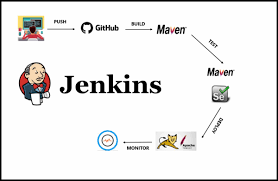
# CI/CD

1. CI/CD stands for continuous integration and continuous deployment.
2. In a production environment, we need to deploy continuously and very frequently.
3. At this scale manual deploy is impossible to keep up the requirement and provide a stable service.
4. So, we use a CI/CD tool to automate these deployments.



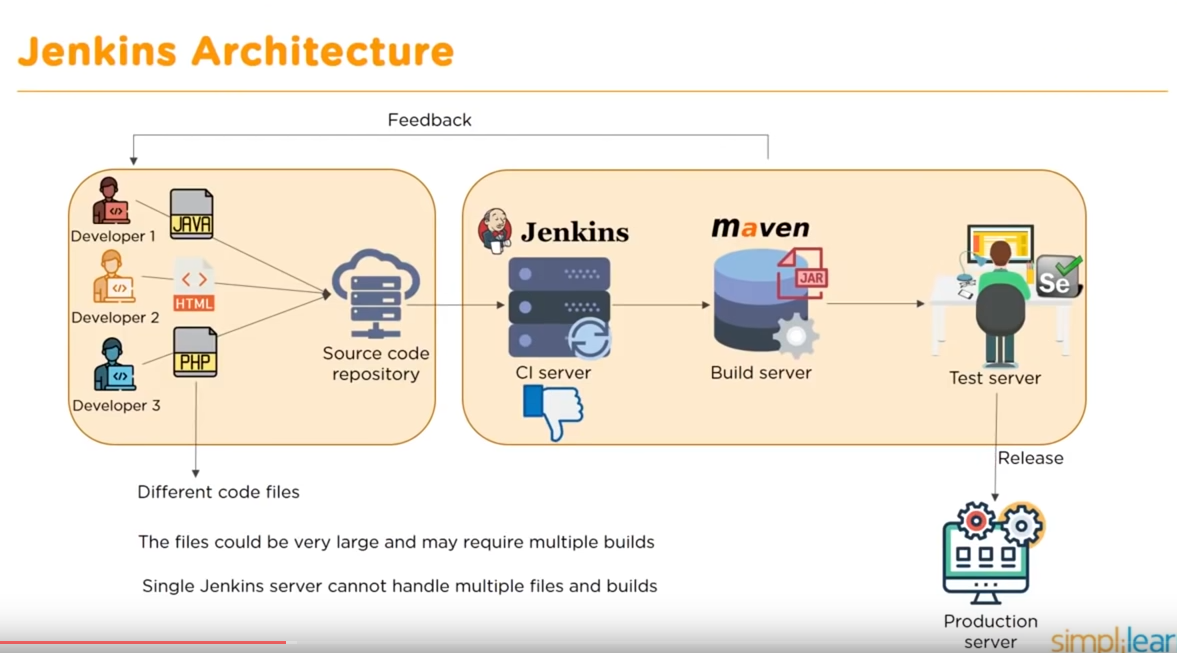
# What are the CI/CD tool

There are very very good open source and paid solutions for CI/CD. One of the very popular tools is Jenkins.



# How does it work?

1. Jenkins uses environment pugin to connect and build our java project.
2. For example, here we will deploy a java project which uses maven. So, Jenkins will need a maven plugin to build this project.z



# Overview

Here is the JAVA web application deployment procedure using Jenkins,

1. Prepare server
   1. Install JDK (java development kit) and set environment for JDK
   2. Install Maven (project management tool)
   3. Install tomcat (Java container)
2. Install jenkins
3. Install necessary plugin
   1. Git
   2. Maven Integration
   3. Deploy to container
4. Deploy a java application
   1. Download a source code from git
   2. Build with maven
   3. Deploy the application to tomcat
   4. Send an email confirmation (if needed)
   5. Add a scheduler/cron job (if needed)
5. Testing
6. Monitoring

# 

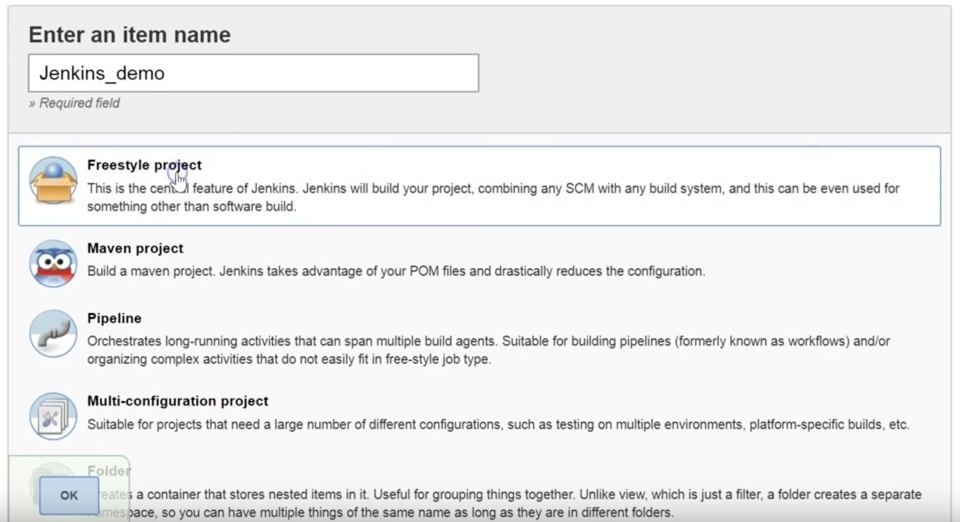
# 

# Work-Flow for deploying a Java code

1. Install necessary softwares at server
   1. Install java to the server
   2. Install tomcat to the server
   3. Install maven to the server
2. Install jenkins
3. Install necessary plugins
   1. git
   2. Maven Integration
   3. Deploy to container
4. Login to the jenkins
5. Create a “New Item”



1. Create “Freestyle project”



1. Select “Git” from “source code management” section

URL: https://github.com/rchidana/mvnwebapp.git

Username:

Password:

1. As our project is created with maven, let's install the plugin “maven”. Now go to the “Build” section and select a build option “Execute shell command” and type “mvn package”. This will create a war file from our source code.
2. Install a plugin named “Deploy to a container”
3. Add a “post build step” > deploy war to a container
4. Select the location.
   1. War files: \*\*/\*.war
   2. Context path: name of the application
   3. Container: tomcat 8,
   4. Credentials: provide access credential of our tomcat
   5. Tomcat URL: http://192.168.0.104:8080
5. Save the configuration
6. Build now
7. Check the console output for log

# 

# Prepare server system’s environment

## A. Install JDK (java development kit)

**Install openJDK:**

[root@server2 ~]# yum install java-1.8.0 java-1.8.0-openjdk-devel -y

**Or Oracle JDK:**

**From rpm:**

[root@server2 ~]# wget https://download.oracle.com/java/17/latest/jdk-17\_linux-x64\_bin.rpm

[root@server2 ~]# yum install jdk-17\_linux-x64\_bin.rpm

[root@server2 ~]# java -version

[root@server2 ~]# which java

/usr/bin/java

**From tar:**

Download the latest version of java from “http://www.oracle.com/technetwork/java/javase/downloads/index.html”.

[root@websrv ~]# mkdir -p /usr/jdk

[root@websrv ~]# tar -zxvf jdk-7u75-linux-x64.tar.gz -C /usr/jdk/

[root@websrv ~]# ls /usr/jdk/jdk1.7.0\_75/

Now, we will set it as our new java location for system uses. First we will check the current java version.

[root@websrv ~]# java -version

Then we will set up new java for our system.

[root@websrv ~]# export JAVA\_HOME=/usr/jdk/jdk1.7.0\_75/

[root@websrv ~]# export PATH=$JAVA\_HOME/bin:$PATH

But it will be removed after a server restart. So, Let's make it permanent.

[root@localhost ~]# vim /etc/profile

done

export JAVA\_HOME=/usr/jdk/jdk1.7.0\_75/

export PATH=$JAVA\_HOME/bin:$PATH

unset i

unset pathmunge

And check new/updated java version

[root@websrv ~]# java -version

## B. Install tomcat

1. Download tomcat tar.gz
2. Unzip the tomcat folder

## C. Install maven

1. Download maven tar.gz (https://dlcdn.apache.org/maven/maven-3/3.8.6/binaries/apache-maven-3.8.6-bin.tar.gz)
2. Unzip the maven folder
3. Set environment path for maven

## D. Install Jenkins

For ubuntu: https://linuxhint.com/install-jenkins-on-ubuntu/

sudo apt-get install openjdk-11-jdk

Installation of a stable version from yum:

[root@jenkins ~]# wget -O /etc/yum.repos.d/jenkins.repo http://pkg.jenkins.io/redhat-stable/jenkins.repo

[root@jenkins ~]# rpm --import http://pkg.jenkins.io/redhat-stable/jenkins.io.key

[root@jenkins ~]# yum update

[root@jenkins ~]# yum install jenkins

[root@jenkins ~]# ln -s /usr/jdk/jdk1.8.0\_321/bin/java /usr/bin/java

[root@app1 ~]# service jenkins restart

[root@app1 ~]# chkconfig jenkins on

If You get This Error:

--> Finished Dependency Resolution

Error: Package: jenkins-2.303.1-1.1.noarch (jenkins)

Requires: daemonize

You could try using --skip-broken to work around the problem

You could try running: rpm -Va --nofiles --nodigest

# yum install epel-release -y

# yum install daemonize -y

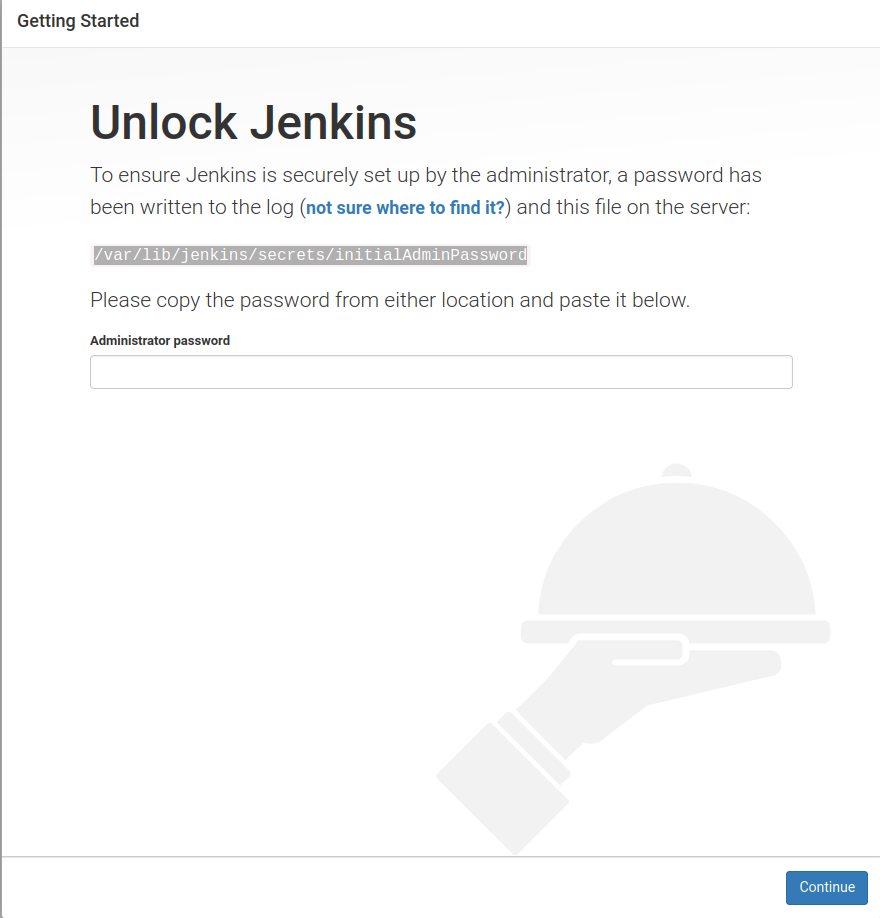
Than you can continue on installing the Jenkins as

# yum install jenkins -

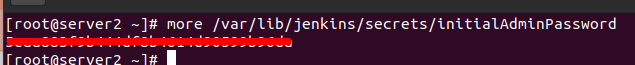
Go to the browser and hit below address,

<http://localhost:8080>

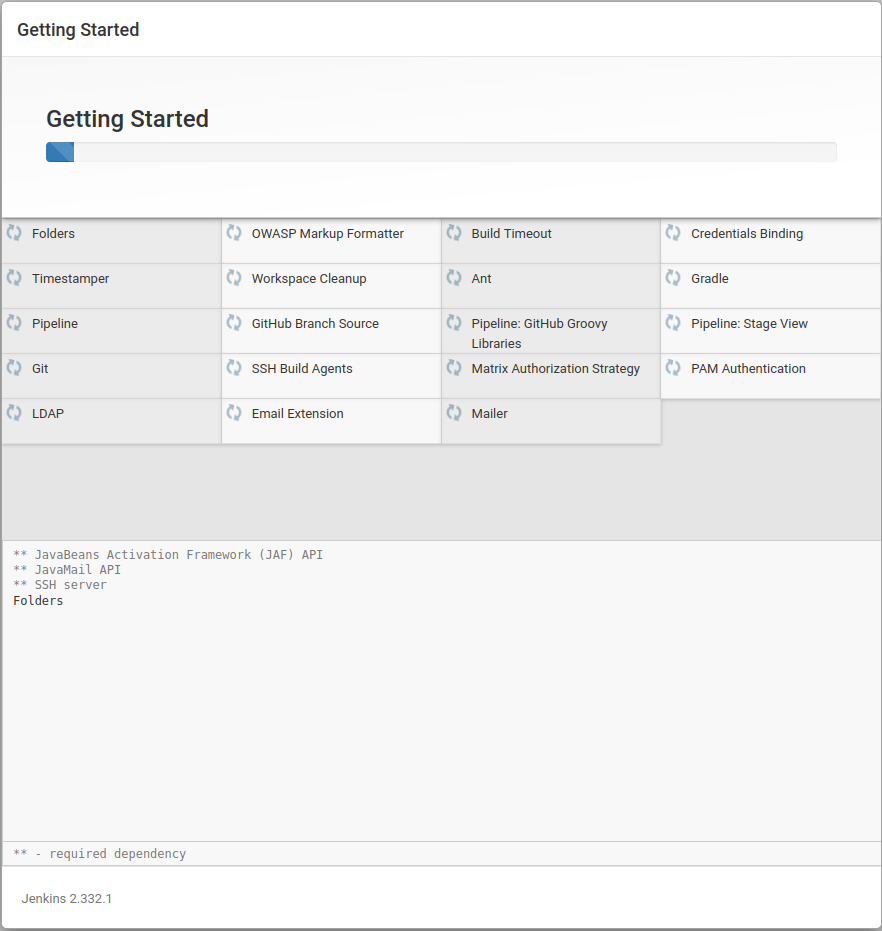
Step #1:



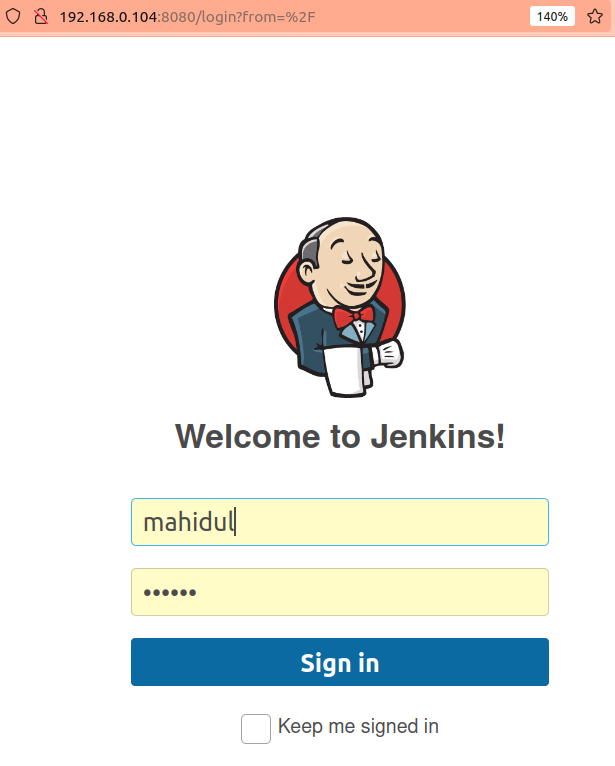
Step #2:



Step #3:



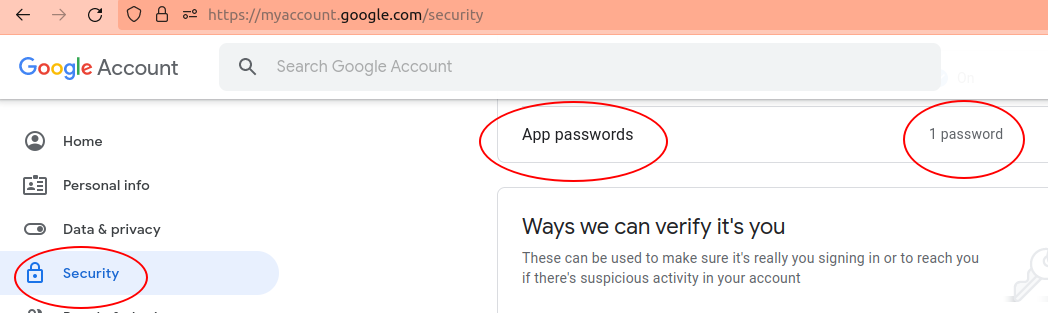
Step #4:



## How to send an email confirmation with jenkins

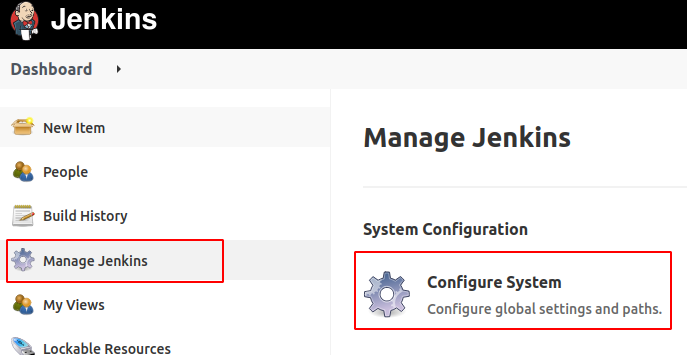
**Configure SMTP for email configuration**

1. Gmail does allow less secure connections. So, we have to create a app password which will be used by the jenkins email client.

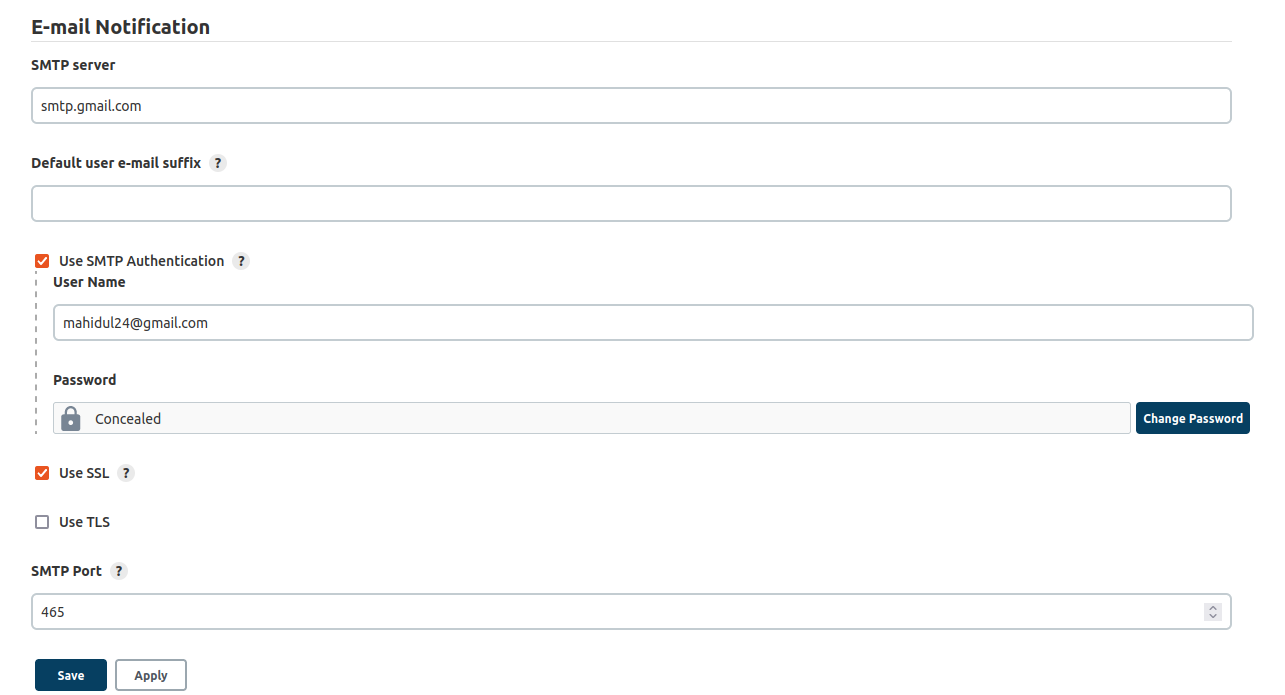


**Configure email client**

1. Go to “**Manage Jenkins**”. Then “**Configure System**”



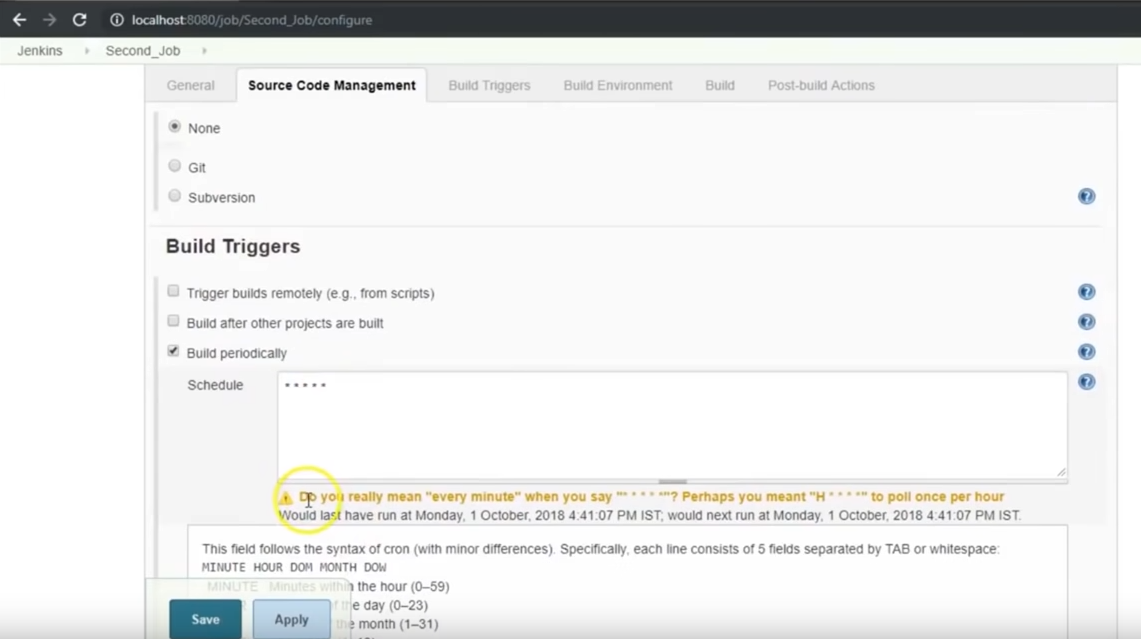
1. Go to “**E-mail Notification**”



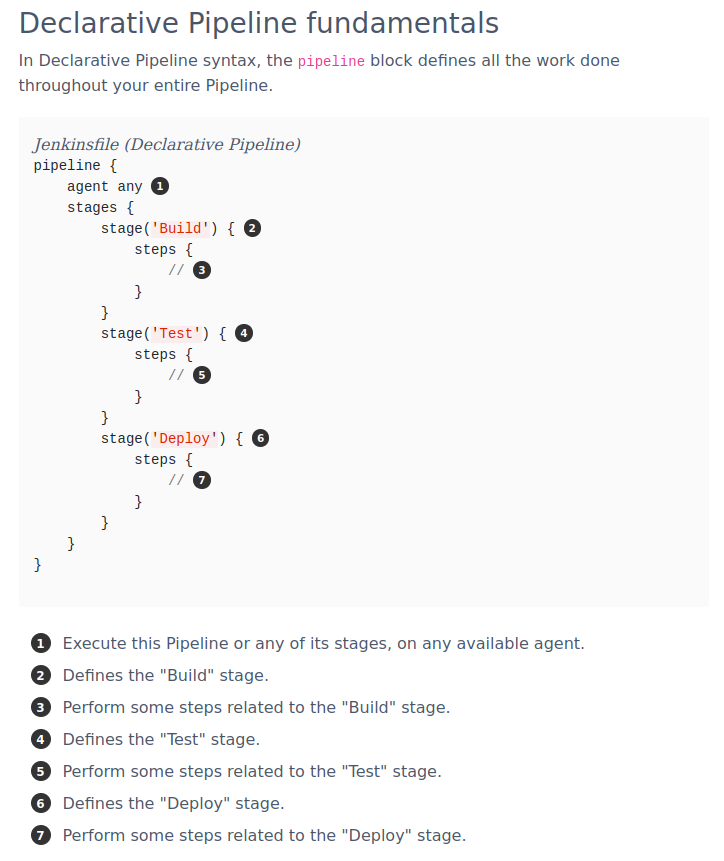
# 

# Add a scheduler/cron job

Go to “Build Trigger” to add a new cron job/schedule



# Jenkins PIPELINE



**Testing a demo pipeline**

pipeline {

agent any

stages {

stage('Build') {

steps {

echo 'Hello world!'

}

}

stage('Test') {

steps {

echo 'service start nginx'

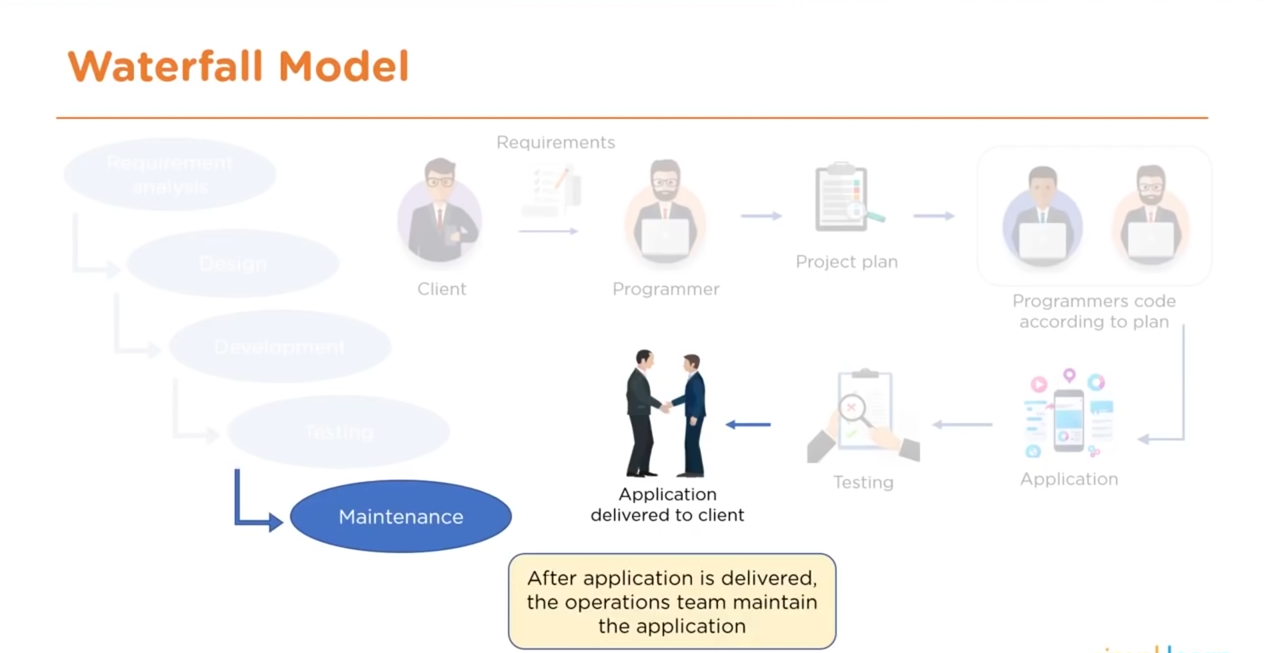
}

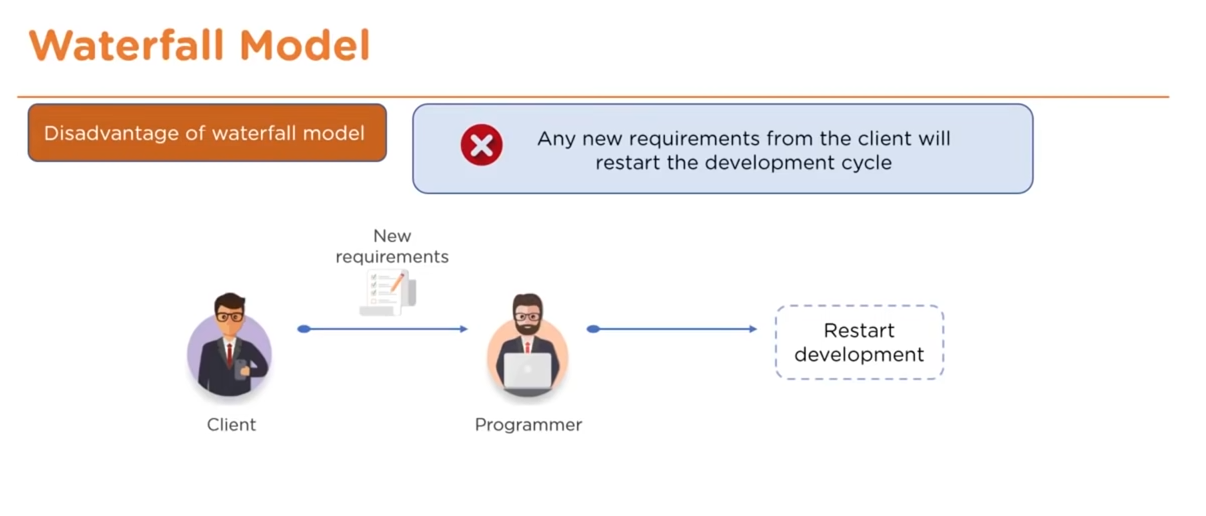
}

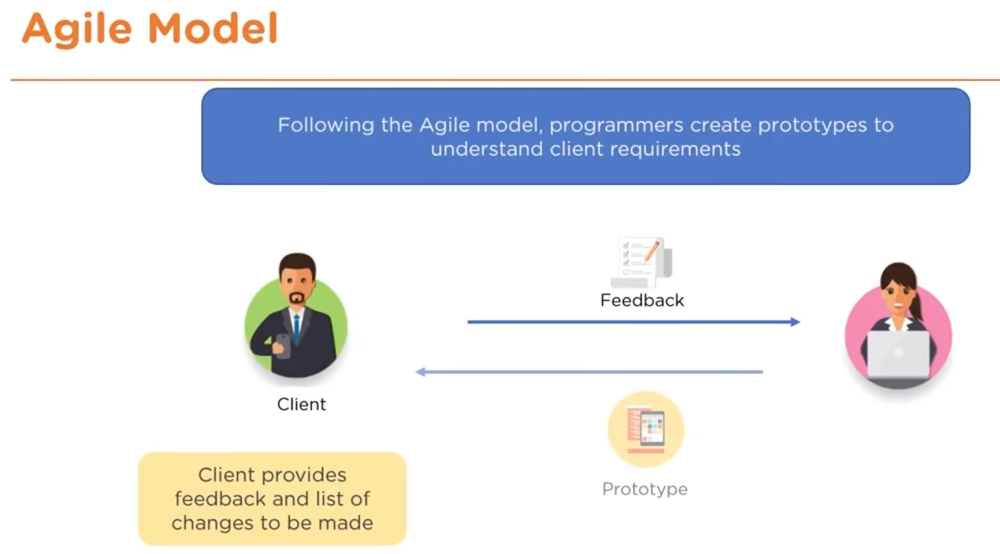
}

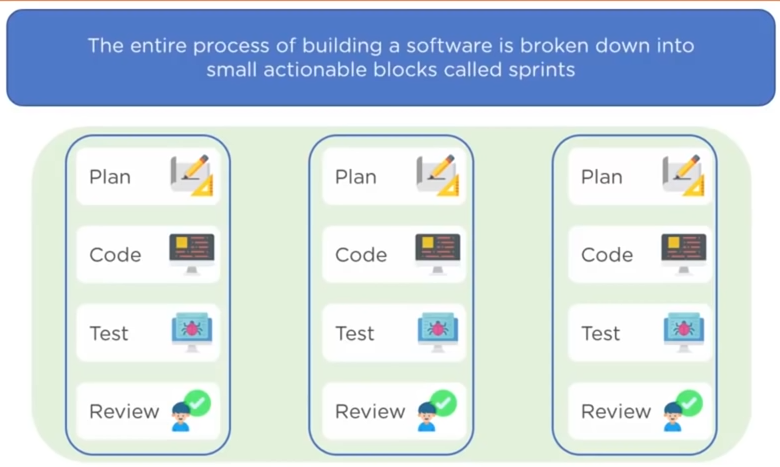
}

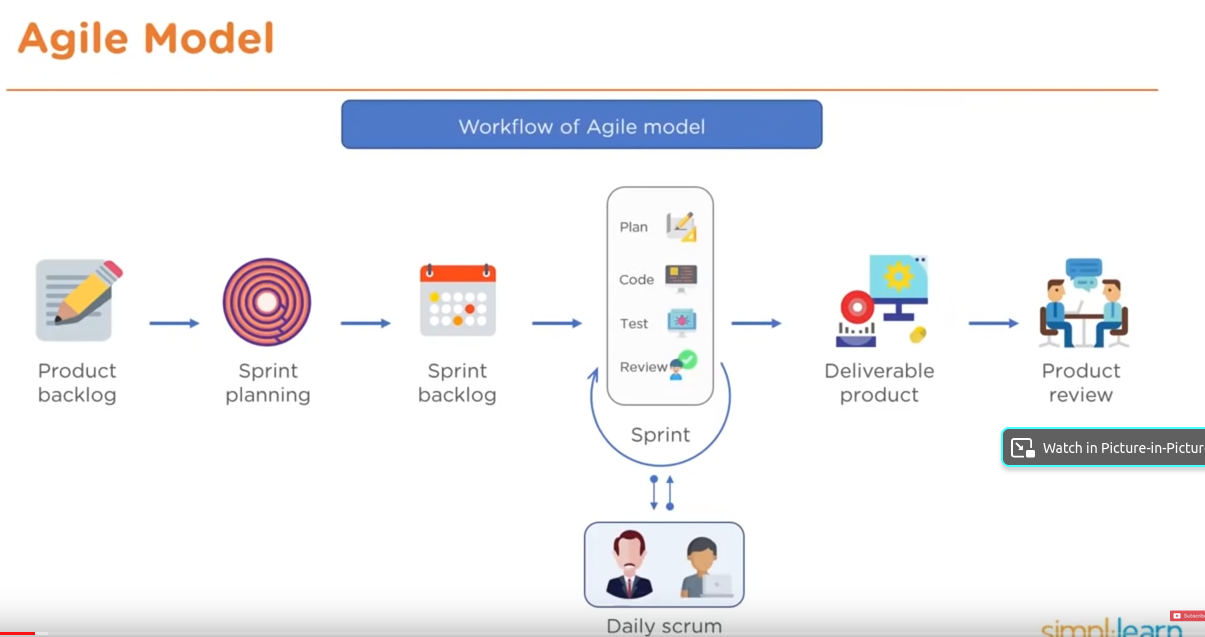
# Advance jenkins features

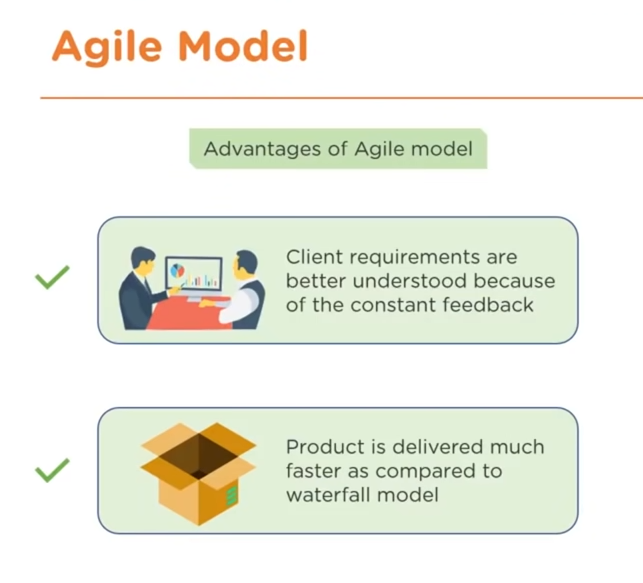


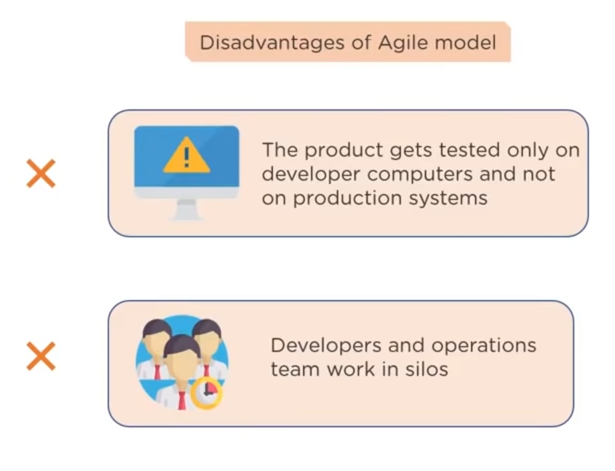


****

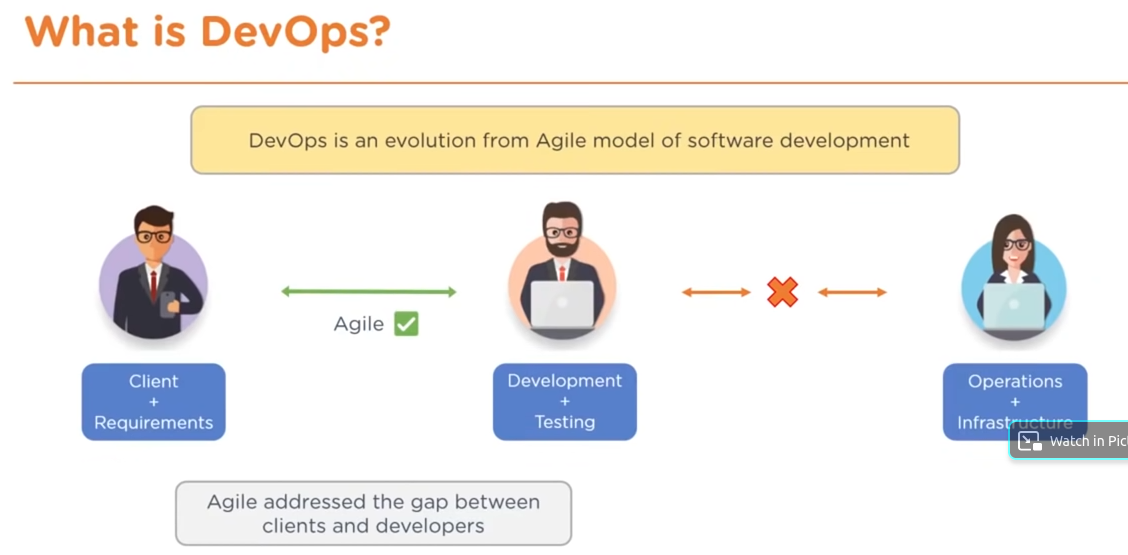


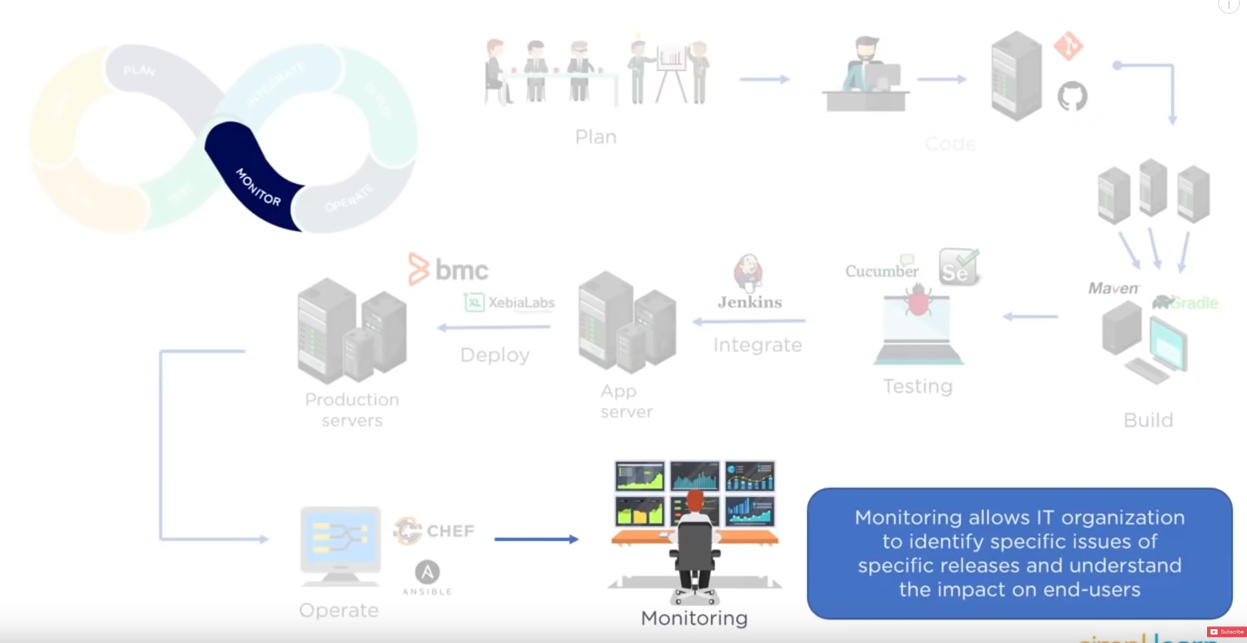


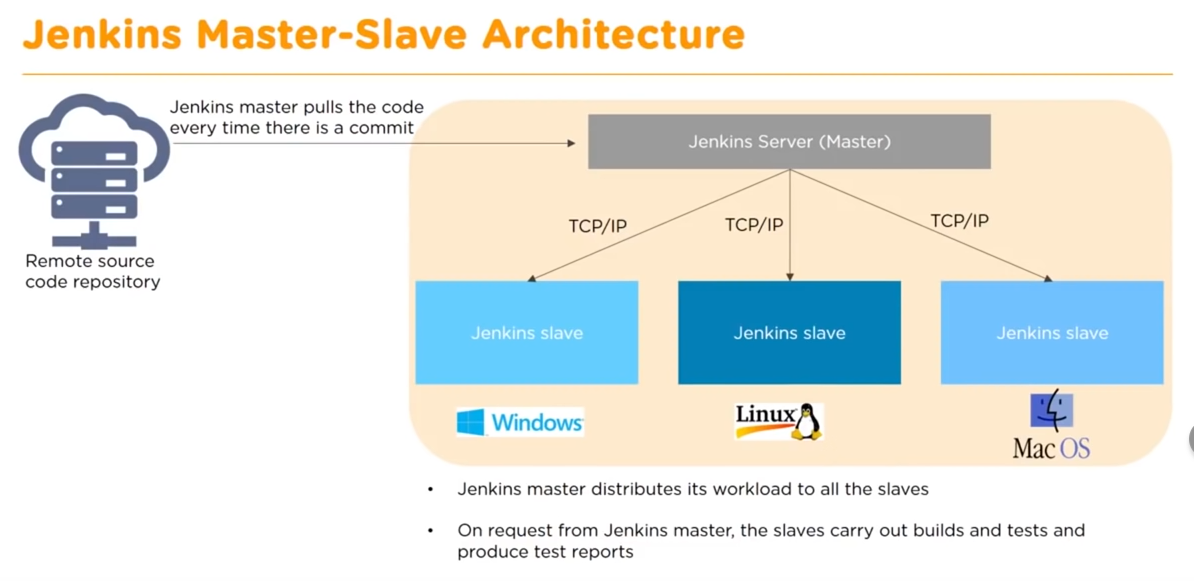


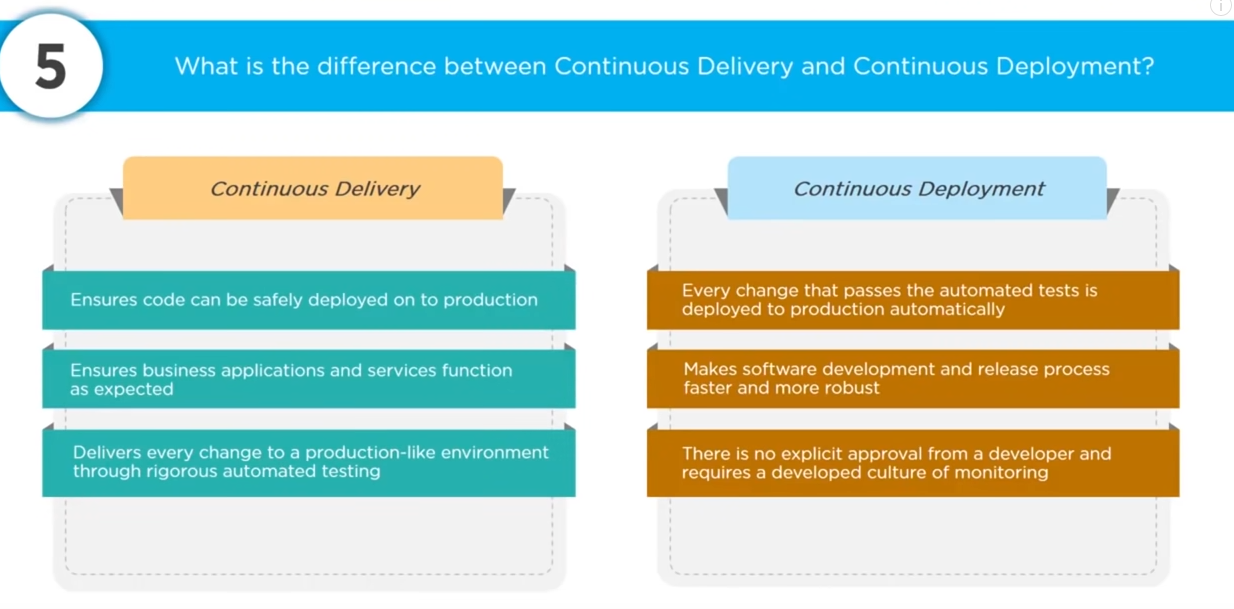


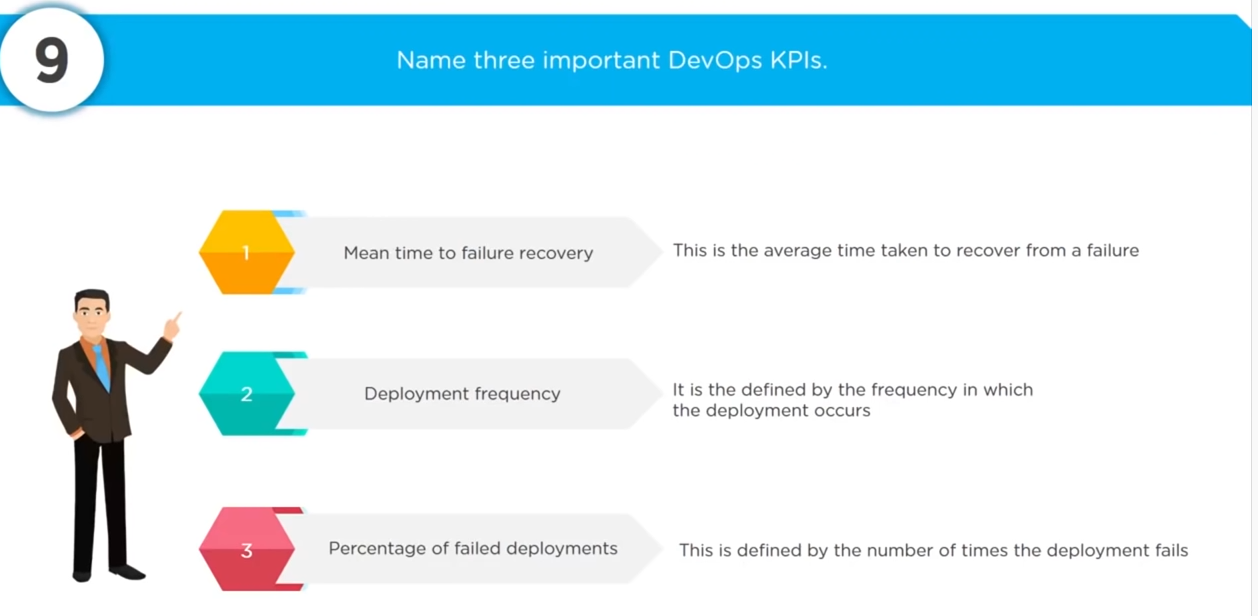


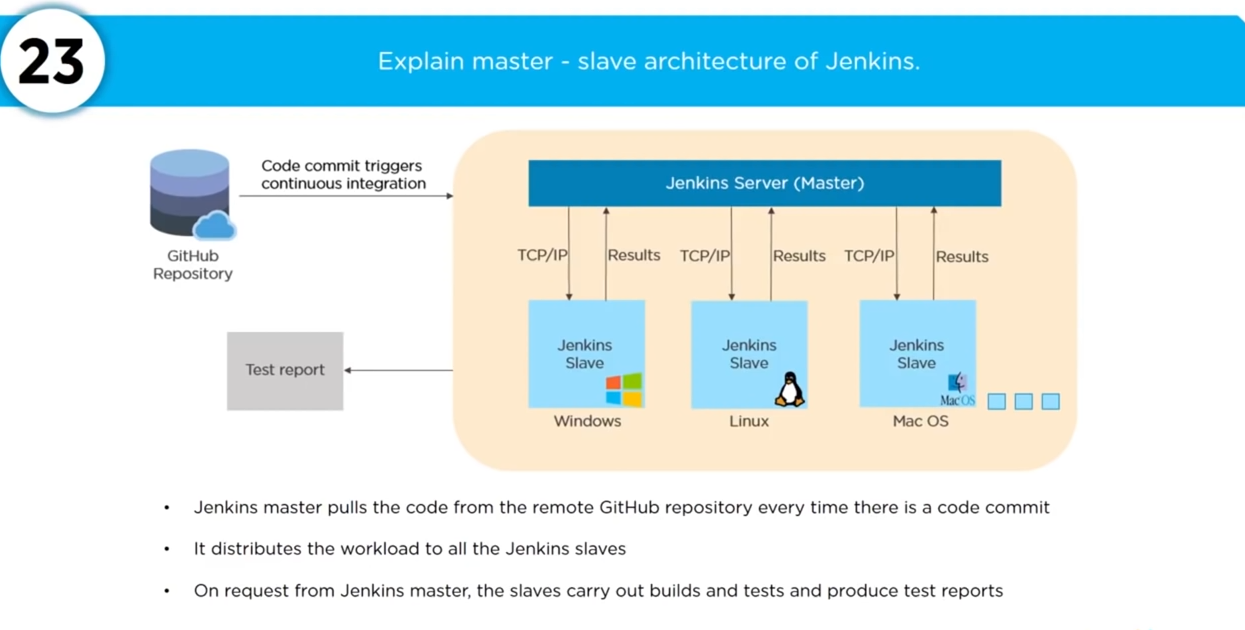












# Reference you need to watch

https://www.youtube.com/watch?v=FX322RVNGj4