Techefx - Spring Boot Microservices and Docker

How to Create Docker Image & Run Spring Boot Microservices In Docker | with Docker Compose | Dockerize Your Container | Monitor Spring Boot Microservices

My Earlier Spring boot Microservices Tutorial Link - https://youtu.be/Z7A_M8HkJG0

In this tutorial, we are going to learn how to deploy spring boot microservices in a docker container; How to orchestrate a docker container using docker compose for running multiple microservices under one network, primarily to enable service to service communications.
I will demonstrate to you, how to set up microservices monitoring using prometheus and data visualisation using grafana. We will also learn to configure various metrics supported by prometheus inside grafana dashboard.
And, we will also integrate distributed tracing using zipkin.

[Objectives]
1. Deploy Microservices in docker container
2. Orchestrate docker using Docker-Compose
3. Configure Microservices Monitoring using Prometheus
4. Configure Prometheus Metrics
5. Configure Grafana for Dashboard Visualization
6. Configure Zipkin for Distributed Tracing

For this tutorial, I am using docker desktop on my machine. You can download docker desktop from docker.com website - **www.docker.com/products/docker-desktop**
It has download links for windows, linux and for mac operating system.
As i am using mac os, I have downloaded the dmg file and set it up in my machine. because i already have it in my machine, I am not going to download it again, and I will directly take you to the setting and configuration page.

enquiry service

For this tutorial, we are going to use two microservices, one is an **enquiry service** and the other is **stock service**. Let me quickly brief you about these services both the services are built on spring boot framework and using spring cloud components.

Enquiry service requires input parameters which are highlighted in RED,
like, product name, product availability, and product units. these are the input parameters, required to call stock service, will see that later. As you can see, there are a couple of additional fields highlighted in BLUE, these are Product Price, discount offer which will come from stock service. total price is calculated based on the product multiplied by the number of units.

Enquiry service calls stock service thru open-feign client library (show the feign client code), which is injected in REST Controller endpoint. If you are not familiar with feign client, let me tell you, you can consider open-feign as an alternate of RestTemplate.
I am running an enquiry service on port 8700.

stock service

Now let's have a quick look at stock service, stock service provides the information about product's stock availability. stock service input parameters are the product name with product availability field, whether it's available or not, as highlighted in RED color.
stock service returns product price and discount offer associated with the product, highlighted in BLUE.
For the demo, I have used **H2 database** to store entity values, and using spring boot's JPA repository with H2.
service is running on port 8800.

For spring cloud components, I am using **Netflix eureka naming server for service registry and using Spring Cloud API gateway server for routing API requests to the destination**.

Prometheus is an open source systems **monitoring and alerting toolkit for your microservices health monitoring.** It has a set of metrics which can be enabled to find the system's health.
In this example, we are going to monitor all these four services, eureka discovery service, enquiry service, stock service and api gateway service. **Prometheus** will collect metrics data from these services through spring cloud **actuator** and send it **grafana** for **visualization**. In this tutorial, we will see how to integrate monitoring on your microservices.

Let's move to another component which is **zipkin**. We all know that **zipkin is a distributed tracing system** that helps gather timely data needed **to troubleshoot latency problems in service** architecture.
We are going to integrate three services, which are **enquiry service, stock service and api gateway t**o zipkin. If you want to know more about zipkin, you can check out my video on Microservices with spring boot and spring cloud with example

#techefx #microservices #springbootmicroservices

My Tags
---
dockerize your microservices
docker compose
spring boot microservices
microservices with docker
microservices monitoring
grafana
prometheus
microservices with grafana
microservices with prometheus
microservices monitoring with grafana and prometheus
run microservices in docker
docker desktop

deploy microservices on docker desktop

microservice with docker
microservice inside docker
docker-compose
docker compose with spring boot

spring boot microservices with docker
microservices with docker example
spring boot microservice with docker desktop
deploy microservices in docker container
microservice with docker tutorial