Spring Cloud Config server:

By using spring cloud config service, we can define all the environments properties, ports and other values in the centralized git Repo and can be access via config server.

To do this we have define the path like below:
spring.cloud.config.server.git.uri`=file:///C:/Users/sreenivasulu.bv/Documents/Spring - Microservice - Master by Sreeni/git-localconfig-repo`

In the same we can configure multiple environments properties files and can be access this info from any of the services with proper configuration.

Once the config server is connected, now how to connect the config server with other services. For the we have renamed the **application. properties** to **bootstrap. properties** and we have to provide the URL of the config sever like below:

```
spring.cloud.config.uri=http://localhost:8888
```

The above set up will fetch us only the default values, if we want to fetch dev or QA we have to set up the active profile we want like below.

```
spring.cloud.config.uri=http://localhost:8888
spring.profiles.active=qa
```

Note: to get the port at which the current application is running

```
    exchangeValue.setPort(Integer.parseInt(environment.
getProperty ("local. server. port"))).
```

RestTemplate()

Rest template is used for calling one service to another service by using the methods available in the rest template.
Name some of the Rest template methods:
RestTemplate().getForEntity(URI, reponseEntityClass.class, urivariables);

GetForObject (): It retrieves an entity using HTTP GET method on the given URL.
exchange (): Executes the HTTP method for the given URI. It returns ResponseEntity. It can communicate using any HTTP method.
headForHeaders (): Retrieves all headers. It uses HTTP HEAD method.
getForEntity (): It retrieves an entity by using HTTP GET method for the given URL. It returns ResponseEntity.
delete (): Deletes the resources at the given URL. It uses HTTP DELETE method.
put(): It creates new resource or update for the given URL using HTTP PUT method.
postForObject(): It creates new resource using HTTP PATCH method and returns an entity.
postForLocation(): It creates new resource using HTTP POST method and returns the location of created new resource.
postForEntity(): It creates news resource using HTTP POST method to the given URI template. It returns ResponseEntity.

Refer below url for more details:
https://howtodoinjava.com/spring-boot2/resttemplate/spring-restful-client-resttemplate-example/

Technologies used in project:

- Java 1.8, Spring boot 2.1.4
- Maven and Hibernate
- **Spring Cloud**
    - **Feign** Rest Client
    - **Ribbon** : load balancing
    - **Eureka**: Naming server
    - **Zuul** : API Gateway Service
    - Spring Cloud Sleuth

Feign:

When we have 2 or more microservices to interact or communicate among themselves, **we use rest te**mplate to communicate between them.

Even though writing the Rest template is not a complex task but it will lead to writing in boiler plate code. To avoid this, we have Feign rest Client.

Dependency for the Feign rest Client is:

```
<dependencies>
        <dependency>
            <groupId>org.sprinframeowrk. cloud</groupid>
            <artifactId> spring-cloud-starter-openfeign</artifactId>
        </dependency>
</ dependencies>
```

Annotation to include in main class:
@EnableFeignClients ("package name"):
Scans for interfaces that declare @FeignClient and configures components in the spring context.
Now we have Feign rest client, then to make it work, we must create a Proxy just like the way we need repository to fetch data from DB.
Create an Interface **CurrencyExchangeProxy** and use the below annotation.

```java
package com.java2cloud.springmicroservices.currencyconversionservice;

import org.springframework.cloud.netflix.ribbon.RibbonClient;
import org.springframework.cloud.openfeign.FeignClient;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;

//@FeignClient(name = "currency-exchange-service")
//@FeignClient(name = "netflix-zuul-api-gateway-server")
//@RibbonClient(name = "currency-exchange-service")
@FeignClient(name = "currency-exchange-service", url =
"localhost:8000")
public interface CurrencyExchangeProxy {
	@GetMapping("/currency-exchange/from/{from}/to/{to}")
	public CurrencyConversionBean retriveExchangeValue(@PathVariable
String from, @PathVariable String to);
}
```

Write the below code in controller:

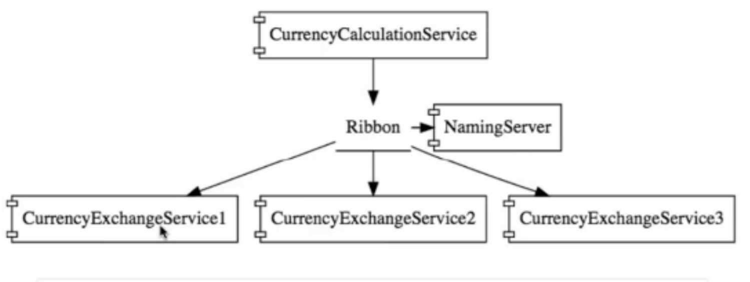
```
@GetMapping("/currency-conversion-
feign/from/{from}/to/{to}/quantity/{quantity}")
	public CurrencyConversionBean convertCurrencyFeign(@PathVariable
String from, @PathVariable String to, @PathVariable BigDecimal
quantity)
{

	CurrencyConversionBean responseBody =
proxy.retriveExchangeValue(from, to);

return new CurrencyConversionBean(responseBody.getId(), from, to,
responseBody.getConversionMultiple(), quantity,
quantity.multiply(responseBody.getConversionMultiple()),
responseBody.getPort());

}
```

Now we can communicate with other rest service using Feign or RestTemplate but we are hard coding the port of the request service to avoid that we have to make use of Ribbon.

Ribbon:

```
@FeignClient (name = "currency-exchange-service", url =
"localhost:8000")
```

From the above code, we have hard coded the url in FeignClient to communicate with currency exchange service, to avoid this and to make random calls to available instances of currency exchange service, that's where Ribbon comes into picture.



Ribbon Load Balancing

We are using Feign to call the Currency Exchange Service. Ribbon can make use of the Feign configuration that we have already done and help us distribute the calls between different instances of the Currency Exchange Service.

What is the dependency for RIBBON ?

```
<dependencies>
        <dependency>
                <groupId>org.springframework.cloud</groupId>
                <artifactId>spring-cloud-starter-netflix-ribbon</artifactId>
        </dependency>
</dependencies>
```

After adding the dependency, the next thing that we'd need to do is to enable Ribbon on the **proxy**. So, I would add in an annotation,

@RibbonClient (name=" currency-exchange-service") (name of the application that we would want to talk to).

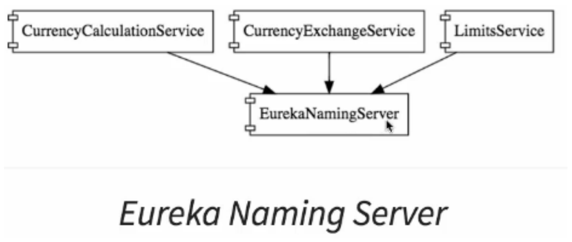
Now the proxy looks like the below:

```
//@FeignClient(name = "currency-exchange-service", url =
"localhost:8000")
@RibbonClient(name = "currency-exchange-service")
public interface CurrencyExchangeProxy {

      //@GetMapping("/currency-exchange/from/{from}/to/{to}")
      @GetMapping("/currency-exchange-service/currency-
exchange/from/{from}/to/{to}")
      public CurrencyConversionBean retriveExchangeValue(@PathVariable
String from, @PathVariable String to);
}
```

We would not want to talk to one instance, but we would want to distribute the load between multiple instances of the Currency Exchange Service. So, what we'll do is, we'll configure it in the application. properties (Currency Conversion Service).

```
currency-exchange-
service.ribbon.listOfServers=http://localhost:8000,http://localhost:80
01
```

Let's assume that I would start up another instance of the currency exchange service let's call it currency exchange service 3 at http://localhost:8002 will ribbon be able to distribute the load ?
Think about it, thing is if I would want ribbon to distribute the load to the new server, I would need to add it to the configuration. So, I would need to change my application configuration whenever a new server is created. And that's not good. What I would want to be able to do is based on the load I would want to be able to increase and decrease the number of instances of these services.
To achieve this, we have to user **Eureka naming Service.**



Eureka naming service:

2 uses of Eureka server is:

a. Registering the services when they start up: - Service registration
b. Find the location of the service: - Service Discovery

Steps to use Eureka naming server. 4 steps.

1. The first thing is we would want to create a component (Service) for the Eureka naming server.
   - Open the Netflix Eureka naming server application class to enable the Eureka server you would need to add in annotation **@EnableEurekaServer**
   - Open Application.properties

```
✓ spring.application.name=netflix-eureka-naming-server
✓ server.port=8761
✓ eureka.client.register-with-eureka=false
✓ eureka.client.fetch-registry=false
```

2. The second thing is we would need to update the currency calculation service to connect the Eureka naming server.

3. The third thing that's we would want to connect the currency exchange application also to talk to the Eureka naming server.
4. The last thing we would do is configure ribbon. Ribbon is currently installed in the currency calculation service. Once the currency exchange service instances register with the naming server. We would then use the ribbon to find those details from the naming server. That would be the last step before we would be able to use the naming.

After setup the Eureka Naming Server. we will connect the Currency Calculation Service and the Currency Exchange Service to the Eureka Naming.

1. Add below dependency in both services.

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

2. Add Annotation @EnableDiscoveryClient in each service.
3. Add the below properties in to each of the services to help them register with Eureka.
```
eureka.client.service-url.default-
zone=http://localhost:8761/eureka
```

Ok Now after configuring all the above steps now all services can communicate with each other by referring to naming server.

So We configure the list of services all that you would need to do to enable ribbon to talk to the naming server using this name is to disable this because in that application we have already configured Eureka in the currency conversion service in the previous step. We already configured the URL for Eureka, so it already knows about Eureka. So, all that I would need to do is disable the list of servers and this exchange service would be starting

API gate Ways: Zuul

Until now we have created 3 simple micro services CurrencyCalculationService, CurrencyExchangeService and the limits.
Typically, in micro services architecture, you'd have 100's of my services that are talking to each other and there are common features that we would need to implement for all these micro services. You would want to make sure that every call to every micro service is authenticated.
A check is done on the authorization whether the user has the right permissions. You also want to implement things like rate limit for a specific client. You would only want to allow certain number of calls per hour or per day they might be limits like that present you'd want all these micro services to be fault tolerant if there is a service that I am depended on and it's not up I should be able to give some default response back and in a typical micro service environment they should also be some kind of service aggregation that should be provided.
Let's say there is an external consumer who wants to call 15 different services as part of one process. It's better to aggregate those 15 services and provide one service call for the external consumer. So, these are kinds of features that are common across all the micro services and these are features which are implemented at the level of API gateways. Instead of allowing micro services to call each other directly what we will do is we would make all the calls go through an API gateway and the API gateway would take care of providing common features like:

- ✓ ***Authentication,***
- ✓ ***Making sure that all service calls are logged,***
- ✓ ***Making sure that these services are Fault Tolerance.***

The API gateways can also provide aggregation services around multiple services, because all calls get routed through the API gateways it a gateway. Also, as a great place for debugging as well as doing analytics.

There are 3 steps to implement the Zuul API Gate Way.

1. Number one is create a component for it (Service creation). So, we would want to create a component for these Zuul API gateways server.

2. The second thing is to decide what should it do when it intercepts a request. So, when a currency calculation service is called to currency

exchange services, an intercepted request should do what, this Zuul API gateway do you need to implement that.

3. The third part is to make sure that all the important requests are configured to pass through. The Zuul API gateway.

Add the below annotations in to Application class of Zuul Api Service

```
@EnableZullProxy
@EnableDiscoveryClient
```

Add the below into properties file

```
spring.application.name=netflix-zuul-api-gateway-server
server.port=8765
eureka.client.service-url.default-zone=http://localhost:8761/eureka
```

Some of the common functionality which is typically implemented in the API gateway:

- ✓ Authentication
- ✓ Authorization rate limits.
- ✓ Service Aggregation
- ✓ Fault Tolerance
- ✓ logging

```
@Component
public class ZuulloggingFilter extends ZuulFilter {

     private Logger logger=LoggerFactory.getLogger(this.getClass());

     @Override
     public boolean shouldFilter() {
          return true.
     }

     @Override
     public Object run() {
          HttpServletRequest request =
     RequestContext.getCurrentContext().getRequest();

            logger.info("Through Zull API gate Way");
                 logger.info("request -> {} request uri-> {}",
     request, request.getRequestURI());
```

```
        return null;
    }

    @Override
    public String filterType() {
        return "pre";
    }

    @Override
    public int filterOrder() {
        return 1;
    }

}
```

Now we would want to implement some functionality in there. What we'll do in this step is we would add some logging to the Zuul API gateway. So, what we'll do is any request that comes through the gateway. What we'll log it. Let's implement that during this step.

Create a new Java class **ZuulloggingFilte, extends ZuulFilter**. Which implements abstract method which are present in this **Zuul filter** which we would need to implement.

And there are four important methods which are present in here should be

1. **Filter** : should this filter be executed or not. You can implement business logic so you can check the request and see certain things and decide if you would want to execute the filter or not. For now, we would want to execute this filter for every request. So, I will return true for should.
2. **Run** : So, what I would want to print is some of the details of the request right. So, I would say

   ```
   logger.info("Through Zull API gate Way");
   logger.info("request -> {} request uri-> {}",
   request, request.getRequestURI());
   ```

   So, if you want log other information feel free to do so and keeping it very simple and I'm just logging in. Couple of things from the request.
3. **Filter type**: Here is filter type the filter type is to when should the filtering. Should filter be happening before the request is executed or after the request has executed. Or do you want to filter only error

requests that have caused an exception to happen in this specific example. What we want to do is we would want to intercept all the requests before execution so I'll say pre filter type we would want to this pre.

4. **Filter order** : If you have multiple filters so let's say you have a Zuul logging filter, Zuul security filter and a lot of other filters you can set a priority order between them. So, you can give a priority order of one for these two for the other one and three for the other one and so on. So, return a one down here. So, the filter order we are setting for this one.

We have 2 services currency conversion and Exchange and if we want these services to execute through this Zuul API gateway.
So, what do we want to do, if we want to execute this request through Zuul API gateway ?
If a consumer directly calls any service by URL the request will not go through this Zuul gateway. So, how do we make the request to go through this Zuul API gateway.
The port at which we configure the Zuul API gate ways is **8765**.
So, the URL for invoking a request through the API gateway would be **http://localhost: 8765**. That's the URL of the API gateway 8765 that's the port.
The next thing that you would need to append in here is the application name. And after that would go your URI of the service.

The complete URL to execute the request through API gate way is:

http://localhost:8765/{application-name}/{uri}
Example: http://localhost:8765/currency-exchange-service/currency-exchange/from/EUR/to/INR

```
//@FeignClient(name = "currency-exchange-service", url =
"localhost:8000")
//@FeignClient(name = "currency-exchange-service")
@FeignClient(name = "netflix-zuul-api-gateway-server")
@RibbonClient(name = "currency-exchange-service")
public interface CurrencyExchangeProxy {
    //@GetMapping("/currency-exchange/from/{from}/to/{to}")
    @GetMapping("/currency-exchange-service/currency-
exchange/from/{from}/to/{to}")
    public CurrencyConversionBean
retriveExchangeValue(@PathVariable String from,
@PathVariable String to);
}
```

So now it would go through the API Server. However, there is another thing that we would need to change. The thing is URI no longer `currency-exchange/from/{from}/to/{to}`.

There is something else that needs to be appended to this. So, you need to append the name of the application which is exposing that service so would need to append in `currency-exchange-service/currency-exchange/from/{from}/to/{to}` URI.

```
http://localhost:8765/ currency-conversion-service
/currency-conversion/from/EUR/to/INR/quantity/200
```

```
currency-conversion/from/{from}/to/{to}/quantity/{quantity}
```

Distributed tracing with Sleuth and ZipKin:
Sleuth:

One of the most important things with implementing a micro services architecture is we would need to have something called distributed tracing. I would want one place where I can go and see what happened with a specific request. I would want to have one single centralized location where I can see the complete chain of what happened with a specific request.

So, the first thing that we'd need to do is to decide where all we would want to use Spring Cloud Sleuth.

What Spring Cloud Sleuth does do?

- It would add unique ID to a request so that you can trace it across multiple components.

- So, let's get started with the API Gateway server.

There are two simple steps in adding Spring Cloud Sleuth to our services to any component.

- Adding a dependency to the pom.xml.

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

Now we have added this in and the only thing that I would want to add in is, to tell what all requests I would want to intercept.

To achieve the above add the below code in to all the components:

```
@Bean
     public Sampler defaultSampler() {
          return  Sampler.ALWAYS_SAMPLE;
     }
```

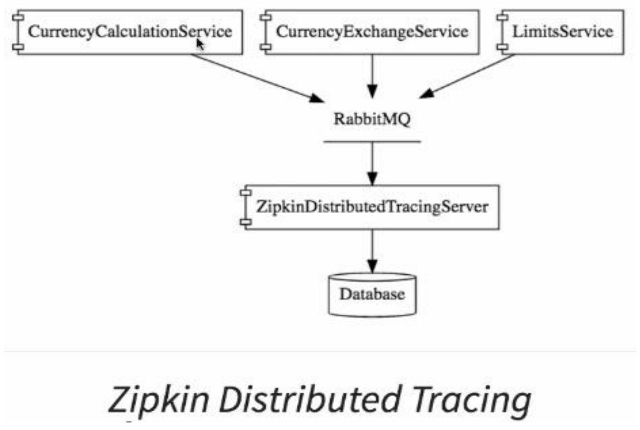
Later add logger into all services and observer the logs with unique ID for that request.

2020-01-05 16:03:24.798 INFO [netflix-zuul-api-gateway-server,**865abd58a151370a**,865abd58a151370a,true] 11948 --- [nio-8765-exec-9] o.s.b.n.ZuulloggingFilter   : ....Through Zuul API gate Way................
2020-01-05 16:03:24.798 INFO [netflix-zuul-api-gateway-server,**865abd58a151370a**,865abd58a151370a,true] 11948 --- [nio-8765-exec-9] o.s.b.n.ZuulloggingFilter   : request -> org.springframework.cloud.netflix.zuul.filters.pre.Servlet30RequestWrapper@d715f5f request uri-> /currency-conversion-service/currency-conversion/from/EUR/to/INR/quantity/10

2020-01-05 16:03:24.843 INFO [currency-conversion-service,**865abd58a151370a**,369834dda544cd15,true] 11328 --- [io-8100-

```
exec-10] c.j.s.c.CurrencyConversoinController     : ..................Through Currency
conversion Way................
2020-01-05 16:03:24.843  INFO [currency-conversion-
service,865abd58a151370a,369834dda544cd15,true] 11328 --- [io-8100-
exec-10] c.j.s.c.CurrencyConversoinController     :
com.java2cloud.springmicroservices.currencyconversionservice.CurrencyCon
versionBean@6e146c0e

2020-01-05 16:03:24.841  INFO [currency-exchange-
service,ede7180f60623b6c,ede7180f60623b6c,true] 3864 --- [nio-8000-
exec-4] c.i.m.c.CurrencyExchangeController       : ...............Through Currency
Exchange Way................
2020-01-05 16:03:24.841  INFO [currency-exchange-
service,ede7180f60623b6c,ede7180f60623b6c,true] 3864 --- [nio-8000-
exec-4] c.i.m.c.CurrencyExchangeController       :
com.in28minutes.microservices.currencyexchangeservice.ExchangeValue@2e
5f58b6
```

ZipKin Distributed Tracing Server:


CurrencyCalculationService
CurrencyExchangeService
LimitsService
RabbitMQ
ZipkinDistributedTracingServer
Database
Zipkin Distributed Tracing

The currency exchange, Currency conversion and zuul API Gateway we saw the same unique ID was being assigned to each request. And I can use that I.D. to trace the request across the logs of these multiple services.

However, the challenge we face is if you want to place that request then we must go and check the logs of the individual applications. One of the solutions to this problem is something called a **centralized log**. What we would need to do is to centralize all the log from all the Micro services.

How to achieve the **centralized log in microservices?**

There are wide variety of solutions for centralized logging. There are things like the **ELK stack,** which is the **Elastics search, Logstash** and **kibana** where all the log from these micro services is consolidated at one place and using elastics search see would be able to search through all the logs.

We would use something called **ZipKin distributed tracing server** to consolidate and get a consolidated view of what is happening across all the Micro services.

Now the question is how to get the log messages from the currency calculation service, currency Exchange and zuul Api to this ZipKin distributed tracing server.

- Using **RabbitMQ**

Till now what is happening: For example, if I'm calling the **currency calculation service** the currency calculation service in turn would invoke the **currency exchange service** and what would happen is the API gateway would be intercepting that So, the request would go through the currency calculation service the **API gateway** and the **currency exchange service,** the **spring cloud sleuth**. **sleuth** will be assigning a unique ID for the request which I can use it to trace the request across multiple applications or multiple components.

All the micro services would put all their log messages on the queue using **rabbitMQ** and I would be able to track them on the **ZipKin distributed tracing server**.

RabbitMQ:
After installing Rabbit MQ and Erlang:

Points to be followed:

- To install Rabbit MQ we got to install erlang first and later RabbitMQ.
- In windows download ZIPKIN sever and run in the background.
- Use the below commands to run the ZipKin.

```
set RABBIT_URI=amqp://localhost
java -jar zipkin-server-2.7.0-exec.jar
```

Restart all services and hit the port : http://localhost:9411/zipkin
Hit any service to see the results in the ZipKin server.

Spring Cloud Bus:

Fault Tolerance with Hystrix:

When we have several smaller micro services interacting with each other It is possible that a couple of my services might be down somewhere in the entire architecture. If any of these micro services are down, then they can pull down the entire chain of micro services that are invoking them or that are dependent on them and that's one of the risks of micro services architecture have.

For example, the currency calculation service depends on the currency exchange service and the currency exchange service depends on the limits service, in that case if the limit service is down then both these services also will not be available and that's not good situation to be in if one of the micro services goes down all the other micro services also go down along with it. your micro services must be better than that and that's where fault tolerance of your micro services comes into picture.

One of the things that you should always consider when a micro service is being created is what if some functionality does not work as expected. Let's say the limits service depends on some of the service and that service is not available then can limit service providers default graceful behavior the functionality might not be perfect but at least the limits service does not go down it at least gives a good enough response back to the currency exchange

service so that it can go ahead and do the work it needs to do. It would prevent the entire chain from going down. Hystrix framework helps us to build fault tolerant micro services.

How do we do that. Let's see that in these steps:

1.
```
      <dependency>
             <groupId>org.springframework. cloud</groupId>
             <artifactId>spring-cloud-starter-hystrix</artifactId>
             <version>1.4.7. RELEASE</version>
      </dependency>
```

2. Once adding the dependency on hystrix we must go to the limits service application and enable it as well.

   `@EnableHystrix` and imported in this would enable hystrix fault tolerant.

3. In all the controllers and the controller methods what we can do is we can add an annotation called:

   `@HystrixCommand(fallbackMethod="fallbackRetrieveConfiguration")`

```
@GetMapping("/fault-tolerance-example")
@HystrixCommand(fallbackMethod="fallbackRetrieveConfiguration")
public LimitConfiguration retriveConfiguration() {
      throw new RuntimeException("No available");
      }

public LimitConfiguration fallbackRetrieveConfiguration() {
      return new LimitConfiguration(999,9);
      }
```

1. What Is Spring Cloud?

Spring Cloud is a Platform, which provides tools for developers to quickly build some of the common patterns in distributed systems (e.g. configuration management, service discovery, circuit breakers, intelligent routing, leadership election, distributed sessions, cluster state).

For typical use cases, Spring Cloud provides the out of the box experiences and a set of extensive features mentioned below:

- Versioned and distributed configuration.
- Discovery of service registration.
- Service to service calls.
- Routing.
- Circuit breakers and load balancing.
- Cluster state and leadership election.
- Global locks and distributed messaging.

2. What Is Spring Boot?

Spring Boot is an open source Java-based framework used to create a micro Service. It is developed by Pivotal Team and is used to build stand-alone and production ready spring applications.

3. How Do You Override a Spring Boot Project's Default Properties?

This can be done by specifying the properties in the application.properties file. For example, in Spring MVC applications, you have to specify the suffix and prefix. This can be done by entering the properties mentioned below in the application.properties file.

- For suffix – `spring.mvc.view.suffix: .jsp`
- For prefix – `spring.mvc.view.prefix: /WEB-INF/`

4. Role of Actuator in Spring Boot

It is one of the most important features, which helps you to access the current state of an application that is running in a production environment. There are multiple metrics which can be used to check the current state. They also provide endpoints for RESTful web services which can be simply used to check the different metrics.

5. How Is Spring Security Implemented in a Spring Boot Application?

Minimal configuration is needed for implementation. All you need to do is add the`spring-boot-starter-security`starter in the pom.xml file. You will also need to create a Spring config class that will override the required method while extending the `WebSecurityConfigurerAdapter` to achieve security in the application. Here is some example code:

```
package
com.gkatzioura.security.securityendpoints.config;
import
org.springframework.context.annotation.Configuration;
import
org.springframework.security.config.annotation.web.buil
ders.HttpSecurity;
import
org.springframework.security.config.annotation.web.conf
iguration.WebSecurityConfigurerAdapter;
@Configuration
public class SecurityConfig extends
WebSecurityConfigurerAdapter {
@Override
protected void configure(HttpSecurity http) throws
Exception {
http.authorizeRequests()
.antMatchers("/welcome").permitAll()
.anyRequest().authenticated()
.and()
.formLogin()
.permitAll()
.and()
.logout()
.permitAll();
}
}
```

6. Which Embedded Containers Are Supported by Spring Boot?

Whenever you are creating a Java application, deployment can be done via two methods:

- Using an application container that is external.
- Embedding the container inside your jar file.

Spring Boot contains Jetty, Tomcat, and Undertow servers, all of which are embedded.

- **Jetty** – Used in a wide number of projects, Eclipse Jetty can be embedded in framework, application servers, tools, and clusters.
- **Tomcat** – Apache Tomcat is an open source JavaServer Pages implementation which works well with embedded systems.
- **Undertow** – A flexible and prominent web server that uses small single handlers to develop a web server.

7. What Do You Mean by End-To-End Testing of Microservices?

End-to-end testing validates all the processes in the workflow to check if everything is working as expected. It also ensures that the system works in a unified manner, thereby satisfying the business requirement.

8. What Is Semantic Monitoring?

It combines monitoring of the entire application along with automated tests. The primary benefit of Semantic Monitoring is to find out the factors which are more profitable to your business.
Semantic monitoring along with service layer monitoring approaches monitoring of microservices from a business point of view. Once an issue is detected, they allow faster isolation and bug triaging, thereby reducing the main time required to repair. It triages the service layer and transaction layer to figure out the transactions affected by availability or poor performance.

9. How Can You Set Up Service Discovery?

There are multiple ways to set up service discovery. I'll choose the one that I think to be most efficient, Eureka by Netflix. It is a hassle-free procedure that does not weigh much on the application. Plus, it supports numerous types of web applications.
Eureka configuration involves two steps – client configuration and server configuration.
Client configuration can be done easily by using the property files. In the classpath, Eureka searches for a **eureka-client.properties** file. It also searches for overrides caused by the environment in property files which are environment specific.

For server configuration, you must configure the client first. Once that is done, the server fires up a client which is used to find other servers. The Eureka server, by default, uses the Client configuration to find the peer server.

10. Why Would You opt for Microservices Architecture?

This is a very common microservices interview question which you should be ready for! There are plenty of pros that are offered by a microservices architecture. Here are a few of them:

- Microservices can adapt easily to other frameworks or technologies.
- Failure of a single process does not affect the entire system.
- Provides support to big enterprises as well as small teams.
- Can be deployed independently and in relatively less time.

11. Why Would You Need Reports and Dashboards in Microservices?

Reports and dashboards are mainly used to monitor and upkeep microservices. There are multiple tools that help to serve this purpose. Reports and dashboards can be used to:

- Find out which microservices expose what resources.
- Find out the services which are impacted whenever changes in a component occur.
- Provide an easy point which can be accessed whenever documentation is required.
- Versions of the components which are deployed.
- To obtain a sense of maturity and compliance from the components.

12. Why Do People Hesitate to Use Microservices?

I have seen many devs fumble over this question. After all, they're getting asked this question when interviewing for a microservices architect role, so acknowledging its cons can be a little tricky. Here are some good answers:

- **They require heavy investment** – Microservices demand a great deal of collaboration. Since your teams are working independently, they should be able to synchronize well at all times.
- **They need heavy architecture set up** – The system is distributed, the architecture is heavily involved.
- **They need excessive planning for handling operations overhead** – You need to be ready for operations overhead if you are planning to use a microservices architecture.

- **They have autonomous staff selection** – Skilled professionals are needed who can support microservices that are distributed heterogeneously.

13. How Does PACT Work?

PACT is an open source tool. It helps in testing the interactions between consumers and service providers. However, it is not included in the contract, increasing the reliability of the application. The consumer service developer starts by writing a test which defines a mode of interaction with the service provider. The test includes the provider's state, the request body, and the response that is expected. Based on this, PACT creates a stub against which the test is executed. The output is stored in a JSON file.

14. Define Domain Driven Design

The main focus is on the core domain logic. Complex designs are detected based on the domain's model. This involves regular collaboration with domain experts to resolve issues related to the domain and improve the model of the application. While answering this microservices interview question, you will also need to mention the core fundamentals of DDD. They are:

- DDD focuses mostly on domain logic and the domain itself.
- Complex designs are completely based on the domain's model.
- To improve the design of the model and fix any emerging issues, DDD constantly works in collaboration with domain experts.

15. What Are Coupling and Cohesion?

Coupling can be the measurement of strength between the dependencies of a component. A good microservices application design always consists of low coupling and high cohesion.

Interviewers will often ask about cohesion. It is also another measurement unit. More like a degree to which the elements inside a module remain bonded together.

It is imperative to keep in mind that an important key to designing microservices is a composition of low coupling along with high cohesion. When loosely coupled, a service knows very little about other services. This keeps the services intact. In high cohesion, it becomes possible to keep all the related logic in a service. Otherwise, the services will try to communicate with each other, impacting the overall performance.

16. What Is OAuth?

Open Authorization Protocol, otherwise known as OAuth, helps to access client applications using third-party protocols like Facebook, GitHub, etc., via HTTP. You can also share resources between different sites without the requirement of credentials.
OAuth allows the account information of the end user to be used by a third-party like Facebook while keeping it secure (without using or exposing the user's password). It acts more like an intermediary on the user's behalf while providing a token to the server for accessing the required information.

17. Why Do We Need Containers for Microservices?

To manage a microservice-based application, containers are the easiest alternative. It helps the user to individually deploy and develop. You can also use Docker to encapsulate microservices in the image of a container. Without any additional dependencies or effort, microservices can use these elements.

18. What Are the Ways to Access RESTful Microservices?

Another one of the frequently asked microservices interview questions is how to access RESTful microservices? You can do that via two methods:

- Using a REST template that is load balanced.
- Using multiple microservices.

19. What Are Some Major Roadblocks for Microservices Testing?

Talking about the cons, here is another one of the microservices interview questions you may be ready for, will be around the challenges faced while testing microservices.

- Testers should have a thorough understanding of all the inbound and outbound processes before they start writing the test cases for integration testing.
- When independent teams are working on different functionalities, collaboration can prove to be quite a struggling task. It can be tough to find an idle time-window to perform a complete round of regression testing.
- With an increasing number of microservices, the complexity of the system also increases.

- During the transition from monolithic architecture, testers must ensure that there is no disruption between the internal communication among the components.

20. Common Mistakes Made While Transitioning to Microservices?

Not only on development, but mistakes also often occur on the process side. And any experienced interviewer will have this in the queue for microservices interview questions. Some of the common mistakes are:

- Often the developer fails to outline the current challenges.
- Rewriting the programs that are already existing.
- Responsibilities, timeline, and boundaries not clearly defined.
- Failing to implement and figure out the scope of automation from the very beginning.

21. What Are the Fundamentals of Microservices Design?

This is probably one of the most frequently asked microservices interview questions. Here is what you need to keep in mind while answering to it:

- Define a scope.
- Combine loose coupling with high cohesion.
- Create a unique service which will act as an identifying source, much like a unique key in a database table.
- Creating the correct API and taking special care during integration.
- Restrict access to data and limit it to the required level.
- Maintain a smooth flow between requests and response.
- Automate most processes to reduce time complexity.
- Keep the number of tables to a minimum level to reduce space complexity.
- Monitor the architecture constantly and fix any flaw when detected.
- Data stores should be separated for each microservice.
- For each microservice, there should be an isolated build.
- Deploy microservices into containers.
- Servers should be treated as stateless.

You can also follow this article explaining 9 Fundamentals to a Successful Microservice Design.

22. Where Do We Use WebMVC Test Annotation?

WebMvcTest is used for unit testing Spring MVC applications. As the name suggests, it focuses entirely on Spring MVC components. For example,
`@WebMvcTest(value = ToTestController.class, secure = false):`
Here, the objective is to only launch `ToTestController`. Until the unit test has been executed, other mappings and controllers will not be launched.

23. What Do You Mean by Bounded Context?

A central pattern is usually seen in domain driven design. Bounded context is the focus of the strategic design section of DDD. It is all about dealing with large teams and models. DDD works with large models by disintegrating them into multiple bounded contexts. While it does that, it also explains the relationship between them explicitly.

24. What Are the Different Types of Two-Factor Authentication?

There are three types of credentials required for performing two-factor authentication.

1. A thing that you know – like password or pin or screen lock pattern.
2. A physical credential that you have – like OTP or phone or an ATM card, in other words, any kind of credential that you have in an external or third-party device.
3. Your physical identity – like voice authentication or biometric security, like a fingerprint or eye scanner.

25. What Is a Client Certificate?

This is a type of digital certificate usually used by client systems for making a request that is authenticated by a remote server. It plays an important role in authentication designs that are mutual and provides strong assurance of the identity of a requester. However, you should have a fully configured backend service for authenticating your client certificate.

26. What Is Conway's Law?

Conway's Law states, "organizations which design systems ... are constrained to produce designs which are copies of the communication structures of these organizations."

The interviewer may ask a counter microservices interview question, like how is Conway's Law related to microservices. Well, some loosely coupled APIs form the architecture of microservices. The structure is well suited to how a small team is implementing components which are autonomous. This architecture makes an organization much more flexible in restructuring its work process.

27. How to Configure Spring Boot Application Logging?

Spring Boot comes with added support for Log4J2, Java Util Logging, and Logback. It is usually pre-configured as console output. They can be configured by only specifying logging.level in the application.properties file.
`logging.level.spring.framework=Debug`

28. How Would You Perform Security Testing on Microservices?

Before answering this microservices interview question, explain to the interviewer that microservices cannot be tested. You will need to test the pieces independently. There are three common procedures:

- **Code scanning** – To ensure that any line of code is bug-free and can be replicated.
- **Flexibility** – The security solution should be flexible so that it can be adjusted as per the requirements of the system.
- **Adaptability** – The security protocols should be flexible and updated to cope up with the new threats by hackers or security breaches.

You can also check out this article explaining the influence of Microservices architecture on security.

29. What Is Idempotence and How Is it Used?

Idempotence refers to a scenario where you perform a task repetitively, but the result remains constant or similar.
Idempotence is mostly used as a data source or a remote service in a way that when it receives more than one set of instructions, it processes only one set of instructions.

Q1. List down the advantages of Microservices Architecture.

Advantages of Microservices Architecture

Advantage	Description
Independent Development	All microservices can be easily developed based on their individual functionality
Independent Deployment	Based on their services, they can be individually deployed in any application
Fault Isolation	Even if one service of the application does not work, the system continues to function
Mixed Technology Stack	Different languages and technologies can be used to build different services of the same application
Granular Scaling	Individual components can scale as per need, there is no need to scale all components together

Q2. What do you know about Microservices?

- **Microservices**, aka ***Microservice Architecture***, is an architectural style that structures an application as a collection of small autonomous services, modeled around a **business domain.**
- In layman terms, you must have seen how bees build their honeycomb by aligning hexagonal wax cells.
- They initially start with a small section using various materials and continue to build a large beehive out of it.
- These cells form a pattern resulting in a strong structure which holds together a section of the beehive.
- Here, each cell is independent of the other, but it is also correlated with the other cells.

- This means that damage to one cell does not damage the other cells, so, bees can reconstruct these cells without impacting the complete beehive.

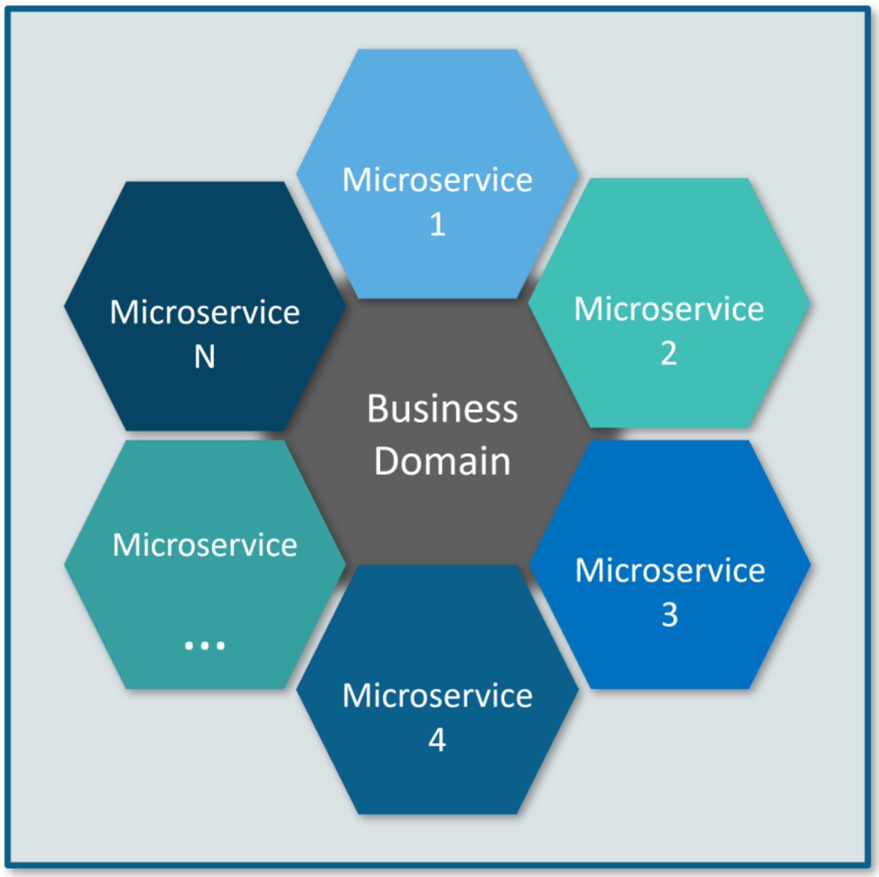


Fig 1: Beehive Representation of Microservices – Microservices Interview Questions

Refer to the above diagram. Here, each hexagonal shape represents an individual service component. Similar to the working of bees, each agile team builds an individual service component with the available frameworks and the chosen technology stack. Just as in a beehive, each service component forms a strong microservice architecture to provide better scalability. Also, issues with each service component can be handled individually by the agile team with no or minimal impact on the entire application.

Q3. What are the features of Microservices?

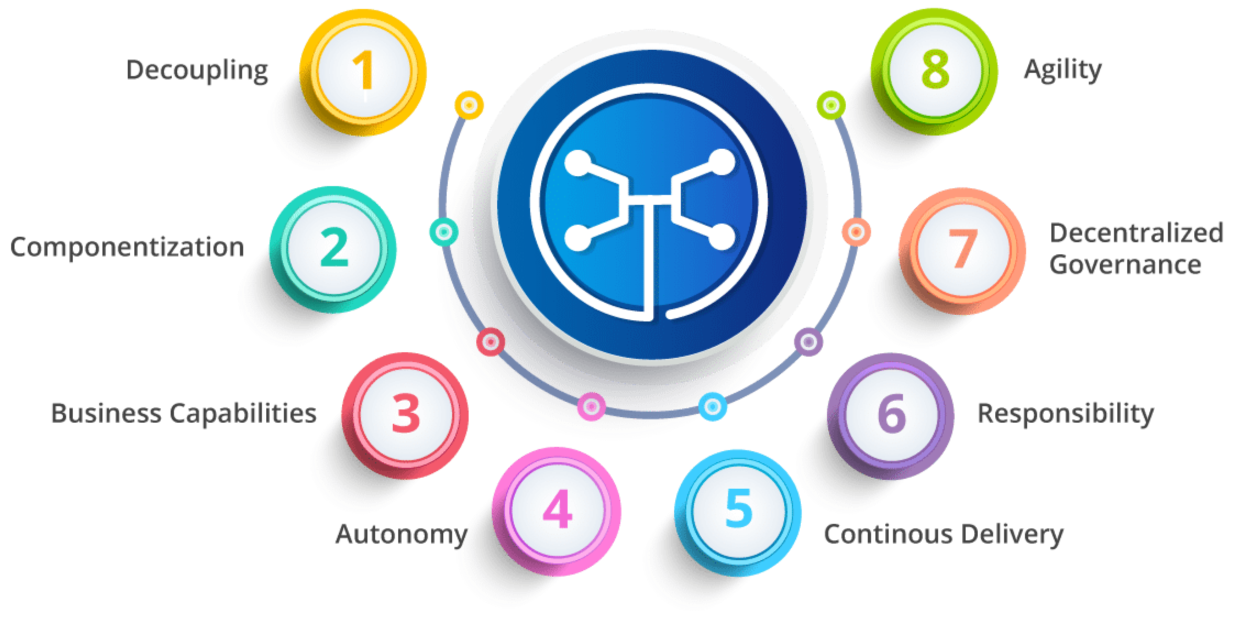


Fig 3: Features of Microservices – Microservices Interview Questions

- **Decoupling** – Services within a system are largely decoupled. So, the application as a whole can be easily built, altered, and scaled
- **Componentization** – Microservices are treated as independent components that can be easily replaced and upgraded
- **Business Capabilities** – Microservices are very simple and focus on a single capability
- **Autonomy** – Developers and teams can work independently of each other, thus increasing speed
- **Continuous Delivery** – Allows frequent releases of software, through systematic automation of software creation, testing, and approval
- **Responsibility** – Microservices do not focus on applications as projects. Instead, they treat applications as products for which they are responsible
- **Decentralized Governance** – The focus is on using the right tool for the right job. That means there is no standardized pattern or any technology pattern. Developers have the freedom to choose the best useful tools to solve their problems
- **Agility** – Microservices support agile development. Any new feature can be quickly developed and discarded again

Q4. What are the best practices to design Microservices?

The following are the best practices to design microservices:

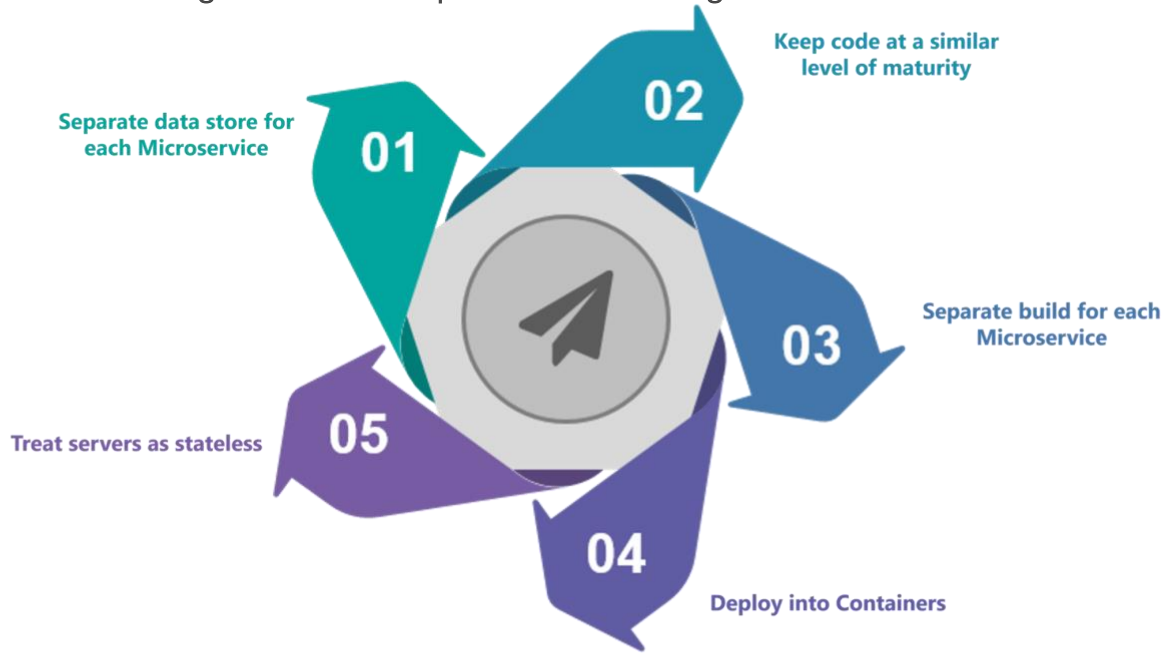

01
Separate data store for each Microservice
02
Keep code at a similar level of maturity
03
Separate build for each Microservice
04
Deploy into Containers
05
Treat servers as stateless

Fig 4: Best Practices to Design Microservices – Microservices Interview Questions

Q5. How does Microservice Architecture work?

A microservice architecture has the following components:

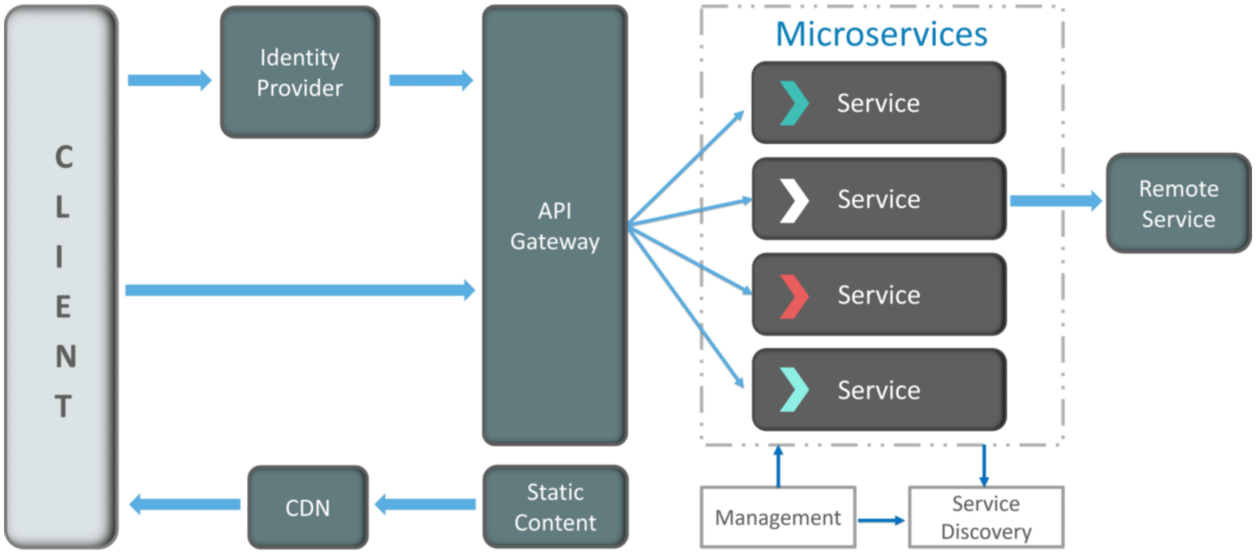


Fig 5: Architecture of Microservices – Microservices Interview Questions

- **Clients** – Different users from various devices send requests.
- **Identity Providers** – Authenticates user or clients identities and issues security tokens.

- **API Gateway** – Handles client requests.
- **Static Content** – Houses all the content of the system.
- **Management** – Balances services on nodes and identifies failures.
- **Service Discovery** – A guide to find the route of communication between microservices.
- **Content Delivery Networks** – Distributed network of proxy servers and their data centers.
- **Remote Service** – Enables the remote access information that resides on a network of IT devices.

Q6. What are the pros and cons of Microservice Architecture?

Pros of Microservice Architecture	Cons of Microservice Architecture
Freedom to use different technologies	Increases troubleshooting challenges
Each microservices focuses on single capability	Increases delay due to remote calls
Supports individual deployable units	Increased efforts for configuration and other operations
Allow frequent software releases	Difficult to maintain transaction safety
Ensures security of each service	Tough to track data across various boundaries
Mulitple services are parallelly developed and deployed	Difficult to code between services

Q7. What is the difference between Monolithic, SOA and Microservices Architecture?

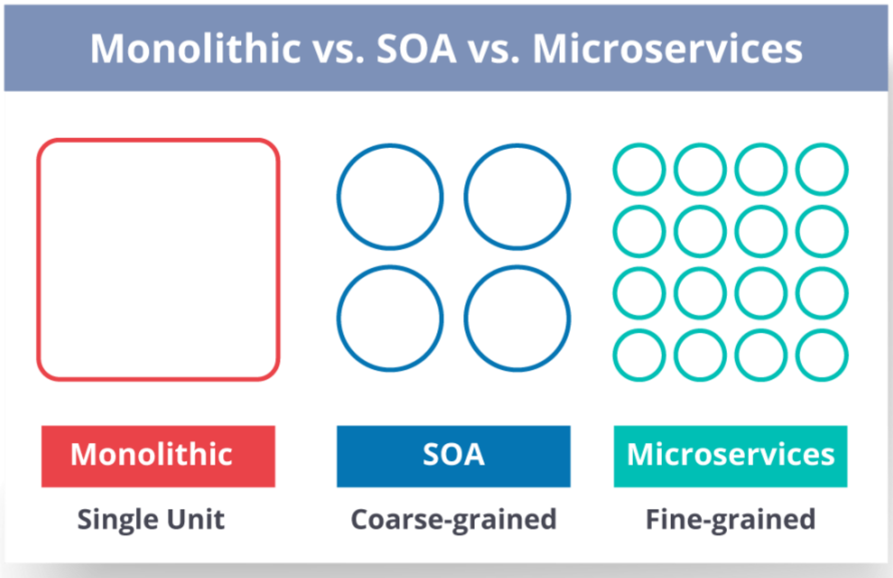


Fig 6: Comparison Between Monolithic SOA & Microservices – Microservices Interview Questions

- **Monolithic Architecture** is similar to a big container wherein all the software components of an application are assembled together and tightly packaged.
- A **Service-Oriented Architecture** is a collection of services which communicate with each other. The communication can involve either simple data passing or it could involve two or more services coordinating some activity.
- **Microservice Architecture** is an architectural style that structures an application as a collection of small autonomous services, modeled around a business domain.

Q8. What are the challenges you face while working Microservice Architectures?

Developing several smaller microservices sounds easy, but the challenges often faced while developing them are as follows.

- **Automate the Components**: Difficult to automate because there are a number of smaller components. So for each component, we have to follow the stages of Build, Deploy and, Monitor.

- **Perceptibility**: Maintaining a large number of components together becomes difficult to deploy, maintain, monitor and identify problems. It requires great perceptibility around all the components.
- **Configuration Management**: Maintaining the configurations for the components across the various environments becomes tough sometimes.
- **Debugging**: Difficult to find out each and every service for an error. It is essential to maintain centralized logging and dashboards to debug problems.

Q9. What are the key differences between SOA and Microservices Architecture?

The key differences between SOA and microservices are as follows:

SOA	Microservices
Follows "**share-as-much-as-possible**" architecture approach	Follows "**share-as-little-as-possible**" architecture approach
Importance is on **business functionality** reuse	Importance is on the concept of "**bounded context**"
They have **common governance** and standards	They focus on **people collaboration** and freedom of other options
Uses **Enterprise Service bus (ESB)** for communication	Simple messaging system
They support **multiple message protocols**	They use **lightweight protocols** such as **HTTP/REST** etc.
Multi-threaded with more overheads to handle I/O	**Single-threaded** usually with the use of Event Loop features for non-locking I/O handling
Maximizes application service reusability	Focuses on **decoupling**
Traditional Relational Databases are more often used	**Modern Relational Databases** are more often used
A systematic change requires modifying the monolith	A systematic change is to create a new service

DevOps / Continuous Delivery is becoming popular, but not yet mainstream	Strong focus on DevOps / Continuous Delivery

Q10. What are the characteristics of Microservices?

You can list down the characteristics of microservices as follows:

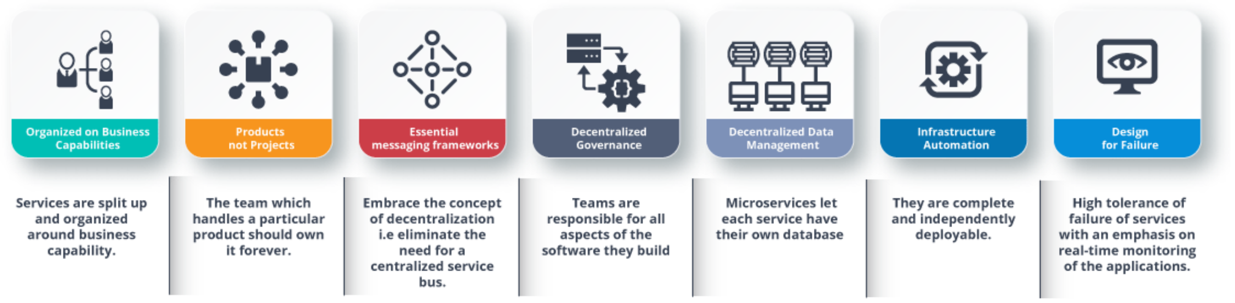

Organized on Business Capabilities
Products not Projects
Essential messaging frameworks
Decentralized Governance
Decentralized Data Management
Infrastructure Automation
Design for Failure
Services are split up and organized around business capability.
The team which handles a particular product should own it forever.
Embrace the concept of decentralization i.e eliminate the need for a centralized service bus.
Teams are responsible for all aspects of the software they build
Microservices let each service have their own database
They are complete and independently deployable.
High tolerance of failure of services with an emphasis on real-time monitoring of the applications.

Fig 7: Characteristics of Microservices – Microservices Interview Questions

Q11. What is Domain Driven Design?

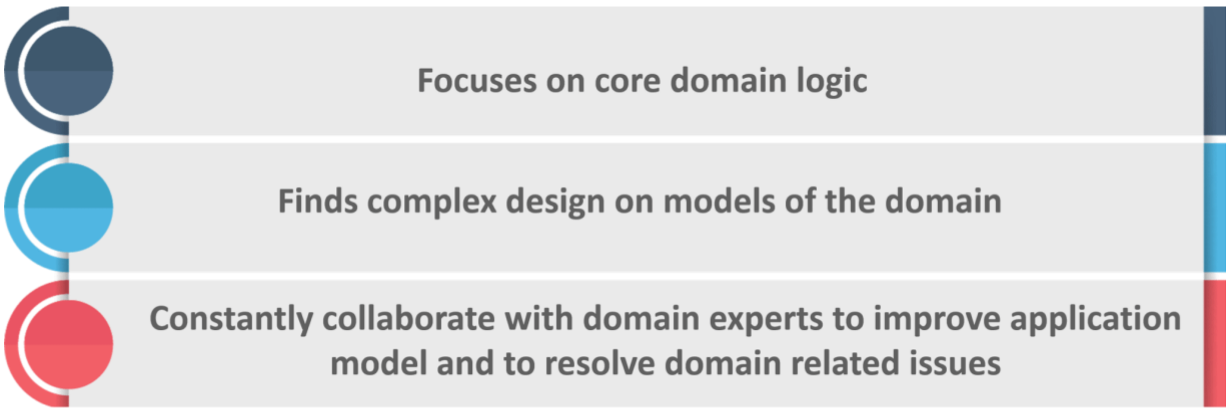

Focuses on core domain logic
Finds complex design on models of the domain
Constantly collaborate with domain experts to improve application model and to resolve domain related issues

Fig 8: Principles of DDD – Microservices Interview Questions

Q12. Why there is a need for Domain Driven Design (DDD)?

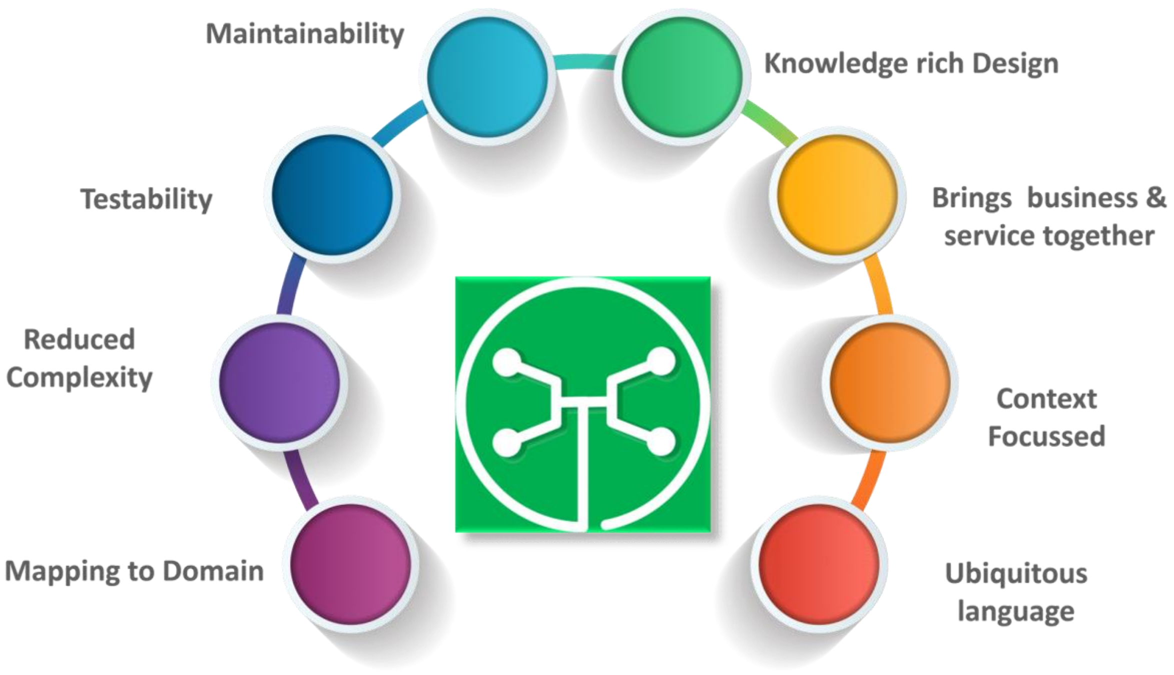


Fig 9: Factors Why we need DDD – Microservices Interview Questions

Q13. What is Ubiquitous language?

If you have to define the **Ubiquitous Language (UL)**, then it is a common language used by developers and users of a specific domain through which the domain can be explained easily.
The ubiquitous language has to be crystal clear so that it brings all the team members on the same page and also translates in such a way that a machine can understand.

Q14. What is Cohesion?

The degree to which the elements inside a module belong together is said to be **cohesion**.

Q15. What is Coupling?

The measure of the strength of the dependencies between components is said to be **coupling**. A good design is always said to have **High Cohesion** and **Low Coupling**.

Subscribe to our youtube channel to get new updates..!

Q16. What is REST/RESTful and what are its uses?

Representational State Transfer (REST)/RESTful web services are an architectural style to help computer systems communicate over the internet. This makes microservices easier to understand and implement.
Microservices can be implemented with or without RESTful APIs, but it's always easier to build loosely coupled microservices using RESTful APIs.

Q17. What do you know about Spring Boot?

It's a knows fact that spring has become more and more complex as new functionalities have been added. If you have to start a new spring project, then you have to add build path or add maven dependencies, configure application server, add spring configuration. So everything has to be done from scratch.
Spring Boot is the solution to this problem. Using spring boot you can avoid all the boilerplate code and configurations. So basically consider yourself as if you're baking a cake spring is like the ingredients that are required to make the cake and spring boot is the complete cake in your hand.

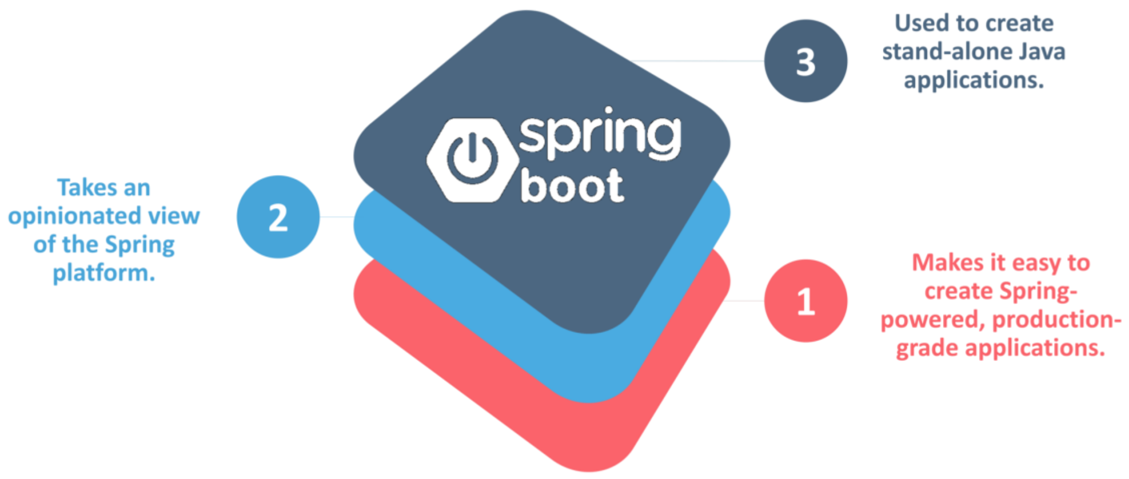

Used to create stand-alone Java applications.
3
spring boot
Takes an opinionated view of the Spring platform.
2
1
Makes it easy to create Spring-powered, production-grade applications.

Fig 10: Factors of Spring Boot – Microservices Interview Questions

Q18. What is an actuator in Spring boot?

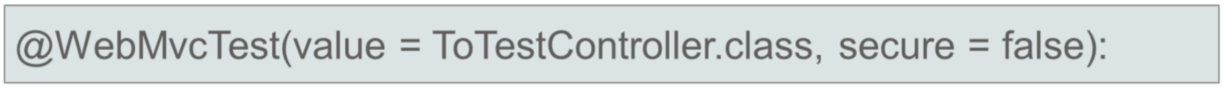
Spring Boot actuator provides restful web services to access the current state of running an application in the production environment. With the help of actuator, you can check various metrics and monitor your application.

Q20. What problems are solved by Spring Cloud?

While developing distributed microservices with Spring Boot we face few issues which are solved by Spring Cloud.

- **The complexity associated with distributed systems –** This includes network issues, Latency overhead, Bandwidth issues, security issues.
- **Ability to handle Service Discovery –** Service discovery allows processes and services in a cluster to find each other and communicate.
- **Solved redundancy issues –** Redundancy issues often occur in distributed systems.
- **Load balancing –** Improves the distribution of workloads across multiple computing resources, such as a computer cluster, network links, central processing units.
- **Reduces performance issues –** Reduces performance issues due to various operational overheads.

Q21. What is the use of WebMvcTest annotation in Spring MVC applications?



```
@WebMvcTest(value = ToTestController.class, secure = false):
```

WebMvcTest annotation is used for unit testing Spring MVC Applications in cases where the test objective is to just focus on Spring MVC Components. In the snapshot shown above, we want to launch only the ToTestController. All other controllers and mappings will not be launched when this unit test is executed.

Q22. Can you give a gist about Rest and Microservices?

REST
Though you can implement microservices in multiple ways, REST over HTTP is a way to implement Microservices. REST is also used in other applications such as web apps, API design, and MVC applications to serve business data.

Microservices
Microservices is an architecture wherein all the components of the system are put into individual components, which can be built, deployed, and scaled individually. There are certain principles and best practices of Microservices that help in building a resilient application.
In a nutshell, you can say that REST is a medium to build Microservices.

Q23. What are different types of Tests for Microservices?

While working with microservices, testing becomes quite complex as there are multiple microservices working together. So, tests are divided into different levels.

- At the **bottom level**, we have **technology-facing tests** like- unit tests and performance tests. These are completely automated.
- At the **middle level**, we have tests for **exploratory testing** like the stress tests and usability tests.
- At the **top level,** we have **acceptance tests** that are few in number. These acceptance tests help stakeholders in understanding and verifying software features.

Q24. What do you understand by Distributed Transaction?

Distributed Transaction is any situation where a single event results in the mutation of two or more separate sources of data which cannot be committed atomically. In the world of microservices, it becomes even more complex as each service is a unit of work and most of the time multiple services have to work together to make a business successful.

Q25. What is an Idempotence and where it is used?

Idempotence is the property of being able to do something twice in such a way that the end result will remain the same i.e. as if it had been done once only.
Usage: Idempotence is used at the remote service, or data source so that, when it receives the instruction more than once, it only processes the instruction once.

Q26. What is Bounded Context?

Bounded Context is a central pattern in Domain-Driven Design. It is the focus of DDD's strategic design section which is all about dealing with large models and teams. DDD deals with large models by dividing them into different Bounded Contexts and being explicit about their inter-relationships.

Fig11: Representation of Two Factor Authentication – Microservices Interview Questions

so suppose a user has to enter only username and password, then that's considered a single-factor authentication.

Q28. What are the types of credentials of Two Factor Authentication?

The three types of credentials are:

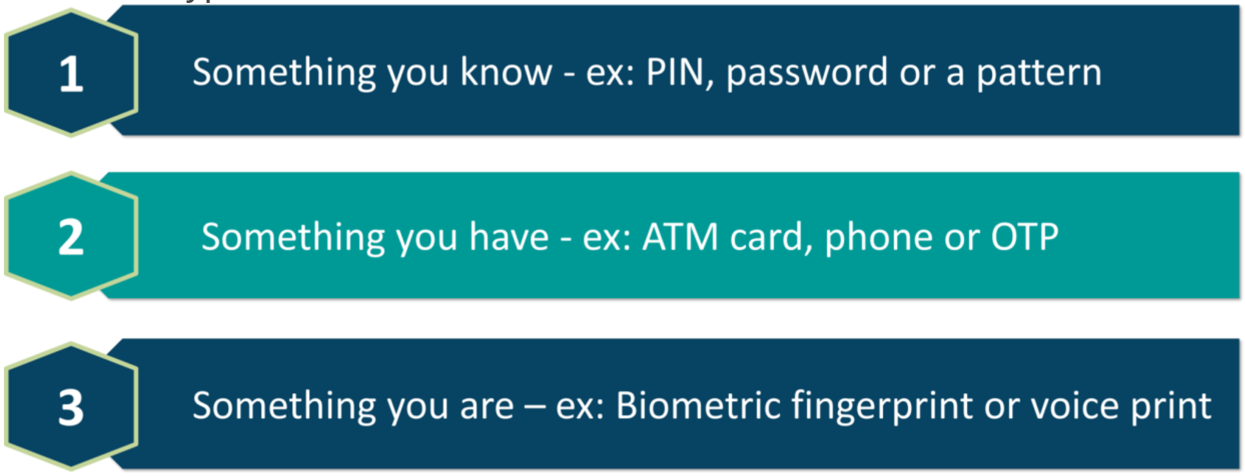

1
Something you know - ex: PIN, password or a pattern
2
Something you have - ex: ATM card, phone or OTP
3
Something you are – ex: Biometric fingerprint or voice print

Fig 12: Types of Credentials of Two Factor Authentication – Microservices Interview Questions

Q29. What are Client certificates?

A type of digital certificate that is used by client systems to make authenticated requests to a remote server is known as the **client certificate**. Client certificates play a very important role in many mutual authentication designs, providing strong assurances of a requester's identity.

Q30. What is the use of PACT in Microservices architecture?

PACT is an open source tool to allow testing interactions between service providers and consumers in isolation against the contract made so that the reliability of Microservices integration increases.

Usage in Microservices:

- Used to implement Consumer Driven Contract in Microservices.
- Tests the consumer-driven contracts between consumer and provider of a Microservice.

Q31. What is OAuth?

OAuth stands for open authorization protocol. This allows accessing the resources of the resource owner by enabling the client applications on HTTP services such as third-party providers Facebook, GitHub, etc. so with this, you can share resources stored on one site with another site without using their credentials.

Q32. What is Conway’s law?

“Any organization that designs a system (defined broadly) will produce a design whose structure is a copy of the organization’s communication structure.” – ***Mel Conway***

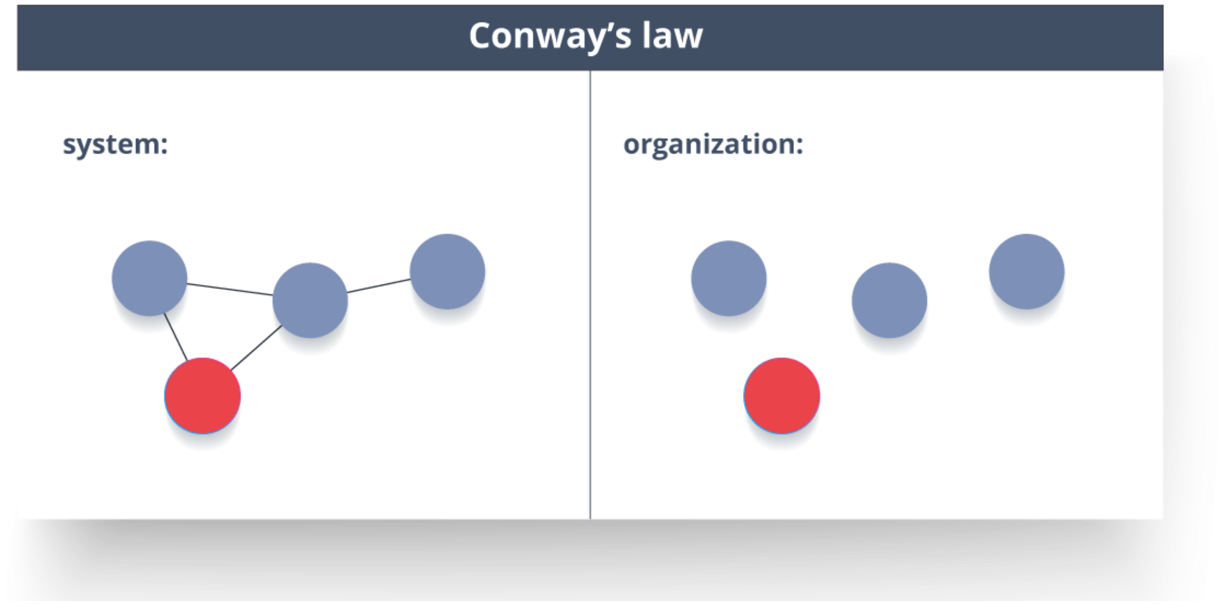


Fig 13: Representation of Conway’s Law – Microservices Interview Questions

This law basically tries to convey the fact that, in order for a software module to function, the complete team should communicate well. Therefore the structure of a system reflects the social boundaries of the organization(s) that produced it.

Q33. What do you understand by Contract Testing?

According to Martin Flower, **contract test** is a test at the boundary of an external service which verifies that it meets the contract expected by a consuming service.
Also, contract testing does not test the behavior of the service in depth. Rather, it tests that the inputs & outputs of service calls contain required attributes and the response latency, throughput iswithin allowed limits.

Q34. What is End to End Microservices Testing?

End-to-end testing validates each and every process in the workflow is functioning properly. This ensures that the system works together as a whole and satisfies all requirements.
In layman terms, you can say that end to end testing is a kind of tests where everything is tested after a particular period.

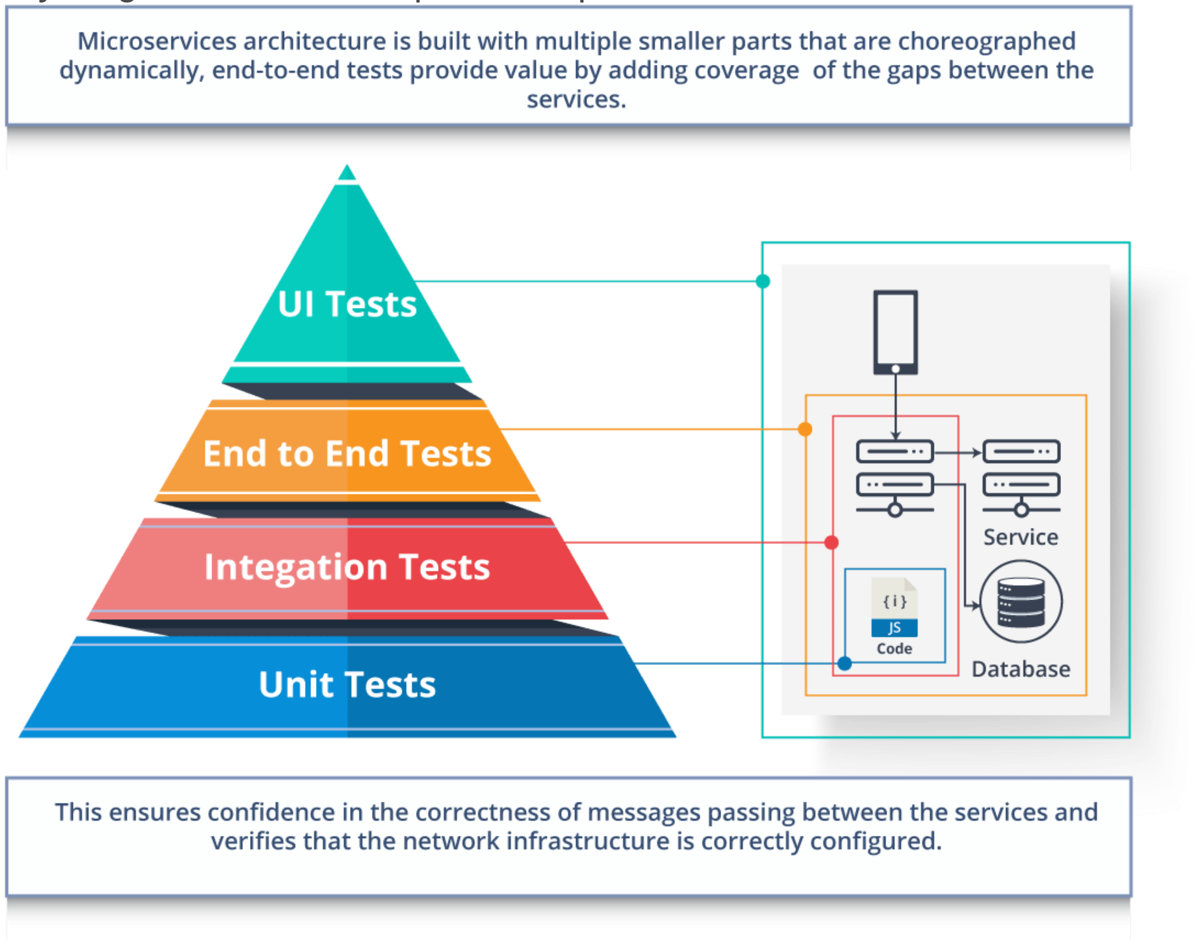

Microservices architecture is built with multiple smaller parts that are choreographed dynamically, end-to-end tests provide value by adding coverage of the gaps between the services.
UI Tests
End to End Tests
Integration Tests
Unit Tests
Service
{ i }
JS
Code
Database
This ensures confidence in the correctness of messages passing between the services and verifies that the network infrastructure is correctly configured.

Fig 14: Hierarchy of Tests – Microservices Interview Questions

Q35. What is the use of Container in Microservices?

Containers are a good way to manage microservice based application to develop and deploy them individually. You can encapsulate your microservice in a container image along with its dependencies, which then can be used to roll on-demand instances of microservice without any additional efforts required.

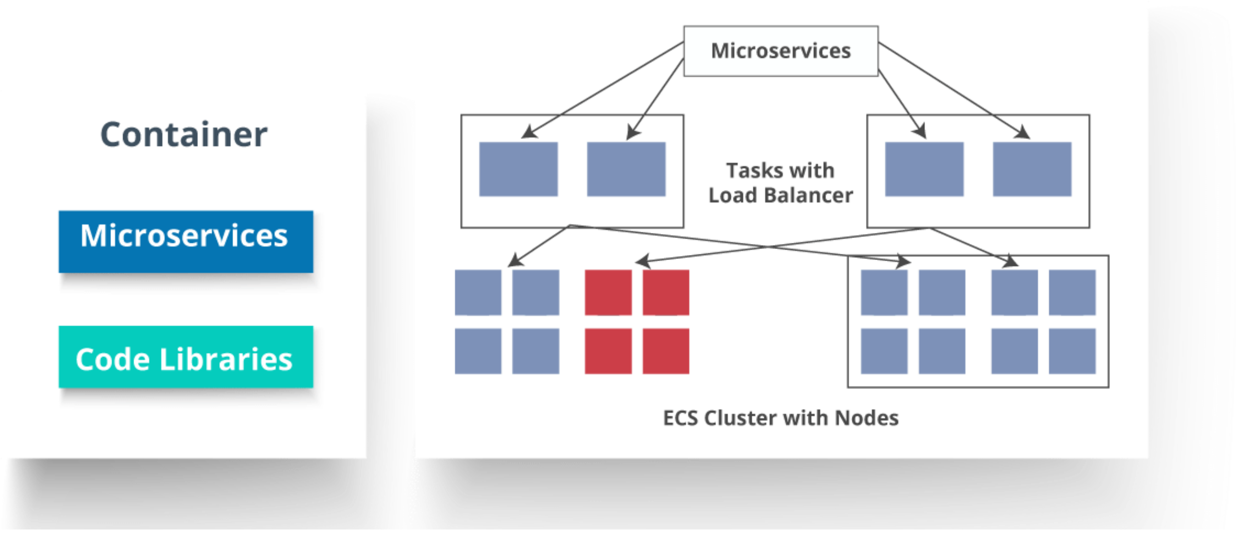


Fig 15: Representation of Containers and How they are used in Microservices – Microservices Interview Questions

Q36. What is DRY in Microservices architecture?

DRY stands for **Don't Repeat Yourself**. It basically promotes the concept of reusing the code. This results in developing and sharing the libraries which in turn result in tight coupling.

Q37. What is a Consumer-Driven Contract (CDC)?

This is basically a pattern for developing Microservices so that they can be used by external systems. When we work on microservices, there is a particular provider who builds it and there are one or more consumers who use Microservice.
Generally, providers specify the interfaces in an XML document. But in Consumer Driven Contract, each consumer of service conveys the interface expected from the Provider.

Q38. What is the role of Web, RESTful APIs in Microservices?

A microservice architecture is based on a concept wherein all its services should be able to interact with each other to build a business functionality. So, to achieve this, each microservice must have an interface. This makes the web API a very important enabler of microservices. Being based on the open networking principles of the Web, RESTful APIs provide the most logical model for building interfaces between the various components of a microservice architecture.

Q39. What do you understand by Semantic monitoring in Microservices architecture?

Semantic monitoring, also known as **synthetic monitoring** combines automated tests with monitoring the application in order to detect business failing factors.

Q40. How can we perform Cross-Functional testing?

Cross-functional testing is a verification of non-functional requirements, i.e. those requirements which cannot be implemented like a normal feature.

Q41. How can we eradicate non-determinism in tests?

Non-Deterministic Tests (NDT) are basically unreliable tests. So, sometimes it may happen that they pass and obviously sometimes they may also fail. As and when they fail, they are made to re-run to pass.
Some ways to remove non-determinism from tests are as follows:

1. Quarantine
2. Asynchronous
3. Remote Services
4. Isolation
5. Time
6. Resource leaks

Q42. What is the difference between Mock or Stub?

Stub

- A dummy object that helps in running the test.
- Provides fixed behavior under certain conditions which can be hard-coded.
- Any other behavior of the stub is never tested.

For example, for an empty stack, you can create a stub that just returns true for empty() method. So, this does not care whether there is an element in the stack or not.

Mock

- A dummy object in which certain properties are set initially.
- The behavior of this object depends on the set properties.
- The object's behavior can also be tested.

For example, for a Customer object, you can mock it by setting name and age. You can set age as 12 and then test for isAdult() method that will return true for age greater than 18. So, your Mock Customer object works for the specified condition.

Q43. What do you know about Mike Cohn's Test Pyramid?

Mike Cohn provided a model called **Test Pyramid.** This describes the kind of automated tests required for software development.

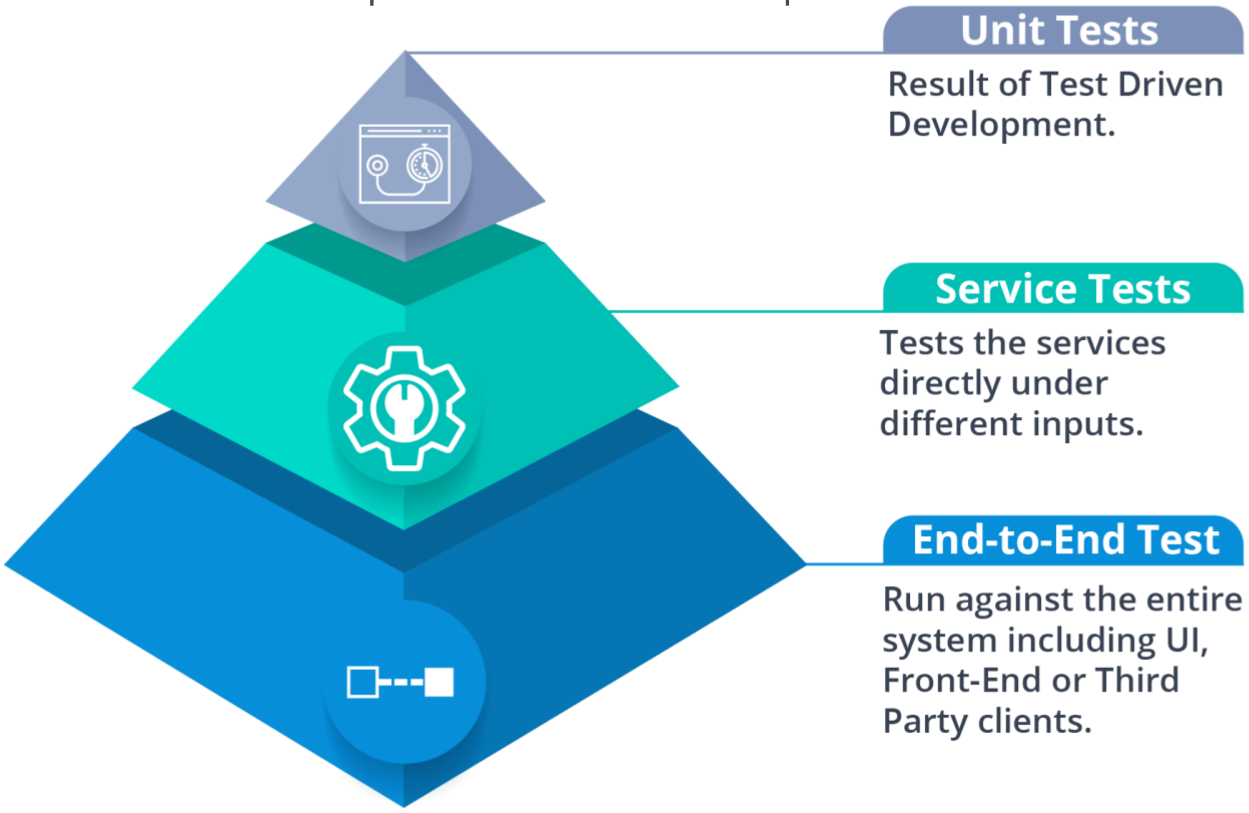
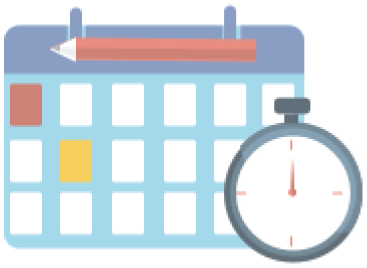

Unit Tests
Result of Test Driven Development.
Service Tests
Tests the services directly under different inputs.
End-to-End Test
Run against the entire system including UI, Front-End or Third Party clients.

Fig 16: Mike Cohn's Test Pyramid – Microservices Interview Questions

As per pyramid, the number of tests at first layer should be highest. At service layer, the number of tests should be less than at the unit test level, but more than at the end-to-end level.

Q44. What is the purpose of Docker?

Docker provides a container environment that can be used to host any application. In this, the software application and the dependencies which support it are tightly-packaged together.
So, this packaged product is called a **Container** and since it is done by Docker, it is called **Docker container!**



See Batch Details

Q45. What is Canary Releasing?

Canary Releasing is a technique to reduce the risk of introducing a new software version in production. This is done by slowly rolling out the change to a small subset of users before giving it out to the entire infrastructure, i.e. making it available to everybody.

Q46. What do you mean by Continuous Integration (CI)?

Continuous Integration (CI) is the process of automating the build and testing of code every time a team member commits changes to version control. This encourages developers to share code and unit tests by merging the changes into a shared version control repository after every small task completion.

Q47. What is Continuous Monitoring?

Continuous monitoring gets into the depth of monitoring coverage, from in-browser front-end performance metrics, through application performance, and down to host virtualized infrastructure metrics.

Q48. What is the role of an architect in Microservices architecture?

An architect in microservices architecture plays the following roles:

- Decides broad strokes about the layout of the overall software system.
- Helps in deciding the zoning of the components. So, they make sure components are mutually cohesive, but not tightly coupled.
- Code with developers and learn the challenges faced in day-to-day life.
- Make recommendations for certain tools and technologies to the team developing microservices.
- Provide technical governance so that the teams in their technical development follow principles of Microservice.

Q49. Can we create State Machines out of Microservices?

As we know that each Microservice owning its own database is an independently deployable program unit, this, in turn, lets us create a State Machine out of it. So, we can specify different states and events for a particular microservice.
For Example, we can define an Order microservice. An Order can have different states. The transitions of Order states can be independent events in the Order microservice.

Q50. What are Reactive Extensions in Microservices?

Reactive Extensions also are known as Rx. It is a design approach in which we collect results by calling multiple services and then compile a combined response. These calls can be synchronous or asynchronous, blocking or non-blocking. Rx is a very popular tool in distributed systems which works opposite to legacy flows.
Hope these Microservices Interview Questions would help you in your Microservices Architect Interviews.
If you wish to learn Microservices and build your own applications, then check out our ***Microservices Architecture Training*** *which comes with instructor-led live training and real-life project experience. This training will help you understand Microservices in depth and help you achieve mastery over the subject.*