg                    latest    c0c5264395bc   46 years ago    611MB
krajesh978/service-registry               latest    c0c5264395bc   46 years ago    611MB
krajesh978/reviewms                       latest    46096d71ca8b   46 years ago    670MB
<none>                                    <none>    3e12b33c2223   46 years ago    1.18GB
<none>                                    <none>    a581b1a926aa   46 years ago    670MB
krajesh978/gateway-ms                     latest    583bdc5b9ca7   46 years ago    593MB
krajesh978/jobappname                     latest    d6229f744334   46 years ago    604MB
paketobuildpacks/builder-noble-java-tiny  latest    7fc39dc25d0d   46 years ago    1.19GB
pack.local/builder/yklliloqxi             latest    5f33e55a5f55   46 years ago    845MB
ghcr.io/openzipkin/zipkin-slim            latest    d95199401718   56 years ago    76.8MB
```

To check what are all the images are running

```
PS D:\Learning\Projects\microservices\msapps\companyms> docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED         STATUS
         PORTS                                                NAMES
c0f0115045ee   krajesh978/companyms:latest          "/cnb/process/web"       7 seconds ago   Up 5 seconds
         0.0.0.0:8082->8082/tcp, [::]:8082->8082/tcp          company
ba5626ed067a   krajesh978/reviewms:latest           "/cnb/process/web"       7 seconds ago   Up 5 seconds
         0.0.0.0:8083->8083/tcp, [::]:8083->8083/tcp          review
6be3c40354a7   krajesh978/jobms:latest              "/cnb/process/web"       7 seconds ago   Up 5 seconds
         0.0.0.0:8081->8081/tcp, [::]:8081->8081/tcp          job
bdb8a408aec2   krajesh978/gateway-ms:latest         "/cnb/process/web"       8 seconds ago   Up 6 seconds
         0.0.0.0:8085->8085/tcp, [::]:8085->8085/tcp          gateway
f3d0ba160866   krajesh978/service-registry:latest   "/cnb/process/web"       8 seconds ago   Up 6 seconds
         0.0.0.0:8761->8761/tcp, [::]:8761->8761/tcp          service-registry
647e03e8598c   postgres                             "docker-entrypoint.s…"   8 seconds ago   Up 7 seconds
         0.0.0.0:5432->5432/tcp, [::]:5432->5432/tcp          postgres_db
891e3c83c9a7   dpage/pgadmin4                       "/entrypoint.sh"         8 seconds ago   Up 7 seconds
         0.0.0.0:5050->80/tcp, [::]:5050->80/tcp              pgadmin_container
382c48190462   ghcr.io/openzipkin/zipkin-slim:latest   "start-zipkin"           8 seconds ago   Up 7 seconds (he
arting)   0.0.0.0:9411->9411/tcp, [::]:9411->9411/tcp          zipkin
```

What is Kubernetes?

Kubernetes

→ *Kubernetes is essentially a platform designed to completely manage the life cycle of containerized applications using methods that provide predictability, scalability, and high availability*

Kubernetes

→ *Kubernetes is essentially a platform designed to completely manage the life cycle of containerized applications using methods that provide predictability, scalability, and high availability*

→ *You can define how your applications should run and the ways they should be able to interact with other applications or the outside world*

Benefits of Kubernetes

Benefits

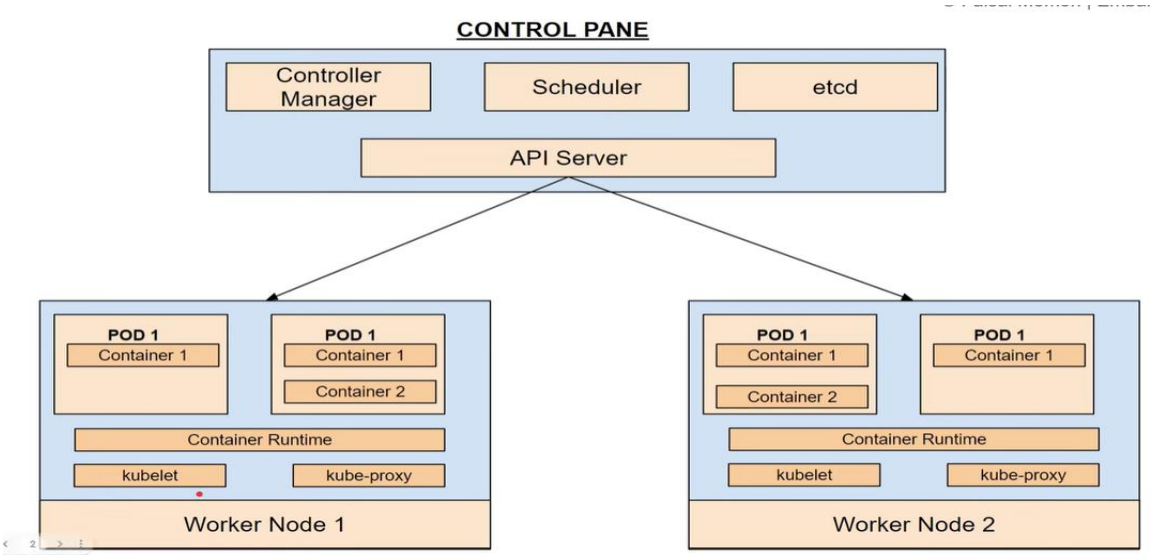
→ *Service Discovery and Load Balancing*

→ *Automated Rollouts and Rollbacks*

→ *Horizontal Scaling*

→ *Self-Healing*

→ *Secret and Configuration Management*



Minikube

Download minikube for local deployment and run minikube in windows powershell

```
Loading personal and system profiles took 625ms.
PS C:\Users\RAJESH K> minikube
minikube provisions and manages local Kubernetes clusters optimized for development workflows.

Basic Commands:
  start            Starts a local Kubernetes cluster
  status           Gets the status of a local Kubernetes cluster
  stop             Stops a running local Kubernetes cluster
  delete           Deletes a local Kubernetes cluster
  dashboard        Access the Kubernetes dashboard running within the minikube cluster
  pause            pause Kubernetes
  unpause          unpause Kubernetes

Images Commands:
  docker-env       Provides instructions to point your terminal's docker-cli to the Docker Engine inside minikube.
(Useful for building docker images directly inside minikube)
  podman-env       Configure environment to use minikube's Podman service
  cache            Manage cache for images
  image            Manage images
```

```
Use "minikube <command> --help" for more information about a given command.
PS C:\Users\RAJESH K> minikube start
* minikube v1.37.0 on Microsoft Windows 11 Pro 10.0.26200.7462 Build 26200.7462
* Automatically selected the docker driver
* Using Docker Desktop driver with root privileges
* Starting "minikube" primary control-plane node in "minikube" cluster
* Pulling base image v0.0.48 ...
* Downloading Kubernetes v1.34.0 preload ...
    > preloaded-images-k8s-v18-v1...:  337.07 MiB / 337.07 MiB  100.00% 12.87 M
    > gcr.io/k8s-minikube/kicbase...:  488.52 MiB / 488.52 MiB  100.00% 10.86 M
* Creating docker container (CPUs=2, Memory=4000MB) ...
! Failing to connect to https://registry.k8s.io/ from inside the minikube container
* To pull new external images, you may need to configure a proxy: https://minikube.sigs.k8s.io/docs/reference/networking
/proxy/
* Preparing Kubernetes v1.34.0 on Docker 28.4.0 ...
* Configuring bridge CNI (Container Networking Interface) ...
* Verifying Kubernetes components...
  - Using image gcr.io/k8s-minikube/storage-provisioner:v5
* Enabled addons: storage-provisioner, default-storageclass
* Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
```

Use docker driver

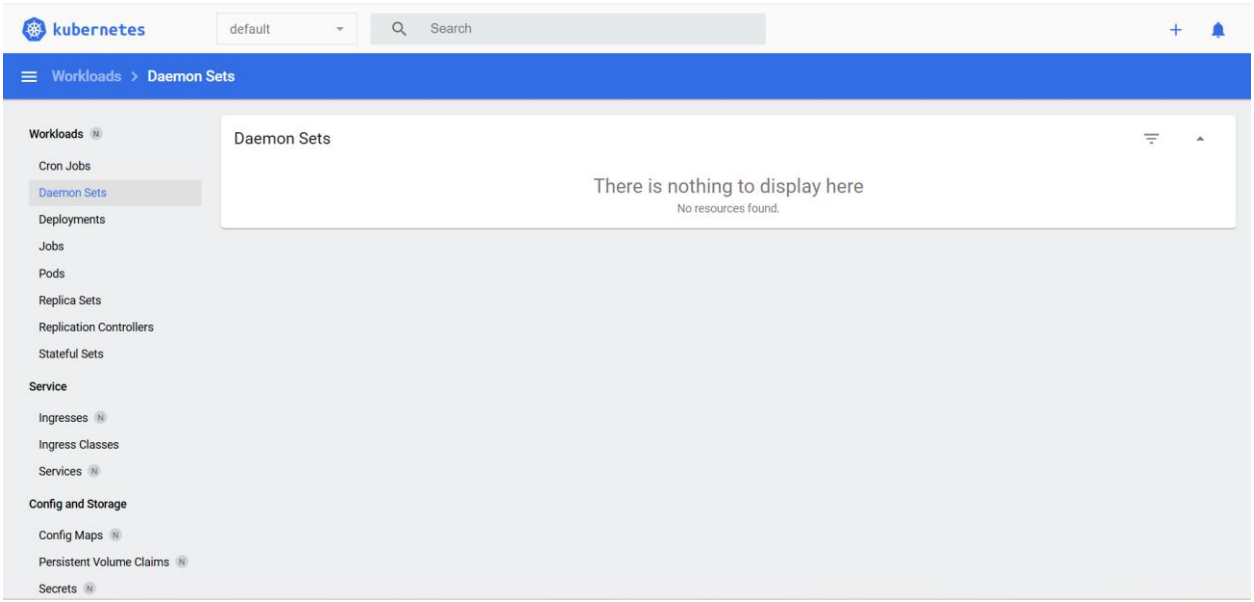
```
PS C:\Users\RAJESH K> minikube start --driver=docker
* minikube v1.37.0 on Microsoft Windows 11 Pro 10.0.26200.7462 Build 26200.7462
* Using the docker driver based on existing profile
* Starting "minikube" primary control-plane node in "minikube" cluster
* Pulling base image v0.0.48 ...
* Updating the running docker "minikube" container ...
! Failing to connect to https://registry.k8s.io/ from inside the minikube container
* To pull new external images, you may need to configure a proxy: https://minikube.sigs.k8s.io/docs/
/proxy/
* Preparing Kubernetes v1.34.0 on Docker 28.4.0 ...
* Verifying Kubernetes components...
  - Using image gcr.io/k8s-minikube/storage-provisioner:v5
* Enabled addons: storage-provisioner, default-storageclass
* Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
```

```
PS C:\Users\RAJESH K> kubectl cluster-info
Kubernetes control plane is running at https://127.0.0.1:64409
CoreDNS is running at https://127.0.0.1:64409/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy
```

We can get minikube dashboard

```
PS C:\Users\RAJESH K> minikube dashboard
* Enabling dashboard ...
  - Using image docker.io/kubernetesui/dashboard:v2.7.0
  - Using image docker.io/kubernetesui/metrics-scraper:v1.0.8
* Some dashboard features require the metrics-server addon. To enable all features please run:

        minikube addons enable metrics-server

* Verifying dashboard health ...
* Launching proxy ...
* Verifying proxy health ...
* Opening http://127.0.0.1:65520/api/v1/namespaces/kubernetes-dashboard/services/http:kubernetes-dashboard:/proxy/
ur default browser...
```

Visual respresentation of minikube dashboard



Pods

What is a Pod

→ A pod groups one or more containers and their shared resources, such as volumes (storage), IP address, and network ports.

→ Containers within a pod run on the same worker node and share the same lifecycle.

→ Pods are ephemeral and can be created, scheduled, and destroyed dynamically

Pod is not...

→ A pod is not a durable entity

→ Pods are not designed for horizontal scaling on their own

Key considerations

→ Designed to be stateless

→ Communicate with each other within the same cluster using localhost

→ Pods are assigned a unique IP address within the cluster

Key considerations

→ Lifecycle and availability of pods are managed by Kubernetes

→ Pods can have associated labels and annotations

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: my-pod
spec:
  containers:
  - name: nginx
    image: nginx:1.14.2
    ports:
    - containerPort: 80
```


```yaml
apiVersion: v1
kind: Pod
metadata:
  name: my-pod
spec:
  containers:
  - name: nginx
    image: nginx:1.14.2
    ports:
    - containerPort: 80
```


```yaml
  - name: redis
    image: redis:6.2.5
```

Create a pod using this command

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: pod
spec:
  containers:
    - name: nginx
      image: nginx:1.14.2
      ports:
        - containerPort: 80
```

```
PS D:\Learning\Projects\microservices\msapps\k8s> kubectl apply -f first-pod.yaml
pod/pod created
```

Pod is immutable

If we are trying to update to a pod it wont create or update

first-pod.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: pod
spec:
  containers:
    - name: nginx
      image: nginx:1.14.2
      ports:
        - containerPort: 80
    - name: redis
      image: redis:8.4.0
```

```
PS D:\Learning\Projects\microservices\msapps\k8s> kubectl apply -f first-pod.yaml
The Pod "pod" is invalid: spec.containers: Forbidden: pod updates may not add or remove containers
```

Get list of pods

```
PS D:\Learning\Projects\microservices\msapps\k8s> kubectl get pods
NAME   READY   STATUS    RESTARTS   AGE
pod    1/1     Running   0          4m34s
```

Pods watch mode

```
PS D:\Learning\Projects\microservices\msapps\k8s> kubectl get pods -w
NAME         READY   STATUS    RESTARTS   AGE
my-new-pod   2/2     Running   0          32s
pod          1/1     Running   0          6m11s
```

Service

A Service in Kubernetes is an abstraction that defines a logical set of Pods and a policy by which to access them, sometimes called a **micro-service**

Why do we need it?

Pods in Kubernetes are ephemeral, they can be created and destroyed

```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  selector:
    app: MyApp
  ports:
    - protocol: TCP
      port: 80
      targetPort: 9376
```

Create a service for pods to communicate

first-pod.yaml | first-service.yaml

```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  selector:
    app: myApp
  ports:
    - protocol: TCP
      port: 80
      targetPort: 9376
```

Commands to create service and get service information

```
PS D:\Learning\Projects\microservices\msapps\k8s> kubectl apply -f first-service.yaml
service/my-service created
PS D:\Learning\Projects\microservices\msapps\k8s> kubectl get service
NAME         TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.96.0.1      <none>        443/TCP   32m
my-service   ClusterIP   10.107.4.48    <none>        80/TCP    20s
```

Exposing your application

Types of Service

→ *ClusterIP*

→ *NodePort*

→ *LoadBalancer*

A ReplicaSet is a Kubernetes object used for managing and scaling a set of identical pod replicas.

Why do you need identical Pods?

→ *High Availability*

→ *Load Balancing*

→ *Scaling*

→ *Rolling Updates*

→ *Service Discovery and Load Balancing*

Why do we need it?

Pods in Kubernetes are ephemeral, they can be created and destroyed

What a ReplicaSet is not

→ *A ReplicaSet is not designed to handle rolling updates or deployments*

→ *It does not provide declarative updates to the pods it manages*

What to keep in mind

→ *ReplicaSets use a selector to identify the pods it manages*

→ *You specify the desired number of replicas*

What to keep in mind

→ *ReplicaSets use a selector to identify the pods it manages*

→ *You specify the desired number of replicas*

→ *If a pod managed by a ReplicaSet fails or gets deleted, the ReplicaSet replaces it automatically to maintain the desired replica count*

→ *ReplicaSets are often used together with Deployments*

→ *When updating the configuration of the pods, such as image versions or resource requirements, it's recommended to use Deployments*

Create replica set

first-pod.yaml | first-service.yaml | replica-set.yaml

```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: my-replicaset
spec:
  replicas: 3
  selector:
    matchLabels:
     app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:1.14.2
          ports:
            - containerPort: 80
```

Commands to create and check replicas of pod

```
PS D:\Learning\Projects\microservices\msapps\k8s> kubectl get service
NAME         TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.96.0.1      <none>        443/TCP   32m
my-service   ClusterIP   10.107.4.48    <none>        80/TCP    20s
PS D:\Learning\Projects\microservices\msapps\k8s> kubectl apply -f replica-set.yaml
replicaset.apps/my-replicaset created
PS D:\Learning\Projects\microservices\msapps\k8s> kubectl get pods -w
NAME                 READY   STATUS    RESTARTS   AGE
my-new-pod           2/2     Running   0          24m
my-replicaset-kzkjj  1/1     Running   0          13s
my-replicaset-mqtb9  1/1     Running   0          13s
my-replicaset-snw7b  1/1     Running   0          13s
pod                  1/1     Running   0          30m
PS D:\Learning\Projects\microservices\msapps\k8s> kubectl get replicaset
NAME           DESIRED   CURRENT   READY   AGE
my-replicaset  3         3         3       2m13s
```

Whenever delete the pod it automatically creates a new one

```
PS D:\Learning\Projects\microservices\msapps\k8s> kubectl delete pod my-replicaset-kzkjj
pod "my-replicaset-kzkjj" deleted from default namespace
PS D:\Learning\Projects\microservices\msapps\k8s> kubectl get pods -w
NAME                 READY   STATUS    RESTARTS   AGE
my-new-pod           2/2     Running   0          28m
my-replicaset-dqvxg  1/1     Running   0          6s
my-replicaset-mqtb9  1/1     Running   0          4m8s
my-replicaset-snw7b  1/1     Running   0          4m8s
pod                  1/1     Running   0          34m
```

Create a deployment

```
PS D:\Learning\Projects\microservices\msapps\k8s> kubectl apply -f first-deployment.yaml
deployment.apps/my-deployment created
PS D:\Learning\Projects\microservices\msapps\k8s> kubectl get deployments
NAME            READY   UP-TO-DATE   AVAILABLE   AGE
my-deployment   0/3     3            0           10s
PS D:\Learning\Projects\microservices\msapps\k8s> kubectl get pods
NAME                             READY   STATUS             RESTARTS   AGE
my-deployment-555bdb55d6-lnkgc   0/1     ImagePullBackOff   0          20s
my-deployment-555bdb55d6-pz786   0/1     ImagePullBackOff   0          20s
my-deployment-555bdb55d6-q6ft8   0/1     ImagePullBackOff   0          20s
my-new-pod                       2/2     Running            0          35m
my-replicaset-dqvxg              1/1     Running            0          7m27s
my-replicaset-mqtb9              1/1     Running            0          11m
my-replicaset-snw7b              1/1     Running            0          11m
pod                              1/1     Running            0          41m
```

irst-pod.yaml | first-service.yaml | replica-set.yaml | first-deployment.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-deployment
spec:
  replicas: 3
  selector:
    matchLabels:
      app: myapp
  template:
    metadata:
      labels:
        app: myapp
    spec:
      containers:
        - name: nginx
          image: nginx:1.14.2
```

We can update and deploy the container using deployment

We can delete deployments, pods and services and replicasets using delete command

```
PS D:\Learning\Projects\microservices\msapps\k8s> kubectl delete deployments --all --all-namesspaces
error: unknown flag: --all-namesspaces
PS D:\Learning\Projects\microservices\msapps\k8s> kubectl delete pod --all --all-namespaces
pod "my-new-pod" deleted from default namespace
pod "pod" deleted from default namespace
pod "etcd-minikube" deleted from kube-system namespace
pod "kube-apiserver-minikube" deleted from kube-system namespace
pod "kube-controller-manager-minikube" deleted from kube-system namespace
pod "kube-proxy-vhjff" deleted from kube-system namespace
pod "kube-scheduler-minikube" deleted from kube-system namespace
pod "storage-provisioner" deleted from kube-system namespace
PS D:\Learning\Projects\microservices\msapps\k8s> kubectl get all
NAME                 TYPE        CLUSTER-IP    EXTERNAL-IP   PORT(S)   AGE
service/kubernetes   ClusterIP   10.96.0.1     <none>        443/TCP   164m
service/my-service   ClusterIP   10.107.4.48   <none>        80/TCP    132m
PS D:\Learning\Projects\microservices\msapps\k8s> kubectl delete service --all --all-namespaces
service "kubernetes" deleted from default namespace
service "my-service" deleted from default namespace
service "kube-dns" deleted from kube-system namespace
service "dashboard-metrics-scraper" deleted from kubernetes-dashboard namespace
service "kubernetes-dashboard" deleted from kubernetes-dashboard namespace
PS D:\Learning\Projects\microservices\msapps\k8s> kubectl get all
No resources found in default namespace.
```

Postgres deployment in Kubernetes

```
PS D:\Learning\Projects\microservices\msapps\k8s> kubectl apply -f .\services\postgres\
configmap/postgres-config unchanged
service/postgres unchanged
statefulset.apps/postgres configured
persistentvolumeclaim/postgres-pc-volume-claim created
PS D:\Learning\Projects\microservices\msapps\k8s> kubectl get all
NAME             READY   STATUS    RESTARTS   AGE
pod/postgres-0   0/1     Pending   0          2m8s

NAME                 TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE
service/kubernetes   ClusterIP   10.96.0.1       <none>        443/TCP    4m10s
service/postgres     ClusterIP   10.105.136.10   <none>        5432/TCP   2m8s

NAME                        READY   AGE
statefulset.apps/postgres   0/1     2m8s
```

dont Api gateway in kuberenetees why because ingress will handle api gateway.

Run zipkin service in Kubernetes

```
Loading personal and system profiles took 860ms.
PS C:\Users\RAJESH K> minikube service zipkin --url
http://127.0.0.1:57163
! Because you are using a Docker driver on windows, the terminal needs to be open to run it.
```

To run job service in Kubernetes

```
PS D:\Learning\Projects\microservices\msapps\k8s> minikube service job --url
* service default/job has no node port
! Services [default/job] have type "ClusterIP" not meant to be exposed, however for local development minikube allows yo
u to access this !
http://127.0.0.1:64969
! Because you are using a Docker driver on windows, the terminal needs to be open to run it.
```

Run review service n kubernetes

```
PS C:\Users\RAJESH K> minikube service review --url
* service default/review has no node port
! Services [default/review] have type "ClusterIP" not meant to be exposed, however for local development minikube allows
 you to access this !
http://127.0.0.1:50424
! Because you are using a Docker driver on windows, the terminal needs to be open to run it.
```

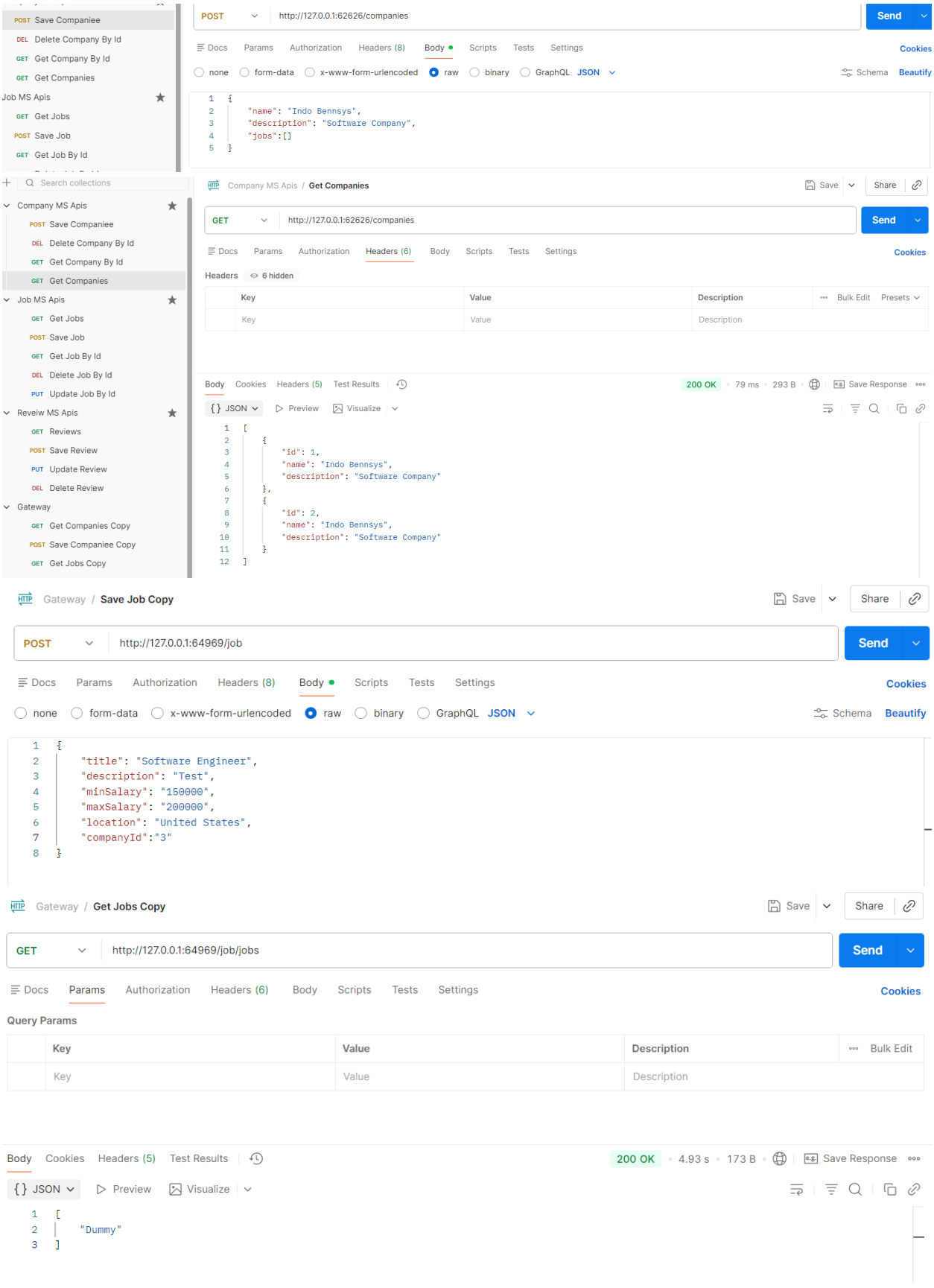
Run company service in Kubernetes

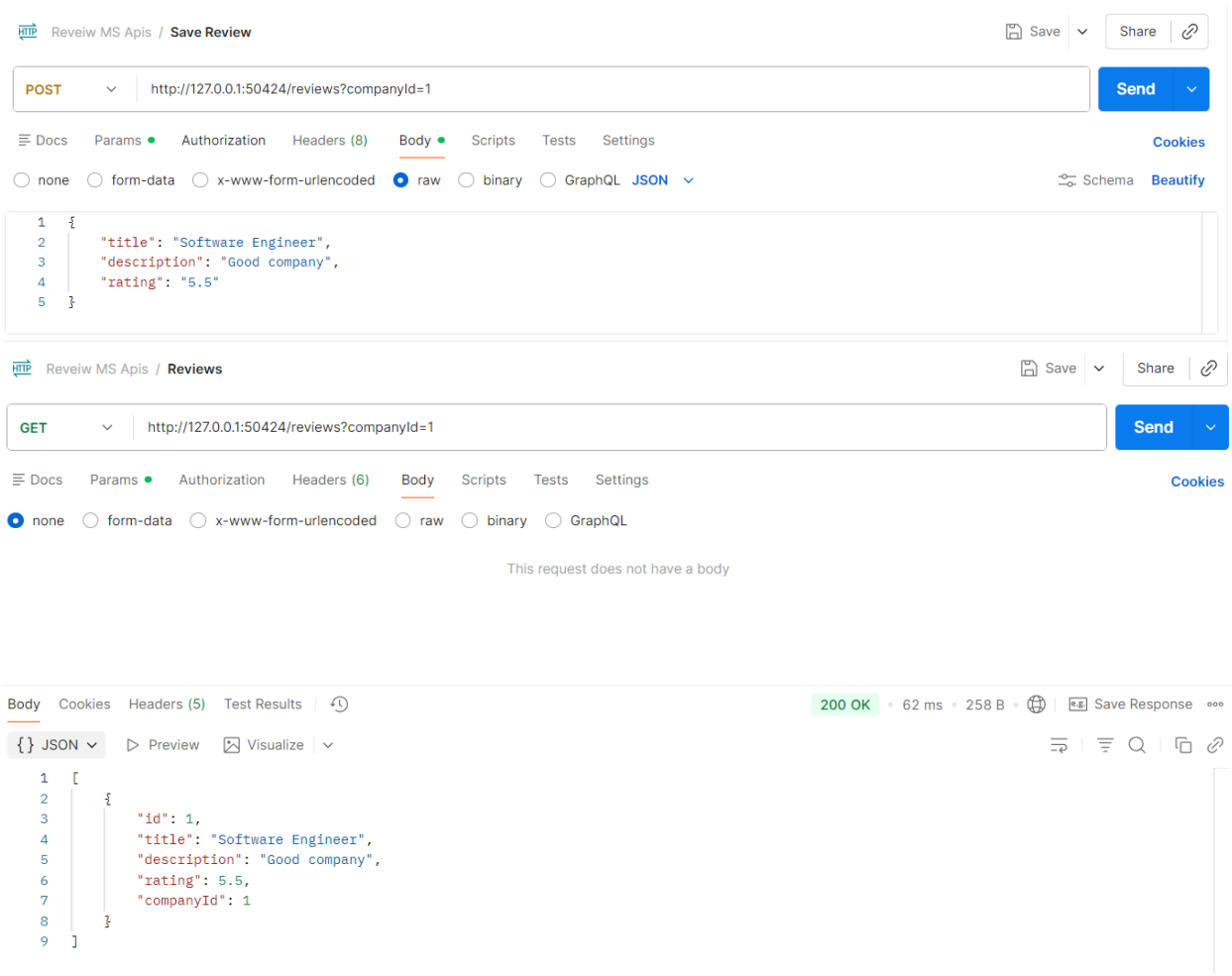
```
Loading personal and system profiles took 1305ms.
PS C:\Users\RAJESH K> minikube service company --url
* service default/company has no node port
! Services [default/company] have type "ClusterIP" not meant to be exposed, however for local development minikube allow
s you to access this !
http://127.0.0.1:62626
! Because you are using a Docker driver on windows, the terminal needs to be open to run it.
```

Deploy all microservices in kubernetes

```
statefulset.apps/postgres    0/1     158m
PS D:\Learning\Projects\microservices\msapps\k8s> minikube service zipkin --url
http://127.0.0.1:58450
! Because you are using a Docker driver on windows, the terminal needs to be open to run
PS D:\Learning\Projects\microservices\msapps\k8s> kubectl apply -f .\bootstrap\company\
deployment.apps/company created
service/company created
PS D:\Learning\Projects\microservices\msapps\k8s> kubectl apply -f .\bootstrap\job\
deployment.apps/job created
service/job created
PS D:\Learning\Projects\microservices\msapps\k8s> kubectl apply -f .\bootstrap\review\
deployment.apps/review created
service/review created
PS D:\Learning\Projects\microservices\msapps\k8s> kubectl get all
NAME                          READY   STATUS             RESTARTS       AGE
pod/company-55c69694f6-q7gfm  1/1     Running            0              46s
pod/job-668d5c7f97-qdl9s      0/1     ContainerCreating  0              41s
pod/postgres-0                0/1     CrashLoopBackOff   14 (25s ago)   158m
pod/review-866d5b7898-vr8fj   0/1     ContainerCreating  0              38s
pod/zipkin-6c97b8f56c-85llk   1/1     Running            0              3m30s
```

PostMan Requests


POST Save Companiee
DEL Delete Company By Id
GET Get Company By Id
GET Get Companies
Job MS Apis
GET Get Jobs
POST Save Job
GET Get Job By Id
POST http://127.0.0.1:62626/companies
Send
Docs Params Authorization Headers (8) Body Scripts Tests Settings
Cookies
none form-data x-www-form-urlencoded raw binary GraphQL JSON
Schema Beautify
{
"name": "Indo Bennsys",
"description": "Software Company",
"jobs":[]
}
Search collections
Company MS Apis
POST Save Companiee
DEL Delete Company By Id
GET Get Company By Id
GET Get Companies
Job MS Apis
GET Get Jobs
POST Save Job
GET Get Job By Id
DEL Delete Job By Id
PUT Update Job By Id
Reveiw MS Apis
GET Reviews
POST Save Review
PUT Update Review
DEL Delete Review
Gateway
GET Get Companies Copy
POST Save Companiee Copy
GET Get Jobs Copy
Company MS Apis / Get Companies
Save Share
GET http://127.0.0.1:62626/companies
Send
Docs Params Authorization Headers (6) Body Scripts Tests Settings
Cookies
Headers 6 hidden
Key Value Description Bulk Edit Presets
Key Value Description
Body Cookies Headers (5) Test Results
200 OK 79 ms 293 B Save Response
{} JSON Preview Visualize
[
{
"id": 1,
"name": "Indo Bennsys",
"description": "Software Company"
},
{
"id": 2,
"name": "Indo Bennsys",
"description": "Software Company"
}
]
Gateway / Save Job Copy
Save Share
POST http://127.0.0.1:64969/job
Send
Docs Params Authorization Headers (8) Body Scripts Tests Settings
Cookies
none form-data x-www-form-urlencoded raw binary GraphQL JSON
Schema Beautify
{
"title": "Software Engineer",
"description": "Test",
"minSalary": "150000",
"maxSalary": "200000",
"location": "United States",
"companyId":"3"
}
Gateway / Get Jobs Copy
Save Share
GET http://127.0.0.1:64969/job/jobs
Send
Docs Params Authorization Headers (6) Body Scripts Tests Settings
Cookies
Query Params
Key Value Description Bulk Edit
Key Value Description
Body Cookies Headers (5) Test Results
200 OK 4.93 s 173 B Save Response
{} JSON Preview Visualize
[
"Dummy"
]


Reveiw MS Apis / Save Review
Save
Share
POST
http://127.0.0.1:50424/reviews?companyId=1
Send
Docs
Params
Authorization
Headers (8)
Body
Scripts
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Schema
Beautify
{
"title": "Software Engineer",
"description": "Good company",
"rating": "5.5"
}
Reveiw MS Apis / Reviews
Save
Share
GET
http://127.0.0.1:50424/reviews?companyId=1
Send
Docs
Params
Authorization
Headers (6)
Body
Scripts
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
This request does not have a body

Body
Cookies
Headers (5)
Test Results
200 OK
62 ms
258 B
Save Response
{} JSON
Preview
Visualize
[
{
"id": 1,
"title": "Software Engineer",
"description": "Good company",
"rating": 5.5,
"companyId": 1
}
]