Microservices Interview Questions

In this post we will look at Spring Boot Microservices questions. Examples are provided with explanation.

Q: What is Spring Boot?
A: Over the years spring has become more and more complex as new functionalities have been added. Just visit the page-https://spring.io/projects (https://spring.io/projects) and we will see all the spring projects we can use in our application for different functionalities. If one has to start a new spring project we have to add build path or add maven dependencies, configure application server, add spring configuration . **So a lot of effort is required to start a new spring project as we have to currently do everything from scratch. Spring Boot is the solution to this problem**. Spring boot has been built on top of existing spring framework. Using spring boot we avoid all the boilerplate code and configurations that we had to do previously. Spring boot thus helps us use the existing Spring functionalities more robustly and with minimum efforts.
More details and miscellaneous examples (/spring/sprboot)
Spring Boot Interview (/spring/SpringBootInterviewQuestions)
Q: What are Microservices ?
A: Microservices is a variant of the service-oriented architecture (SOA) architectural style that structures an application as a collection of loosely coupled services. In a microservices architecture, services should be fine-grained and the protocols should be lightweight. The benefit of decomposing an application into different smaller services is that it improves modularity and makes the application easier to understand, develop and test. It also parallelizes development by enabling small autonomous teams to develop, deploy and scale their respective services independently. It also allows the architecture of an individual service to emerge through continuous refactoring. Microservices-based architectures enable continuous delivery and deployment.

Q: What is Spring Cloud ?
A: Spring Cloud Stream App Starters are Spring Boot based Spring Integration applications that provide integration with external systems. Spring Cloud Task. A short-lived microservices framework to quickly build applications that perform finite amounts of data processing.
Spring Cloud (/spring/springcloud)

Q: What are the advantages of using Spring Cloud ?
A: When developing distributed microservices with Spring Boot we face the following issues-

- **Complexity associated with distributed systems-**
  This overhead includes network issues, Latency overhead, Bandwidth issues, security issues.
- **Service Discovery-**
  Service discovery tools manage how processes and services in a cluster can find and talk to one another. It involves a directory of services, registering services in that directory, and then being able to lookup and connect to services in that directory.
- **Redundancy-**
  Redundancy issues in distributed systems.
- **Loadbalancing-**
  Load balancing improves the distribution of workloads across multiple computing resources, such as computers, a computer cluster, network links, central processing units, or disk drives.
- **Performance issues-**
  Performance issues due to various operational overheads.
- **Deployment complexities-**
  Requirement of Devops skills.

Q: How will you monitor multiple microservices for various indicators like health?
A: Spring Boot provides actuator endpoints to monitor metrics of individual microservices. These endpoints are very helpful for getting information about applications like if they are up, if their components like database etc are working good. But a major drawback or difficulty about using actuator enpoints is that we have to individually hit the enpoints for applications to know their status or health. Imagine microservices involving 50 applications, the admin will have to hit the actuator endpoints of all 50 applications. To help us deal with this situation, we will be using open source project located at https://github.com/codecentric/spring-boot-admin (https://github.com/codecentric/spring-boot-admin). Built on top of Spring Boot Actuator, it provides a web UI to enable us visualize the metrics of multiple applications.
Spring Boot Admin Example (/spring/boot-admin)

Q: What does one mean by Service Registration and Discovery ? How is it implemented in Spring Cloud
A: When we start a project, we usally have all the configurations in the properties file. As more and more services are developed and deployed, adding and modifying these properties become more complex. Some services might go down, while some the location might change. This manual changing of properties may create issues.
Eureka Service Registration and Discovery helps in such scenarios. As all services are registered to the Eureka server and lookup done by calling the Eureka Server, any change in service locations need not be handled and is taken care of
Microservice Registration and Discovery with Spring cloud using Netflix Eureka.
(/spring/spring_eurekaregister)

Q: What does one mean by Load Balancing ? How is it implemented in Spring Cloud
A: In computing, load balancing improves the distribution of workloads across multiple computing resources, such as computers, a computer cluster, network links, central processing units, or disk drives. Load balancing aims to optimize resource use, maximize throughput, minimize response time, and avoid overload of any single resource. Using multiple components with load balancing instead of a single component may increase reliability and availability through redundancy. Load balancing usually involves dedicated software or hardware, such as a multilayer switch or a Domain Name System server process.
In SpringCloud this can be implemented using Netflix Ribbon.
Spring Cloud- Netflix Eureka + Ribbon Simple Example (/spring/spring_ribbon)

Q: How to achieve server side load balancing using Spring Cloud?
A: Server side load balancingcan be achieved using Netflix Zuul.
Zuul is a JVM based router and server side load balancer by Netflix.
It provides a single entry to our system, which allows a browser, mobile app, or other user interface to consume services from multiple hosts without managing cross-origin resource sharing (CORS) and authentication for each one. We can integrate Zuul with other Netflix projects like Hystrix for fault tolerance and Eureka for service discovery, or use it to manage routing rules, filters, and load balancing across your system.
Spring Cloud- Netflix Zuul Example (/spring/spring-cloud-netflix-zuul-tutorial)

Q: In which business scenario to use Netflix Hystrix ?
A: Hystrix is a latency and fault tolerance library designed to isolate points of access to remote systems, services and 3rd party libraries, stop cascading failure and enable resilience in complex distributed systems where failure is inevitable.
Usually for systems developed using Microservices architecture, there are many microservices involved. These microservices collaborate with each other.
Consider the following microservices-

Suppose if the microservice 9 in the above diagram failed, then using the traditional approach we will propagate an exception. But this will still cause the whole system to crash anyways.
This problem gets more complex as the number of microservices increase. The number of microservices can be as high as 1000. This is where hystrix comes into picture-
We will be using two features of Hystrix-

- Fallback method
- Circuit Breaker

Spring Cloud- Netflix Eureka + Ribbon + Hystrix Simple Example (/spring/spring_hystrix)

Q: What is Spring Cloud Bus? Need for it?
A: Consider the scenario that we have multiple applications reading the properties using the Spring Cloud Config and the Spring Cloud Config in turn reads these properties from GIT.
Consider the below example where multiple employee producer modules are getting the property for Eureka Registration from Employee Config Module.

What will happen if suppose the eureka registration property in GIT changes to point to another Eureka server. In such a scenario we will have to restart the services to get the updated properties. There is another way of using Actuator Endpoint **/refresh**. But we will have to individually call this url for each of the modules. For example if Employee Producer1 is deployed on port 8080 then call **http://localhost:8080/refresh**. Similarly for Employee Producer2 **http://localhost:8081/refresh** and so on. This is again cumbersome. This is where Spring Cloud Bus comes into picture.

The Spring Cloud Bus provides feature to refresh configurations across multiple instances. So in above example if we refresh for Employee Producer1, then it will automatically refresh for all other required modules. This is particularly useful if we are having multiple microservice up and running. This is achieved by connecting all microservices to a single message broker. Whenever an instance is refreshed, this event is subscribed to all the microservices listening to this broker and they also get refreshed. The refresh to any single instance can be made by using the endpoint **/bus/refresh**

Spring Cloud Tutorial - Publish Events Using Spring Cloud Bus (/spring/cloud-stream-bus)

Q: What is Spring Cloud Data Flow? Need for it?

A: Spring Cloud Data Flow is a toolkit to build real-time data integration and data processing pipelines by establishing message flows between Spring Boot applications that could be deployed on top of different runtimes.

Long lived applications require Stream Applications while Short lived applications require Task Applications. In this example we make use of Stream Applications. Previously we had already developed Spring Cloud Stream applications to understand the concept of Spring Cloud Stream Source (/spring/cloud-stream-rabbitmq-1) and Spring Cloud Sink (/spring/cloud-stream-rabbitmq-2) and their benefit.

Pipelines consist of Spring Boot apps, built using the Spring Cloud Stream or Spring Cloud Task microservice frameworks. SCDF can be accessed using the REST API exposed by it or the web UI console.
We can make use of metrics, health checks, and the remote management of each microservice application
Also we can scale stream and batch pipelines without interrupting data flows. With SCDF we build data pipelines for use cases like data ingestion, real-time analytics, and data import and export. SCDF is composed of the following Spring Projects-

Spring Cloud Tutorial - Stream Processing Using Spring Cloud Data Flow (/spring/cloud-data-flow)

Q:What is Docker? How to deploy Spring Boot Microservices to Docker?
A: What is Docker (/devOps/docker)
Deploying Spring Based WAR Application to Docker (/devOps/docker/docker-war)
Deploying Spring Based JAR Application to Docker (/devOps/docker/docker-jar)

Q: How to deploy multiple microservices to docker?
A: Deploying Multiple Spring Boot Microservices using Docker Networking (/devOps/docker/docker-networking)