# Top 50 Microservices Interview Questions You Must Prepare In 2023

### **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 still 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 particular 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.



### **Q3. What are the features of Microservices?**

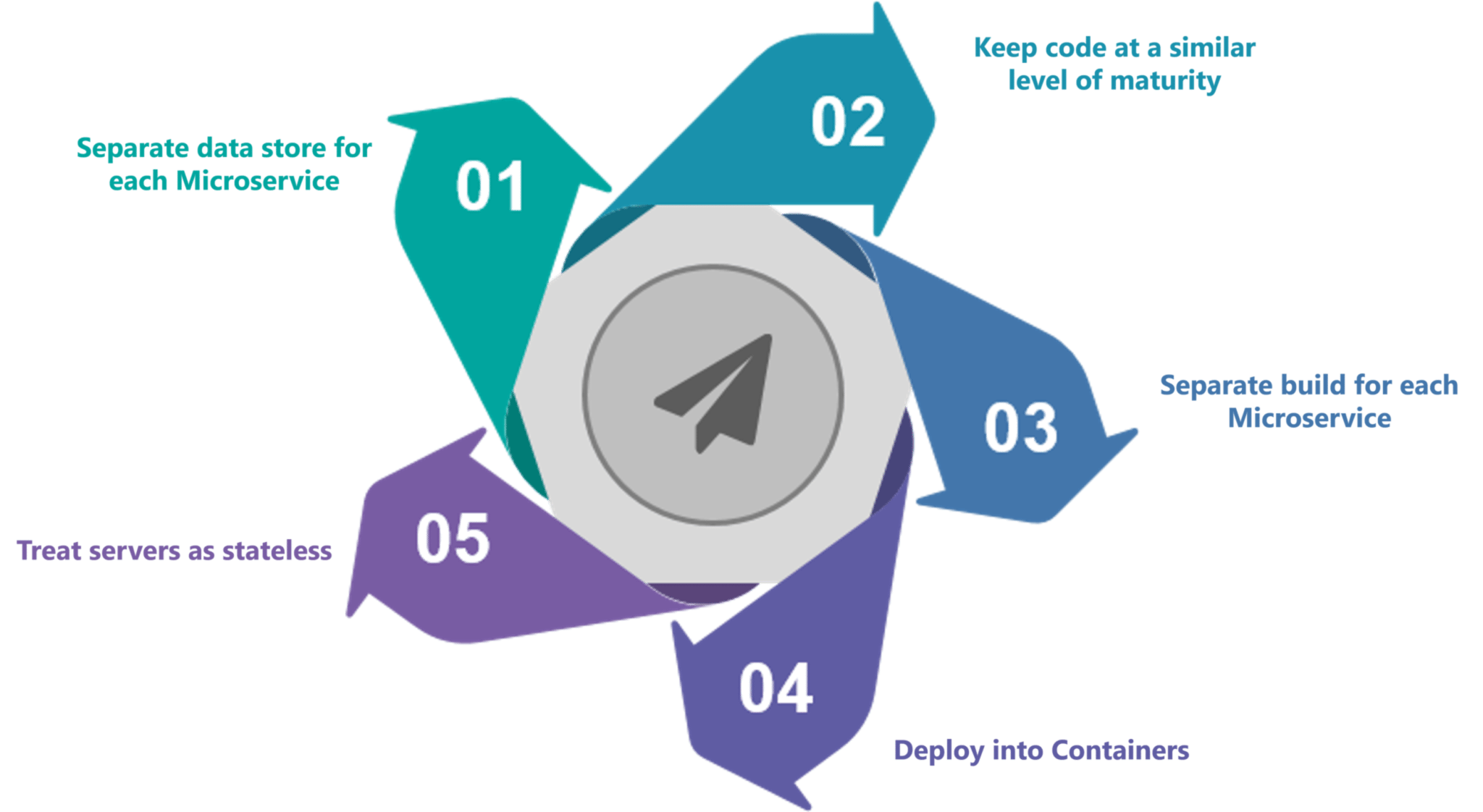


**Fig 3:**Features of Microservices – Microservices Interview Questions

* **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
* **Continous 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?**

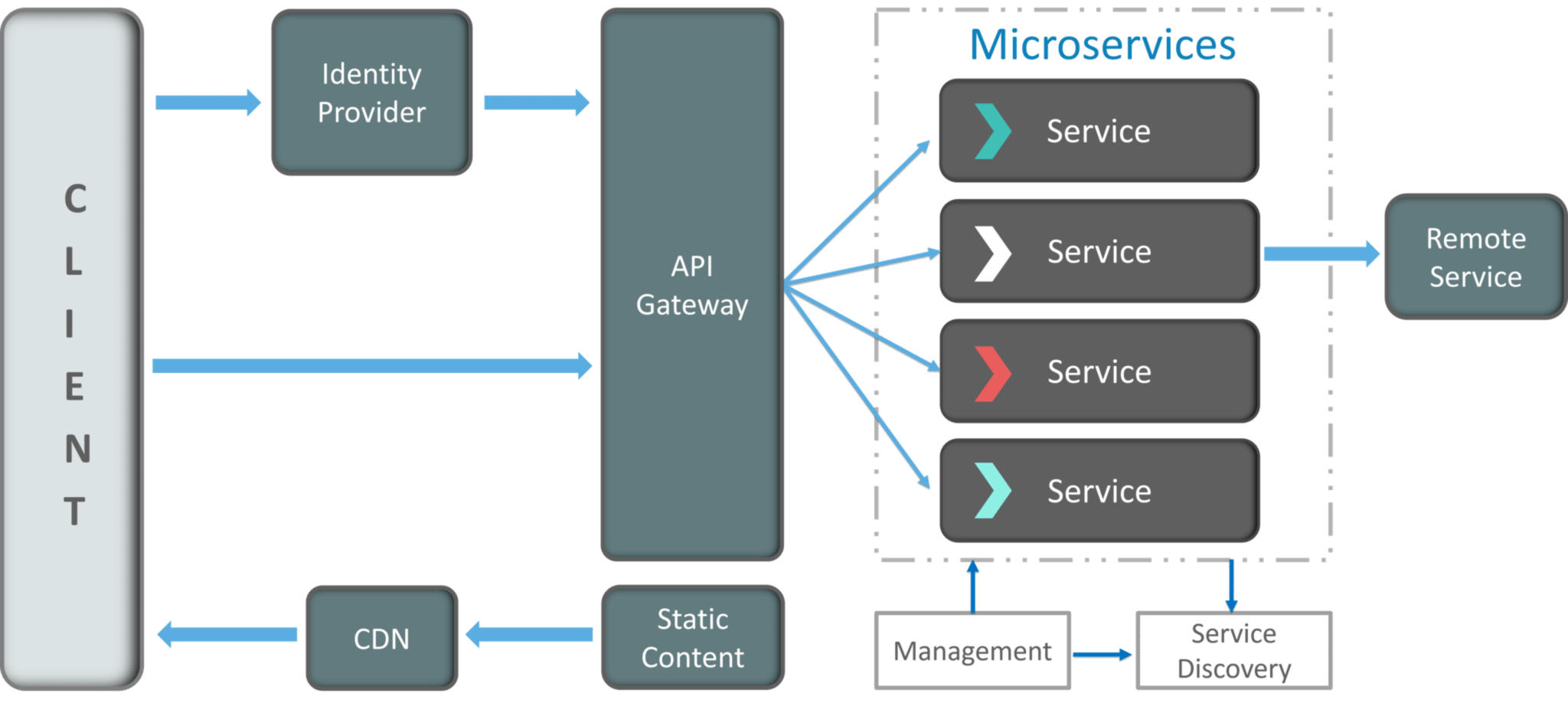
The following are the best practices to design microservices:



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

### **Q5. How does Microservice Architecture work?**

A microservice architecture has the following components:



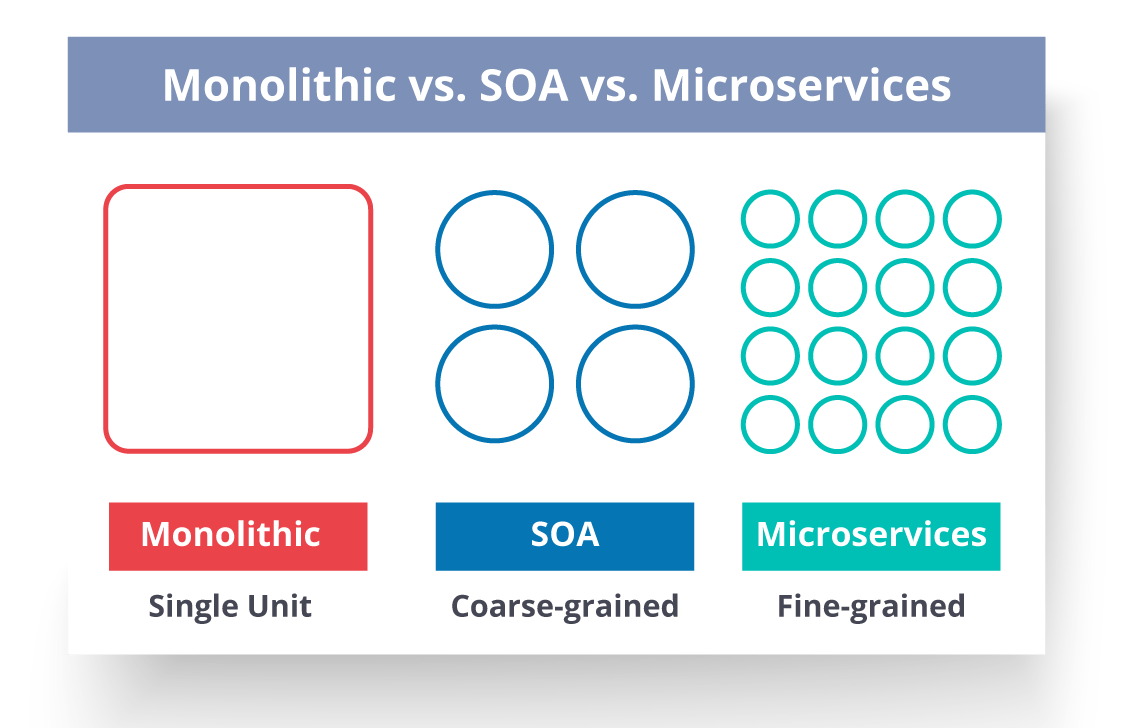
**Fig 5:**Architecture of Microservices – Microservices Interview Questions

* **Clients** – Different users from various devices send requests.
* **Identity Providers** – Authenticates user or client’s 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 |
| Multiple 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 a number of 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.

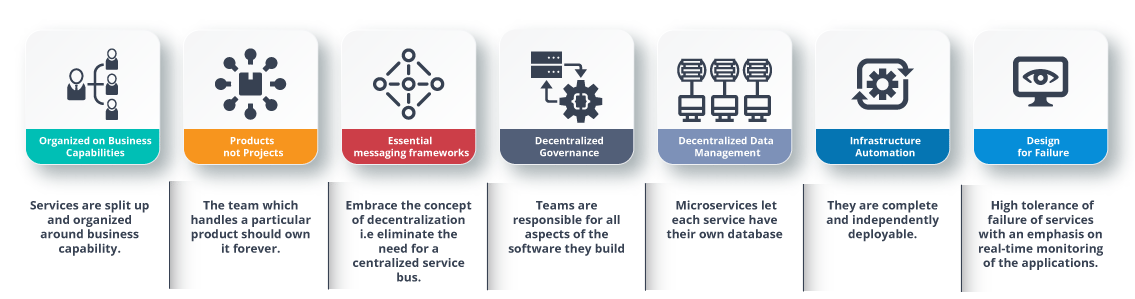
### Q9. What are the key differences between SOA and Microservices Architecture?

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?**

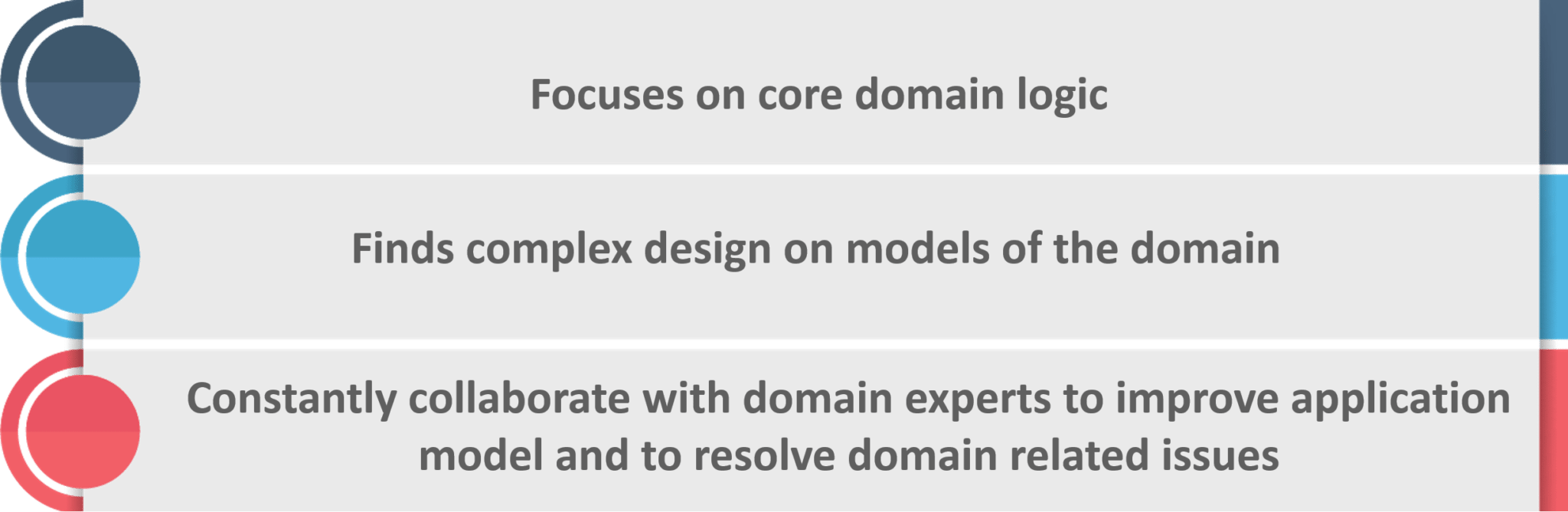
You can list down the characteristics of microservices as follows:



**Fig 7:**Characteristics of Microservices – Microservices Interview Questions

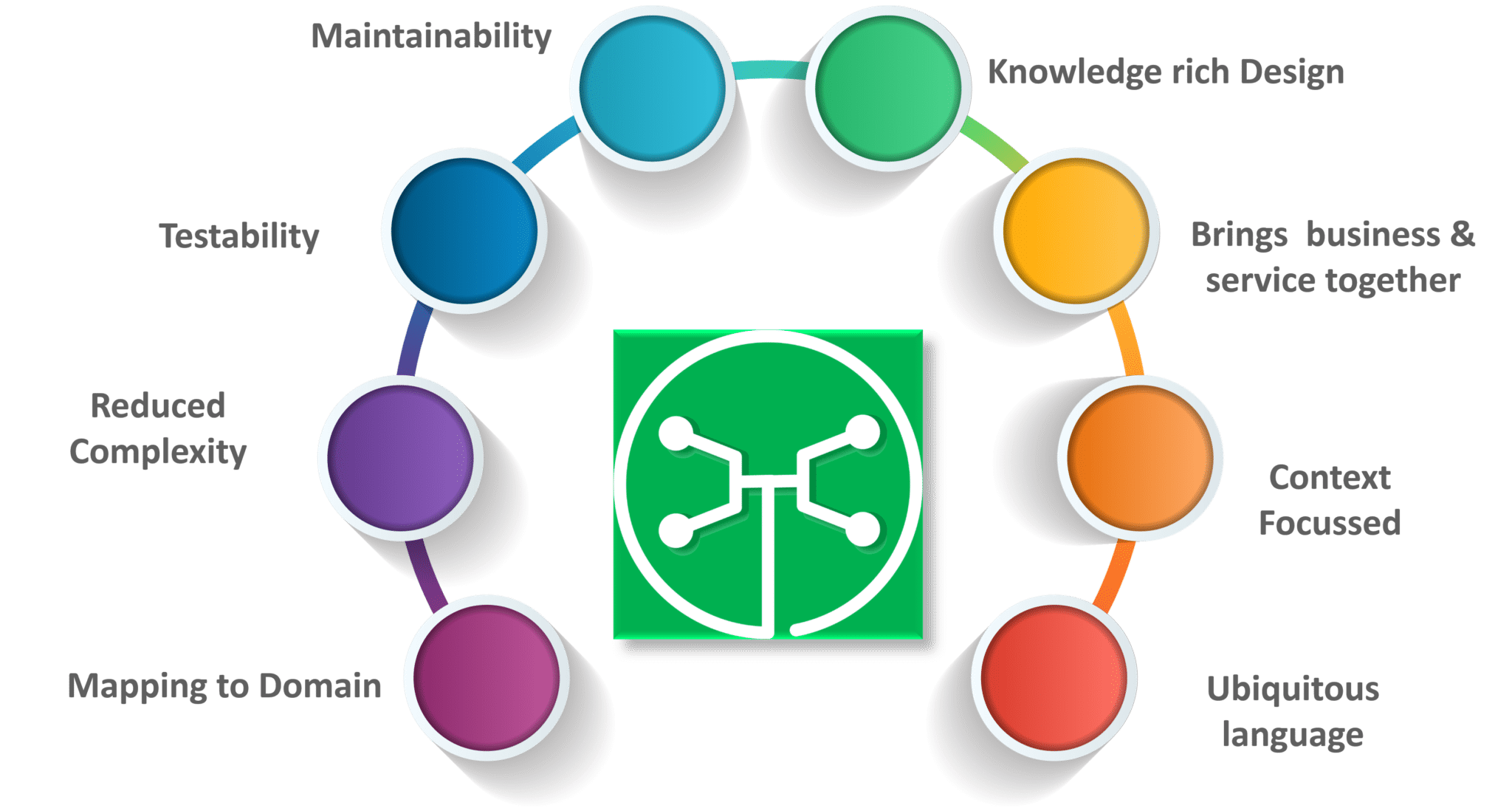
## ****Microservices Interview Questions and Answers in 2022 | Edureka****

### **Q11. What is Domain Driven Design?**



**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.

[[](https://www.edureka.co/microservices-architecture-training?utm_source=blogbanner&utm_campaign=curriculum)](https://www.edureka.co/microservices-architecture-training?utm_source=blogbanner&utm_campaign=curriculum" \t "_blank)

### [Microservices Certification Training Course](https://www.edureka.co/microservices-architecture-training?utm_source=blogbanner&utm_campaign=curriculum" \t "_blank)

[Explore Curriculum](https://www.edureka.co/microservices-architecture-training?utm_source=blogbanner&utm_campaign=curriculum" \t "_blank)

### **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?**

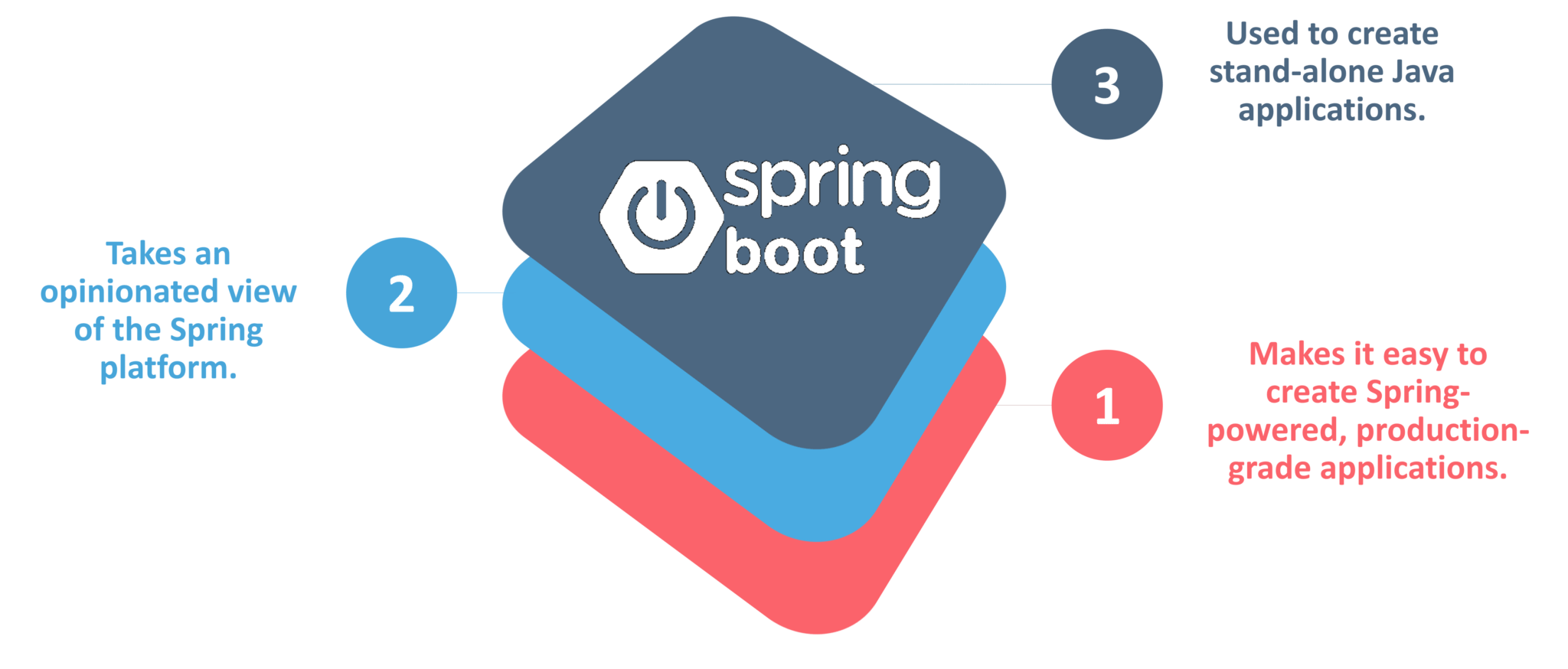
**Representational State Transfer (REST)/RESTful** web services are an architectural style to help computer systems communicate over the internet. This makes microservices easier to understand and implement.

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.



**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.

### **Q19. What is Spring Cloud?**

According to the official website of Spring Cloud, Spring Cloud 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).

### **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** 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.

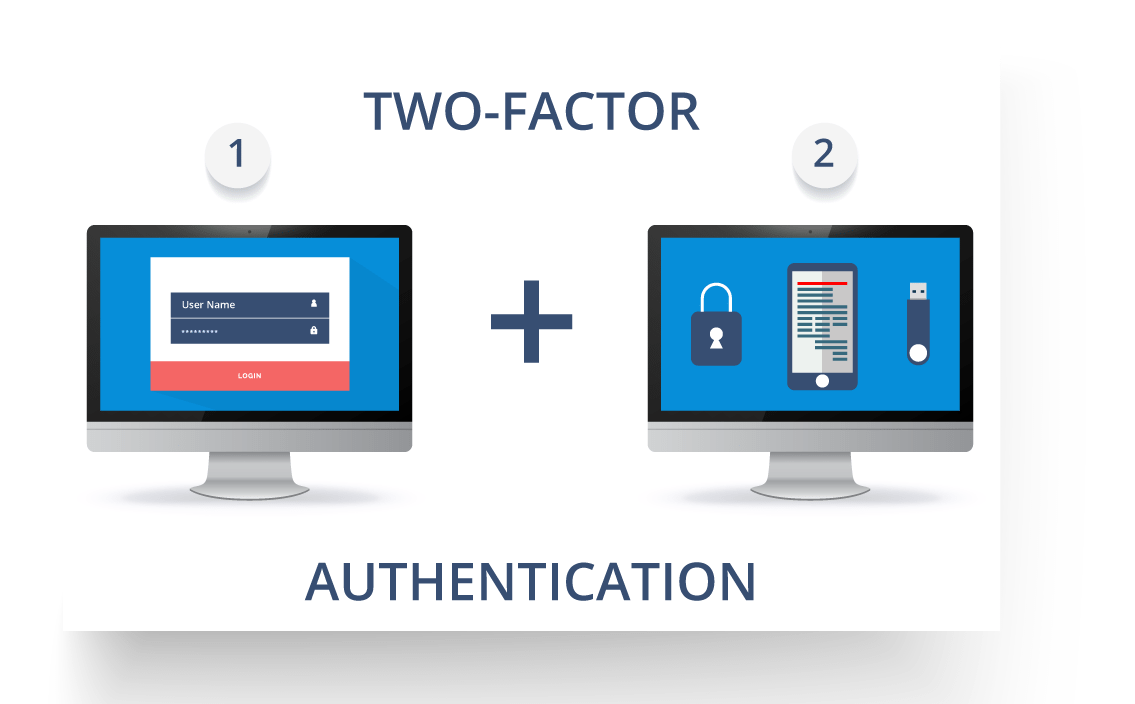
**Usage**: Idempotence is used at the remote service, or data source so that, when it receives the instruction more than once, it only processes the instruction once.

### **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.

### **Q27. What is Two Factor Authentication?**

Two-factor authentication enables the second level of authentication to an account log-in process.

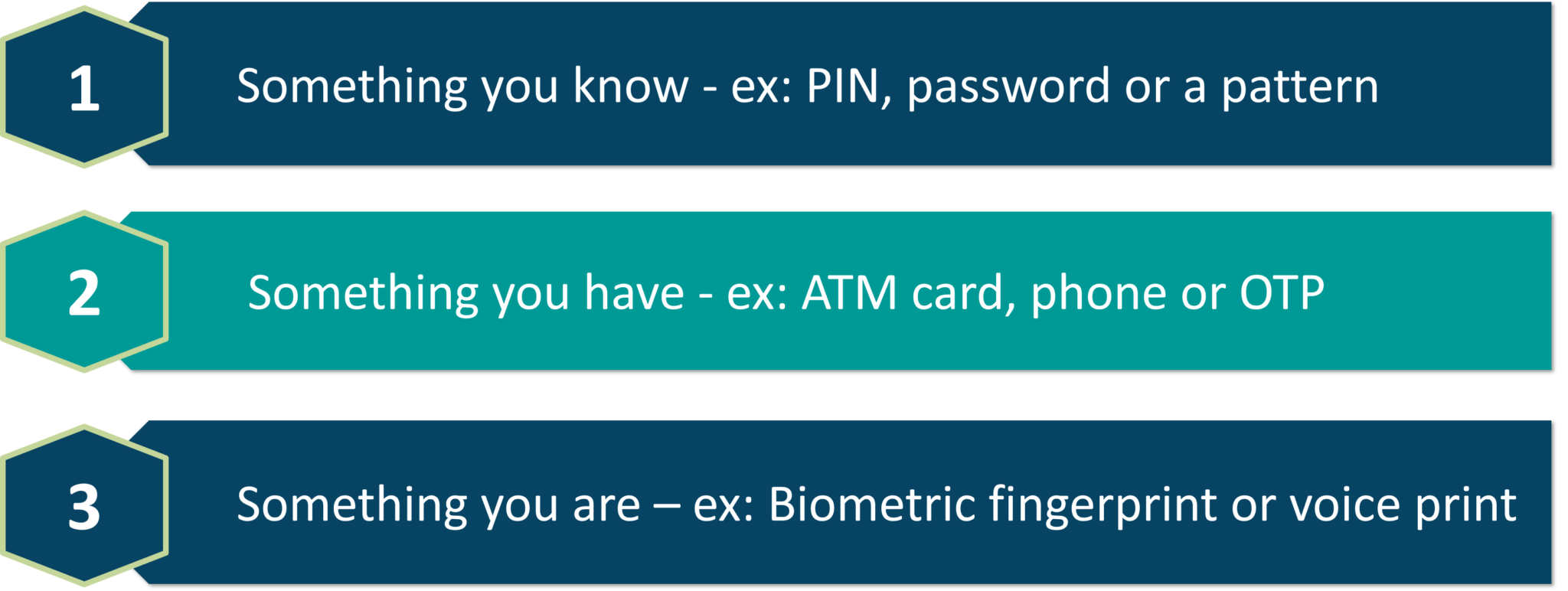


**Fig11:**Representation of Two Factor Authentication – Microservices Interview Questions

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:



**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?**

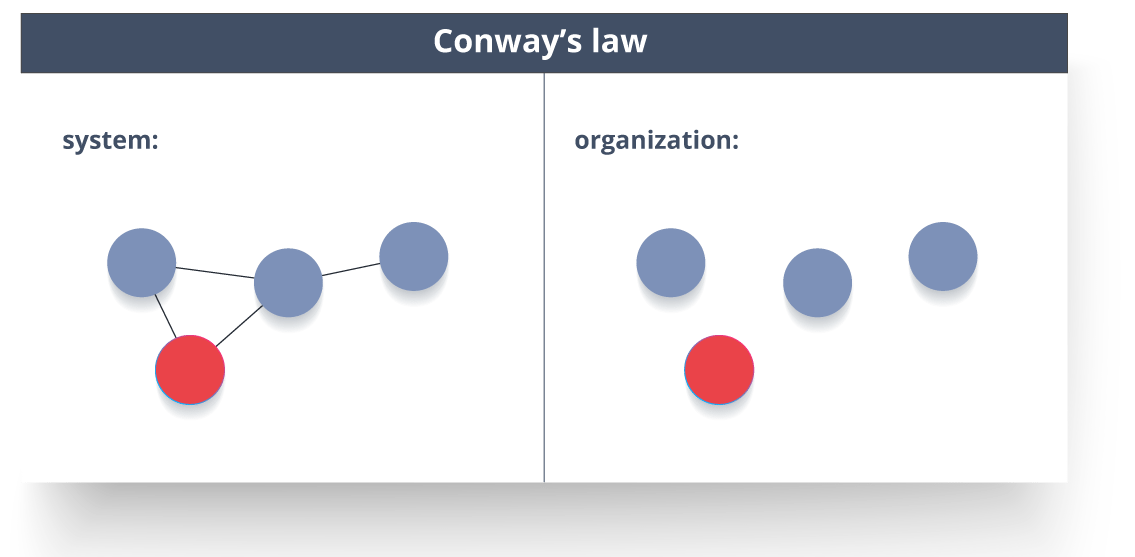
**PACT**is an open source tool to allow testing interactions between service providers and consumers in isolation against the contract made so that the reliability of Microservices integration increases.

### **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?**

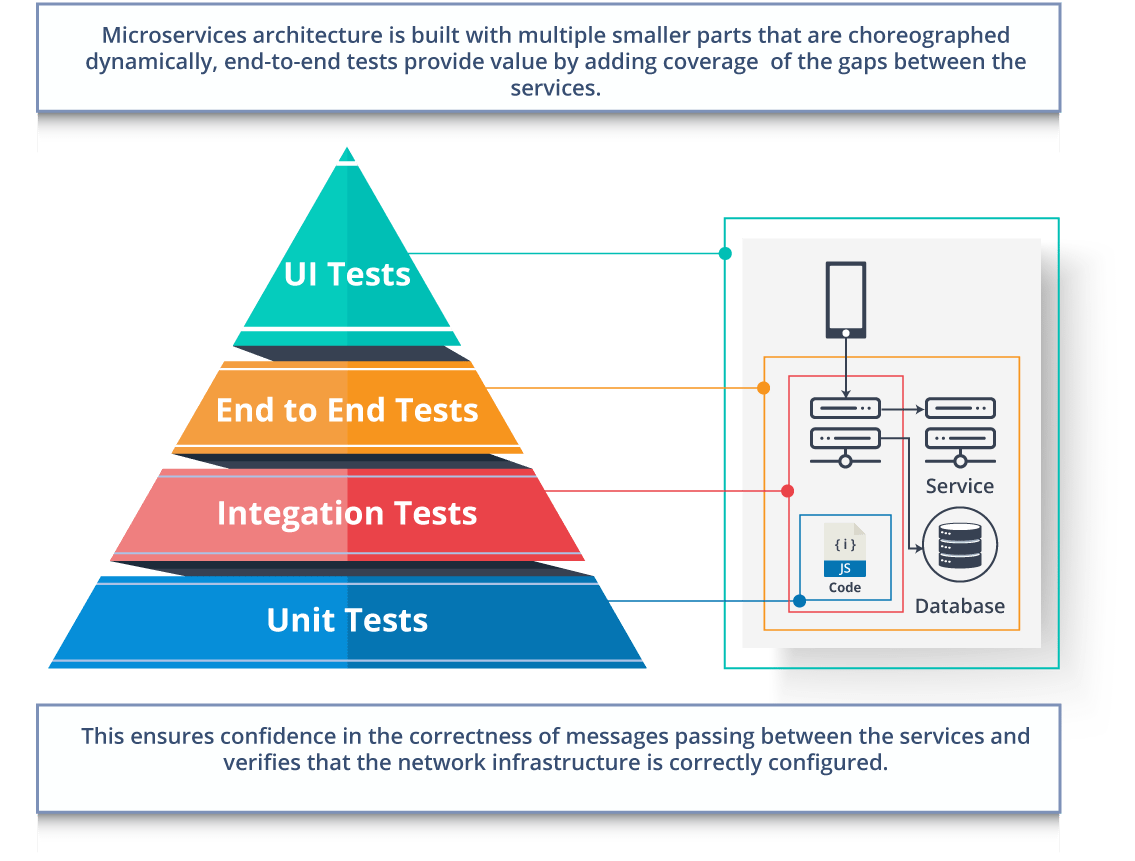
According to Martin Flower,**contract test**is a test at the boundary of an external service which verifies that it meets the contract expected by a consuming service.

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 is within 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.

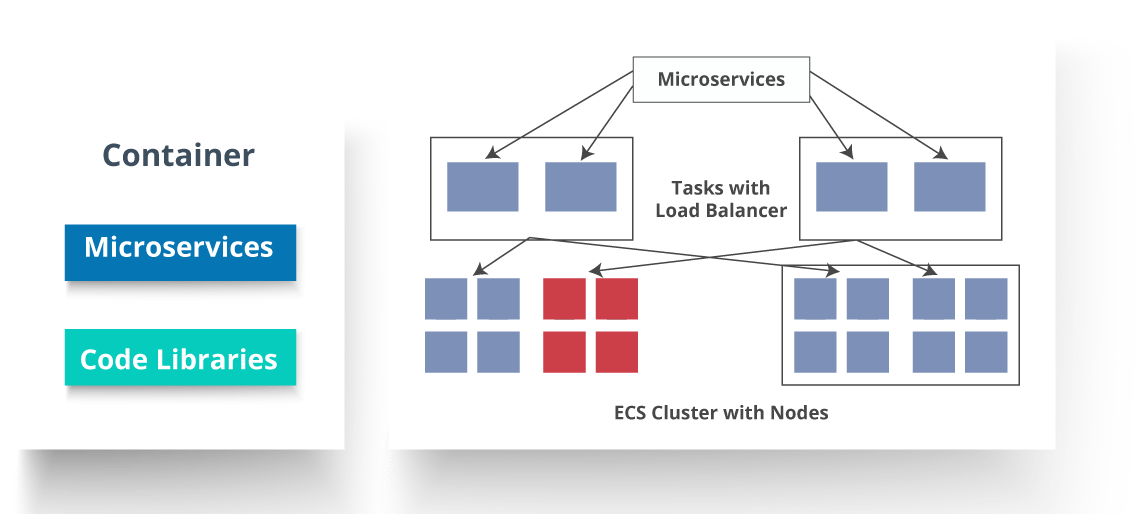
In layman terms, you can say that end to end testing is a kind of tests where everything is tested after a particular period.



**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.

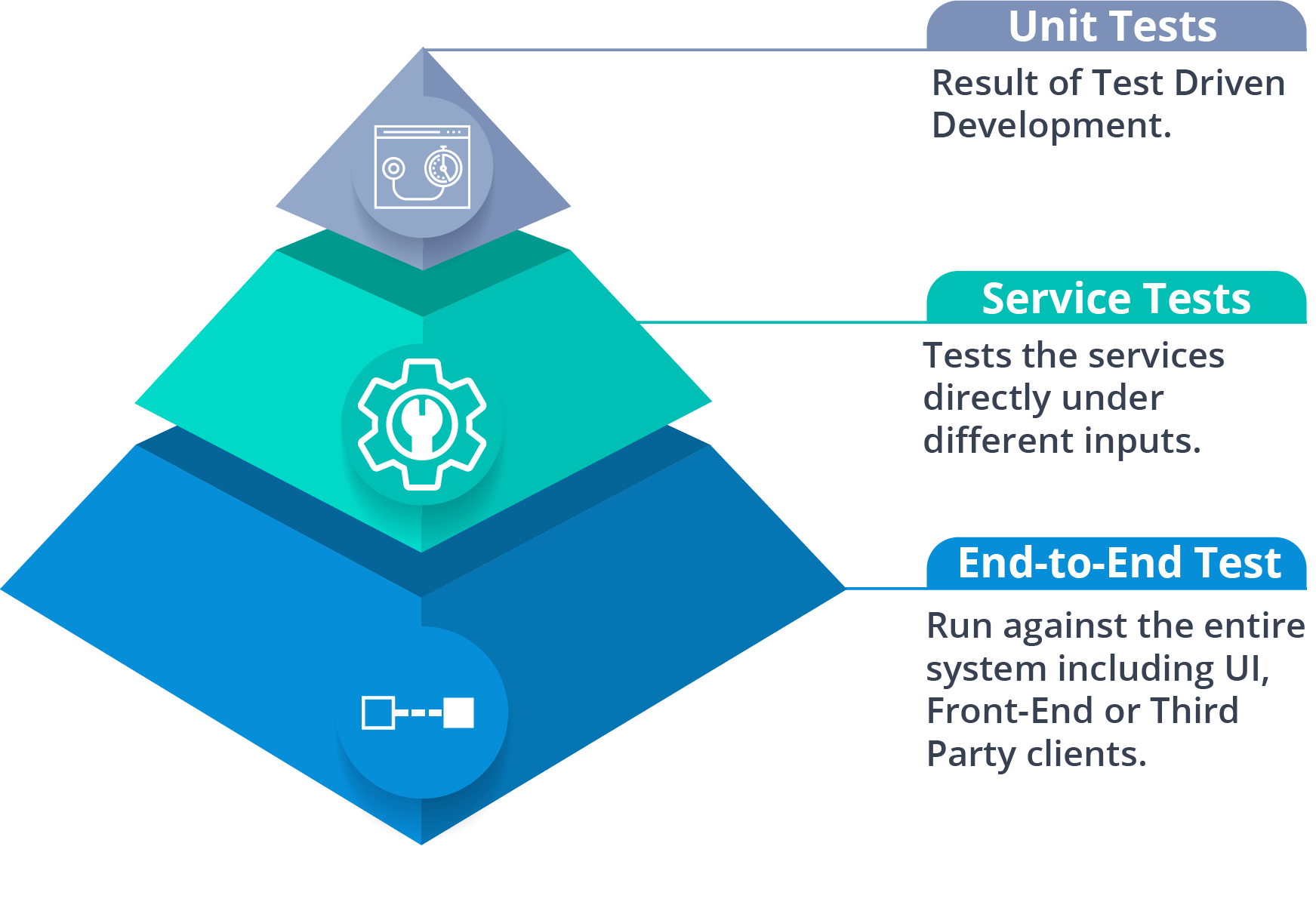
#### **Mock**

* 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.



**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!**

[[](https://www.edureka.co/microservices-architecture-training?utm_source=blogbanner&utm_campaign=batches)](https://www.edureka.co/microservices-architecture-training?utm_source=blogbanner&utm_campaign=batches" \t "_blank)

### [Microservices Certification Training Course](https://www.edureka.co/microservices-architecture-training?utm_source=blogbanner&utm_campaign=batches" \t "_blank)

[Weekday / Weekend BatchesSee Batch Details](https://www.edureka.co/microservices-architecture-training?utm_source=blogbanner&utm_campaign=batches" \t "_blank)

### **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.

[image[*source*](https://www.shutterstock.com/)]

**Q #1) What are microservices?**

**Answer:** Think of it as an approach towards [software development](https://www.softwaretestinghelp.com/software-development-tools/). What this approach does is aims at having small autonomous services that focus on a common business goal. The image above has a number of business domains and surroundings that are certain microservices. Each one only focuses on particular business functionality.

If a particular business domain has a set of 20 functionalities, each of the microservices will only focus on one functionality. 20 microservices means 20 corresponding functionalities, which is a far more much-simplified architecture. This approach explains the architecture.

**Q #2) What are the advantages of microservices architecture?**

**Answer: Advantages include:**

**a) Independent development:** When you talk about microservices, you’ll have to relate it to this point because we are talking about having individual microservice for individual functionality. That means for each functionality, you can develop a separate service and you can manage it and control it totally independently.

Apart from that, it also ensures independent deployment. This means that if you have developed a single service, you can deploy that service. You do not have to wait for the complete architecture or other services to be ready and up for deployment, even if there is a certain amount of change in a particular service.

**b) Fault isolation:** If a particular microservice goes down, it does not affect the functioning of other services. If one service fails, it does not affect the functioning of other services in any way.

**For example,** think of an architecture comprising 20 applications or 20 services working parallelly. Since these applications are interdependent, as is the case with a monolithic architecture, it takes the whole system down. This is not the case with microservices.

**c) Mixed acknowledges stack:** Since you have different microservices and you have an independent approach towards development, you can pick the technology that suits the needs of that service. And accordingly, you can have a mixed technology stack for a complete architecture.

**d) Granular scaling:** This is an important point. Scaling can be a huge problem since you have to take into consideration the complete architecture. But since you’re having small services, you can actually scale those independently.

**Q #3) What are the features of microservices?**

**Answer: Features are:**

* **Decoupling:** Applications can be easily decoupled or rather separated to have individual functionality, which implies it assures easy development, maintenance, and deployment.
* **Componentization:** In componentization, every microservice is treated as an individual component and it is responsible or manages all the intricacies as far as a particular service is concerned. It focuses only on that. It is loosely coupled with other services. That is why each of them can be thought of as a single component or a container.
* **Business capabilities:** We can focus business capabilities much better way since you are setting up smaller goals and you focus on individual functionalities. It becomes easier to develop applications to meet these requirements as far as your business is concerned.
* **Autonomy:** The developers are now free because, as a result of small team clusters, they’re free to develop the applications. This implies that the other teams and developers are not interdependent on each other. This assures speeded allotment of software.
* [**Continuous delivery**](https://www.softwaretestinghelp.com/what-is-continuous-delivery/)**:** Since you have so many features which we just discussed, this ensures that you have constant and frequent releases of software, and a system can be continuously updated and modified in alignment with the needs of a particular company or a particular business domain.
* **Responsibility:** In this approach, every domain or every project is treated as a product. This means that a team will take responsibility for the so-called product and they will bring it to life, and it is that responsibility, i.e. builds it, test it and carry on with the whole lifecycle.
* **Agility:** Since you have a decentralized architecture, you can easily build applications and discard them if they’re no longer needed.

**Q #4) What are the characteristics of microservices?**

**Answer:**

**a) Organized on business capabilities:** Since you have individual microservices, it means that you have an application that works on its own. You also have a database that corresponds to individual microservices, which means if you have 10 microservices, you will have 10 individual databases that are corresponding or answering only to that particular microservice.

This gives you a complete decentralized structure, and this also means better organization because every microservices is very clear what it is supposed to do. It has all the resources in that single container, i.e. this is to be done and this is what we are focused on. That is how this organization or better management comes into the picture.

**b) Products and not projects:** Suppose you have 10 microservices, you’ll be having small clusters of the team that would build up this application and also maintains it. What Amazon thinks is you should have a two-pizzas team. This means you should have such a small team that can survive on two pizzas, and you should also be able to maintain the software.

Every responsibility as far as software and application are concerned is given to that size of a team that can survive on two pizzas. This ensures modularity. When you talk about microservices, basically you’re talking about a product. When you talk about a team focused on a particular functionality, that functionality is not a project for them, it is more of a product for them.

**c) Smart endpoints and dumb pipes:** This means that you have different applications and they’re quite decentralized and there has to be a way of communication. You should have smart endpoints and good ways to integrate your applications with each other.

**d) Decentralized governance:** Discussed earlier.

**e) Decentralized data management:** This means you are free to have an individual approach where you have your own database and your single database in your microservices treated as an individual entity or a container.

**f) Infrastructure automation:**Since we already have so many microservices, it is important that there is proper automation. This ensures speedy development and also maintenance.

**g) Design for failure:** Applications are built such that they do not hamper the progress of any applications, even when they fail. Despite that, you need to have a mechanism that takes care of failure as well.

There is the need to have real-time monitoring and analysis, which ensures that if an application goes down, the fastest possible action is taken so that the application runs again and does not hamper the working of any other applications.

**Q #5) What are the best practices to design microservices?**

**Answer:**

* **Separate data store for each microservices** because we have a database that corresponds to a particular microservice.
* **Keep code at a similar level of maturity:** This gives developers the freedom to build applications the way they want to.
* **A similar level of understanding** means a similar level from a business domain perspective. All the teams are on a similar page, even though they’re working on different things.
* **Separate build for each microservice:** All teams are on the same level when they’re working separately for each microservice. These get deployed in containers.
* **Treat servers as stateless:** Containers read servers as stateless, this helps in better communication.
* **Deploy into containers:**Microservices should be deployed into containers.

**Q #6) What is DDD (Domain Driven Design)?**

**Answer:**This is an approach that actually helps you [collaborate with all the teams](https://www.softwaretestinghelp.com/best-team-collaboration-tool/) together, and to a great extent, makes the development of complex applications easier.

**The three main pillars of DDD principles are:**

* Focus on the core domain and domain logic.
* Trace the complex design of the business domain’s models.
* Resolve business domain problems, make the application domain better by having regular consultations with the business domain experts.

At times, you let go of what is right for you. Rather, you focus only on the business domain. This means that the focus should be on what the business domain needs, and this is what we would include in this architecture. This kind of approach is called domain-driven design.

DDD tries to bring again everything on the same page through the use of a ubiquitous language, which helps you achieve this domain-driven design approach. A ubiquitous language should be designed in such a way that it focuses around a particular domain, and the architecture that is developed out of it is referred to as a domain-driven architecture.

**Q #7) What is a ubiquitous language?**

**Answer:**Suppose we have a ticket reservation system for a flight. The ubiquitous language will be all the terminologies that would be related to airplanes, flight attendants, customers, etc. The manager should be able to comprehend the architecture.

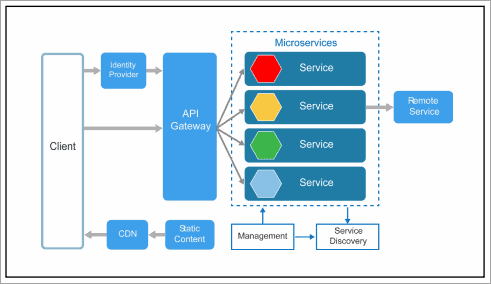
He should be able to understand the terminologies that are being used, and the same applies to the developers, too. This means that we are bringing every team on the same page, and a language that does that is referred to as a ubiquitous language.

**Q #8) Why is there a need for a Domain-Driven Design (DDD)?**

**Answer:**

* Mapping to a domain: We are mapping our architecture to a particular domain.
* Testability:[Testing the application](https://www.softwaretestinghelp.com/application-testing-into-the-basics-of-software-testing/) becomes easier.
* Maintainability: Maintaining the application also becomes easier because everybody is on the same page.
* Knowledge-rich design: These applications are very heavy on knowledge.
* It brings business and service together.
* Context focused: Very much focused on a particular domain.
* Uses ubiquitous focussed.
* Reduced complexity.

**Q #9) How does microservices architecture work?**

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2022/01/client.png)

[image [*source*](https://dzone.com/)]

There is no particular pattern for this kind of architecture because it can be very complex at times. If you have several microservices, all of those will have different architectures, but the base remains the same. That means no matter how complex or how big the project is, the architecture will never become complex.

But yes, it might become big because we might think of having a big number of microservices, maybe in thousands. But let’s assume you have an application that is huge, no matter how big it becomes.

It would never get complicated because we are developing small applications. Each of these small applications will have an architecture that is more or less like the one in the image above.

**The architecture comprises:**

* Client and identity provider
* API Gateway
* Static content
* SDN – content delivery networks
* Microservices
* Management
* Service discovery portal or a model
* User
* Remote service
* Identity provider, remote service, and microservices.

**Steps of understanding working:**

1. If a client wants to use a particular service, the client would send in a request. It is the job of an identity provider to verify whether or not the user is valid.
2. Once the identity provider authenticates a user, the request is then forwarded to API Gateway. Since a client cannot directly communicate with a service, there has to be an intermediary or some intermediate approach that lets the client communicate.
3. That’s where API Gateway comes in. It finds directories that a client is looking for and the request is forwarded to that service.
4. The service communicates with other services and sees how a solution is generated for the client.
5. Once the solution is generated, it is sent back to the client using CDN, i.e. Content Delivery Network.
6. The static content is whatever content that is entered and is in the form of static, is held by static content.
7. The management and the service discovery portals have a particular task as well. **For example,** I have several services. The management portal places those services on respective nodes.
8. The service discovery, on the other hand, keeps track of all these services, makes note of services that have failed and all those things.
9. This record is communicated back to the management portal, and it is the job of this management portal to resolve if there is any failure in the corresponding architecture or any of the services.

These components all work in tandem, and this is how they bring the architecture together. Apart from that, there are some other concepts that are not there in the architecture displayed on the image above, e.g. the messaging model.

There is an approach where messaging also occurs, **for instance,** if the client is waiting for a particular reply from a service. In that case, we have a means of communication and that’s through synchronous message passing. However, if the client won’t be waiting for a response from the service, we have an asynchronous means of communication.

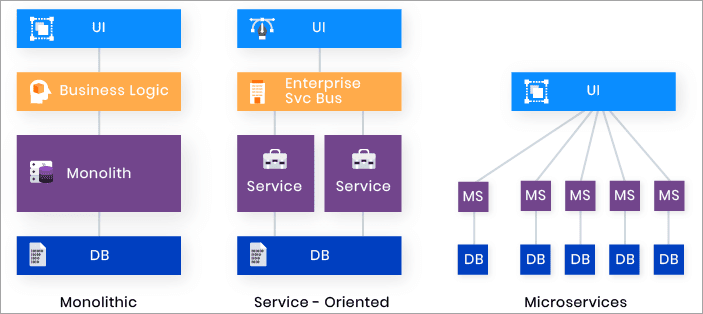
**Q #10) What are the advantages and disadvantages of microservice architecture?**

**Answer: This is explained in the table below:**

| **Advantages** | **Disadvantages** |
| --- | --- |
| The liberty to use various technologies | Chances of troubleshooting is increased |
| Each microservice focuses on a specific business domain | Delay increased due to remote calls |
| Units that can be deployed individually are supported | Efforts of configurations including other efforts are increased |
| Software are released more frequently | Transaction safety isn’t easy to maintain |
| Each service security is ensured | It isn’t easy to track data from various service boundaries |
| Supports the development and deployment of many services parallelly | Coding between services is a challenge |

**Q #11) What are the distinguishing factors between Monolithic, Microservices Architecture, and SOA?**

**Answer:**

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2022/01/monolithic.png)

[image [*source*](https://rubygarage.org/)]

**Monolithic architecture**

This is where a single container holds the complete architecture of all your services or databases.

There was a single database that corresponded to all these services and all the interdependencies of those were inside one single container. All the advantages that we discussed become a disadvantage because, since you had a single container or a single architecture, deploying was a huge issue.

A single failure would mean that you would have to work on the application and then deploy the complete architecture. There was no fault isolation because if something failed, everything depended on it and there was a need to repair that part and that decision had to wait for that issue to rectify to resume work again.

Monolithic architecture had quite a few other problems:

* Scaling was a problem.
* Isolation was a problem.
* Then there was no individual development.
* Deployment was a big issue.
* Hard to develop since it was one complex block.
* There is too much dependency on each other – making it poor for software development.

**S.O.A**

It is similar to microservices, but the difference is that the monolithic architecture is divided into smaller units and these smaller units also have subunits. This means that you do not have complete modularity. The modularity is there, but to a smaller extent.

When complete modularity is needed, the option is to choose microservices. SOA has full features and each of these features has individual modules.

**Microservices**

This is not the case with microservices, which have individual modules or complete modularity.

**Q #12) What are the main distinguishing factors between microservices architecture and SOA?**

**Answer:**

**There are a few differences stated below:**

* SOA focuses on sharing almost everything with other modules. But when you talk about microservices, the aim is to share as little as possible.
* The approach is completely different. Apart from that, you have modular applications in SOA, but there are still a lot of interdependencies. That is not the case with microservices.
* When talking about intercommunication between services, in SOA you need a middleware for communication. But in microservices, that are completely independent and you do not need a middleware for that, they directly communicate to the rest of APIs.

**Q #13) What are the challenges with microservices architecture?**

**Answer:**

* **Automating the components:** We can automate all the components, but maintaining these components is a huge task, and automation to work smoothly means constant attention.
* **Configuration management:** In regards to versatility and [configuration management](https://www.softwaretestinghelp.com/configuration-management-in-devops/), you need to understand that this is nothing but an approach that deals with the complete design that you are talking about. And configuring these microservices and having clear architecture can pose a problem.
* **Perceptibility**
* **Debugging:** Debugging or debugging is another issue because we have fault isolation, and if some application fails, it does not affect the working of any other application. But that also means that you also have to be continuously monitoring.

These applications need individual monitoring, and that is why when you talk about system failures, dealing with such failures can be a problem in the overall architecture. Even if it does not affect the functioning of other applications, still, that application has gone down and needs to be up and running.

**Q #14) What is cohesion?**

**Answer:**Microservices architecture is a design pattern. Cohesion and coupling are important because a good design will always have high cohesion and low coupling. To understand this, we need to understand what cohesion and coupling are.

Cohesion is nothing but the bond between an application or an intra bond within an application. **For example,** a molecule, inside a molecule, we have atoms. These atoms, how closely are they bonded to each other?

This bond is what is referred to as cohesion. Cohesion should be high when you talk about a microservice. That means we are referring to the fact that the interdependency inside a container, the database, and the service that is running how closely or how bonded are they with each other?

If they are closely bonded, that is good for the microservice. When you take a look at it from a bigger picture, you are talking about different microservices.

The bond between these microservices should be below. The interbond between two applications in microservices should be loose. This is referred to as low coupling. The interbond inside a container is cohesion, while interbond between two different applications or containers is coupling. That is why when you want a good design, you need high cohesion and low coupling.

**Q #15) What is**[**REST/RESTful**](https://www.softwaretestinghelp.com/restful-web-services-interview-question/)**and what is it used for?**

**Answer:**This is an API, that is Application Protocol Interfaces.

REST stands for Representational State Transfer. These are more or less used for integrating with your applications much better. When talking about a decentralized architecture and to build such an architecture, you need to have a medium through which you can communicate with all these services efficiently and the rest enables us to achieve that.

**Q #16) What is actuator in spring boot?**

**Answer:**An actuator provides real-time analysis. It allows us to monitor an application when it is in its production state. It provides all the information about a particular application. That means you can monitor all of its metrics, i.e. it gives you information about an application and its running state.

**Q #17)**[**What is Spring Boot?**](https://www.softwaretestinghelp.com/spring-boot-interview-questions/)

**Answer:**Spring is an approach towards web services development. It gives you all the ingredients that are needed as far as a web application is concerned. Think of it as a shop where you get all the ingredients for web development. Spring Boot is the customized version of Spring. To get a clear picture, think of the process of cooking and eating food.

As long as you have spring, you have all the ingredients. Even so, you’ll have to prepare your food and eat it. Spring Boot is a customized approach where we have the food ready for us to eat in the fridge. Spring Boot means the food. All the ingredients and spices are already well mixed together. All that is required of us is to place the food inside the oven to heat for us to eat.

**Q #18) What is Spring Cloud?**

**Answer:**It is something that lets you get real-time analysis and perform a finite amount of data processing. It is nothing but an API that is provided by spring, and it helps you get rid of various complexity as far as architecture is concerned.

**Q #19) What problems are solved by Spring Cloud?**

**Answer:**

* Reduces the complexity associated with distributed systems, or the complexity that is associated with distributed systems.
* Service Discovery becomes very easy.
* Helps in load balancing.
* It takes care of redundant code and application development.
* Help in improving performance.

**Q #20) What is the difference between REST and Microservices?**

**Answer: Differences include:**

* REST is a means to implement Microservices.
* It is an architectural style that can develop large-scale applications that can be scaled up quite easily.
* It is used in API design as well as in web apps.
* We need to follow particular patterns to make microservices loosely coupled.
* Through REST we can build Microservices.

**Q #21) What are the different types of tests for Microservices?**

**Answer:**

* [Unit testing](https://www.softwaretestinghelp.com/unit-testing/) – testing the individual functional modules/units
* [Performance testing](https://www.softwaretestinghelp.com/introduction-to-performance-testing-loadrunner-training-tutorial-part-1/)
* [Acceptance test](https://www.softwaretestinghelp.com/what-is-acceptance-testing/)
* [Exploratory test](https://www.softwaretestinghelp.com/what-is-exploratory-testing/) – experimental testing aimed at learning how an application works

Acceptance tests and performance tests are for the stakeholders. They are usually performed when presenting or giving your end product. They just test the working of an application, thus, they occur less compared to the other two tests.

**Q #22) What is a distributed transaction?**

**Answer:**When you have a number of applications or services, these services will commit changes. Thus, the need to make changes to a particular plan request. There is a need to make sure that all the applications are on the same page.

There is a managing entity that ensures that there is a commitment for a particular service from all the applications and all the services. When that commitment arrives, it is then that a transaction is completed and this kind of transaction is referred to as a distributed transaction.

**Q #23) What is an Idempotence and where is it used?**

**Answer:**There are certain results where you would want uniformity. That means if there would be a particular application that would execute 10 times, that should not vary my end result. This implies that no matter how many times the application executes, the end result should be the same each time.

To guarantee this, we have something called an Idempotence law which ensures that uniformity.

**Q #24) What is bounded context?**

**Answer:**We are talking about microservices and modularity. To give you an example, on the organizational level, we have different teams working on different things, like the marketing team focuses on marketing the product, the sales team sells the product, the development team develops the product, and the maintenance team that maintains the product, so you’ll be having different teams.

The different teams would have different concerns and different interdependencies. Each of these teams should exist in a single module. The same applies to application development as well. Each module should have its own independence. This kind of approach is referred to as a bounded context. That means your application is bounded within these limits.

**Q #25) What is two-factor authentication?**

**Answer:**An example would be ATM transactions. When you give in your credit card details, you are actually issuing in your credentials, that is your card number, your name, and your CVS.

After entering those details, you are also expected to enter the OTP that the ATM application uses to reverify that the details that you’ve given are correct; you are the owner of the card and that the number entered is registered with your service. This is where the second step of the authentication happens.

This kind of authentication is known as two-factor authentication.