**Spring Boot WebFlux | Write Integration Test Using Mockito & Junit | JavaTechie (** **<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(https://www.youtube.com/watch?v=XtutSW2zCNw)
* 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 the 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 requests from the client.

**Why use API Gateway: (Problem with direct API call from UI to many microservices)**

\* Many hardcode URL

\* No Security

\* UI Code changes require when changes in backend API

\* Load Balancer

\* Rerouting Issue

### **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=AqITZLJ5eZ4**](https://www.youtube.com/watch?v=AqITZLJ5eZ4)

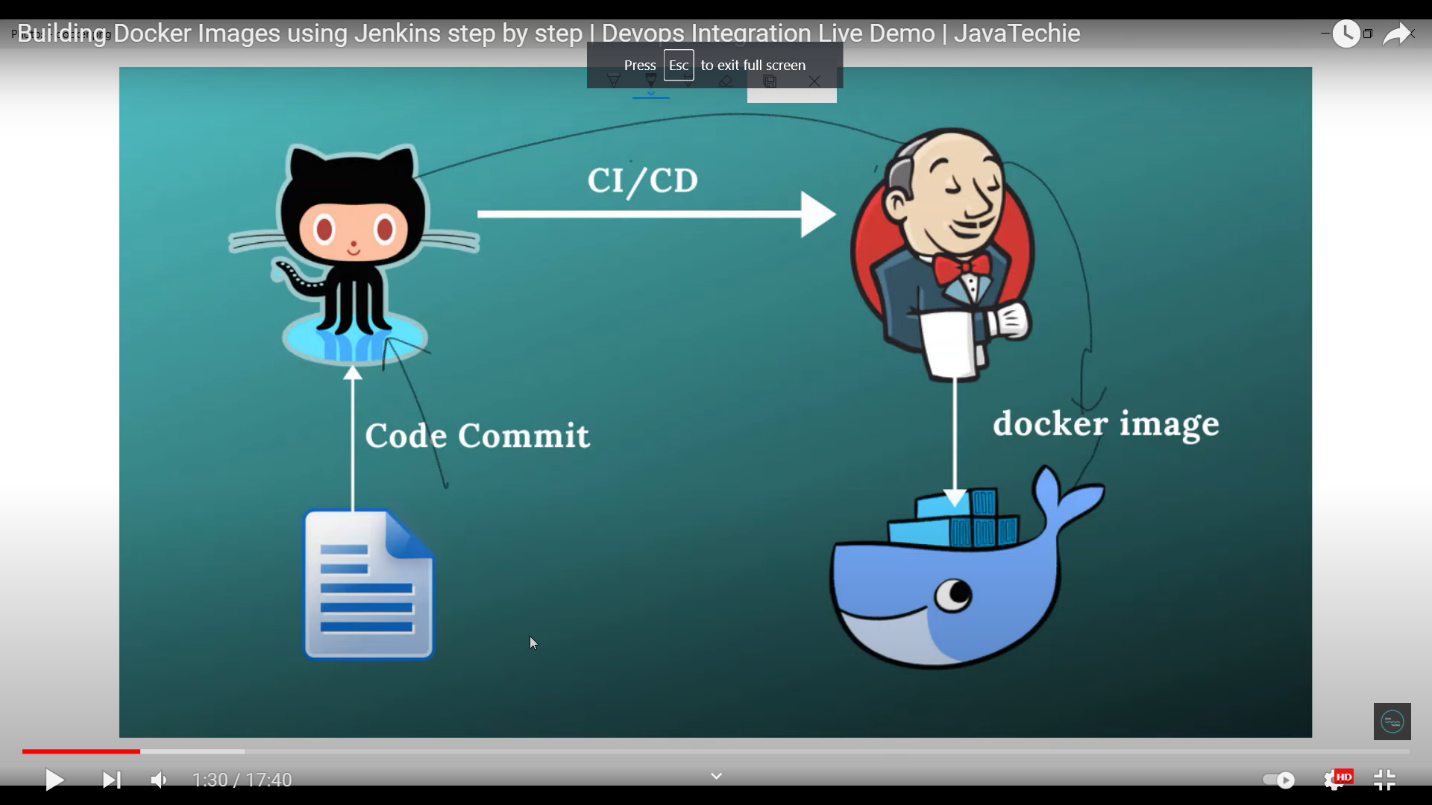
Jenkins is an open-source automation tool written in Java with plugins built for Continuous Integration purposes.

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**

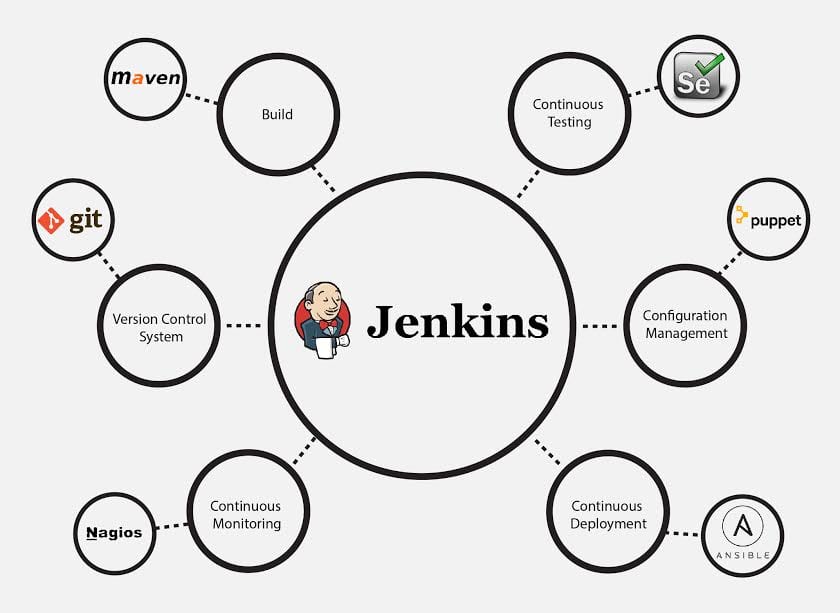
**JOB Schedule:**

Jenkins used a [cron expression](https://en.wikipedia.org/wiki/Cron" \l "CRON_expression), and the different fields are:

1. MINUTES Minutes in one hour (0-59)
2. HOURS Hours in one day (0-23)
3. DAYMONTH Day in a month (1-31)
4. MONTH Month in a year (1-12)
5. DAYWEEK Day of the week (0-7) where 0 and 7 are sunday

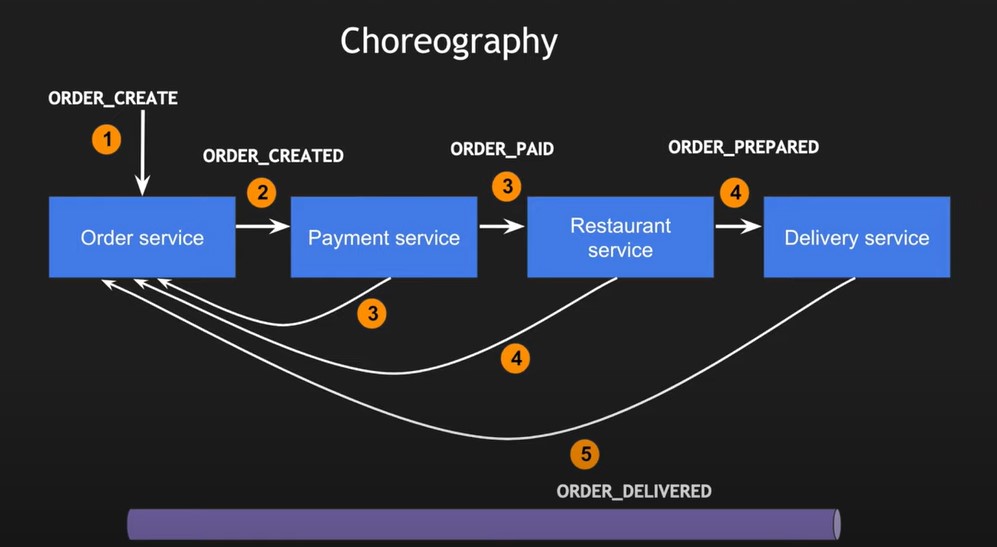
If you want to schedule your build every 5 minutes, this will do the job : \*/5 \* \* \* \*

If you want to schedule your build every day at 8h00, this will do the job : 0 8 \* \* \*



**Microservice Design Pattern:**

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

****

**=======================================================================================**

**resilience4j**

<https://medium.com/bliblidotcom-techblog/resilience4j-circuit-breaker-implementation-on-spring-boot-9f8d195a49e0>

<https://www.youtube.com/watch?v=b6R4dElDtRc>

<dependency>

<groupId>io.github.resilience4j</groupId>

<artifactId>resilience4j-spring-boot2</artifactId>

<version>1.7.0</version>

</dependency>

1. Circuit Breaker(Fault Tolerence)
2. Rate Limiter (Block too frequent Request)
3. Time Limiter(set a time limit when calling remote operation)
4. Retry Mechenism(Automatically retry a failed remote operation)
5. Bulthead(Avoid too many concurrent request)
6. Cache (Store result of costly remote operations)

[Resilience4j](https://github.com/resilience4j/resilience4j) is a lightweight fault tolerance library inspired by Netflix Hystrix, but designed for functional programming.  
Resilience4j provides higher-order functions (decorators) to enhance any functional interface, lambda expression or method reference with a Circuit Breaker, Rate Limiter, Retry or Bulkhead. You can stack more than one decorator on any functional interface, lambda expression or method reference.

**Spring Cloud Stream:**

Spring Cloud Stream is a framework built on top of Spring Boot and Spring Integration that **helps in creating event-driven or message-driven microservices**.

Spring Cloud Stream is a framework for building highly scalable event-driven microservices connected with shared messaging systems.

<**dependency**> <**groupId**>org.springframework.cloud</**groupId**> <**artifactId**>spring-cloud-stream-test-support</**artifactId**> <**version**>3.1.3</**version**> <**scope**>test</**scope**> </**dependency**>