**1. What are the major features in different versions of Spring Framework?**

|  |  |  |
| --- | --- | --- |
| **Features of Spring Framework** | | |
| **Version** | **Logo** | **Feature** |
| **Spring 2.5** | spring 2.5 logo - Spring Interview Questions - Edureka! | This version was released in 2007. It was the first version which supported annotations. |
| **Spring 3.0** | spring 3.0 logo - Spring Interview Questions - Edureka! | This version was released in 2009. Itmade full-fledged use of improvements in Java5 and also provided support to JEE6. |
| **Spring 4.0** | Spring 4.0 logo - Spring Interview Questions - Edureka! | This version was released in 2013. This was the first version to provide full support to Java 8. |

**2. What is a Spring Framework?**

* Spring is a powerful open source, application framework created to reduce the complexity of enterprise application development.
* It is light-weighted and loosely coupled.
* It has layered architecture, which allows you to select the components to use, while also providing a cohesive framework for J2EE application development.
* Spring framework is also called the framework of frameworks as it provides support to various other frameworks such as Struts, Hibernate, Tapestry, EJB, JSF etc.

**3. List the advantages of Spring Framework.**

* Because of Spring Frameworks layered architecture, you can use what you need and leave which you don’t.
* Spring Framework enables POJO (Plain Old Java Object) Programming which in turn enables continuous integration and testability.
* JDBC is simplified due to Dependency Injection and Inversion of Control.
* It is open-source and has no vendor lock-in.

**4. What are the different features of Spring Framework?**

Following are some of the major features of Spring Framework :

* **Lightweight:** Spring is lightweight when it comes to size and transparency.
* **Inversion of control (IOC):** The objects give their dependencies instead of creating or looking for dependent objects. This is called Inversion Of Control.
* **Aspect oriented Programming (AOP):** Aspect oriented programming in Spring supports cohesive development by separating application business logic from system services.
* **Container:**Spring Framework creates and manages the life cycle and configuration of the application objects.
* **MVC Framework:** Spring Framework’s MVC web application framework is highly configurable. Other frameworks can also be used easily instead of Spring MVC Framework.
* **Transaction Management:** Generic abstraction layer for transaction management is provided by the Spring Framework. Spring’s transaction support can be also used in container less environments.
* **JDBC Exception Handling:** The JDBC abstraction layer of the Spring offers an exception hierarchy, which simplifies the error handling strategy.

**5. How many modules are there in Spring Framework and what are they?**

There are around 20 modules which are generalized into Core Container, Data Access/Integration, Web, AOP (Aspect Oriented Programming), Instrumentation and Test.

* **Spring Core Container –**This layer is basically the core of Spring Framework.It contains the following modules :

1. Spring Core
2. Spring Bean
3. SpEL (Spring Expression Language)
4. Spring Context

* **Data Access/Integration –**This layer provides support to interact with the database. It contains the following modules :

1. JDBC (Java DataBase Connectivity)
2. ORM (Object Relational Mapping)
3. OXM (Object XML Mappers)
4. JMS (Java Messaging Service)
5. Transaction

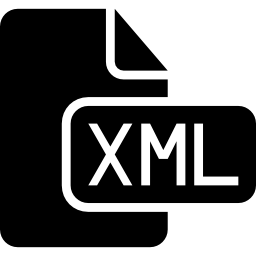
* **Web –**This layer provides support to create web application. It contains the following modules :

1. Web
2. Web – MVC
3. Web – Socket
4. Web – Portlet

* **Aspect Oriented Programming (AOP) –** In this layer you can use Advices, Pointcuts etc., to decouple the code.
* **Instrumentation –**This layer provides support to class instrumentation and classloader implementations.
* **Test –**This layer provides support to testing with JUnit and TestNG.

Few Miscellaneous modules are given below:

* **Messaging –**This module provides support for STOMP. It also supports an annotation programming model that is used for routing and processing STOMP messages from WebSocket clients.
* **Aspects –**This module provides support to integration with AspectJ.

**6. What is a Spring configuration file?**

A Spring configuration file is an XML file. This file mainly contains the classes information. It describes how those classes are configured as well as introduced to each other. The XML configuration files, however, are verbose and more clean. If it’s not planned and written correctly, it becomes very difficult to manage in big projects.

**7. What are the different components of a Spring application?**

A Spring application, generally consists of following components:

* Interface: It defines the functions.
* Bean class: It contains properties, its setter and getter methods, functions etc.
* Spring Aspect Oriented Programming (AOP): Provides the functionality of cross-cutting concerns.
* Bean Configuration File: Contains the information of classes and how to configure them.
* User program: It uses the function.

**8. What are the various ways of using Spring Framework?**

Spring Framework can be used in various ways. They are listed as follows:

1. As a Full-fledged Spring web application.
2. As a third-party web framework, using Spring Frameworks middle-tier.
3. For remote usage.
4. As Enterprise Java Bean which can wrap existing POJOs (Plain Old Java Objects).

The next section of Spring Interview Questions is on *Dependency Injection and IoC container*.

*[Powered by Edureka](https://www.edureka.co/mymock-interview-service" \t "_blank)*[[](https://www.edureka.co/mymock-interview-service)](https://www.edureka.co/mymock-interview-service" \t "_blank)

**[80% INTERVIEW REJECTIONS HAPPEN IN FIRST 90 SECONDS](https://www.edureka.co/mymock-interview-service" \t "_blank)**

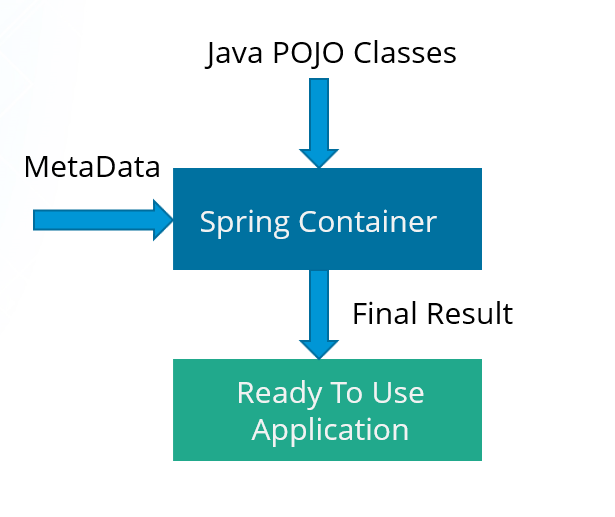
**[Take Mock Interview](https://www.edureka.co/mymock-interview-service" \t "_blank)**

* **[Get Interviewed by Industry Experts](https://www.edureka.co/mymock-interview-service" \t "_blank)**
* **[Personalized interview feedback](https://www.edureka.co/mymock-interview-service" \t "_blank)**

[BOOK A SLOT](https://www.edureka.co/mymock-interview-service" \t "_blank)

**Dependency Injection/ IoC Container – Spring Interview Questions**

**9. What is Spring IOC Container?**

****

At the core of the Spring Framework, lies the Spring container. The container creates the object, wires them together, configures them and manages their complete life cycle. The Spring container makes use of Dependency Injection to manage the components that make up an application. The container receives instructions for which objects to instantiate, configure, and assemble by reading the configuration metadata provided. This metadata can be provided either by XML, Java annotations or Java code.

**10. What do you mean by Dependency Injection?**

In Dependency Injection, you do not have to create your objects but have to describe how they should be created. You don’t connect your components and services together in the code directly, but describe which services are needed by which components in the configuration file. The IoC container will wire them up together.

**11. In how many ways can Dependency Injection be done?**

In general, dependency injection can be done in three ways, namely :

* Constructor Injection
* Setter Injection
* Interface Injection

In Spring Framework, only constructor and setter injections are used.

**12. Differentiate between constructor injection and setter injection.**

Constructor Injection vs Setter Injection

|  |  |
| --- | --- |
| **Constructor Injection** | **Setter Injection** |
| There is no partial injection. | There can be partial injection. |
| It doesn’t override the setter property. | It overrides the constructor property. |
| It will create a new instance if any modification is done. | It will not create new instance if any modification is done. |
| It works better for many properties. | It works better for few properties. |

**13. How many types of IOC containers are there in spring?**

1. **BeanFactory**: BeanFactory is like a factory class that contains a collection of beans. It instantiates the bean whenever asked for by clients.
2. **ApplicationContext**: The ApplicationContext interface is built on top of the BeanFactory interface. It provides some extra functionality on top BeanFactory.

**14. Differentiate between BeanFactory and ApplicationContext.**

BeanFactory vs ApplicationContext

|  |  |
| --- | --- |
| **BeanFactory** | **ApplicationContext** |
| It is an interface defined in org.springframework.beans.factory.**BeanFactory** | It is an interface defined in org.springframework.context.**ApplicationContext** |
| It uses Lazy initialization | It uses Eager/ Aggressive initialization |
| It explicitly provides a resource object using the syntax | It creates and manages resource objects on its own |
| It doesn’t supports internationalization | It supports internationalization |
| It doesn’t supports annotation based dependency | It supports annotation based dependency |

**15.  List some of the benefits of IoC.**

Some of the benefits of IoC are:

* It will minimize the amount of code in your application.
* It will make your application easy to test because it doesn’t require any singletons or JNDI lookup mechanisms in your unit test cases.
* It promotes loose coupling with minimal effort and least intrusive mechanism.
* It supports eager instantiation and lazy loading of the services.

Let’s move on to the next section of Spring Interview Questions, that is *Spring Beans Interview Questions*.

**Spring Beans – Spring Interview Questions**

**16. Explain Spring Beans?**

* They are the objects that form the backbone of the user’s application.
* Beans are managed by the Spring IoC container.
* They are instantiated, configured, wired and managed by a Spring IoC container
* Beans are created with the configuration metadata that the users supply to the container.

**17. How configuration metadata is provided to the Spring container?**

Configuration metadata can be provided to Spring container in following ways:

* **XML-Based configuration:**In Spring Framework, the dependencies and the services needed by beans are specified in configuration files which are in XML format. These configuration files usually contain a lot of bean definitions and application specific configuration options. They generally start with a bean tag. For example:

|  |  |
| --- | --- |
| 1  2  3 | <bean id="studentbean" class="org.edureka.firstSpring.StudentBean">   <property name="name" value="Edureka"></property>  </bean> |

* **Annotation-Based configuration**: Instead of using XML to describe a bean wiring, you can configure the bean into the component class itself by using annotations on the relevant class, method, or field declaration. By default, annotation wiring is not turned on in the Spring container. So, you need to enable it in your Spring configuration file before using it. For example:

|  |  |
| --- | --- |
| 1  2  3  4 | <beans>  <context:annotation-config/>  <!-- bean definitions go here -->  </beans> |

* **Java-based configuration:**The key features in Spring Framework’s new Java-configuration support are @Configuration annotated classes and @Bean annotated methods.

1. @Bean annotation plays the same role as the <bean/> element.

 2.@Configuration classes allows to define inter-bean dependencies by simply calling other @Bean methods in the same class.

For example:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | @Configuration  public class StudentConfig  {  @Bean  public StudentBean myStudent()  { return new StudentBean(); }  } |

**18. How many bean scopes are supported by Spring?**

The Spring Framework supports five scopes. They are:

* **Singleton:**This provides scope for the bean definition to single instance per Spring IoC container.
* **Prototype:**This provides scope for a single bean definition to have any number of object instances.
* **Request:**This provides scope for a bean definition to an HTTP-request.
* **Session:**This provides scope for a bean definition to an HTTP-session.
* **Global-session:**This provides scope for a bean definition to an Global HTTP-session.

The last three are available only if the users use a web-aware ApplicationContext.

**19. What is the Bean life cycle in Spring Bean Factory Container?**

Bean life cycle in Spring Bean Factory Container is as follows:

1. The Spring container instantiates the bean from the bean’s definition in the XML file.
2. Spring populates all of the properties using the dependency injection, as specified in the bean definition.
3. The factory calls setBeanName() by passing the bean’s ID, if the bean implements the BeanNameAware interface.
4. The factory calls setBeanFactory() by passing an instance of itself, if the bean implements the BeanFactoryAware interface.
5. preProcessBeforeInitialization() methods are called if there are any BeanPostProcessors associated with the bean.
6. If an init-method is specified for the bean, then it will be called.
7. Finally, postProcessAfterInitialization() methods will be called if there are any BeanPostProcessors associated with the bean.

To understand it in better way check the below diagram:



**20. Explain inner beans in Spring.**

A bean can be declared as an inner bean only when it is used as a property of another bean. For defining a bean, the Spring’s XML based configuration metadata provides the use of <bean> element inside the <property> or <constructor-arg>. Inner beans are always anonymous and they are always scoped as prototypes. For example, let’s say we have one Student class having reference of Person class. Here we will be creating only one instance of Person class and use it inside Student.

Here’s a Student class followed by bean configuration file:

Student.java

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | public class Student  {  private Person person;  //Setters and Getters  }  public class Person  {  private String name;  private String address;  //Setters and Getters  } |

studentbean.xml

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | <bean id=“StudentBean" class="com.edureka.Student">  <property name="person">  <!--This is inner bean -->  <bean class="com.edureka.Person">  <property name="name" value=“Scott"></property>  <property name="address" value=“Bangalore"></property>  </bean>  </property>  </bean> |

**21. Define Bean Wiring.**

When beans are combined together within the Spring container, it’s called wiring or bean wiring. The Spring container needs to know what beans are needed and how the container should use dependency injection to tie the beans together, while wiring beans.



**22. What do you understand by auto wiring and name the different modes of it?**

The Spring container is able to autowire relationships between the collaborating beans. That is, it is possible to let Spring resolve collaborators for your bean automatically by inspecting the contents of the BeanFactory.  
Different modes of bean auto-wiring are:

1. **no:** This is default setting which means no autowiring. Explicit bean reference should be used for wiring.
2. **byName:** It injects the object dependency according to name of the bean. It matches and wires its properties with the beans defined by the same names in the XML file.
3. **byType:**It injects the object dependency according to type. It matches and wires a property if its type matches with exactly one of the beans name in XML file.
4. **constructor:**It injects the dependency by calling the constructor of the class. It has a large number of parameters.
5. **autodetect:**First the container tries to wire using autowire by *constructor*, if it can’t then it tries to autowire by *byType*.

**23. What are the limitations with auto wiring?**

Following are some of the limitations you might face with auto wiring:

* **Overriding possibility:**You can always specify dependencies using <constructor-arg> and <property> settings which will override autowiring.
* **Primitive data type:**Simple properties such as primitives, Strings and Classes can’t be autowired.
* **Confusing nature:**Always prefer using explicit wiring because autowiring is less precise.

In the next section, we will discuss on *Spring Annotations Interview Questions*.

**Spring Annotations – Spring Interview Questions**

**24. What do you mean by  Annotation-based container configuration?**

Instead of using XML to describe a bean wiring, the developer moves the configuration into the component class itself by using annotations on the relevant class, method, or field declaration. It acts as an alternative to XML setups. For example:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | @Configuration  public class AnnotationConfig  {  @Bean  public MyDemo myDemo()   { return new MyDemoImpll(); }  } |

**25. How annotation wiring can be turned on in Spring?**

By default, Annotation wiring is not turned on in the Spring container. Thus, to use annotation based wiring we must enable it in our Spring configuration file by configuring **<context:annotation-config/>** element. For example:

|  |  |
| --- | --- |
| 1  2  3  4 | <beans xmlns="<a href="http://www.springframework.org/schema/beans">http://www.springframework.org/schema/beans</a>" xmlns:xsi="<a href="http://www.w3.org/2001/XMLSchema-instance">http://www.w3.org/2001/XMLSchema-instance</a>" xmlns:context="<a href="http://www.springframework.org/schema/context">http://www.springframework.org/schema/context</a>">  <context:annotation-config/>  <beans ………… />  </beans> |

**26. What’s the difference between @Component, @Controller, @Repository & @Service annotations in Spring?**

**@Component:** This marks a java class as a bean. It is a generic stereotype for any Spring-managed component. The component-scanning mechanism of spring now can pick it up and pull it into the application context.

**@Controller:** This marks a class as a Spring Web MVC controller. Beans marked with it are automatically imported into the Dependency Injection container.

**@Service:** This annotation is a specialization of the component annotation. It doesn’t provide any additional behavior over the @Component annotation. You can use @Service over @Component in service-layer classes as it specifies intent in a better way.

*[Powered by Edureka](https://www.edureka.co/mymock-interview-service" \t "_blank)*[[](https://www.edureka.co/mymock-interview-service)](https://www.edureka.co/mymock-interview-service" \t "_blank)

**[Mock interviews for you](https://www.edureka.co/mymock-interview-service" \t "_blank)**

* **[Interviews by Industry Experts](https://www.edureka.co/mymock-interview-service" \t "_blank)**
* **[Personalized detailed interview feedback](https://www.edureka.co/mymock-interview-service" \t "_blank)**
* **[Access to exclusive and curated content](https://www.edureka.co/mymock-interview-service" \t "_blank)**

[SCHEDULE NOW](https://www.edureka.co/mymock-interview-service" \t "_blank)

**@Repository:** This annotation is a specialization of the @Component annotation with similar use and functionality. It provides additional benefits specifically for DAOs. It imports the DAOs into the DI container andmakes the unchecked exceptions eligible for translation into Spring DataAccessException.

**27. What do you understand by @Required annotation?**

@Required is applied to bean property setter methods. This annotation simply indicates that the affected bean property must be populated at the configuration time with the help of an explicit property value in a bean definition or with autowiring. If the affected bean property has not been populated, the container will throw BeanInitializationException.

For example:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | public class Employee  {  private String name;  @Required  public void setName(String name)  {this.name=name; }  public string getName()  { return name; }  } |

**28. What do you understand by @Autowired annotation?**

The **@Autowired** annotation provides more accurate control over where and how autowiring should be done. This annotation is used to autowire bean on the setter methods, constructor, a property or methods with arbitrary names or multiple arguments. By default, it is a type driven injection.

For Example:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | public class Employee  {  private String name;  @Autowired  public void setName(String name)  {this.name=name; }  public string getName()  { return name; }  } |

**29. What do you understand by @Qualifier annotation?**

When you create more than one bean of the same type and want to wire only one of them with a property  you can use the **@Qualifier** annotation along with **@Autowired** to remove the ambiguity by specifying which exact bean should be wired.

For example, here we have two classes, Employee and EmpAccount respectively. In EmpAccount, using @Qualifier its specified that bean with id emp1 must be wired.

Employee.java

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | public class Employee  {  private String name;  @Autowired  public void setName(String name)  { this.name=name; }  public string getName()  { return name; }  } |

EmpAccount.java

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | public class EmpAccount  {  private Employee emp;  @Autowired  @Qualifier(emp1)  public void showName()  {  System.out.println(“Employee name : ”+emp.getName);  }  } |

**30.  What do you understand by @RequestMapping annotation?**

@RequestMapping annotation is used for mapping a particular HTTP request method to a specific class/ method in controller that will be handling the respective request. This annotation can be applied at both levels:

* **Class level**: Maps the URL of the request
* **Method level**: Maps the URL as well as HTTP request method

Next section of Spring Interview Questions is on *Data Access*.

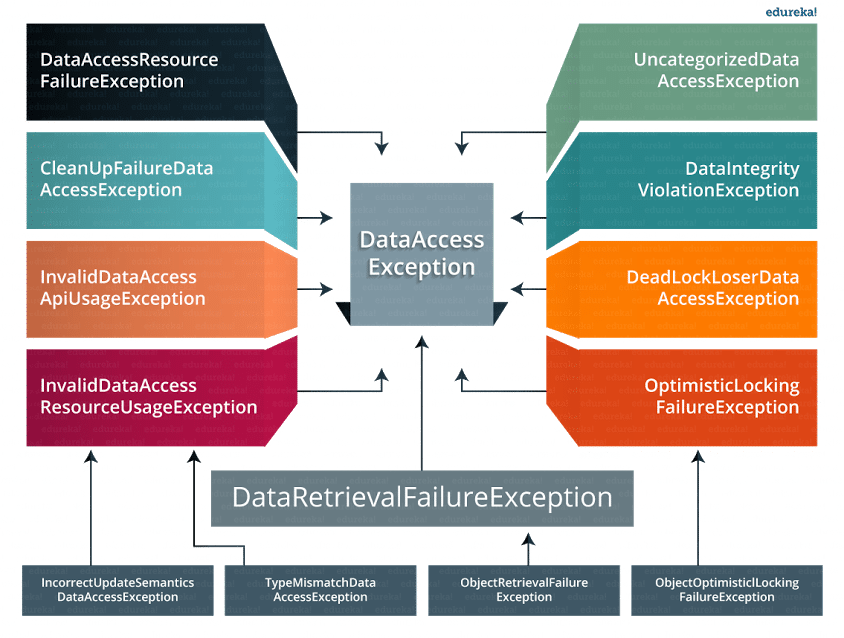
**Data Access – Spring Interview Questions**

**31. Describe Spring DAO support?**

The Data Access Object (DAO) support in Spring makes it easy to work with data access technologies like JDBC, Hibernate or JDO in a consistent way. This allows one to switch between the persistence technologies easily. It also allows you to code without worrying about catching exceptions that are specific to each of these technology.

**32. Name the exceptions thrown by the Spring DAO classes.**

See the below diagram, it depicts all the Spring DAO classes in the hierarchical order.



**33.  Which classes are present in spring JDBC API?**

Classes present in JDBC API are as follows:

1. JdbcTemplate
2. SimpleJdbcTemplate
3. NamedParameterJdbcTemplate
4. SimpleJdbcInsert
5. SimpleJdbcCall

**34. What are the ways by which Hibernate can be accessed using Spring?**

There are two ways by which we can access Hibernate using Spring:

1. Inversion of Control with a Hibernate Template and Callback
2. Extending HibernateDAOSupport and Applying an AOP Interceptor node

**35. Name the types of transaction management that Spring supports.**

Two types of transaction management are supported by Spring. They are:

1. **Programmatic transaction management:**In this, the transaction is managed with the help of programming. It provides you extreme flexibility, but it is very difficult to maintain.
2. **Declarative transaction management:**In this, the transaction management is separated from the business code. Only annotations or XML based configurations are used to manage the transactions.

**36. What are the different ORM’s supported by Spring?**

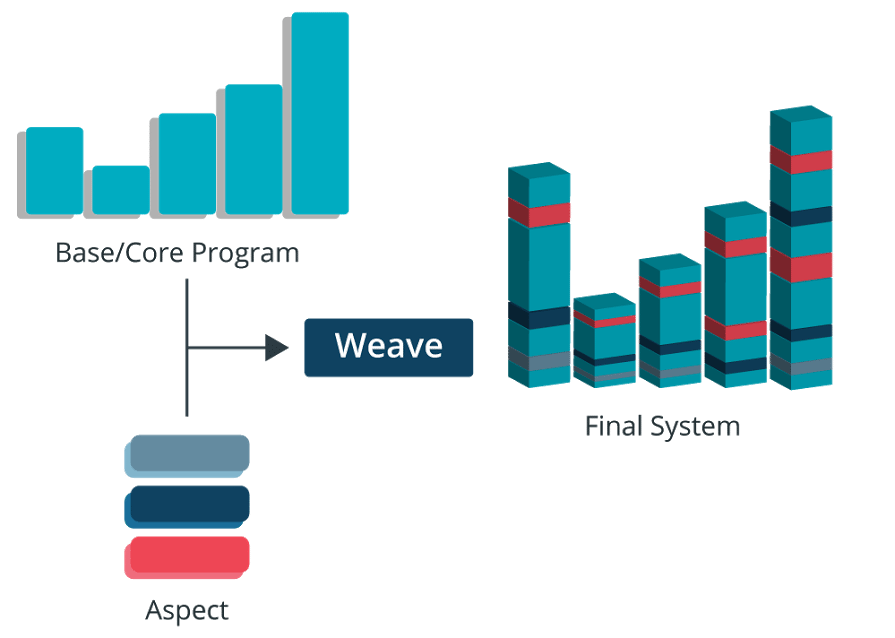
Different ORM’s supported by Spring are depicted via the below diagram:

The next section of Spring interview questions discusses on *Spring AOP Interview Questions*.

**Aspect Oriented Programming (AOP) – Spring Interview Questions**

**37. Describe AOP.**

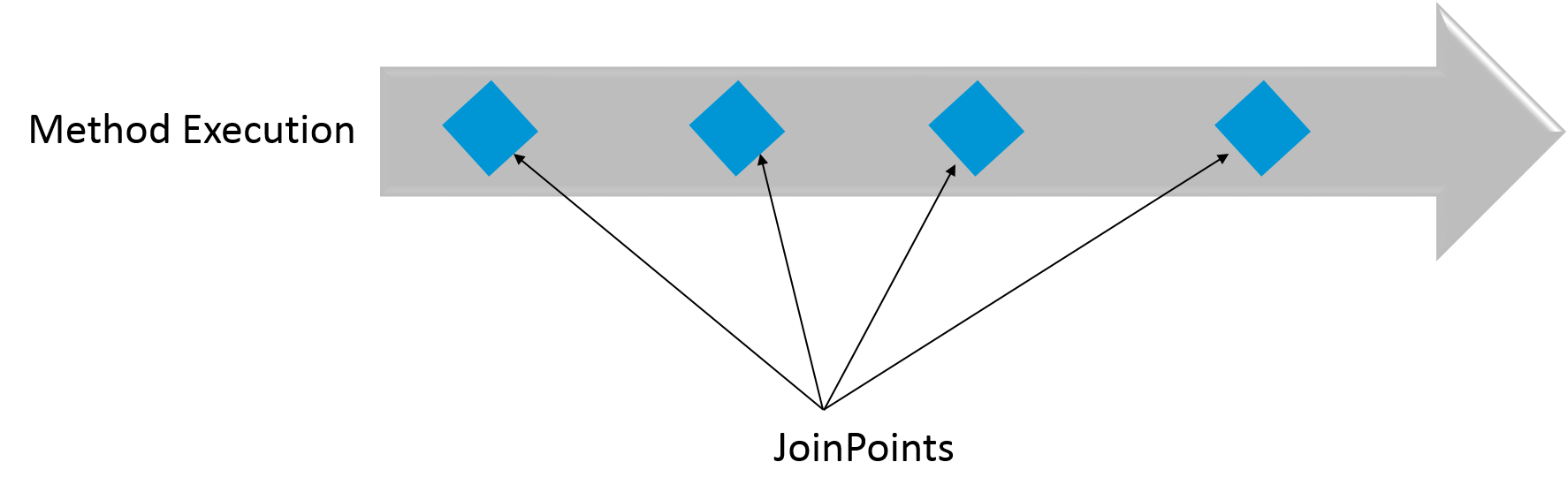
Aspect-oriented programming or AOP is a programming technique which allows programmers to modularize crosscutting concerns or behavior that cuts across the typical divisions of responsibility. Examples of cross-cutting concerns can be logging and transaction management. The core of AOP is an *aspect*. It encapsulates behaviors that can affect multiple classes into reusable modules.



**38. What do you mean by Aspect?**

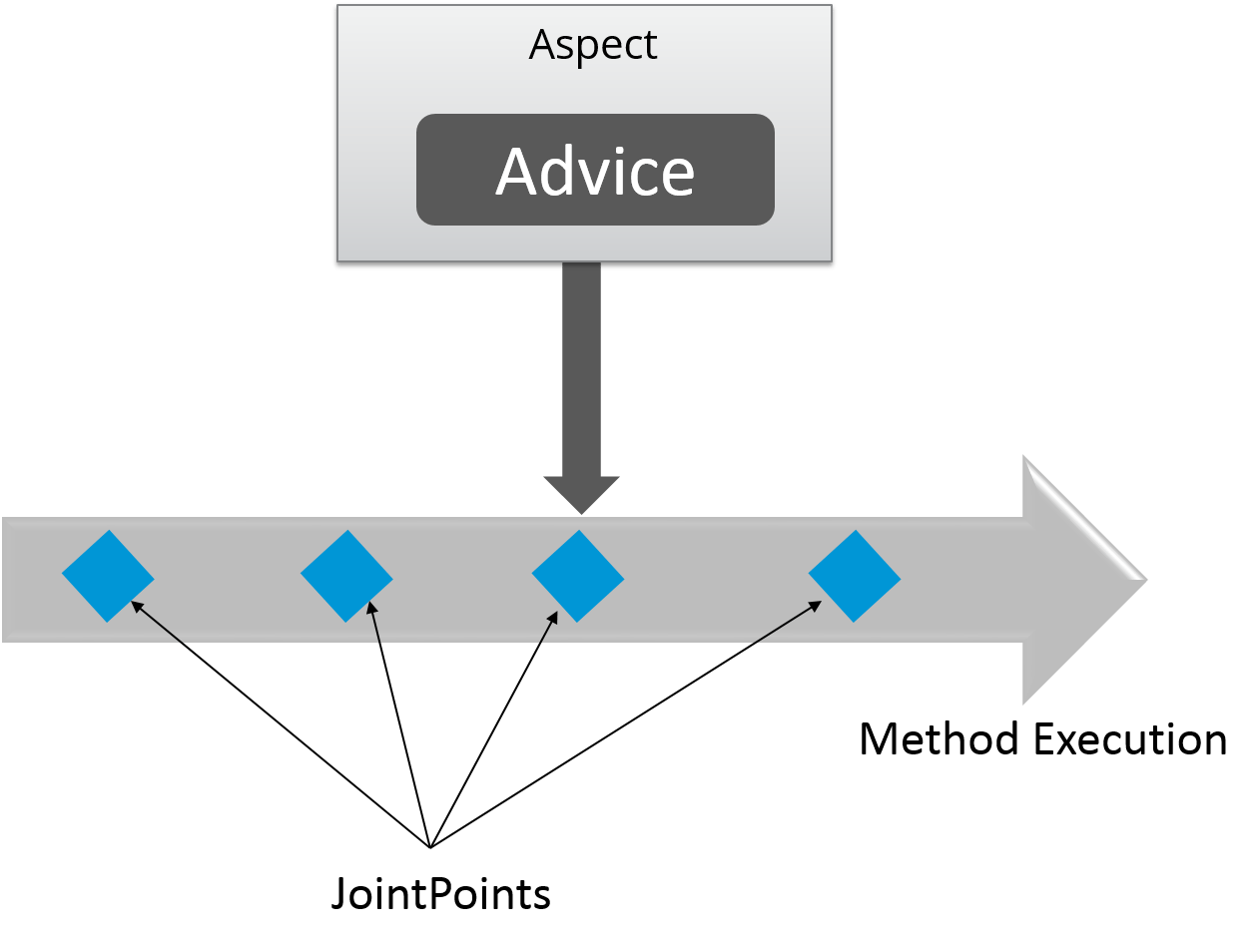
Aspect is a modularization of concern which cuts across multiple objects. Transaction management is a good example of a crosscutting concern in J2EE applications. Aspects are implemented using regular classes or regular classes annotated with the @Aspect annotation in Spring Framework.

**39. Explain JoinPoint.**

A point during the execution of a program is called JoinPoint, such as the execution of a method or the handling of an exception. In Spring AOP, a joinpoint always represents a method execution.

**40. What is an Advice?**

An Action taken by an aspect at a particular joinpoint is known as an Advice. Spring AOP uses an advice as an interceptor, maintaining a chain of interceptors “around” the join point.



**41. What are the different types of Advices?**

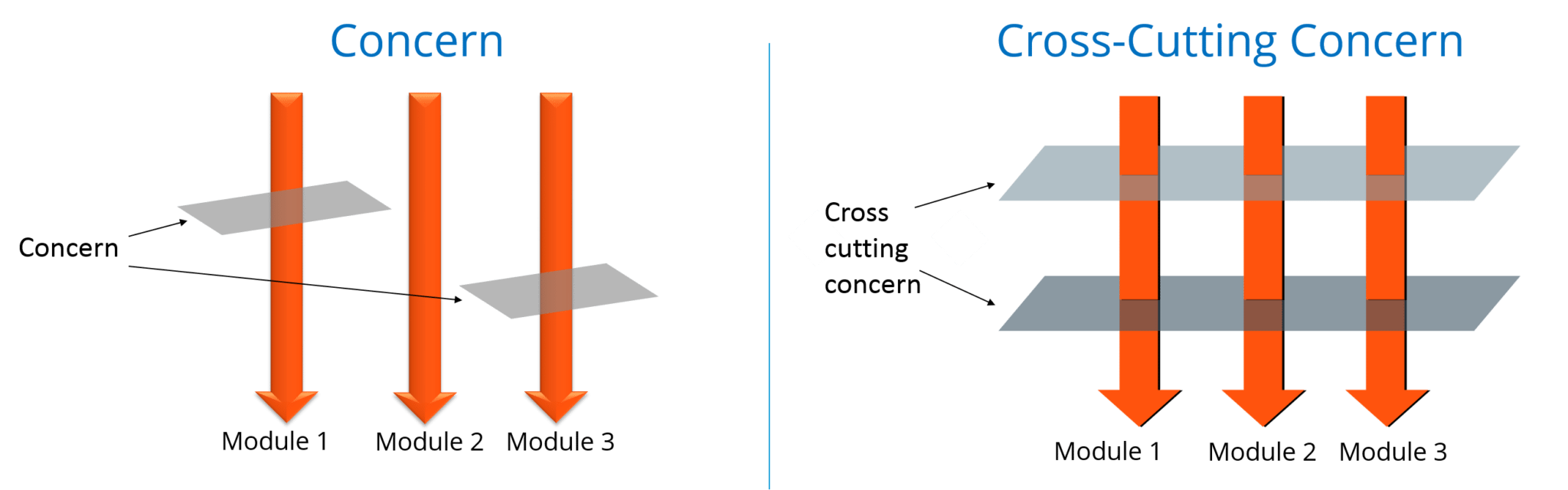
Different types of Advices in Spring AOP are:

1. **Before:** These types of advices execute before the joinpoint methods and are configured using **@Before**annotation mark.
2. **After returning:**These types of advices execute after the joinpoint methods completes executing normally and are configured using @AfterReturning annotation mark.
3. **After throwing:** These types of advices execute only if joinpoint method exits by throwing an exception and are configured using @AfterThrowing annotation mark.
4. **After (finally):**These types of advices execute after a joinpoint method, regardless of the method’s exit whether normally or exceptional return and are configured using @After annotation mark.
5. **Around:**These types of advices execute before and after a joinpoint and are configured using @Around annotation mark.

**42. Point out the difference between concern and cross-cutting concern in Spring AOP?**

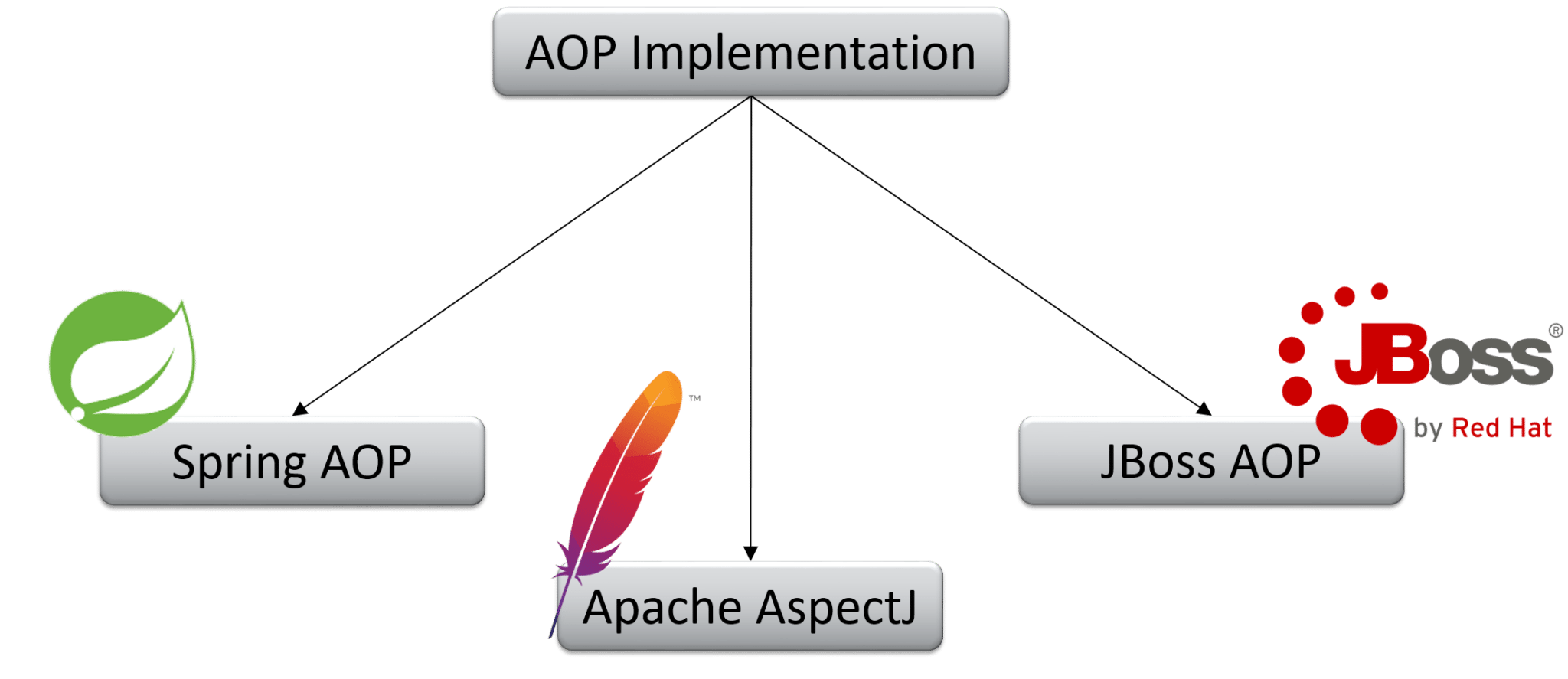
The concern is the behavior we want to have in a particular module of an application. It can be defined as a functionality we want to implement.

The cross-cutting concern is a concern which is applicable throughout the application. This affects the entire application. For example, logging, security and data transfer are the concerns needed in almost every module of an application, thus they are the cross-cutting concerns.



**43. What are the different AOP implementations?**

Different AOP implementations are depicted by the below diagram:



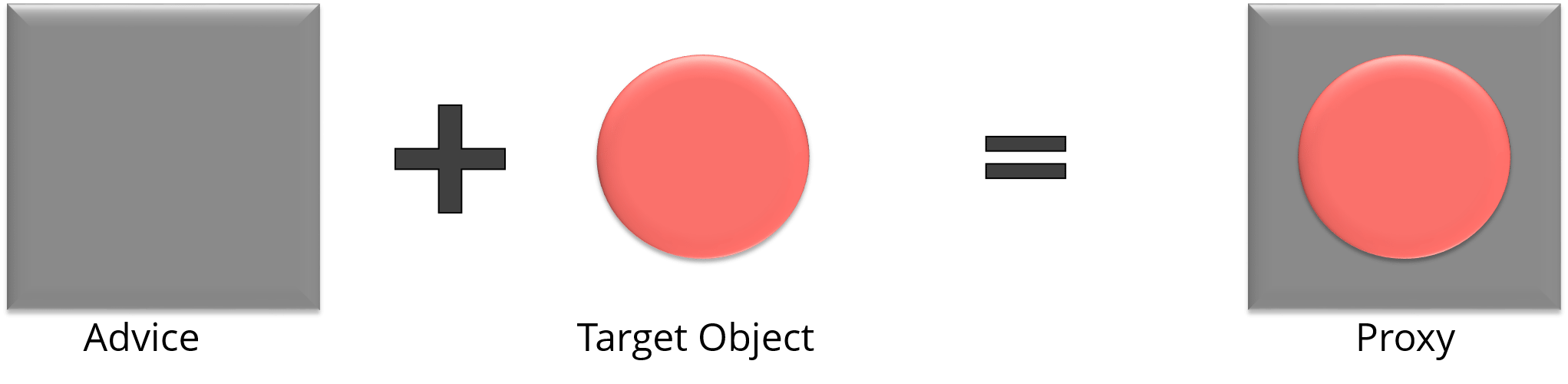
**44. What are the difference between Spring AOP and AspectJ AOP?**

Spring AOP vs AspectJ AOP

|  |  |
| --- | --- |
| **Spring AOP** | **AspectJ AOP** |
| Runtime weaving through proxy is done | Compile time weaving through AspectJ Java tools is done |
| It supports only method level PointCut | It suports field level Pointcuts |
| It is DTD based | It is schema based and Annotation configuration |

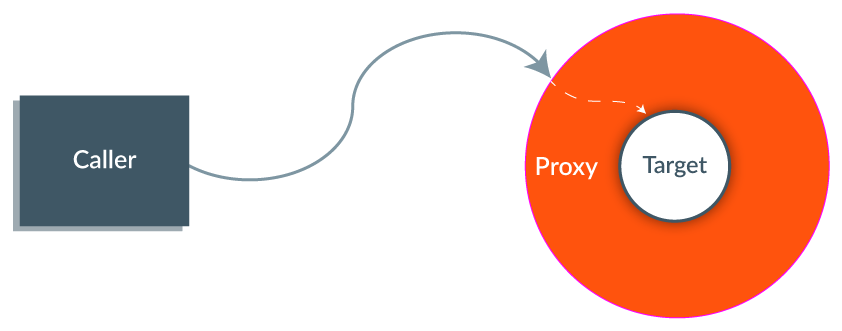
**45. What do you mean by Proxy in Spring Framework?**

An object which is created after applying advice to a target object is known as a Proxy. In case of client objects the target object and the proxy object are the same.



**46. In Spring, what is Weaving?**

The process of linking an aspect with other application types or objects to create an advised object is called Weaving. In Spring AOP, weaving is performed at runtime. Refer the below diagram:

The last section of Spring interview questions is on *Spring MVC Interview Questions*.

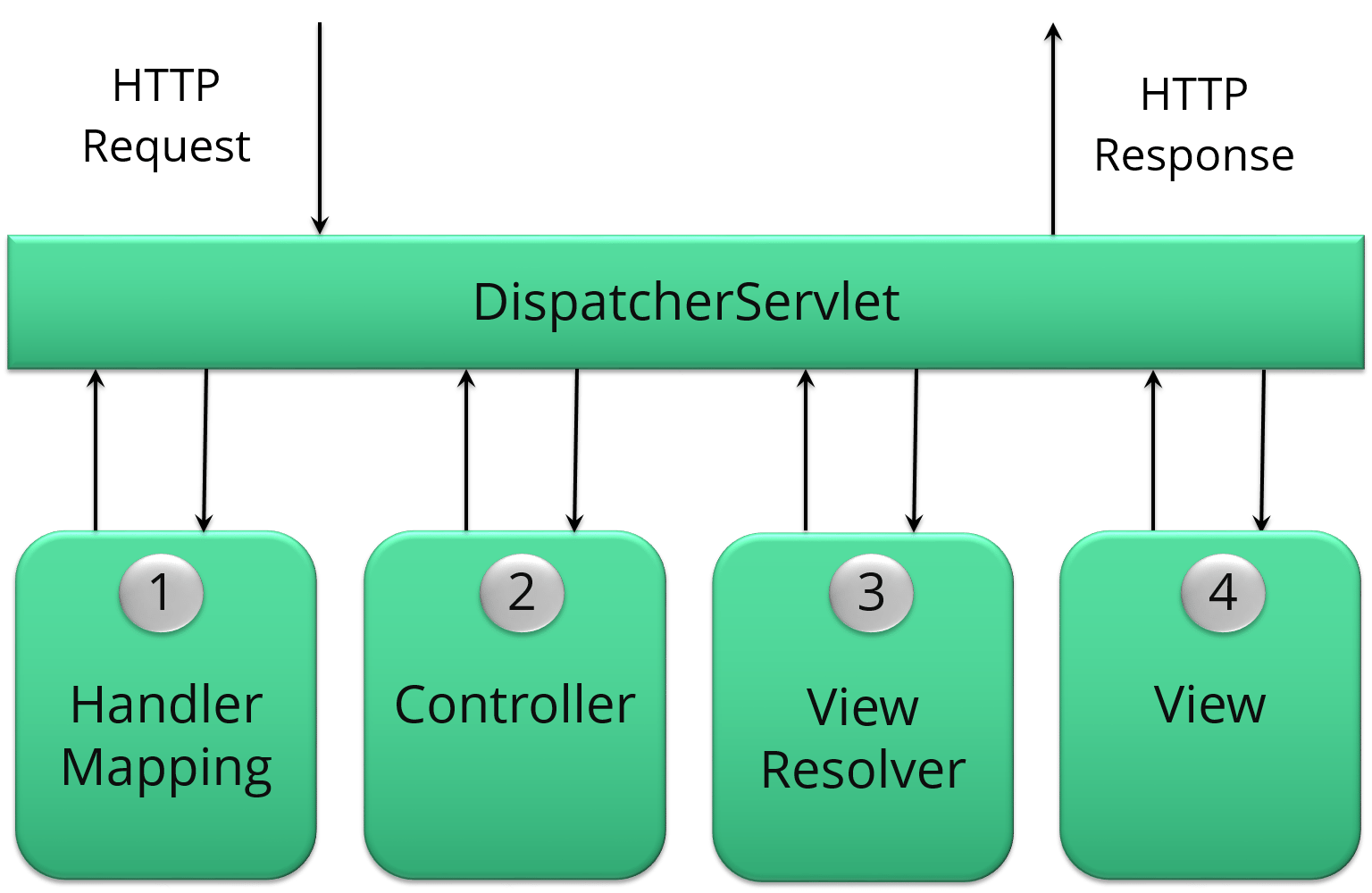
**MVC (Model-View-Controller) – Spring Interview Questions**

**47. What do you mean by Spring MVC framework?**

The Spring web MVC framework provides model-view-controller architecture and ready to use components that are used to develop flexible and loosely coupled web applications. The MVC pattern helps in separating the different aspects of the application like input logic, business logic and UI logic, while providing a loose coupling between all these elements.

**48. Describe DispatcherServlet.**

The DispatcherServlet is the core of Spring Web MVC framework. It handles all the HTTP requests and responses. The DispatcherServlet receives the entry of handler mapping from the configuration file and forwards the request to the controller. The controller then returns an object of Model And View. The DispatcherServlet checks the entry of view resolver in the configuration file and calls the specified view component.

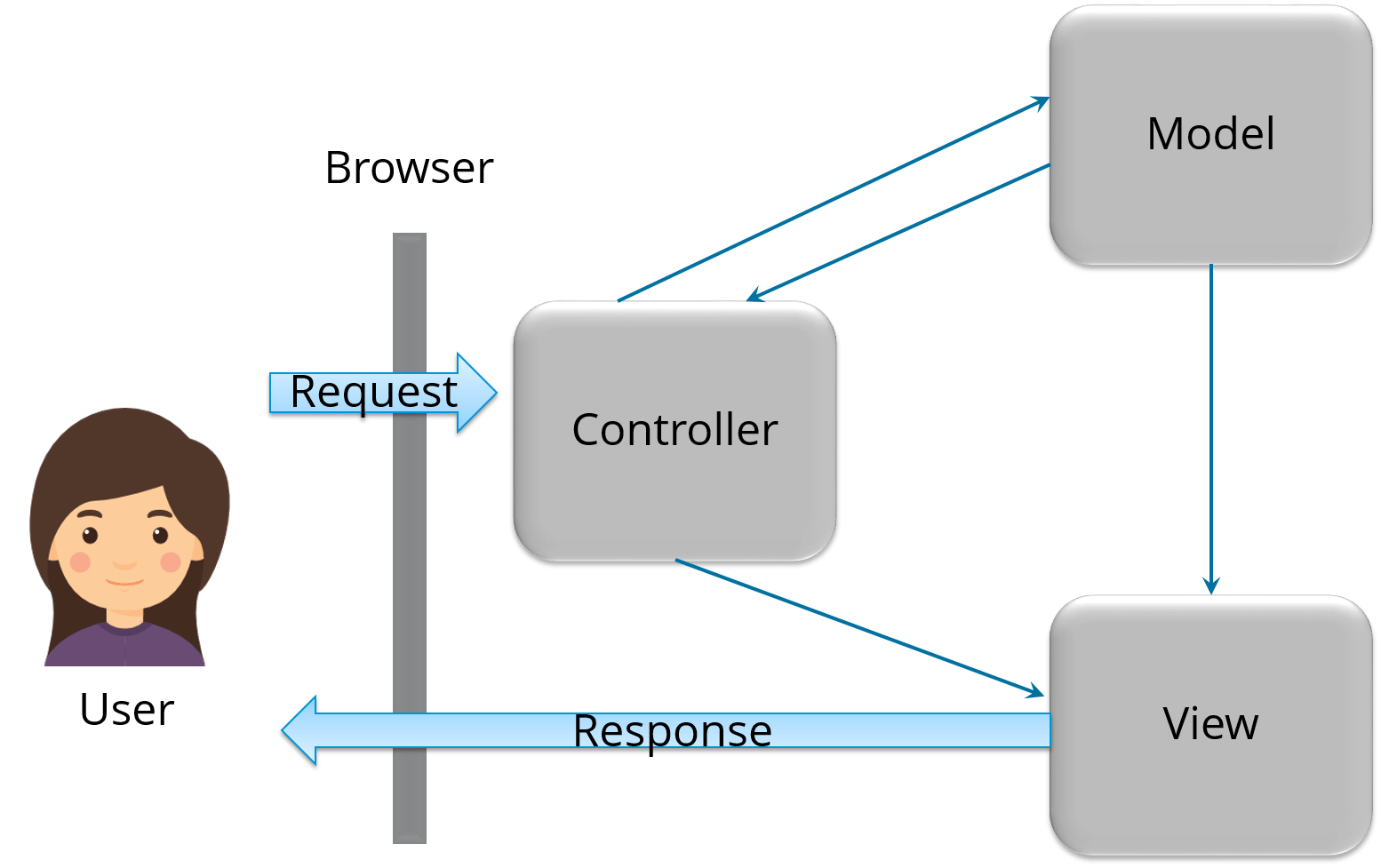


**49. Explain WebApplicationContext.**

The WebApplicationContext is an extension of the plain ApplicationContext. It has some extra features that are necessary for web applications. It differs from a normal ApplicationContext in terms of its capability of resolving themes and in deciding which servlet it is associated with.

**50. In Spring MVC framework, what is controller?**

Controllers provide access to the application behavior. These behaviors are generally defined through a service interface. Controllers interpret the user input and transform it into a model which is represented to the user by the view. In Spring, controller is implemented in a very abstract way. It also enables you to create a wide variety of controllers.



**Hibernate Interview Questions for beginners**

**Q1. What is Hibernate?**

[*Hibernate*](https://www.edureka.co/blog/what-is-hibernate-in-java/) is one of the most popular [*Java frameworks*](https://www.edureka.co/blog/java-frameworks/) that simplify the development of Java application to interact with the database. It is an Object-relational mapping (ORM) tool. Hibernate also provides a reference implementation of Java API.

It is referred as a framework which comes with an abstraction layer and also handles the implementations internally. The implementations include tasks like writing a query for [*CRUD*](https://www.edureka.co/blog/node-js-mysql-tutorial/)operations or establishing a connection with the databases, etc.

Hibernate develops persistence logic, which stores and processes the data for longer use. It is a lightweight tool and most importantly **open-sourced**which gives it an edge over other frameworks.

**Q2. What are the major advantages of Hibernate Framework?**

* It is open-sourced and lightweight.
* Performance of Hibernate is very fast.
* Helps in generating database independant queries.
* Provides facilities to automatically create a table.
* It provides query statistics and database status.

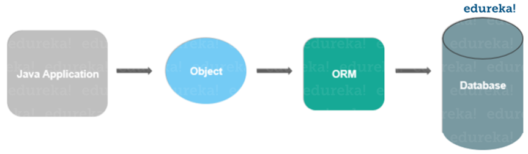
**Q3. What are the advantages of using Hibernate over JDBC?**

Major advantages of using Hibernate over JDBC are:

1. Hibernate eliminates a lot of boiler-plate code that comes with [*JDBC API*](https://www.edureka.co/blog/connect-mysql-database-in-java/), the code looks cleaner and readable.
2. This Java framework supports [*inheritance*](https://www.edureka.co/blog/inheritance-in-java/), associations, and collections. These features are actually not present in JDBC.
3. HQL (Hibernate Query Language) is more object-oriented and close to Java. But for JDBC, you need to write native SQL queries.
4. Hibernate implicitly provides transaction management whereas, in JDBC API, you need to write code for transaction management using *commit* and *rollback*.
5. JDBC throws SQLException that is a checked exception, so you have to write a lot of try-catch block code. Hibernate wraps JDBC exceptions and throw *JDBCException*or *HibernateException* which are the unchecked exceptions, so you don’t have to write code to handle it has built-in transaction management which helps in removing the usage of try-catch blocks.

**Q4. What is an ORM tool?**

It is basically a technique that maps the object that is stored in the database. An ORM tool helps in simplifying data creation, manipulation, and access. It internally uses the Java API to interact with the databases.



**Q5. Why use Hibernate Framework?**

Hibernate overcomes the shortcomings of other technologies like [*JDBC*](https://www.edureka.co/blog/connect-mysql-database-in-java).

* It overcomes the database dependency faced in the JDBC.
* Changing of the databases cost a lot working on JDBC, hibernate overcomes this problem with flying colors.
* Code portability is not an option while working on JDBC. This is easily handled by Hibernate.
* Hibernate strengthens the object level relationship.
* It overcomes the [*exception-handling*](https://www.edureka.co/blog/java-exception-handling) part which is mandatory while working on JDBC.
* It reduces the length of code with increased readability by overcoming the boilerplate problem.

**Q6. What are the different functionalities supported by Hibernate?**

* Hibernate is an ORM tool.
* Hibernate uses Hibernate Query Language(HQL) which makes it database-independent.
* It supports auto DDL operations.
* This Java framework also has an *Auto Primary Key Generation* support.
* Supports cache memory.
* Exception handling is not mandatory in the case of Hibernate.

**Q7. What are the technologies that are supported by Hibernate?**

Hibernate supports a variety of technologies, like:

* XDoclet Spring
* [*Maven*](https://www.edureka.co/blog/create-selenium-maven-project/)
* Eclipse Plug-ins
* *J2EE*

**Q8. What is HQL?**

HQL is the acronym of Hibernate Query Language. It is an Object-Oriented Query Language and is independent of the database.

**Q9. How to achieve mapping in Hibernate?**

Association mappings are one of the key features of Hibernate. It supports the same associations as the relational database model. They are:

* One-to-One associations
* Many-to-One associations
* Many-to-Many associations

You can map each of them as a uni- or bidirectional association.

**Q10. Name some of the important interfaces of Hibernate framework?**

Hibernate interfaces are:

* **SessionFactory** (org.hibernate.SessionFactory)
* **Session** (org.hibernate.Session)
* **Transaction** (org.hibernate.Transaction)

**Q11. What is One-to-One association in Hibernate?**

In this type of mapping,  you only need to model the system for the entity for which you want to navigate the relationship in your query or domain model. You need an entity attribute that represents the association, so annotate it with an *@OneToOne*annotation.

**Q12. What is One-to-Many association in Hibernate?**

In this type of association, one object can be associated with multiple/different objects. Talking about the mapping, the One-to-Many mapping is implemented using a [*Set Java*](https://www.edureka.co/blog/sets-in-java/) collection that does not have any redundant element. This *One-to-Many* element of the set indicates the relation of one object to multiple objects.

**Q13. What is Many-to-Many association in Hibernate?**

Many-to-Many mapping requires an entity attribute and a *@ManyToMany* annotation. It can either be unidirectional and bidirectional. In **Unidirectional**, the attributes model the association and you can use it to navigate it in your domain model or JPQL queries. The annotation tells Hibernate to map a Many-to-Many association. The **bidirectional** relationship, mapping allows you to navigate the association in both directions.

**Q14. How to integrate Hibernate and Spring?**

[*Spring*](https://www.edureka.co/blog/spring-tutorial/) is also one of the most commonly used Java frameworks in the market today. Spring is a JavaEE Framework and Hibernate is the most popular ORM framework. This is why Spring Hibernate combination is used in a lot of enterprise applications.

Following are the steps you should follow to integrate Spring and Hibernate.

[[](https://www.edureka.co/java-j2ee-training-course)](https://www.edureka.co/java-j2ee-training-course" \t "_blank)

**[Java, J2EE & SOA Certification Training](https://www.edureka.co/java-j2ee-training-course" \t "_blank)**

* *[Instructor-led Sessions](https://www.edureka.co/java-j2ee-training-course" \t "_blank)*
* *[Real-life Case Studies](https://www.edureka.co/java-j2ee-training-course" \t "_blank)*
* *[Assignments](https://www.edureka.co/java-j2ee-training-course" \t "_blank)*
* *[Lifetime Access](https://www.edureka.co/java-j2ee-training-course" \t "_blank)*

[Explore Curriculum](https://www.edureka.co/java-j2ee-training-course" \t "_blank)

1. Add Hibernate-entity manager, Hibernate-core and Spring-ORM dependencies.
2. Create Model classes and corresponding DAO implementations for database operations. The DAO classes will use *SessionFactory* that will be injected by the Spring Bean configuration.
3. Note that you don’t need to use Hibernate Transaction Management, as you can leave it to the Spring declarative transaction management using *@Transactional annotation.*

**Q15. What do you mean by Hibernate Configuration File?**

*Hibernate Configuration File* mainly contains database-specific configurations and are used to initialize *SessionFactory*. Some important parts of the Hibernate Configuration File are Dialect information, so that hibernate knows the database type and mapping file or class details.

**Hibernate Interview Questions for intermediate**

**Q16. Mention some important annotations used for Hibernate mapping?**

Hibernate supports JPA annotations. Some of the major annotations are:

1. **javax.persistence.Entity:** This is used with model classes to specify they are entity beans.
2. **javax.persistence.Table:** It is used with entity beans to define the corresponding table name in the database.
3. **javax.persistence.Access:** Used to define the access type, field or property. The default value is field and if you want Hibernate to use the getter/setter methods then you need to set it to a property.
4. **javax.persistence.Id:** Defines the primary key in the entity bean.
5. **javax.persistence.EmbeddedId:** It defines a composite primary key in the entity bean.
6. **javax.persistence.Column:** Helps in defining the column name in the database table.
7. **javax.persistence.GeneratedValue:**It defines the strategy to be used for the generation of the primary key. It is also used in conjunction with *javax.persistence.GenerationType* enum.

**Q17. What is Session in Hibernate and how to get it?**

Hibernate Session is the interface between Java application layer and Hibernate. It is used to get a physical connection with the database. The *Session* object created is lightweight and designed to be instantiated each time an interaction is needed with the database. This *Session* provides methods to create, read, update and delete operations for a constant object. To get the Session, you can execute HQL queries, SQL native queries using the *Session* object.

**Q18. What is Hibernate SessionFactory?**

SessionFactory is the factory class that is used to get the Session objects. The SessionFactory is a heavyweight object so usually, it is created during application startup and kept for later use. This *SessionFactory* is a thread-safe object which is used by all the threads of an application. If you are using multiple databases then you would have to create multiple *SessionFactory* objects.

**Q19. What is the difference between openSession and getCurrentSession?**

This *getCurrentSession()* method returns the session bound to the context and for this to work, you need to configure it in Hibernate configuration file. Since this session object belongs to the context of Hibernate, it is okay if you don’t close it. Once the *SessionFactory* is closed, this session object gets closed.

*openSession()* method helps in opening a new session. You should close this session object once you are done with all the database operations. And also, you should open a new session for each request in a multi-threaded environment.

**Q20. What do you mean by Hibernate configuration file?**

The following steps help in configuring Hibernate file:

1. First, identify the POJOs (Plain Old Java Objects) that have a database representation.
2. Identify which properties of POJOs need to be continued.
3. Annotate each of the POJOs in order to map the Java objects to columns in a database table.
4. Create a database schema using the schema export tool which uses an existing database, or you can create your own database schema.
5. Add Hibernate Java libraries to the application’s classpath.
6. Create a Hibernate *XML configuration file* that points to the database and the mapped classes.
7. In the Java application, you can create a Hibernate Configuration object that refers to your XML configuration file.
8. Also, build a Hibernate SessionFactory object from the Configuration object.
9. Retrieve the Hibernate Session objects from the SessionFactory and write down the data access logic for your application (create, retrieve, update, and delete).

**Q21. What are the key components of a Hibernate configuration object?**

The configuration provides 2 key components, namely:

* *Database Connection:* This is handled by one or more configuration files.
* *Class Mapping setup*: It helps in creating the connection between Java classes and database tables.

**Q22. Discuss the Collections in Hibernate**

Hibernate provides the facility to persist the Collections. A [*Collection*](https://www.edureka.co/blog/java-collections/)basically can be a List, Set, Map, Collection, Sorted Set, Sorted Map. java.util.List, java.util.Set, java.util.Collection, etc, are some of the real interface types to declared the persistent collection-value fields. Hibernate injects persistent Collections based on the type of interface. The collection instances generally behave like the types of value behavior.

**Q23. What are the collection types in Hibernate?**

There are five collection types in hibernate used for one-to-many relationship mappings.

* Bag
* Set
* List
* Array
* Map

**Q24. What is a Hibernate Template class?**

When you integrate Spring and Hibernate, Spring ORM provides two helper classes – HibernateDaoSupport and HibernateTemplate. The main reason to use them was to get two things, the Session from Hibernate and Spring Transaction Management. However, from Hibernate 3.0.1, you can use the SessionFactory getCurrentSession() method to get the current session. The major advantage of using this Template class is the **exception translation** but that can be achieved easily by using *@Repository* annotation with service classes.

**Q25. What are the benefits of using Hibernate template?**

The following are the benefits of using this Hibernate template class:

* Automated Session closing ability.
* The interaction with the Hibernate Session is simplified.
* Exception handling is automated.

**Q26. Which are the design patterns that are used in Hibernate framework?**

There are a few design patterns used in Hibernate Framework, namely:

* *Domain Model Pattern:* An object model of the domain that incorporates both behavior as well as data.
* *Data Mapper:*A layer of the map that moves data between objects and a database while keeping it independent of each other and the map itself.
* *Proxy Pattern:*It is used for lazy loading.
* *Factory Pattern:*Used in SessionFactory.

**Q27. Define Hibernate Validator Framework**

Data validation is considered as an integral part of any application. Also, data validation is used in the presentation layer with the use of Javascript and the server-side code before processing. It occurs before persisting it in order to make sure it follows the correct format. Validation is a cross-cutting task, so we should try to keep it apart from the business logic. This Hibernate Validator provides the reference implementation of bean validation specs.

**Q28. What is Dirty Checking in Hibernate?**

Hibernate incorporates *Dirty Checking*feature that permits developers and users to avoid time-consuming write actions. This Dirty Checking feature changes or updates fields that need to be changed or updated, while keeping the remaining fields untouched and unchanged.

**Q29. How can you share your views on mapping description files?**

* Mapping description files are used by the Hibernate to configure functions.
* These files have the **\*.hbm** extension, which facilitates the mapping between database tables and Java class.
* Whether to use mapping description files or not this entirely depends on business entities.

**Q30. What is meant by Light Object Mapping?**

The means that the syntax is hidden from the business logic using specific design patterns. This is one of the valuable levels of ORM quality and this Light Object Mapping approach can be successful in case of applications where there are very fewer entities, or for applications having data models that are metadata-driven.

**Hibernate Interview Questions for experienced**

**Q31. What is meant by Hibernate tuning?**

Optimizing the performance of Hibernate applications is known as Hibernate tuning.

The performance tuning strategies for Hibernate are:

1. SQL Optimization
2. Session Management
3. Data Caching

**Q32. What is Transaction Management in Hibernate? How does it work?**

Transaction Management is a property which is present in the Spring framework. Now, what role does it play in Hibernate?

Transaction Management is a process of managing a set of commands or statements. In hibernate, Transaction Management is done by transaction interface. It maintains abstraction from the transaction implementation (JTA, JDBC). A transaction is associated with Session and is instantiated by calling *session.beginTransaction()*.

**Q33. How do you integrate Hibernate with Struts2 or Servlet web applications?**

You can integrate any Struts application with Hibernate. There are no extra efforts required.

1. Register a custom *ServletContextListener*.
2. In the *ServletContextListener* class, first, initialize the Hibernate Session, store it in the servlet context.
3. Action class helps in getting the Hibernate Session from the servlet context, and perform other Hibernate task as normal.

**Q34. What are the different states of a persistent entity?**

It may exist in one of the following 3 states:

* Transient: This is not associated with the Session and has no representation in the database.
* Persistent: You can make a transient instance persistent by associating it with a Session.
* Detached: If you close the Hibernate Session, the persistent instance will become a detached instance.

**Q35. How can the primary key be created by using Hibernate?**

A Primary key is a special relational database table column designated to uniquely identify all table records. It is specified in the configuration file *hbm.xml*. The generator can also be used to specify how a Primary key can be created in the database.

|  |  |
| --- | --- |
| 1  2  3  4 | <id name="ClassID" type="string" >  <column name= "columnID" length="10" >  <generator/>  </id> |

**Q36. Explain about Hibernate Proxy and how it helps in Lazy loading?**

* Hibernate uses a proxy object in order to support *Lazy loading*.
* When you try loading data from tables, Hibernate doesn’t load all the mapped objects.
* After you reference a child object through getter methods, if the linked entity is not present in the session cache, then the proxy code will be entered to the database and load the linked object.
* It uses Java assist to effectively and dynamically generate sub-classed implementations of your entity objects.

**Q37. How can we see Hibernate generated SQL on console?**

In order to view the SQL on a console, you need to add following in Hibernate configuration file to enable viewing SQL on the console for debugging purposes:

|  |  |
| --- | --- |
| 1 | <property name="show\_sql">true</property> |

**Q38. What is Query Cache in Hibernate?**

Hibernate implements a separate cache region for queries resultset that integrates with the Hibernate second-level cache. This is also an optional feature and requires a few more steps in code.

***Note:****This is only useful for queries that are run frequently with the same parameters.*

**Q39. What is the benefit of Native SQL query support in Hibernate?**

Hibernate provides an option to execute Native SQL queries through the use of the *[SQLQuery](https://www.edureka.co/blog/insert-query-sql/" \t "_blank)*object. For normal scenarios, it is however not the recommended approach because you might lose other benefits like Association and Hibernate first-level caching.

Native SQL Query comes handy when you want to execute database-specific queries that are not supported by Hibernate API such query hints or the*Connect* keyword in Oracle Database.

**Q40. What is Named SQL Query?**

Hibernate provides another important feature called Named Query using which you can define at a central location and use them anywhere in the code.

You can create named queries for both HQL as well as for Native SQL. These Named Queries can be defined in Hibernate mapping files with the help of JPA annotations @NamedQuery and @NamedNativeQuery.

**Q41. When do you use merge() and update() in Hibernate?**

This is one of the tricky Hibernate Interview Questions asked.

*update():* If you are sure that the Hibernate Session does not contain an already persistent instance with the same *id* .  
*merge():*  Helps in merging your modifications at any time without considering the state of the Session.

**Q42. Difference between get() vs load() method in Hibernate?**

This is one of the most frequently asked Hibernate Interview Questions. The key difference between the get() and load() method is:

*load():*It will throw an exception if an object with an ID passed to them is not found.  
*get():*Will return null.

*load():*It can return proxy without hitting the database unless required.  
*get():*It always goes to the database.

So sometimes using *load()* can be faster than the *get()* method.

**Q43. Difference between the first and second level cache in Hibernate?**

The *first-level cache* is maintained at Session level while the *second level cache* is maintained at a SessionFactory level and is shared by all sessions.

**Q44. Difference between Session and SessionFactory in Hibernate?**

This is yet another popular Hibernate Interview Question asked.

* *A Session* is a single-threaded, short-lived object. It provides the first-level cache.
* SessionFactory is immutable and shared by all Session. It also lives until the Hibernate is running. It also provides the second-level cache.

**Q45. Difference between save() and saveOrUpdate() method of Hibernate?**

Even though *save()* and *saveOrUpdate()* method is used to store an object into Database, the key difference between them is that *save()* can only **Insert** records but *saveOrUpdate()* can either Insert or Update records.

**Q46. Difference between sorted and ordered collection in Hibernate?**

*sorted collection* sort the data in JVM’s heap memory using Java’s collection framework sorting methods. *The ordered collection* is sorted using order by clause in the database itself.

***Note:****A sorted collection is more suited for small dataset but for a large dataset, it’s better to use ordered collection to avoid*

[[](https://www.edureka.co/java-j2ee-training-course)](https://www.edureka.co/java-j2ee-training-course" \t "_blank)

**[Java, J2EE & SOA Certification Training](https://www.edureka.co/java-j2ee-training-course" \t "_blank)**

[Weekday / Weekend BatchesSee Batch Details](https://www.edureka.co/java-j2ee-training-course" \t "_blank)

**Q47. Difference between the transient, persistent and detached state in Hibernate?**

***Transient* state:** New objects are created in the Java program but are not associated with any Hibernate Session.

***Persistent state:*** An object which is associated with a Hibernate session is called Persistent object. While an object which was earlier associated with Hibernate session but currently it’s not associate is known as a detached object. You can call save() or persist() method to store those object into the database and bring them into the Persistent state.

***Detached state***: You can re-attach a detached object to Hibernate sessions by calling either update() or saveOrUpdate() method.

**Q48. Difference between managed associations and Hibernate associations?**

***Managed associations:*** Relate to container management persistence and are bi-directional.

***Hibernate Associations:*** These associations are unidirectional.

**Q49. What are the best practices that Hibernate recommends for persistent classes?**

* All Java classes that will be persisted need a default constructor.
* All classes should contain an ID in order to allow easy identification of your objects within Hibernate and the database. This property maps to the primary key column of a database table.
* All attributes that will be persisted should be declared private and have **getXXX** and **setXXX** methods defined in the JavaBean style.
* A central feature of Hibernate, proxies, depends upon the persistent class being either non-final, or the implementation of an interface that declares all public methods.
* All classes that do not extend or implement some specialized classes and interfaces required by the EJB framework.

**Q50. What are the best practices to follow with Hibernate framework?**

* Always check the primary key field access, if it’s generated at the database layer then you should not have a setter for this.
* By default hibernate set the field values directly, without using setters. So if you want Hibernate to use setters, then make sure proper access is defined as @Access(value=AccessType.PROPERTY).
* If access type is property, make sure annotations are used with getter methods and not setter methods. Avoid mixing of using annotations on both filed and getter methods.
* Use native sql query only when it can’t be done using HQL, such as using the database-specific feature.
* If you have to sort the collection, use ordered list rather than sorting it using Collection API.
* Use named queries wisely, keep it at a single place for easy debugging. Use them for commonly used queries only. For entity-specific query, you can keep them in the entity bean itself.
* For web applications, always try to use JNDI DataSource rather than configuring to create a connection in hibernate.
* Avoid Many-to-Many relationships, it can be easily implemented using bidirectional One-to-Many and Many-to-One relationships.
* For collections, try to use Lists, maps and sets. Avoid array because you don’t get benefit of lazy loading.
* Do not treat exceptions as recoverable, roll back the Transaction and close the Session. If you do not do this, Hibernate cannot guarantee that the in-memory state accurately represents the persistent state.
* Prefer DAO pattern for exposing the different methods that can be used with entity bean
* Prefer lazy fetching for associations

#### ****Q1.****What is a Docker?

**Answer:**  
A Docker is defined as the platform for containerizing the applications to isolate it from each other in order to ensure high availability and more efficiency irrespective of the environments such as Development, Testing or Production. All the application related dependencies such as libraries, jar files, server related configurations, infrastructure-related elements will be packaged and formed as container called containerized application which does not need any dependency and works independently. It ensures the application to be run irrespective of the external factors. Containers in Docker have support from Docker Engine and Host Operating System to support all the operational or infrastructural related dependencies.

#### ****Q2.****What are the components of Docker Architecture and explain?

**Answer:**  
This is the common Docker Interview Questions asked in an interview. The Docker works on client-server architecture. The Docker client establishes communication with the Docker Daemon. The Docker client and Daemon can run on the same system. A Docket client can also be connected to a remote Docker Daemon. The different types of Docker [components in a Docker architecture](https://www.educba.com/docker-architecture/) are–

1. **Docker Client:** This performs Docker build pull and run operations to establish communication with the Docker Host. The Docker command uses Docker API to call the queries to be run.
2. **Docker Host**: This component contains Docker Daemon, Containers and its images. The images will be the kind of metadata for the applications which are containerized in the containers. The Docker Daemon establishes a connection with Registry.
3. **Registry:** This component will be storing the Docker images. The public registries are Docker Hub and Docker Cloud which can be s used by anyone.

#### ****Q3.****What is Docker Container?

**Answer:**  
A Docker Container is a form of encapsulation to the application which holds all the dependencies which share the kernel with other containers in the duration of running the isolated processes on the host operating system. A Docker container can be created by creating a Docker image. These Docker images can be run after that using [Docker commands](https://www.educba.com/docker-commands-cheat-sheet/). Docker containers are the instances of the Docker images at the runtime. Docker images can be stored in any public hosts or private hosts like Docker hub. Docker Image is a set of files which can be run in an isolated process.

#### ****Q4.****What are Docker Image and Docker Hub?

**Answer:**  
The Docker Image is a set of files and a combination of parameters which will allow creating the instances to run in separate containers as an isolated process. The Docker hub is a kind of repository to the images where these images can be stored and this access is public. The Docker run command can be used to create the instance called container which can be run using the Docker image. Docker hub is the largest public repository of the image containers which is being maintained by the community of developers and individual contributors.

#### ****Q5.****What are the different functionalities and applications of using Docker?

**Answer:**  
The different functionalities and applications of using and implementing Docker are as below:

1. It simplifies the configuration and provides ease at infrastructure level configuration.
2. It manages the code pipeline easily which provides consistent environment and stability in the application.
3. It enables the isolation of the application.
4. It improves the productivity of the developer by allowing the developer to solely concentrate on business logic.
5. It enables a lot of debugging capabilities that provides extreme useful functionalities to implement.
6. It enables rapid deployment in the form of[virtualization at the operating system level](https://www.educba.com/virtualization-interview-questions/).
7. It reduces the utilization of multiple servers in the form of containerization.

#### ****Q6.****What is a Docker Registry?

**Answer:**  
A Docker Registry is a place where all the Docker Images will be stored and Docker Cloud and Docker Hub are the public registries where these images can be hosted upon. The Docker hub is the default storage for the Docker Images. An own registry can also be set up as per the requirement. Docker Data Center (DDC) can also be used which includes DTR (Docker Trusted Registry). Docker store will provide the feature of buying and selling the Docker images.

#### ****Q7.****What is the lifecycle of Docker Container?

**Answer:**  
This is the most popular Docker Interview Questions asked in an interview. The life cycle of the Docker container is as below:

1. Create a container.
2. Run the Docker container.
3. Pause the Container.
4. Unpause the Container.
5. Start the Container.
6. Stop the Container.
7. Restart the Container.
8. Kill the Container.
9. Destroy the Container.

#### ****Q8.****What are Docker Objects?

**Answer:**  
The Docker Objects are Docker Images, Services, and [Docker](https://www.educba.com/docker-alternatives/) Containers. A Docker Image is a read-only template with the configuration or runtime instructions for the Docker container. The Services allow scaling the containers across the different Docker Daemons. These all [together work as Swarm](https://www.educba.com/swarm-intelligence-applications/).

#### ****Q9.****What are Docker Namespaces?

**Answer:**  
The Namespaces in Docker is a technology which provides isolated workspaces called the Container. Namespaces provide a layer of isolation for the Docker containers

#### ****Q10.****What are the important Docker commands

|  |  |
| --- | --- |
| **Command** | **Description** |
| **dockerd** | To launch Docker daemon. |
| **build** | To build an image file for docker. |
| **create** | To create a new container. |
| **kill** | To kill a container. |
| **commit** | To create a new image from container changes. |

# [Top 50 AWS Interview Questions & Answers](https://career.guru99.com/top-15-aws-interview-questions/)

### ****1) Explain what AWS is?****

AWS stands for Amazon Web Service; it is a collection of remote computing services also known as a cloud computing platform.  This new realm of cloud computing is also known as IaaS or Infrastructure as a Service.

### ****2) Mention what the key components of AWS are?****

The key components of AWS are

* **Route 53:**A DNS web service
* **Simple E-mail Service:**It allows sending e-mail using RESTFUL API call or via regular SMTP
* **Identity and Access Management:**It provides enhanced security and identity management for your AWS account
* **Simple Storage Device or (S3):**It is a storage device and the most widely used AWS service
* **Elastic Compute Cloud (EC2):**It provides on-demand computing resources for hosting applications. It is handy in case of unpredictable workloads
* **Elastic Block Store (EBS):**It offers persistent storage volumes that attach to EC2 to allow you to persist data past the lifespan of a single Amazon EC2 instance
* **CloudWatch:**To monitor AWS resources, It allows administrators to view and collect key Also, one can set a notification alarm in case of trouble.

### ****3) Explain what S3 is?****

S3 stands for Simple Storage Service. You can use S3 interface to store and retrieve any amount of data, at any time and from anywhere on the web.  For S3, the payment model is “pay as you go.”

### ****4) What is AMI?****

AMI stands for Amazon Machine Image.  It’s a template that provides the information (an operating system, an application server, and applications) required to launch an instance, which is a copy of the AMI running as a virtual server in the cloud.  You can launch instances from as many different AMIs as you need.

### ****5) Mention what the relationship between an instance and AMI is?****

From a single AMI, you can launch multiple types of instances.  An instance type defines the hardware of the host computer used for your instance. Each instance type provides different computer and memory capabilities.  Once you launch an instance, it looks like a traditional host, and we can interact with it as we would with any computer.

An AMI includes the following things

* A template for the root volume for the instance
* Launch permissions decide which AWS accounts can avail the AMI to launch instances
* A block device mapping that determines the volumes to attach to the instance when it is launched

### ****7) How can you send a request to Amazon S3?****

Amazon S3 is a REST service, and you can send a request by using the REST API or the AWS SDK wrapper libraries that wrap the underlying Amazon S3 REST API.

### ****8) Mention what the difference between Amazon S3 and EC2 is?****

The difference between EC2 and Amazon S3 is that

|  |  |
| --- | --- |
| **EC2** | **S3** |
| * It is a cloud web service used for hosting your application | * It is a data storage system where any amount of data can be stored |
| * It is like a huge computer machine which can run either Linux or Windows and can handle application like PHP, Python, Apache or any databases | * It has a REST interface and uses secure HMAC-SHA1 authentication keys |

### ****9) How many buckets can you create in AWS by default?****

By default, you can create up to 100 buckets in each of your AWS accounts.

### ****10) Explain can you vertically scale an Amazon instance? How?****

Yes, you can vertically scale on Amazon instance. For that

* Spin up a new larger instance than the one you are currently running
* Pause that instance and detach the root webs volume from the server and discard
* Then stop your live instance and detach its root volume
* Note the unique device ID and attach that root volume to your new server
* And start it again

### ****11) Explain what T2 instances is?****

T2 instances are designed to provide moderate baseline performance and the capability to burst to higher performance as required by the workload.

### ****12) In VPC with private and public subnets, database servers should ideally be launched into which subnet?****

With private and public subnets in VPC, database servers should ideally launch into private subnets.

### ****13) Mention what the security best practices for Amazon EC2 are?****

For secure Amazon EC2 best practices, follow the following steps

* Use AWS identity and access management to control access to your AWS resources
* Restrict access by allowing only trusted hosts or networks to access ports on your instance
* Review the rules in your security groups regularly
* Only open up permissions that you require
* Disable password-based login, for example, launched from your AMI

### ****14) Explain how the buffer is used in Amazon web services?****

The buffer is used to make the system more robust to manage traffic or load by synchronizing different component.  Usually, components receive and process the requests in an unbalanced way. With the help of buffer, the components will be balanced and will work at the same speed to provide faster services.

### ****15) While connecting to your instance what are the possible connection issues one might face?****

The possible connection errors one might encounter while connecting instances are

* Connection timed out
* User key not recognized by the server
* Host key not found, permission denied
* An unprotected private key file
* Server refused our key or No supported authentication method available
* Error using MindTerm on Safari Browser
* Error using Mac OS X RDP Client

### ****16) What are key-pairs in AWS?****

Key-pairs are secure login information for your virtual machines. To connect to the instances, you can use key-pairs which contain a public-key and private-key.

### ****17)  What are the different types of instances?****

Following are the types of instances:

* General purpose
* Computer Optimized
* Memory Optimized
* Storage Optimized
* Accelerated Computing

### ****18) Is the property of broadcast or multicast supported by Amazon VPC?****

No, currently Amazon VPI not provide support for broadcast or multicast.

### ****19) How many Elastic IPs is allows you to create by AWS?****

5 VPC Elastic IP addresses are allowed for each AWS account.

### ****20) Explain default storage class in S3****

The default storage class is a Standard frequently accessed.

### ****21) What are the roles?****

Roles are used to providing permissions to entities which you can trust within your AWS account. Roles are very similar to users. However,  with roles, you do not require to create any username and password to work with the resources.

### ****22) What are the edge locations?****

Edge location is the area where the contents will be cached. So, when a user is trying to accessing any content, the content will automatically be searched in the edge location.

### ****23) What is VPC?****

VPC stands for Virtual Private Cloud. It allows you to customize your networking configuration. It is a network which is logically isolated from another network in the cloud. It allows you to have your IP address range,  internet gateways, subnet and security groups.

### ****24) Explain snowball****

Snowball is a data transport option. It used source appliances to a large amount of data into and out of AWS. With the help of snowball, you can transfer a massive amount of data from one place to another. It helps you to reduce networking costs.

### ****25) What is a redshift?****

Redshift is a big data warehouse product. It is fast and powerful, fully managed data warehouse service in the cloud.

### ****26) What are the advantages of auto-scaling?****

Following are the advantages of autoscaling

* Offers fault tolerance
* Better availability
* Better cost management

### ****27) What is meant by subnet?****

A large section of IP Address divided into chunks is known as subnets.

### ****28) Can you establish a Peering connection to a VPC in a different region?****

No, It’s only possible between VPCs in the same region.

### ****29) What is SQL?****

Simple Queues Services also known as SQL. It is distributed queuing service which acts as a mediator for two controllers.

### ****30) How many subnets can you have per VPC?****

You can have 200 subnets per VPC.

### ****31) DNS  and Load Balancer service comes under which type of cloud service?****

DNS and Load Balancer and DNS services come under IAAS-storage cloud service.

### ****32) What is the role of AWS CloudTrail?****

CloudTrail is a specially designed tool for logging and tracking API calls. It helps to audit all S3 bucket accesses.

### ****33) When EC2 officially launched?****

EC2 officially launched in the year 2006.

### ****34) What is SimpleDB?****

SimpleDB is a data repository of structure record which encourages data doubts and indexing both S3 and EC2are called SimpleDB.

### ****35) Explain Amazon ElasticCache****

Amazon Elasticcache is a web service which makes it easy to deploy, scale and store data in the cloud.

### ****36) What is AWS Lambda?****

Lambda is an Amazon compute service which allows you to run code in the  AWS Cloud without managing servers.

### ****37) Name the types of AMI provided by AWS****

The types of AMI provided by AWS are:

1. Instance store backed
2. EBS backed

### ****38) Name the AWS service exists only to redundantly cache data and images?****

AWS Edge locations are service which redundantly cache data and images.

### ****39) Explain Geo Restriction in CloudFront****

A Geo-restriction feature helps you to prevent users of specific geographic locations from accessing content which you’re distributing through a CloudFront web distribution.

### ****40) What is Amazon EMR?****

EMR is a survived cluster stage which helps you to interpret the working of data structures before the intimation.  Apache Hadoop and Apache Spark on the Amazon Web Services helps you to investigate a large amount of data. You can prepare data for the analytics goals and marketing intellect workloads using Apache Hive and using other relevant open source designs.

### ****41) What is boot time taken for the instance stored backed AMI?****

The boot time for an Amazon instance store-backend AMI is less than 5 minutes.

### ****42) Do you need an internet gateway to use peering connections?****

Yes, the Internet gateway is needed to use VPC (virtual private cloud peering) connections.

### ****43) How to connect EBS volume to multiple instances?****

We can’t be able to connect EBS volume to multiple instances.  Although, you can connect various EBS Volumes to a single instance.

### ****44) List different types of cloud services****

Various types of cloud services are:

* Software as a Service (SaaS),
* Data as a Service (DaaS)
* Platform as a Service (PaaS)
* Infrastructure as a Service (IaaS).

### ****45) State the difference between An Instance  and AMI****

AMI is a template consisting software configuration part. For example Operating systems, applications, application server if you start an instance, a duplicate of the AMI in a row as an attendant in the cloud.

### ****46) What are the different types of Load Balancer in AWS services?****

Two types of Load balancer are:

1. Application Load Balancer
2. Classic Load Balancer

### ****47) In which situation you will select provisioned IOPS over standard RDS storage?****

You should select provisioned IOPS storage over standard RDS storage if you want to perform batch-related workloads.

### ****48) What are the important features of Amazon cloud search?****

Important features of the Amazon cloud are:

* Boolean searches
* Prefix Searches
* Range searches
* Entire text search
* AutoComplete advice

### ****49) Can vertically scaling is allows in  Amazon Instance?****

Yes, you can vertically estimate one Amazon instance.

### ****50) What is the use of lifecycle hooks in Autoscaling?****

Lifecycle hooks are used for autoscaling to put an additional wait time to a scale in or scale out event.

### ****51) What are various layers of Cloud Architecture explained in AWS training?****

Different layers of cloud architecture are:

* Cloud controller
* Cluster controller
* Storage Controller
* Node Controller

### ****52) What are the storage class available in Amazon s3?****

Storage classes available with Amazon s3 are:

* Amazon S3 standard
* Amazon S3 standard-infrequent Access
* Amazon S3 Reduced Redundancy Storage
* Amazon Glacier

### ****53) Name some of the DB engines which can be used in AWS RDS****

1. MS-SQL DB
2. MariaDB
3. MYSQL DB
4. OracleDB
5. PostgreDB

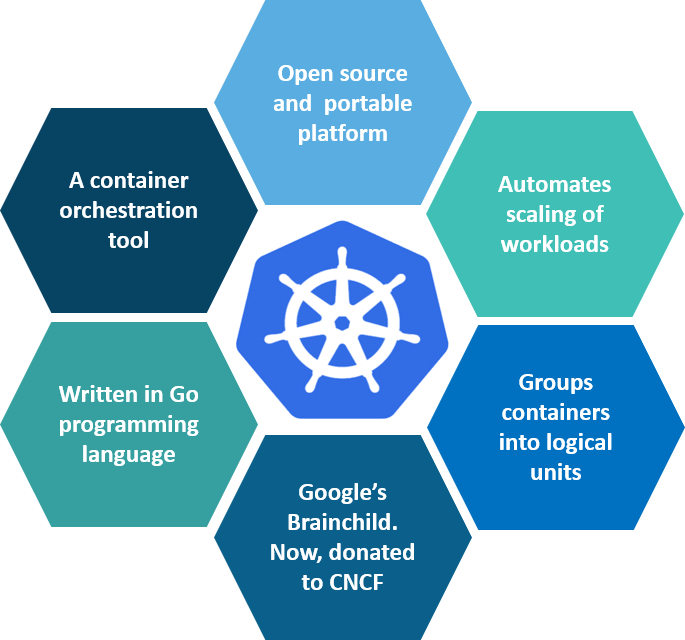
**Basic Kubernetes Interview Questions**

This section of questions will consist of all those basic questions that you need to know related to the working of Kubernetes.

**Q1. How is Kubernetes different from Docker Swarm?**

|  |  |  |
| --- | --- | --- |
| **Features** | **Kubernetes** | **Docker Swarm** |
| **Installation & Cluster Config** | Setup is very complicated, but once installed cluster is robust. | Installation is very simple, but the cluster is not robust. |
| **GUI** | GUI is the Kubernetes Dashboard. | There is no GUI. |
| **Scalability** | Highly scalable and scales fast. | Highly scalable and scales 5x faster than Kubernetes. |
| **Auto-scaling** | Kubernetes can do auto-scaling. | Docker swarm cannot do auto-scaling. |
| **Load Balancing** | Manual intervention needed for load balancing traffic between different containers and pods. | Docker swarm does auto load balancing of traffic between containers in the cluster. |
| **Rolling Updates & Rollbacks** | Can deploy rolling updates and does automatic rollbacks. | Can deploy rolling updates, but not automatic rollback. |
| **DATA Volumes** | Can share storage volumes only with the other containers in the same pod. | Can share storage volumes with any other container. |
| **Logging & Monitoring** | In-built tools for logging and monitoring. | 3rd party tools like ELK stack should be used for logging and monitoring. |

**Q2. What is Kubernetes?**



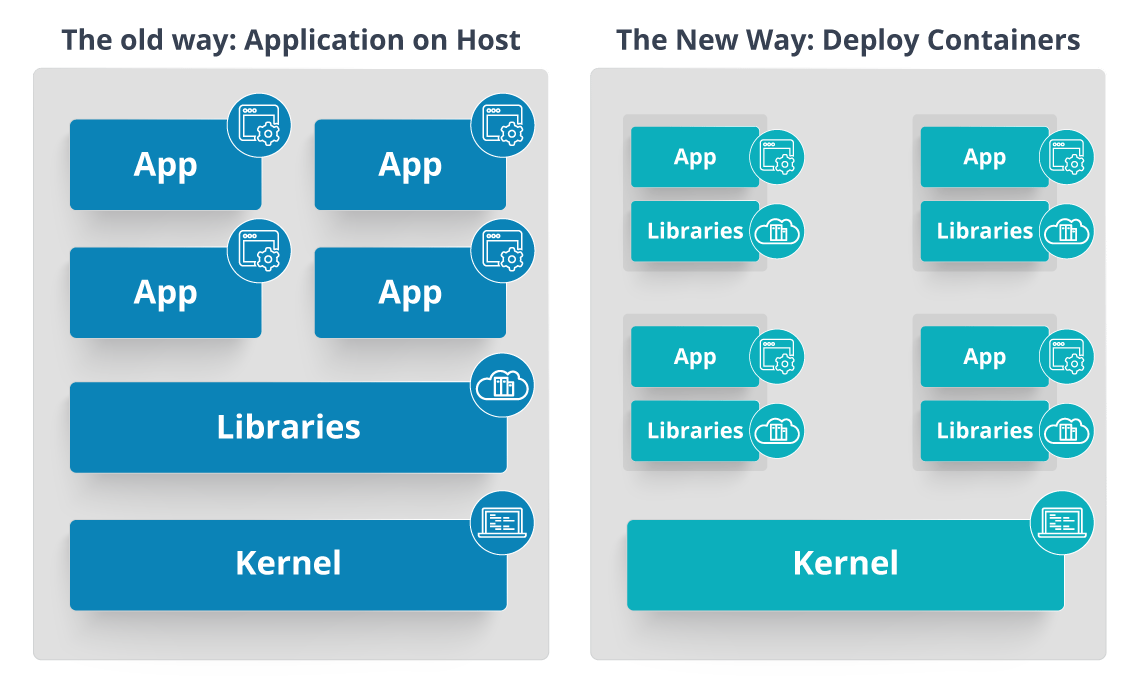
**Fig 1:** What is Kubernetes – Kubernetes Interview Questions

Kubernetes is an open-source container management tool which holds the responsibilities of container deployment, scaling & descaling of containers & load balancing. Being the Google’s brainchild, it offers excellent community and works brilliantly with all the cloud providers. So, we can say that Kubernetes is not*a containerization platform, but it is a multi-container management solution.*

**Q3. How is Kubernetes related to Docker?**

It’s a known fact that Docker provides the lifecycle management of containers and a Docker image builds the runtime containers. But, since these individual containers have to communicate, Kubernetes is used.  So, Docker builds the containers and these containers communicate with each other via Kubernetes. So, containers running on multiple hosts can be manually linked and orchestrated using Kubernetes.

**Q4. What is the difference between deploying applications on hosts and containers?**



**Fig 2:** Deploying Applications On Host vs Containers – Kubernetes Interview Questions

Refer to the above diagram. The left side architecture represents deploying applications on hosts. So, this kind of architecture will have an operating system and then the operating system will have a kernel which will have various libraries installed on the operating system needed for the application. So, in this kind of framework you can have n number of applications and all the applications will share the libraries present in that operating system whereas while deploying applications in containers the architecture is a little different.

This kind of architecture will have a kernel and that is the only thing that’s going to be the only thing common between all the applications. So, if there’s a particular application which needs Java then that particular application we’ll get access to Java and if there’s another application which needs Python then only that particular application will have access to Python.

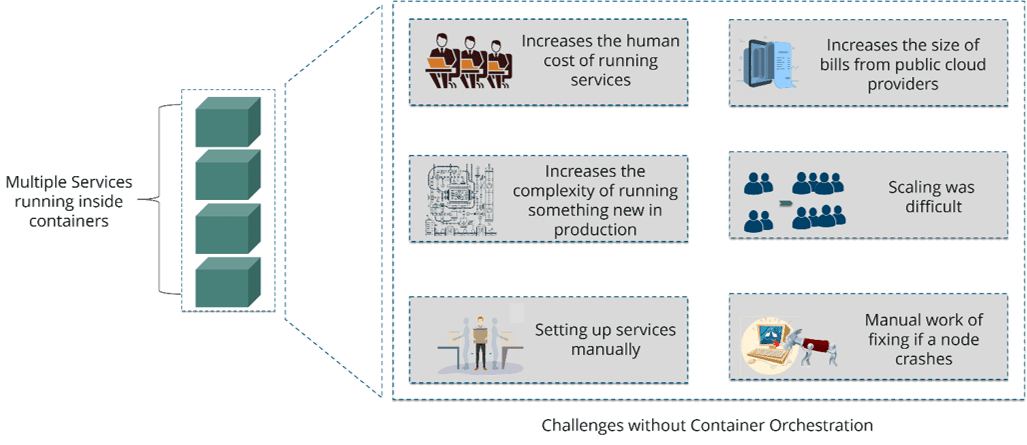
The individual blocks that you can see on the right side of the diagram are basically containerized and these are isolated from other applications. So, the applications have the necessary libraries and binaries isolated from the rest of the system, and cannot be encroached by any other application.

**Q5. What is Container Orchestration?**

Consider a scenario where you have 5-6 microservices for an application. Now, these microservices are put in individual containers, but won’t be able to communicate without container orchestration. So, as orchestration means the amalgamation of all instruments playing together in harmony in music, similarly container orchestration means all the services in individual containers working together to fulfill the needs of a single server.

**Q6. What is the need for Container Orchestration?**

Consider you have 5-6 microservices for a single application performing various tasks, and all these microservices are put inside containers. Now, to make sure that these containers communicate with each other we need container orchestration.

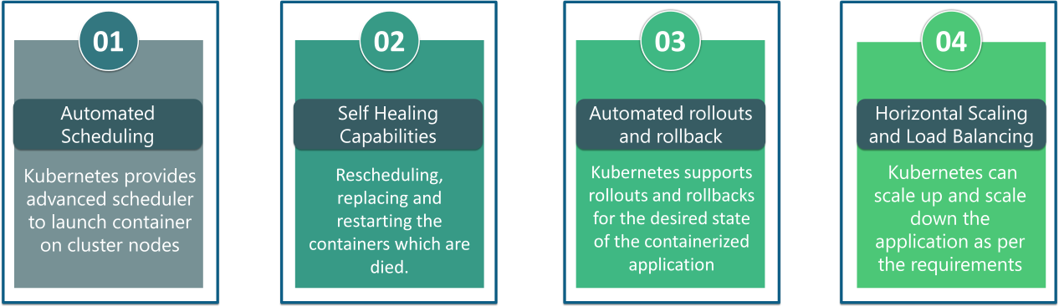


**Fig 3:** Challenges Without Container Orchestration – Kubernetes Interview Questions

As you can see in the above diagram, there were also many challenges that came into place without the use of container orchestration. So, to overcome these challenges the container orchestration came into place.

**Q7. What are the features of Kubernetes?**

The features of Kubernetes, are as follows:



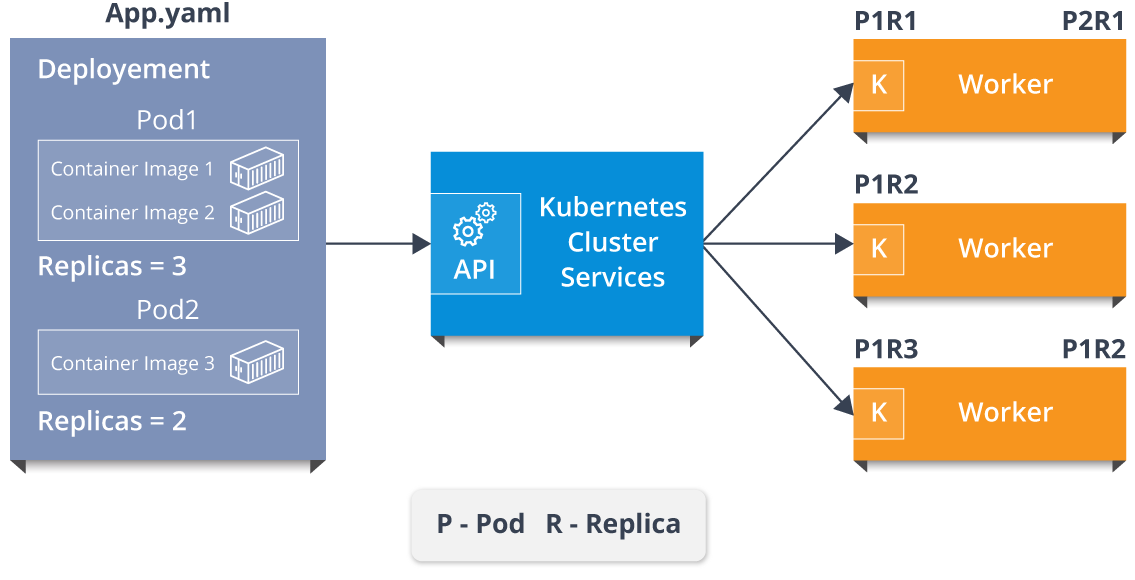
**Fig 4:** Features Of Kubernetes – Kubernetes Interview Questions

**Q8. How does Kubernetes simplify containerized Deployment?**

As a typical application would have a cluster of containers running across multiple hosts, all these containers would need to talk to each other. So, to do this you need something big that would load balance, scale & monitor the containers. Since Kubernetes is cloud-agnostic and can run on any public/private providers it must be your choice simplify containerized deployment.

**Q9. What do you know about clusters in Kubernetes?**

The fundamental behind Kubernetes is that we can enforce the desired state management, by which I mean that we can feed the cluster services of a specific configuration, and it will be up to the cluster services to go out and run that configuration in the infrastructure.



**Fig 5:** Representation Of Kubernetes Cluster – Kubernetes Interview Questions

So, as you can see in the above diagram, the deployment file will have all the configurations required to be fed into the cluster services. Now, the deployment file will be fed to the API and then it will be up to the cluster services to figure out how to schedule these pods in the environment and make sure that the right number of pods are running.

So, the API which sits in front of services, the worker nodes & the Kubelet process that the nodes run, all together make up the Kubernetes Cluster.

**Q10. What is Google Container Engine?**

**Google Container Engine (GKE)**is an open source management platform for Docker containers and the clusters. This Kubernetes based engine supports only those clusters which run within the Google’s public cloud services.

**Q11.  What is Heapster?**

Heapster is a cluster-wide aggregator of data provided by Kubelet running on each node. This container management tool is supported natively on Kubernetes cluster and runs as a pod, just like any other pod in the cluster. So, it basically discovers all nodes in the cluster and queries usage information from the Kubernetes nodes in the cluster, via on-machine Kubernetes agent.

**Q12.  What is Minikube?**

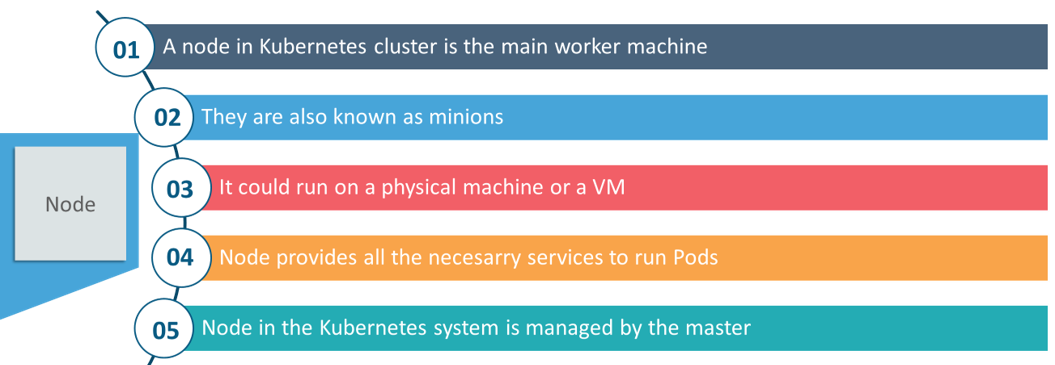
Minikube is a tool that makes it easy to run Kubernetes locally. This runs a single-node Kubernetes cluster inside a virtual machine.

**Q13.  What is** **Kubectl?**

Kubectl is the platform using which you can pass commands to the cluster. So, it basically provides the CLI to run commands against the Kubernetes cluster with various ways to create and manage the Kubernetes component.

**Q14.  What is Kubelet?**

This is an agent service which runs on each node and enables the slave to communicate with the master. So, Kubelet works on the description of containers provided to it in the PodSpec and makes sure that the containers described in the PodSpec are healthy and running.

**Q15. What do you understand by a node in Kubernetes?**  


**Fig 6:**Node In Kubernetes – Kubernetes Interview Questions

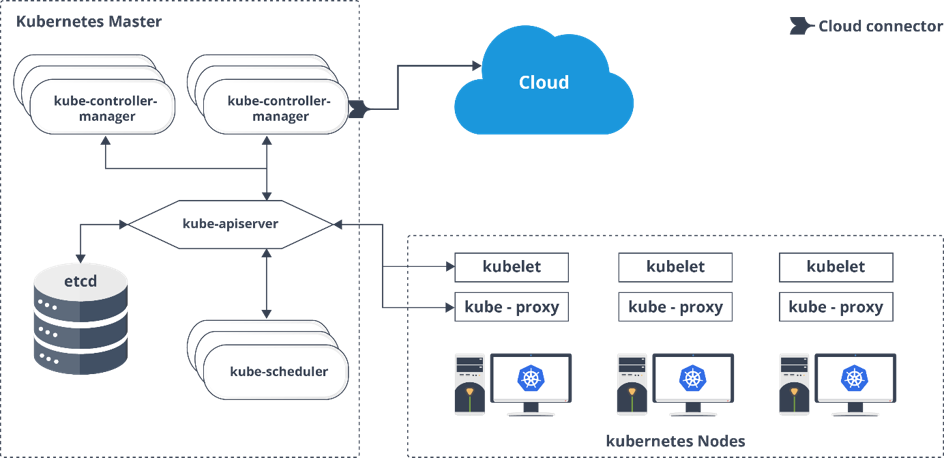
Interested in learning Kubernetes? [Learn Now](https://www.edureka.co/kubernetes-certification)

**Architecture-Based Kubernetes Interview Questions**

This section of questions will deal with the questions related to the architecture of Kubernetes.

**Q1. What are the different components of Kubernetes Architecture?**

The Kubernetes Architecture has mainly 2 components – the master node and the worker node. As you can see in the below diagram, the master and the worker nodes have many inbuilt components within them. The master node has the kube-controller-manager, kube-apiserver, kube-scheduler, etcd. Whereas the worker node has kubelet and kube-proxy running on each node.



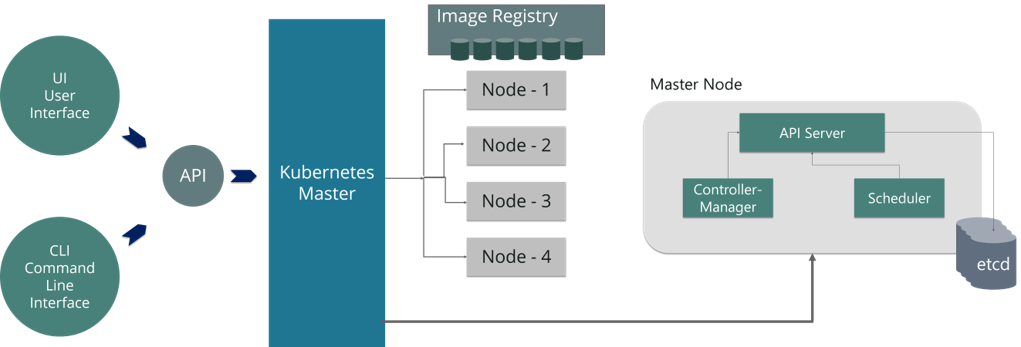
**Fig 7:** Architecture Of Kubernetes – Kubernetes Interview Questions

**Q2. What do you understand by Kube-proxy?**

Kube-proxy can run on each and every node and can do simple TCP/UDP packet forwarding across backend network service. So basically, it is a network proxy which reflects the services as configured in Kubernetes API on each node. So, the Docker-linkable compatible environment variables provide the cluster IPs and ports which are opened by proxy.

**Q3.  Can you brief on the working of the master node in Kubernetes?**

Kubernetes master controls the nodes and inside the nodes the containers are present. Now, these individual containers are contained inside pods and inside each pod, you can have a various number of containers based upon the configuration and requirements. So, if the pods have to be deployed, then they can either be deployed using user interface or command line interface. Then, these pods are scheduled on the nodes and based on the resource requirements, the pods are allocated to these nodes. The kube-apiserver makes sure that there is communication established between the Kubernetes node and the master components.



**Fig 8:** Representation Of Kubernetes Master Node – Kubernetes Interview Questions

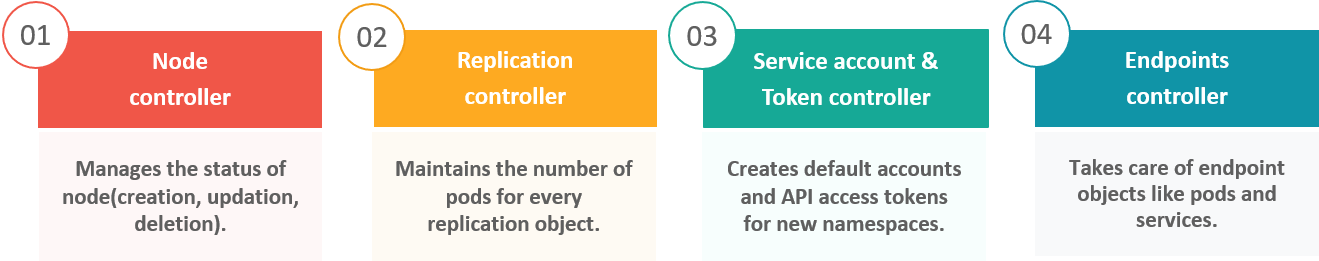
**Q4.  What is the role of kube-apiserver and kube-scheduler?**

The kube – apiserver follows the scale-out architecture and, is the front-end of the master node control panel. This exposes all the APIs of the Kubernetes Master node components and is responsible for establishing communication between Kubernetes Node and the Kubernetes master components.

The kube-scheduler is responsible for distribution and management of workload on the worker nodes. So, it selects the most suitable node to run the unscheduled pod based on resource requirement and keeps a track of resource utilization. It makes sure that the workload is not scheduled on nodes which are already full.

**Q5.  Can you brief about the Kubernetes controller manager?**

Multiple controller processes run on the master node but are compiled together to run as a single process which is the Kubernetes Controller Manager. So, Controller Manager is a daemon that embeds controllers and does namespace creation and garbage collection. It owns the responsibility and communicates with the API server to manage the end-points.

So, the different types of controller manager running on the master node are :  


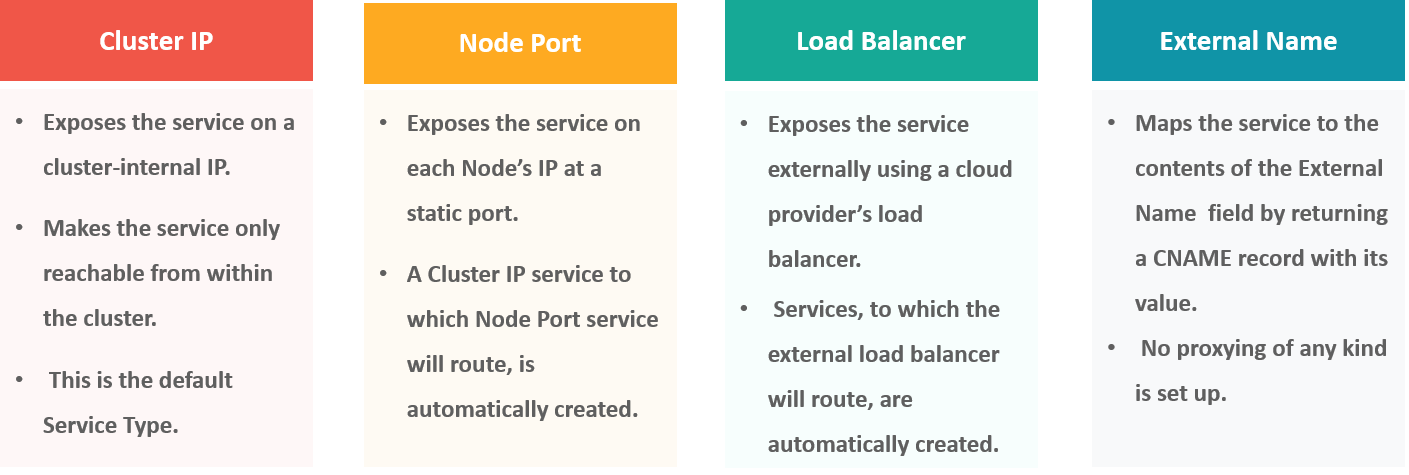
**Fig 9:** Types Of Controllers – Kubernetes Interview Questions

**Q6.  What is ETCD?**

Etcd is written in Go programming language and is a distributed key-value store used for coordinating between distributed work. So, Etcd stores the configuration data of the Kubernetes cluster, representing the state of the cluster at any given point in time.

**Q7. What are the different types of services in Kubernetes?**

The following are the different types of services used:



**Fig 10:** Types Of Services – Kubernetes Interview Questions

**Q8. What do you understand by load balancer in Kubernetes?**

A load balancer is one of the most common and standard ways of exposing service. There are two types of load balancer used based on the working environment i.e. either the Internal Load Balancer or the External Load Balancer. The Internal Load Balancer automatically balances load and allocates the pods with the required configuration whereas the External Load Balancer directs the traffic from the external load to the backend pods.

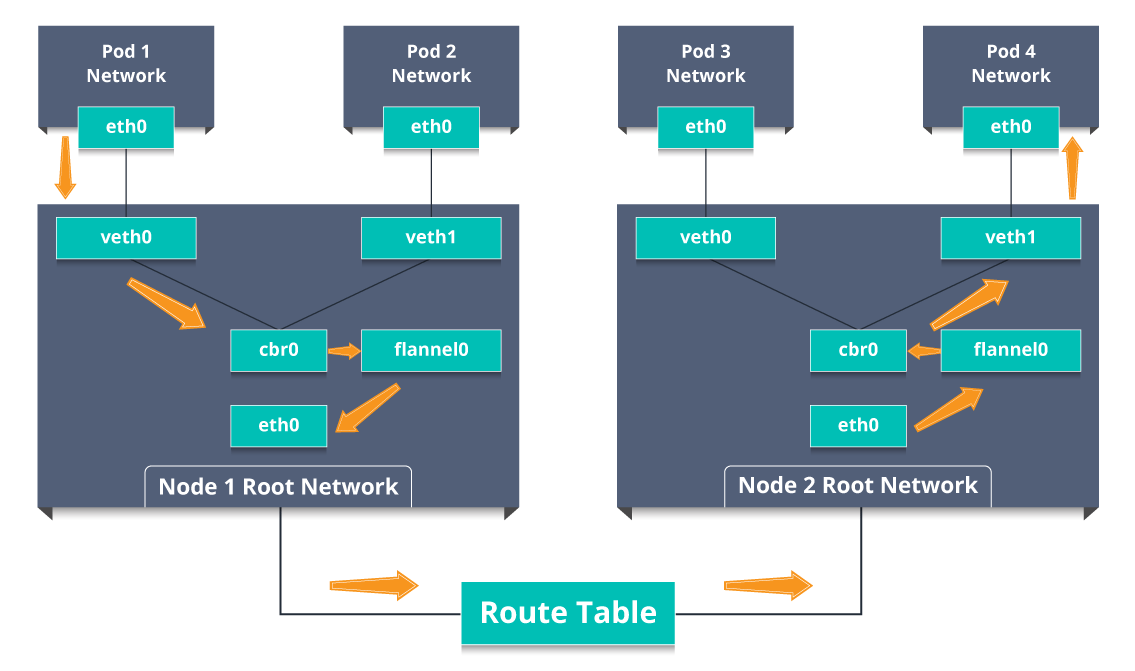
**Q9. What is Ingress network, and how does it work?**

Ingress network is a collection of rules that acts as an entry point to the Kubernetes cluster. This allows inbound connections, which can be configured to give services externally through reachable URLs, load balance traffic, or by offering name-based virtual hosting. So, Ingress is an API object that manages external access to the services in a cluster, usually by HTTP and is the most powerful way of exposing service.

Now, let me explain to you the working of Ingress network with an example.

There are 2 nodes having the pod and root network namespaces with a Linux bridge. In addition to this, there is also a new virtual ethernet device called flannel0(network plugin) added to the root network.

Now, suppose we want the packet to flow from pod1 to pod 4. Refer to the below diagram.



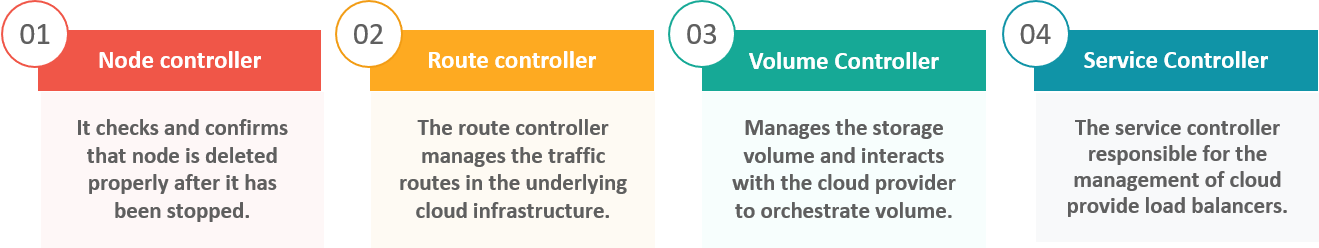
**Fig 11:** Working Of Ingress Network – Kubernetes Interview Questions

* So, the packet leaves pod1’s network at eth0 and enters the root network at veth0.
* Then it is passed on to cbr0, which makes the ARP request to find the destination and it is found out that nobody on this node has the destination IP address.
* So, the bridge sends the packet to flannel0 as the node’s route table is configured with flannel0.
* Now, the flannel daemon talks to the API server of Kubernetes to know all the pod IPs and their respective nodes to create mappings for pods IPs to node IPs.
* The network plugin wraps this packet in a UDP packet with extra headers changing the source and destination IP’s to their respective nodes and sends this packet out via eth0.
* Now, since the route table already knows how to route traffic between nodes, it sends the packet to the destination node2.
* The packet arrives at eth0 of node2 and goes back to flannel0 to de-capsulate and emits it back in the root network namespace.
* Again, the packet is forwarded to the Linux bridge to make an ARP request to find out the IP that belongs to veth1.
* The packet finally crosses the root network and reaches the destination Pod4.

**Q10.  What do you understand by Cloud controller manager?**

The Cloud Controller Manager is responsible for persistent storage, network routing, abstracting the cloud-specific code from the core Kubernetes specific code, and managing the communication with the underlying cloud services. It might be split out into several different containers depending on which cloud platform you are running on and then it enables the cloud vendors and Kubernetes code to be developed without any inter-dependency. So, the cloud vendor develops their code and connects with the Kubernetes cloud-controller-manager while running the Kubernetes.

The various types of cloud controller manager are as follows:

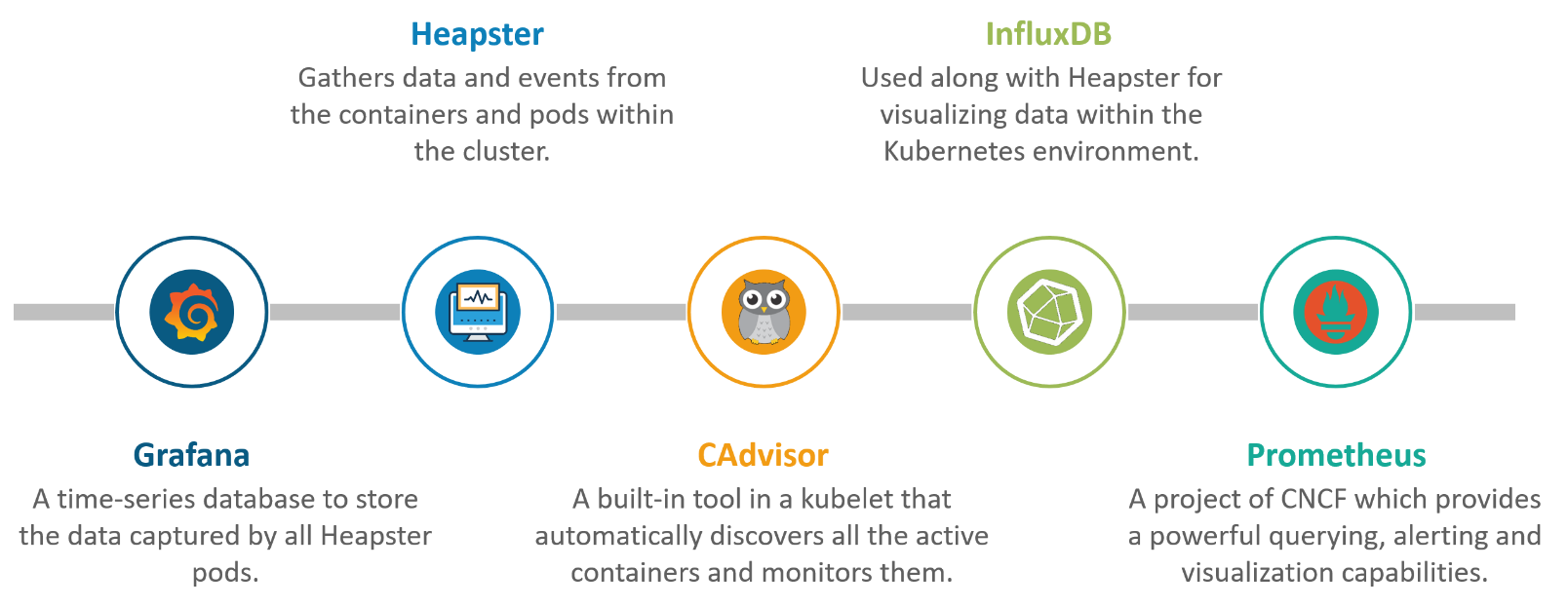


**Fig 12:** Types Of Cloud Controller Manager – Kubernetes Interview Questions

**Q11. What is Container resource monitoring?**

As for users, it is really important to understand the performance of the application and resource utilization at all the different abstraction layer, Kubernetes factored the management of the cluster by creating abstraction at different levels like container, pods, services and whole cluster. Now, each level can be monitored and this is nothing but Container resource monitoring.

The various container resource monitoring tools are as follows:



**Fig 13:** Container Resource Monitoring Tools – Kubernetes Interview Questions

**Q12. What is the difference between a replica set and replication controller?**

Replica Set and Replication Controller do almost the same thing. Both of them ensure that a specified number of pod replicas are running at any given time. The difference comes with the usage of selectors to replicate pods. Replica Set use Set-Based selectors while replication controllers use Equity-Based selectors.

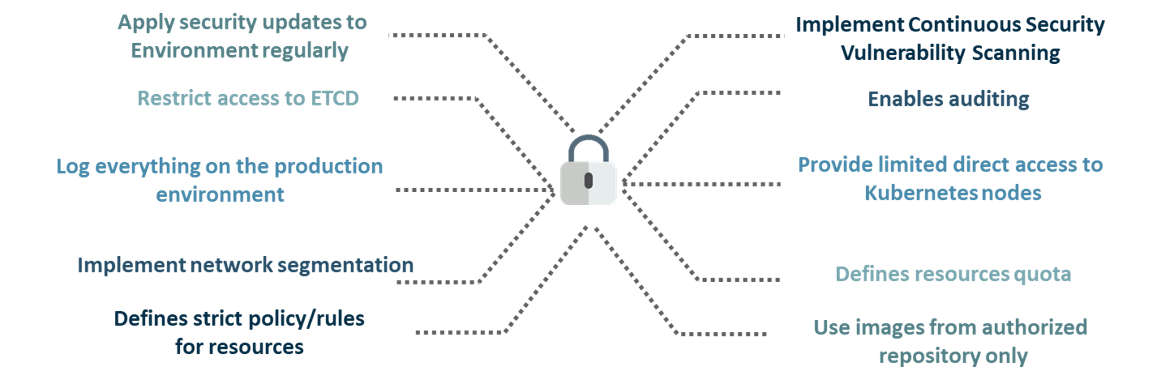
* **Equity-Based Selectors:**This type of selector allows filtering by label key and values. So, in layman terms, the equity-based selector will only look for the pods which will have the exact same phrase as that of the label.  
  **Example**: Suppose your label key says app=nginx, then, with this selector, you can only look for those pods with label app equal to nginx.
* **Selector-Based Selectors:**This type of selector allows filtering keys according to a set of values. So, in other words, the selector based selector will look for pods whose label has been mentioned in the set.  
  **Example:** Say your label key says app in (nginx, NPS, Apache). Then, with this selector, if your app is equal to any of nginx, NPS, or Apache, then the selector will take it as a true result.

**Q13. What is a Headless Service?**

Headless Service is similar to that of a ‘Normal’ services but does not have a Cluster IP. This service enables you to directly reach the pods without the need of accessing it through a proxy.

**Q14. What are the best security measures that you can take while using Kubernetes?**

The following are the best security measures that you can follow while using Kubernetes:

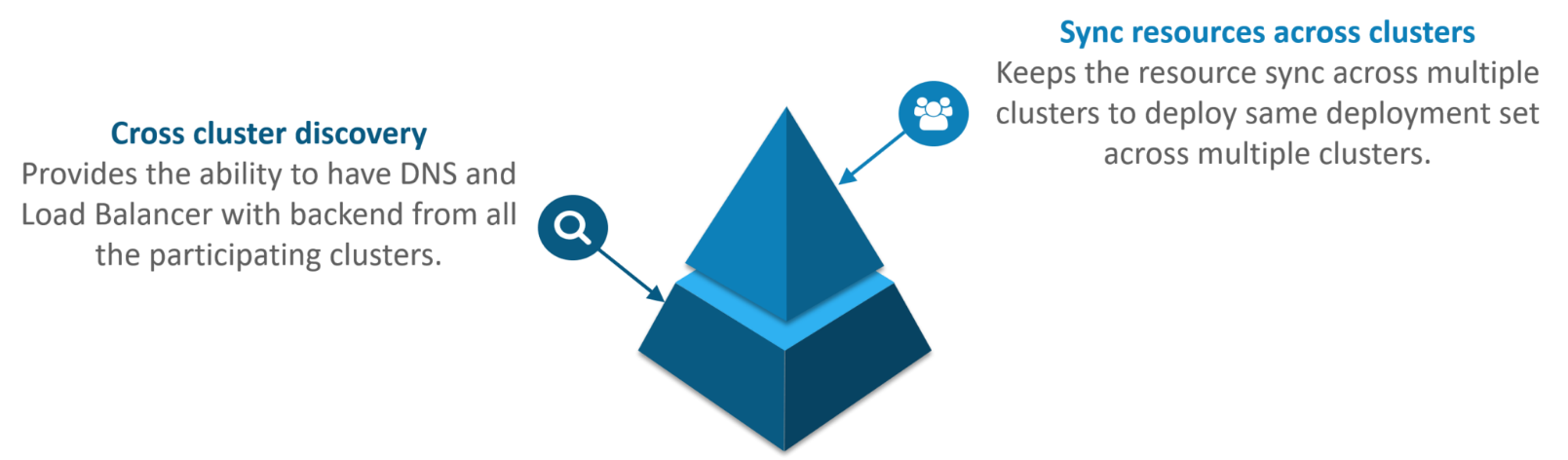


**Fig 14:** Best Security Measures – Kubernetes Interview Questions

**Q15. What are federated clusters?**

Multiple Kubernetes clusters can be managed as a single cluster with the help of federated clusters. So, you can create multiple Kubernetes clusters within a data center/cloud and use federation to control/manage them all at one place.

The federated clusters can achieve this by doing the following two things. Refer to the below diagram.



**Fig 15:** Federated Clusters – Kubernetes Interview Questions

**Scenario-Based Interview Questions**

This section of questions will consist of various scenario based questions that you may face in your interviews.

**Scenario 1:** Suppose a company built on monolithic architecture handles numerous products. Now, as the company expands in today’s scaling industry, their monolithic architecture started causing problems.

*How do you think the company shifted from monolithic to microservices and deploy their services containers?*

**Solution:**

As the company’s goal is to shift from their monolithic application to microservices, they can end up building piece by piece, in parallel and just switch configurations in the background. Then they can put each of these built-in microservices on the Kubernetes platform. So, they can start by migrating their services once or twice and monitor them to make sure everything is running stable. Once they feel everything is going good, then they can migrate the rest of the application into their Kubernetes

Next

**Scenario 2:**Consider a multinational company with a very much distributed system, with a large number of data centers, virtual machines, and many employees working on various tasks.

*How do you think can such* *a company manage all the tasks in a consistent way with Kubernetes?*

**Solution:**

As all of us know that I.T. departments launch thousands of containers, with tasks running across a numerous number of nodes across the world in a distributed system.

In such a situation the company can use something that offers them agility, scale-out capability, and DevOps practice to the cloud-based applications.

So, the company can, therefore, use Kubernetes to customize their scheduling architecture and support multiple container formats. This makes it possible for the affinity between container tasks that gives greater efficiency with an extensive support for various container networking solutions and container storage.

**Scenario 3:**Consider a situation, where a company wants to increase its efficiency and the speed of its technical operations by maintaining minimal costs.

*How do you think the company will try to achieve this?*

**Solution:**

The company can implement the DevOps methodology, by building a CI/CD pipeline, but one problem that may occur here is the configurations may take time to go up and running. So, after implementing the CI/CD pipeline the company’s next step should be to work in the cloud environment. Once they start working on the cloud environment, they can schedule containers on a cluster and can orchestrate with the help of Kubernetes. This kind of approach will help the company reduce their deployment time, and also get faster across various environments.

**Scenario 4:** Suppose a company wants to revise it’s deployment methods and wants to build a platform which is much more scalable and responsive.

*How do you think this company can achieve this to satisfy their customers?*

**Solution:**

In order to give millions of clients the digital experience they would expect, the company needs a platform that is scalable, and responsive, so that they could quickly get data to the client website. Now, to do this the company should move from their private data centers (if they are using any) to any cloud environment such as AWS. Not only this, but they should also implement the microservice architecture so that they can start using Docker containers. Once they have the base framework ready, then they can start using the best orchestration platform available i.e. Kubernetes. This would enable the teams to be autonomous in building applications and delivering them very quickly.

**Scenario 5:** Consider a multinational company with a very much distributed system, looking forward to solving the monolithic code base problem.

*How do you think the company can solve their problem?*

**Solution**

Well, to solve the problem, they can shift their monolithic code base to a microservice design and then each and every microservices can be considered as a container. So, all these containers can be deployed and orchestrated with the help of Kubernetes.

**Scenario 6:**All of us know that the shift from monolithic to microservices solves the problem from the development side, but increases the problem at the deployment side.

*How can the company solve the problem on the deployment side?*

**Solution**

The team can experiment with container orchestration platforms, such as Kubernetes and run it in data centers. So, with this, the company can generate a templated application, deploy it within five minutes, and have actual instances containerized in the staging environment at that point. This kind of Kubernetes project will have dozens of microservices running in parallel to improve the production rate as even if a node goes down, then it can be rescheduled immediately without performance impact.

**Scenario 7:**Suppose a company wants to optimize the distribution of its workloads, by adopting new technologies.

*How can the company achieve this distribution of resources efficiently?*

**Solution**

The solution to this problem is none other than Kubernetes. Kubernetes makes sure that the resources are optimized efficiently, and only those resources are used which are needed by that particular application. So, with the usage of the best container orchestration tool, the company can achieve the distribution of resources efficiently.

**Scenario 8:**Consider a carpooling company wants to increase their number of servers by simultaneously scaling their platform.

*How do you think will the company deal with the servers and their installation?*

**Solution**

The company can adopt the concept of containerization. Once they deploy all their application into containers, they can use Kubernetes for orchestration and use container monitoring tools like Prometheus to monitor the actions in containers. So, with such usage of containers, giving them better capacity planning in the data center because they will now have fewer constraints due to this abstraction between the services and the hardware they run on.

**Scenario 9:**Consider a scenario where a company wants to provide all the required hand-outs to its customers having various environments.

*How do you think they can achieve this critical target in a dynamic manner?*

**Solution**

The company can use Docker environments, to put together a cross-sectional team to build a web application using Kubernetes. This kind of framework will help the company achieve the goal of getting the required things into production within the shortest time frame. So, with such a machine running, the company can give the hands-outs to all the customers having various environments.

**Scenario 10**: Suppose a company wants to run various workloads on different cloud infrastructure from bare metal to a public cloud.

*How will the company achieve this in the presence of different interfaces?*

**Solution**

The company can decompose its infrastructure into microservices and then adopt Kubernetes. This will let the company run various workloads on different cloud infrastructures.