**Spring Boot WebFlux | Write Integration Test Using Mockito & Junit | JavaTechie (** [**https://www.youtube.com/watch?v=Zf7NB7yKisI**](https://www.youtube.com/watch?v=Zf7NB7yKisI) **)**

Components of Microservices

There are the following components of microservices:

* Spring Cloud Config Server
* Netflix Eureka Naming Server
* Hystrix Server
* Netflix ZuulAPI Gateway Server
* Netflix Ribbon
* Zipkin Distributed Tracing Server

### **Spring Cloud Config Server**

Spring Cloud Config Server provides the HTTP resource-based API for external configuration in the distributed system. We can enable the Spring Cloud Config Server by using the annotation **@EnableConfigServer**.

### **Netflix Eureka Naming Server**

Netflix Eureka Server is a discovery server. It provides the REST interface to the outside for communicating with it. A microservice after coming up, register itself as a discovery client. The Eureka server also has another software module called **Eureka Client**. Eureka client interacts with the Eureka server for service discovery. The Eureka client also balances the client requests.

* @EnableEurekaClient
* @EnableEurekaServer

### **Hystrix Server**

Hystrix server acts as a fault-tolerance robust system. It is used to avoid complete failure of an application. It does this by using the **Circuit Breaker mechanism**. If the application is running without any issue, the circuit remains closed. If there is an error encountered in the application, the Hystrix Server opens the circuit. The Hystrix server stops the further request to calling service. It provides a highly robust system.

### **Netflix Zuul API Gateway Server**

Netflix Zuul Server is a gateway server from where all the client request has passed through. It acts as a unified interface to a client. It also has an inbuilt load balancer to load the balance of all incoming request from the client.

### **Netflix Ribbon**

Netflix Ribbon is the client-side Inter-Process Communication (IPC) library. It provides the client-side balancing algorithm. It uses a Round Robin Load Balancing:

* Load balancing
* Fault tolerance
* Multiple protocols(HTTP, TCP, UDP)
* Caching and Batching

### **Zipkin Distributed Server**

Zipkin is an open-source project. That provides a mechanism for sending, receiving, and visualization traces.

**One thing you need to be focused on that is port number.**

|  |  |
| --- | --- |
| **Application** | **Port** |
| Spring Cloud Config Server | 8888 |
| Netflix Eureka Naming Server | 8761 |
| Netflix Zuul API gateway Server | 8765 |
| Zipkin distributed Tracing Server | 9411 |

**Jenkins (CI/CD tool)** [**https://www.youtube.com/watch?v=mszE-OCI2V4**](https://www.youtube.com/watch?v=mszE-OCI2V4)

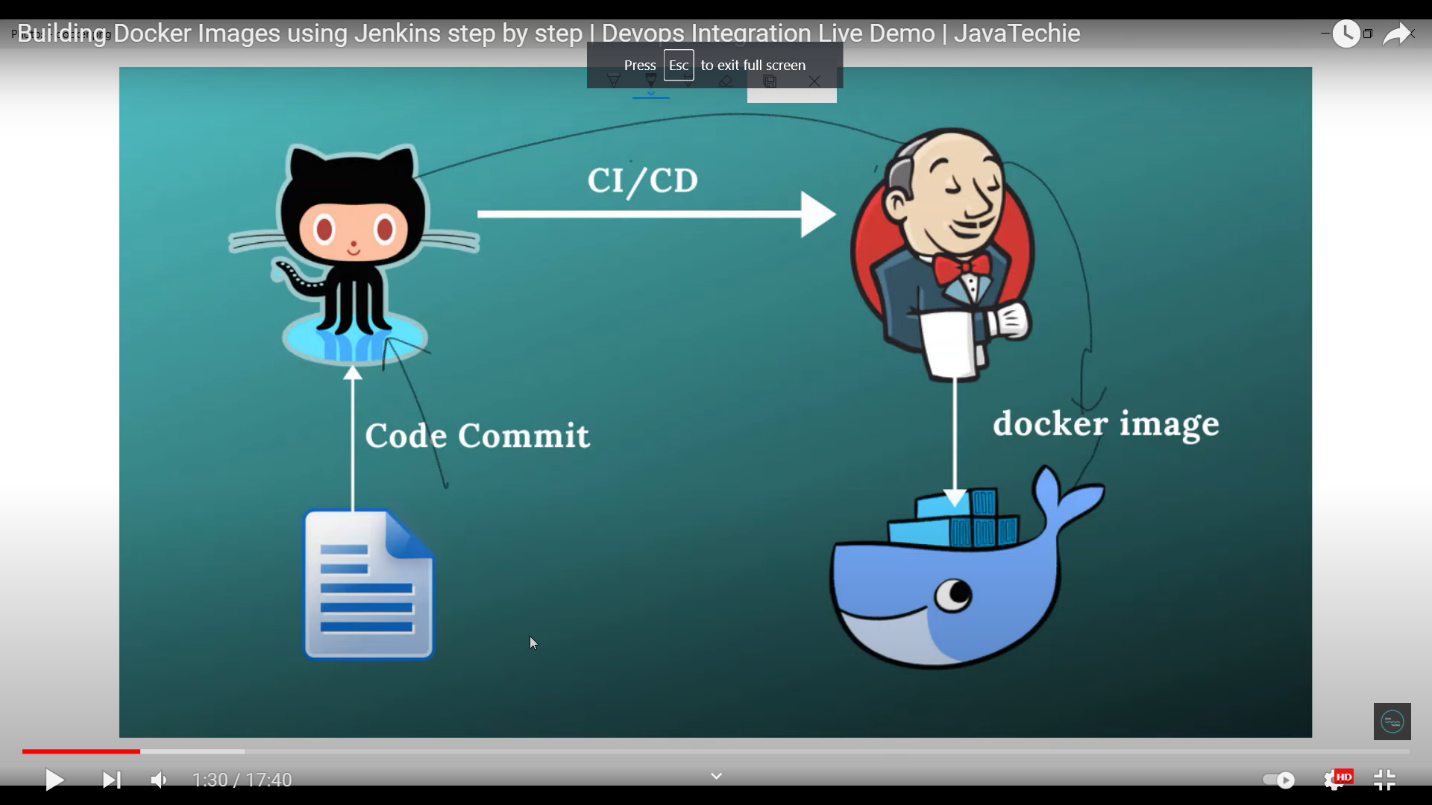
Jenkins is an open-source automation tool written in Java with plugins built for Continuous Integration purposes.

Jenkins is used to build and test your software projects continuously making it easier for developers to integrate changes to the project, and making it easier for users to obtain a fresh build.

It also allows you to continuously deliver your software by integrating with a large number of testing and deployment technologies.

With Jenkins, organizations can accelerate the software development process through automation.

Jenkins integrates development life-cycle processes of all kinds, including build, document, test, package, stage, deploy, static analysis, and much more.

****

**Step to run Jenkins**

**Step 1: run Jenkins war file**

**Java -jar jenkin.war**

**Step 2: Create maven project**

**Step 3: Project commit in git.**

**Step 4: open browser**

**->Then localhost:8080**

**-> create new job**

**-> Entry Item name(Project Name)**

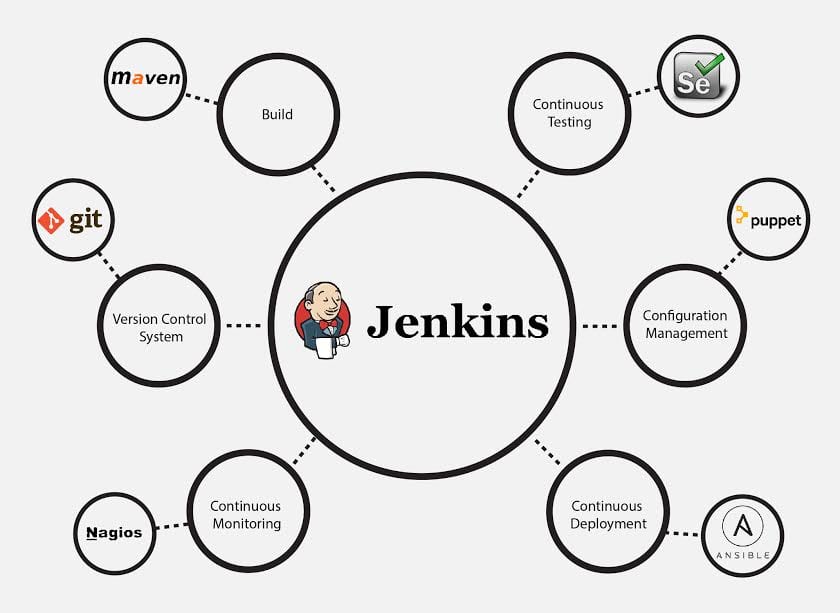
**->click on freestyle project**

**->select github project and give git url**

**-> In source code management enter Repository url**

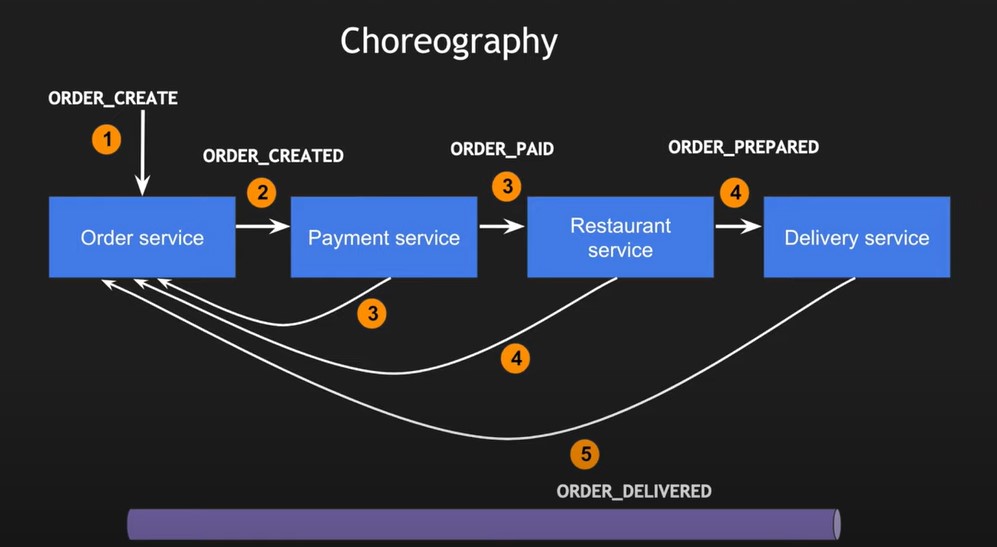
**-> In build trigger click POLL SCM**

**-> And set job crown job**



**Microservice Design Pattern:**

* **API Gateway Design Pattern**
* **Circuit Breaker Pattern**
* **Saga Design pattern**
* **Choreography (Event Based)**
* **Orchestration (command based)**

****