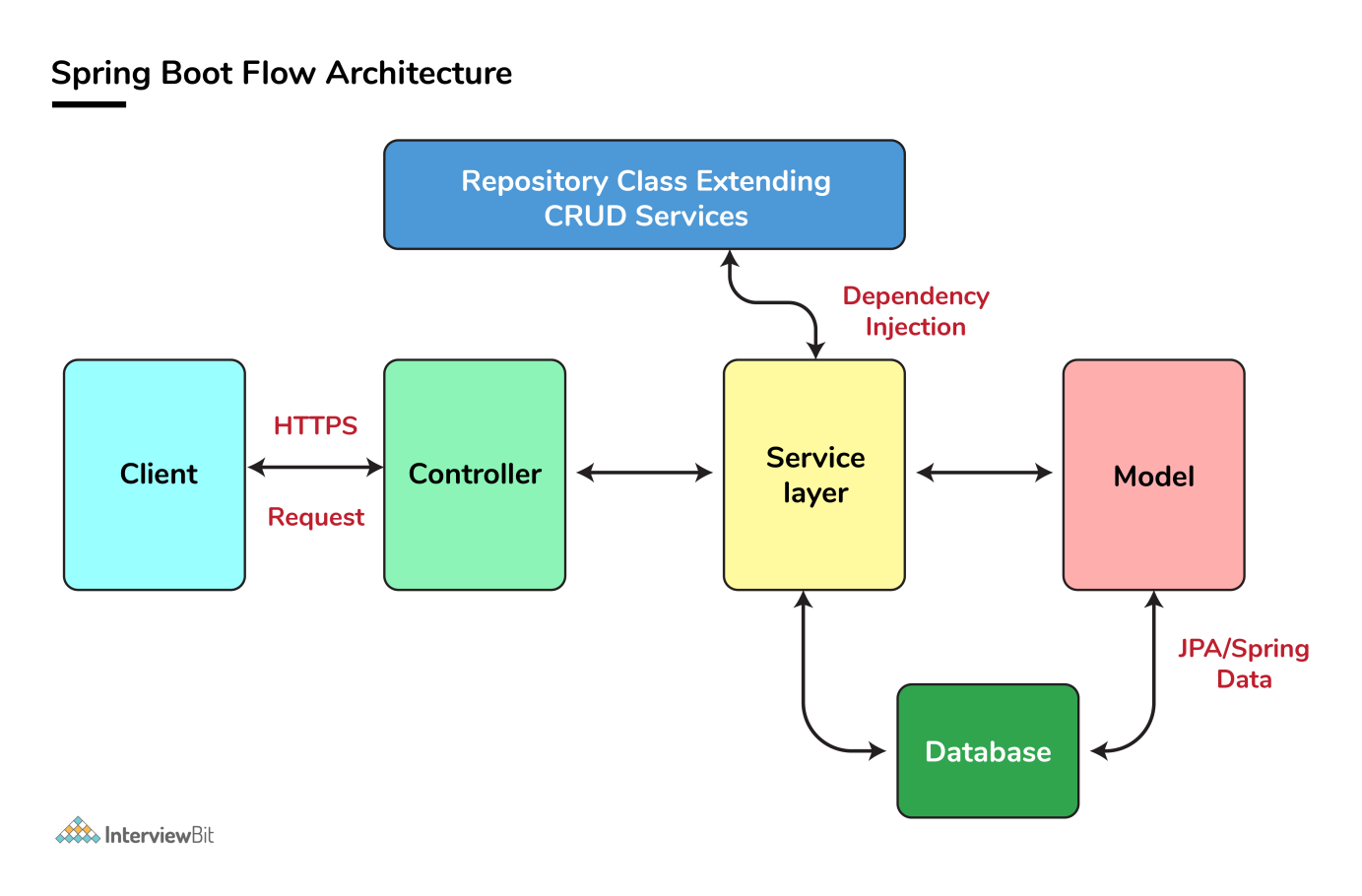
**Definitions:**

* Spring Boot is an open source Java-based framework used to build stand-alone and production ready spring applications , generally used in micro-services architecture.
* Micro Service is an architecture that allows the developers to develop and deploy services independently. Each service running has its own process and this achieves the lightweight model to support business applications.
* Dependency Injection is a fundamental aspect of the Spring framework, through which the Spring container “injects” objects into other objects or “dependencies” to achieve loose coupling.
* Inversion of Control is a principle in software engineering which transfers the control of objects or portions of a program to a container or framework.
* The objects that form the backbone of your application and that are managed by the Spring IoC container are called beans.
* If one bean class is dependent on another bean class then the bean dependencies need to be explicitly defined in your configuration class. But you can let the Spring IoC container to inject the dependencies into dependent bean classes without been defined in your configuration class. This is called as autowiring.
* The controller class is responsible for processing incoming REST API requests, preparing a model, and returning the view to be rendered as a response.
* Aspect-oriented programming (AOP) is an approach to programming that allows global properties of a program to determine how it is compiled into an executable program.
* Spring Profiles provide a way to segregate parts of your application configuration and make it only available in certain environments.
* Hibernate is a Java framework that simplifies the development of Java application to interact with the database. Hibernate ORM is an object–relational mapping tool for the Java programming language. It provides a framework for mapping an object-oriented domain model to a relational database.



**Advantages of spring boot:**

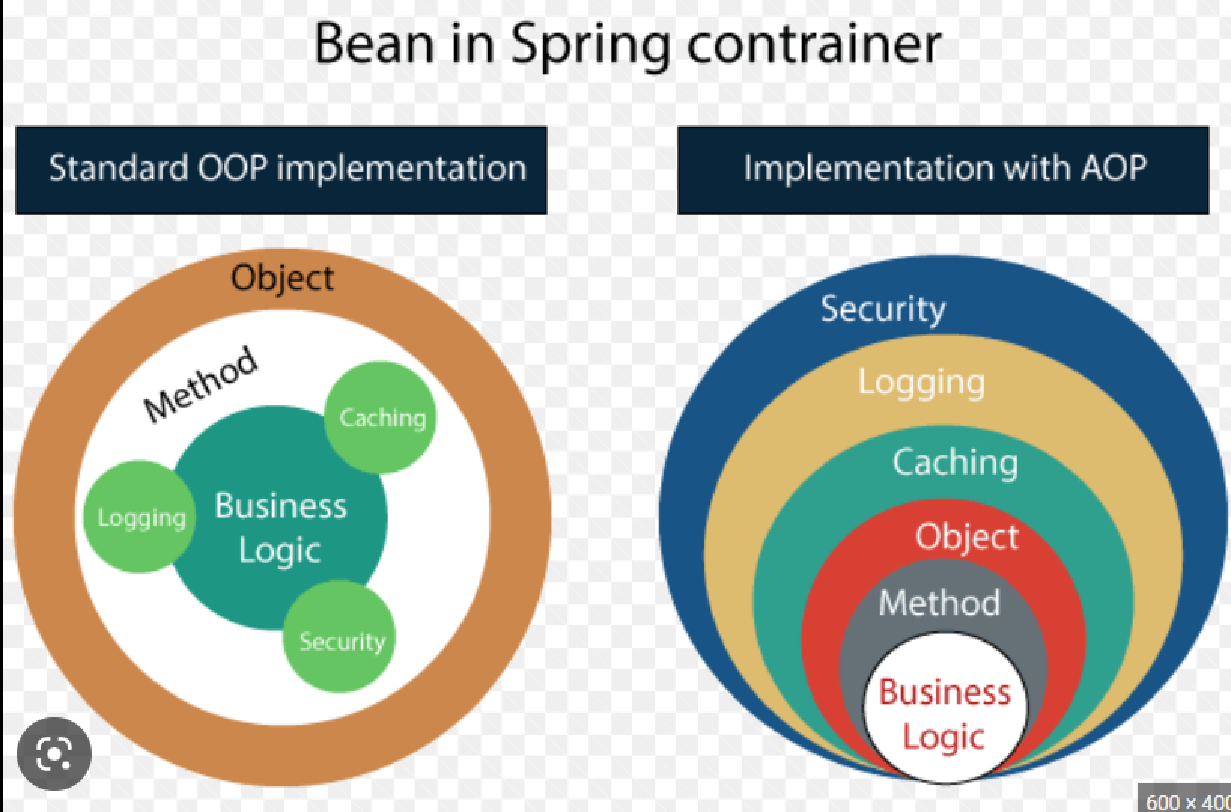
* It provides a flexible way to configure Java Beans, XML configurations, and Database Transactions.
* It provides a powerful batch processing and manages REST endpoints.
* In Spring Boot, everything is auto configured; no manual configurations are needed.
* It offers annotation-based spring application
* Eases dependency management
* It includes Embedded Tomcat server.
* For testing purpose we need to create mok object of the class that can only be possible if the objects are loosly coupled.
* An actuator is an additional feature of Spring that helps you to monitor and manage your application when you push it to production.
* SOAP – Simple Object Access Protocol

**Spring Bean Scopes**

* singleton - only one instance of the spring bean will be created for the spring container. This is the default spring bean scope.
* prototype – A new instance will be created every time the bean is requested from the spring container.
* request – This is same as prototype scope, however it’s meant to be used for web applications. A new instance of the bean will be created for each HTTP request.
* session – A new bean will be created for each HTTP session by the container.
* global-session – This is used to create global session beans for Portlet applications.

Autowring types : byName , byType(default) , constructor , autodetect.

**Aspect oriented programming:**

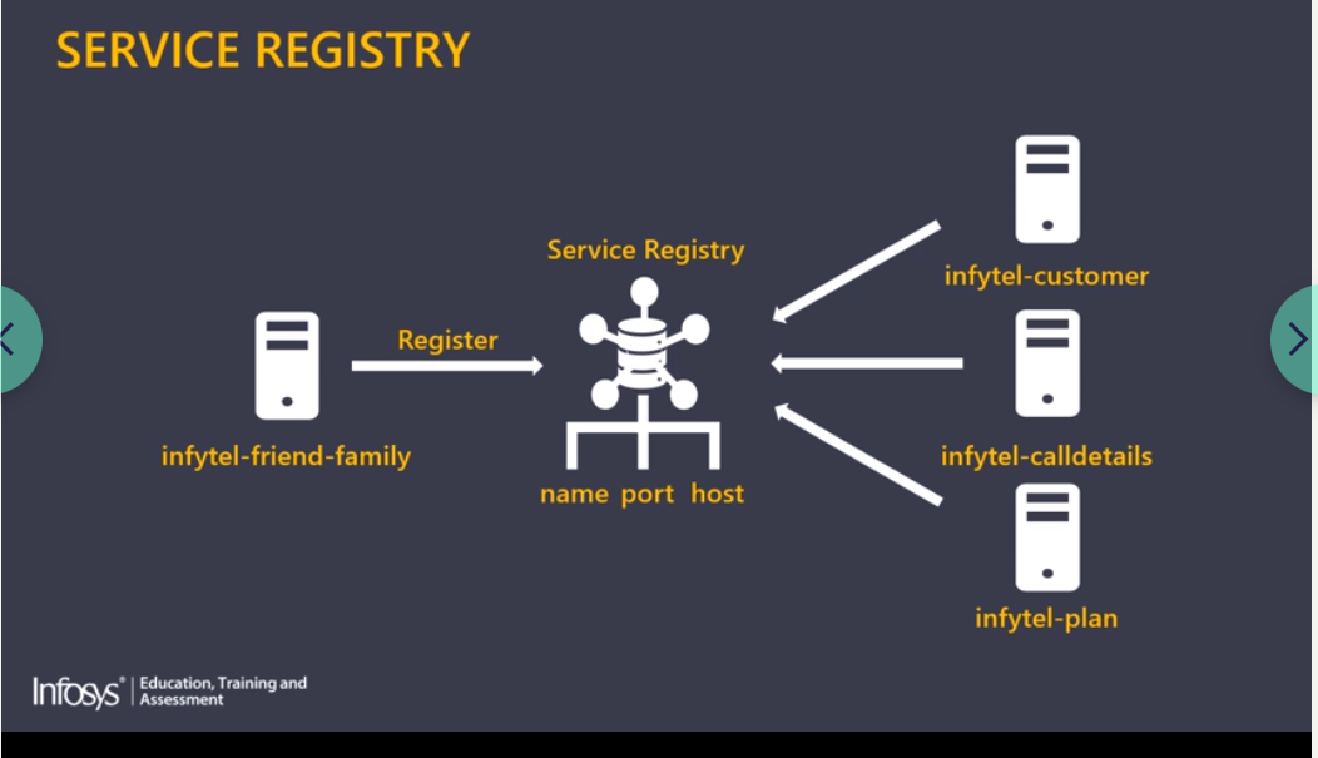
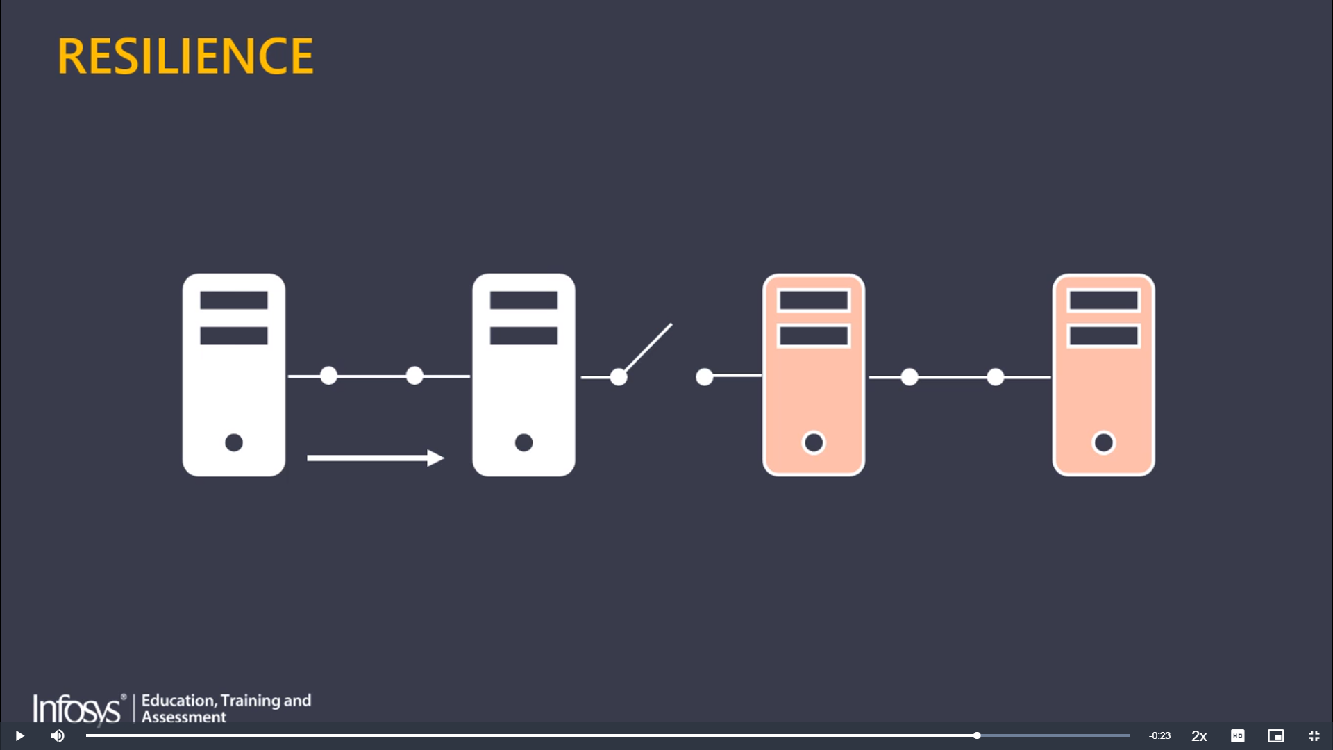
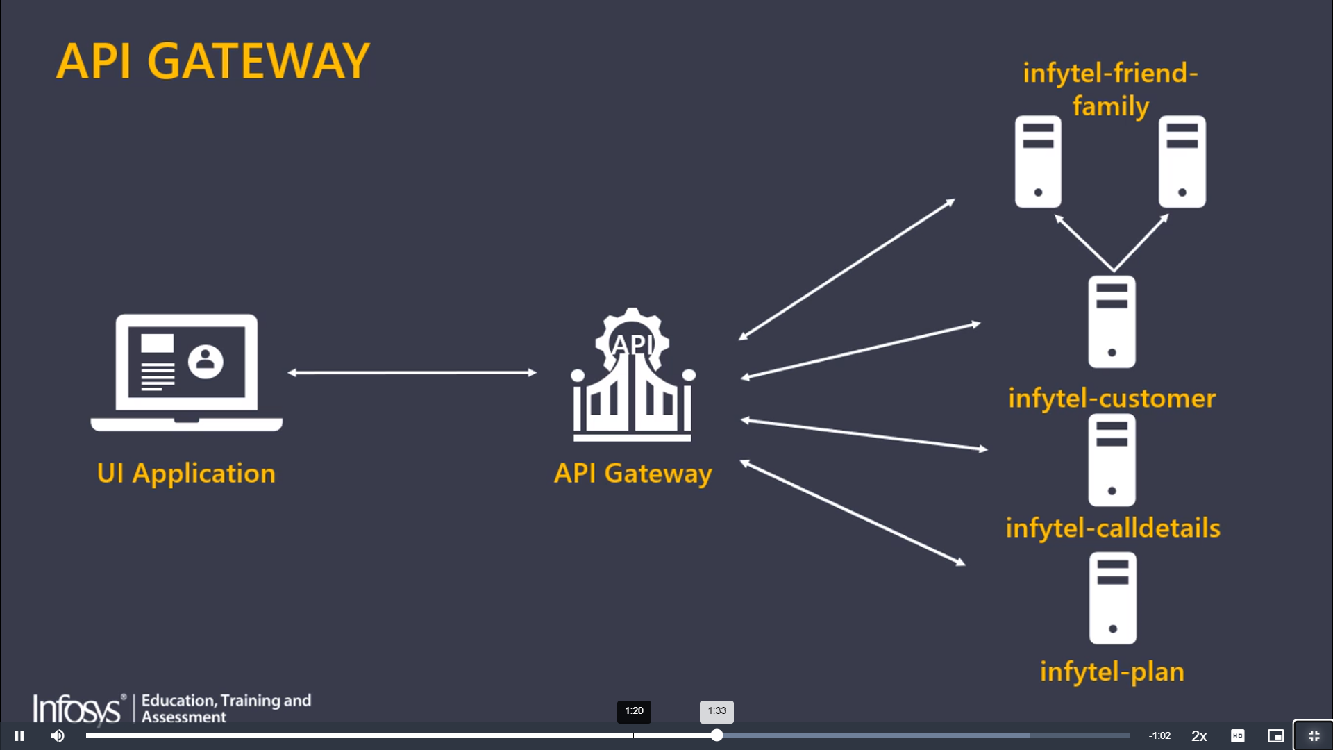
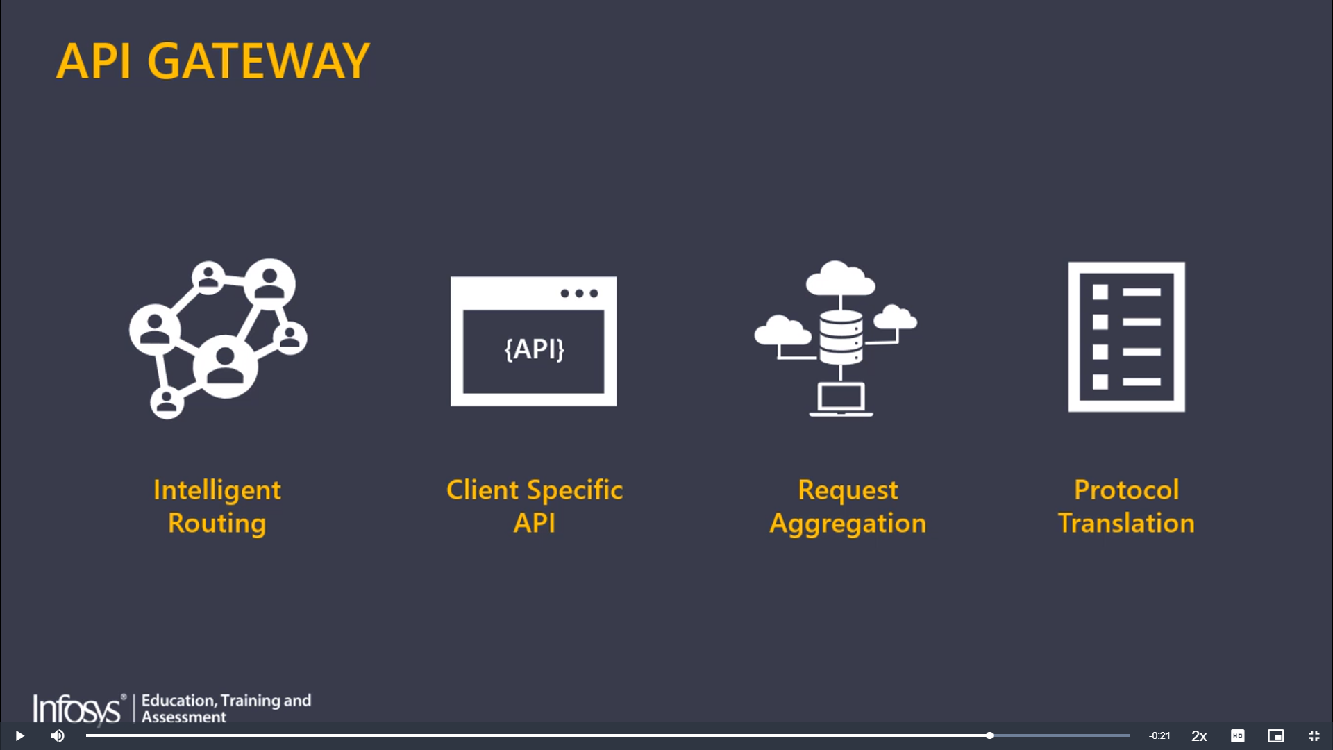
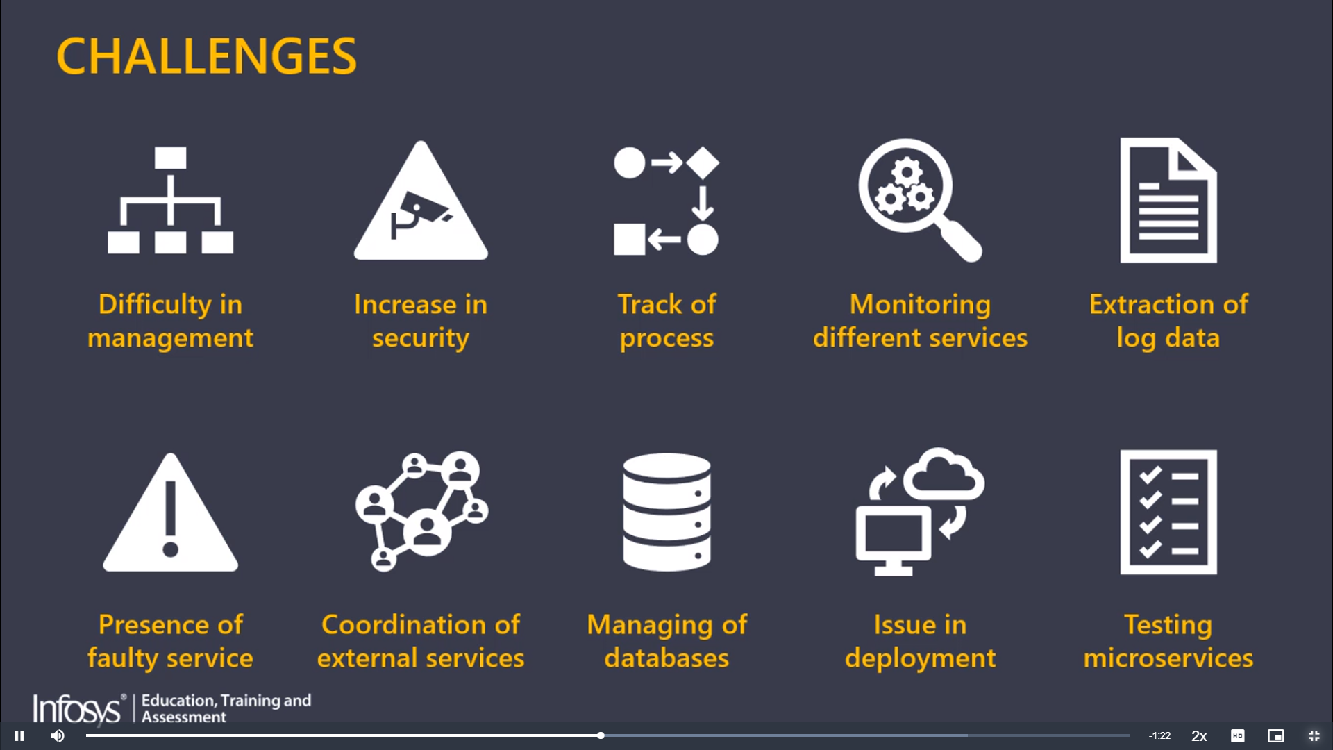
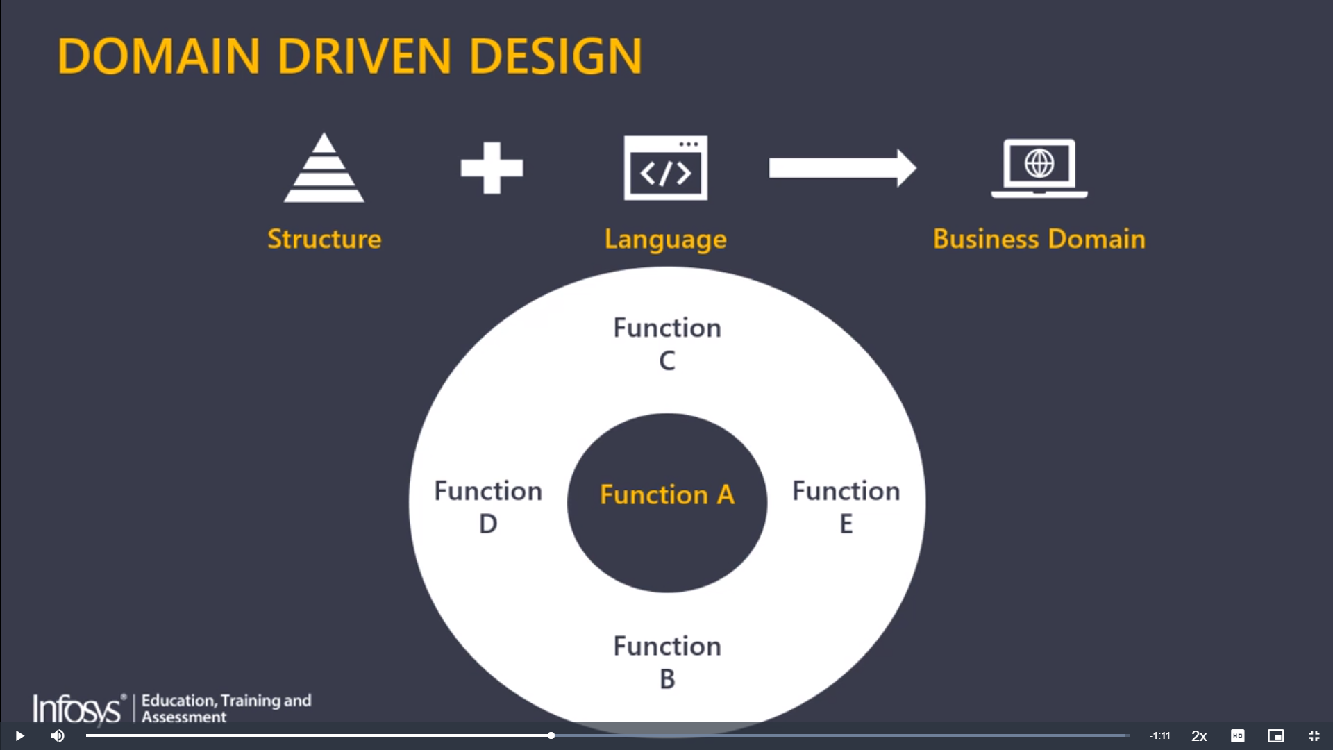
****

****

**Restful API architecture constraints:**

* Uniform Interface
* Stateless
* Cache-able
* Client-Server
* Layered System
* Code on Demand

**Microservices Basics information:**

* Distributed and centralized configuration: suppose we have 4 ms and all the conf. details are stored in service itself, but the problem with this approach is that we need to change into all 4 services and it needs to be redeployed if there is any kind of changes in one property file, hence we need to store all this properties files into external conf. server. ex : spring consul.
* Service Registery:
* 
* Load balancing refers to efficiently distributing incoming network traffic across a group of backend servers,
* Resilience: ex : resilience4j
* 
* Api gateway:
* 
* 
* Declarative client helps in integrating with load balancing api's and circuit breaker api's to provide load balancing and fall-back mechanism. ex: fegin
* **Challenges in MS:**
* ****
* ****

Spring Boot offers two models for building web applications: the traditional Servlet-based model and the reactive model. The Servlet-based model uses synchronous, blocking I/O with Spring MVC, suitable for traditional web apps with moderate concurrency. Key components include controllers, models, views, and the DispatcherServlet. The reactive model, built on Project Reactor and Spring WebFlux, uses non-blocking, event-driven paradigms with components like Mono and Flux. This model is ideal for high-throughput, low-latency applications, real-time systems, and microservices. The choice between these models depends on application requirements for performance, scalability, and concurrency.

Spring Security

CSRF – cross side request forgery

*@SpringBootApplication = @Configuration + @EnableAutoConfiguration + @ComponentScan*

RestTemplate is a synchronous client in the Spring Framework used for making HTTP requests to RESTful web services. It simplifies the process of interacting with HTTP servers by handling the boilerplate code required for sending and receiving HTTP requests and responses.

Ideal for microservices that require high throughput, low latency, and the ability to handle a large number of simultaneous requests without blocking threads.

The Three-Phase Commit protocol extends the Two-Phase Commit protocol by adding an additional phase to handle network partitions and failures more gracefully. ()

When dealing with common properties or shared functionality across multiple microservices we can do following:

Use an API Gateway to handle common cross-cutting concerns such as authentication, logging, and request routing.

* **Use shared libraries or modules** to encapsulate common functionality.
* **Centralize configuration management** with tools like Spring Cloud Config or Consul.
* **Leverage an API Gateway** to handle cross-cutting concerns.
* **Define and share common data models or API contracts** to maintain consistency.
* **Use service templates or boilerplates** to standardize the development of new microservices.

Behavior-Driven Development (BDD) is a software development approach that enhances collaboration between developers, testers, and non-technical stakeholders by focusing on the behavior of an application from the end-user’s perspective. It extends Test-Driven Development (TDD) by incorporating natural language statements that describe the desired behavior of the system.

**Service discovery** is a key concept in microservices architecture and distributed systems, used to enable services to dynamically find and communicate with each other. It addresses the challenge of locating services in a network where services can be started, stopped, or scaled dynamically.