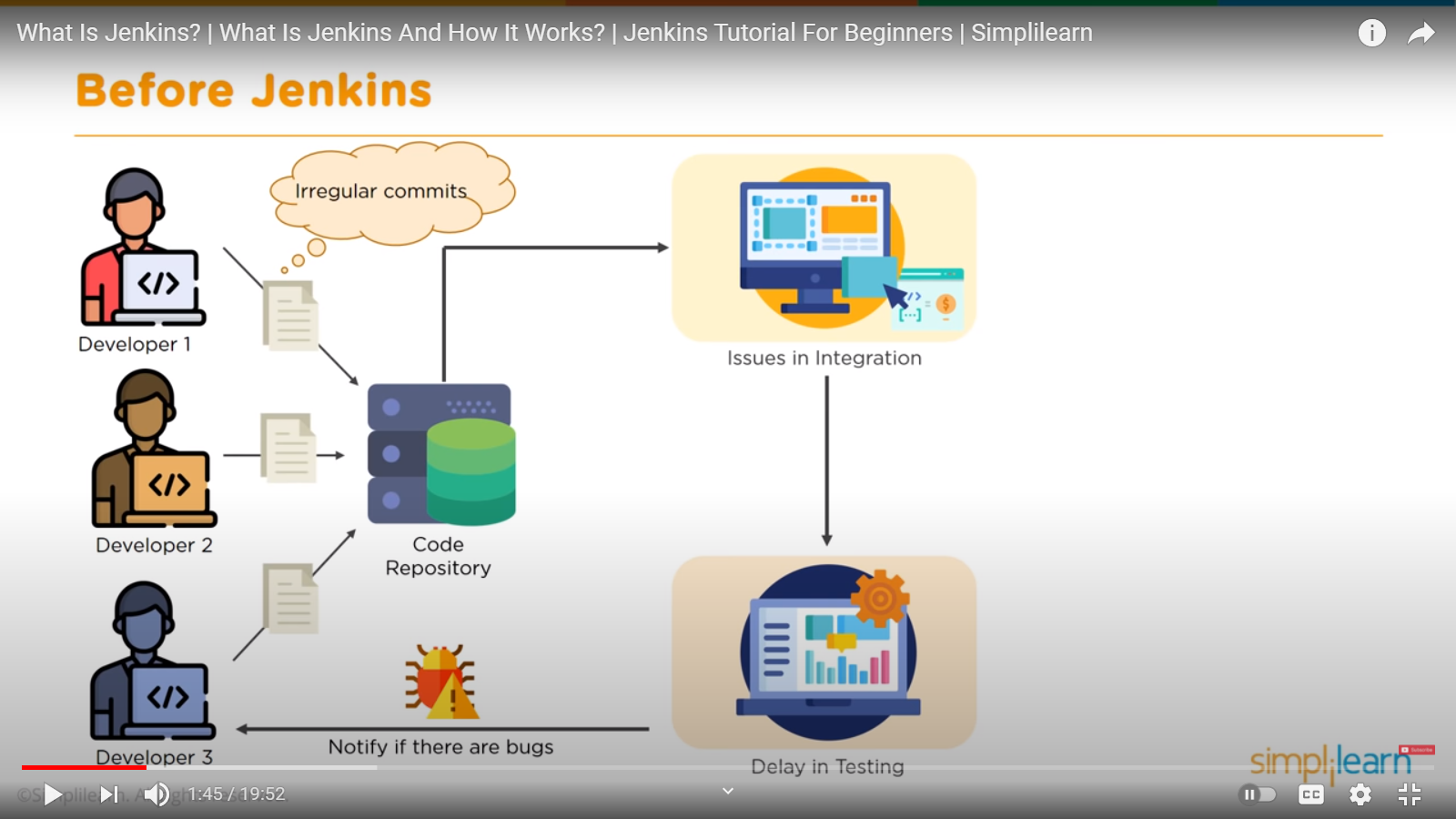
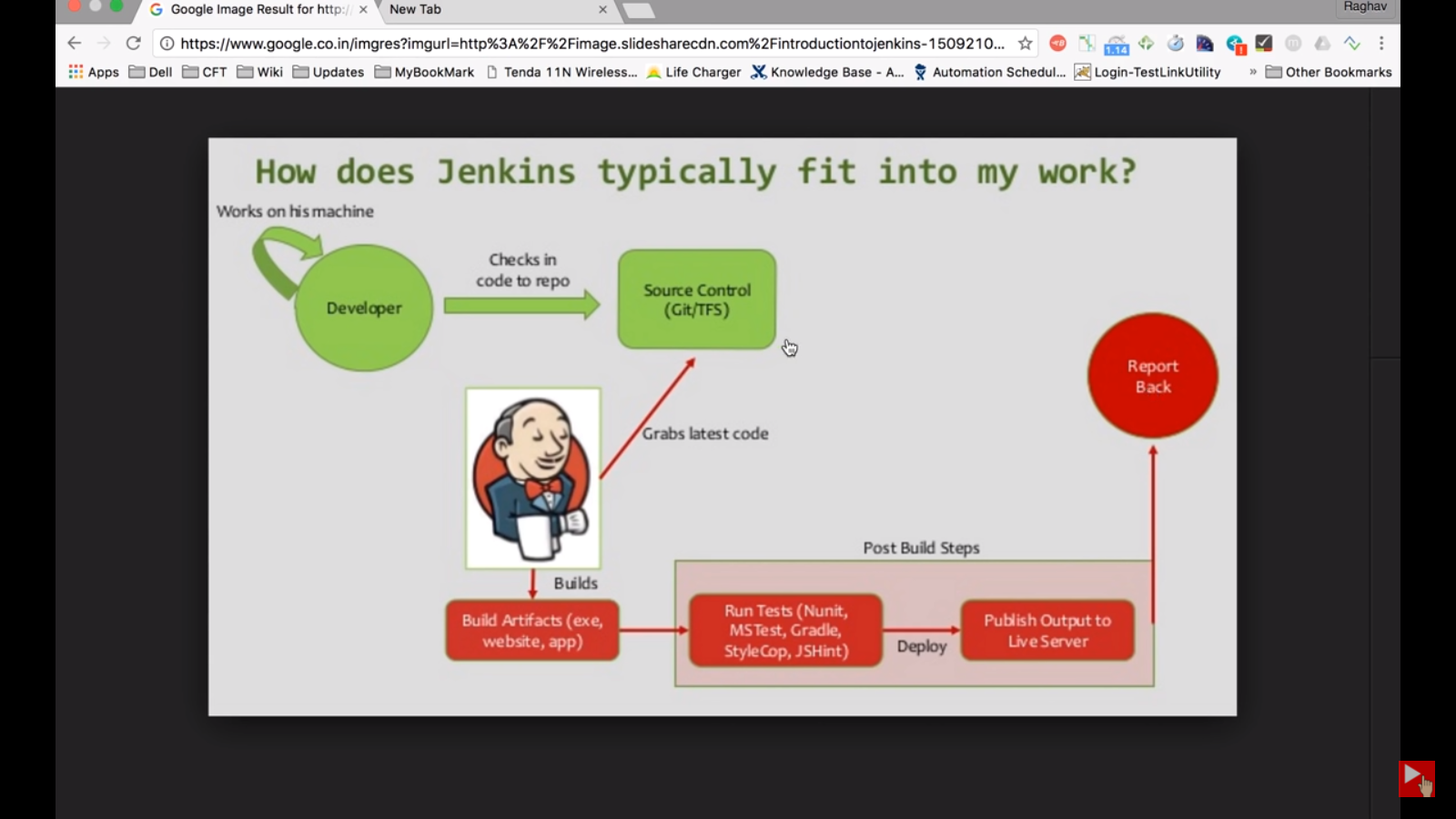
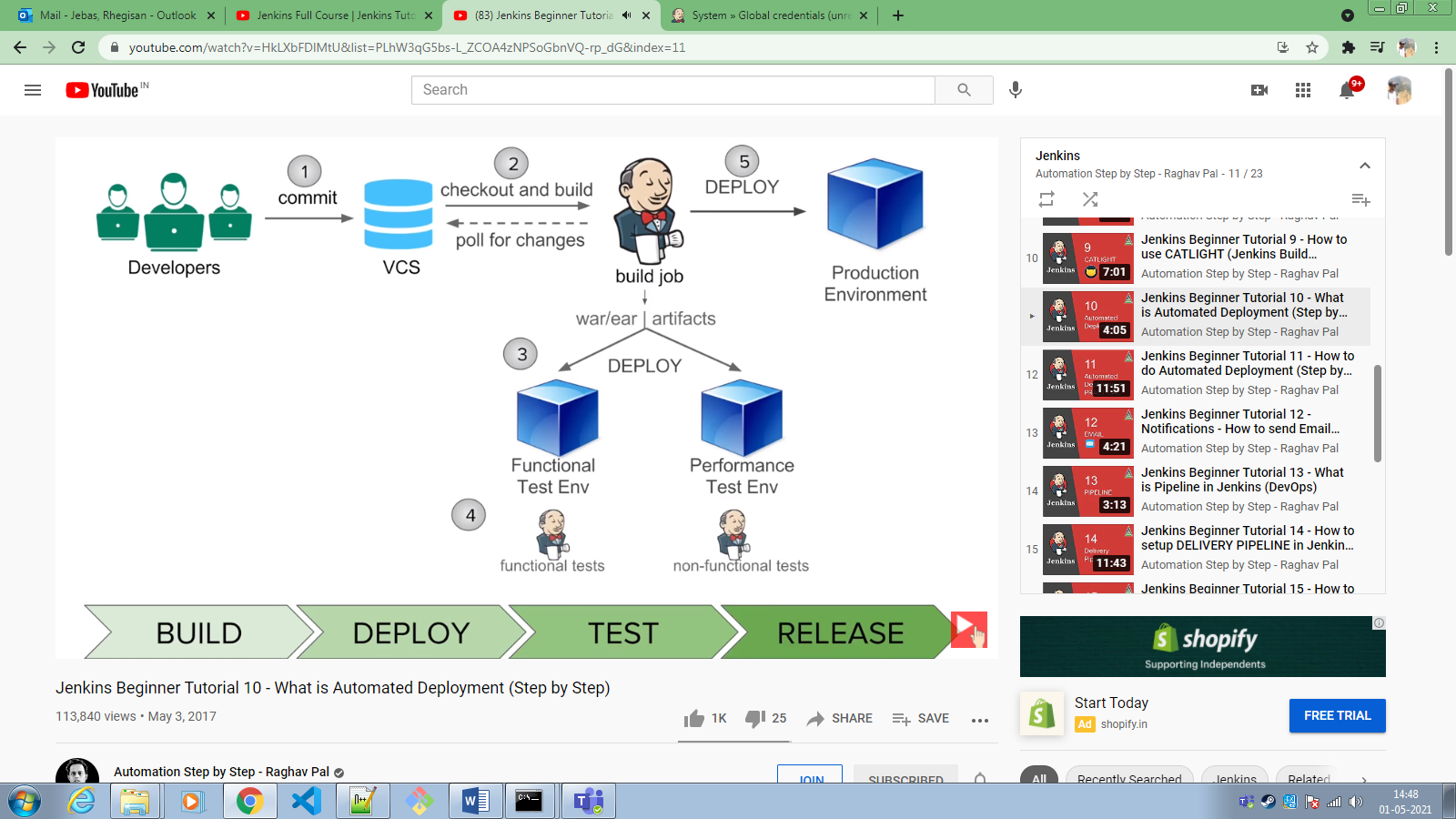
JENKINS NOTES

**BEFORE JENKINS:**

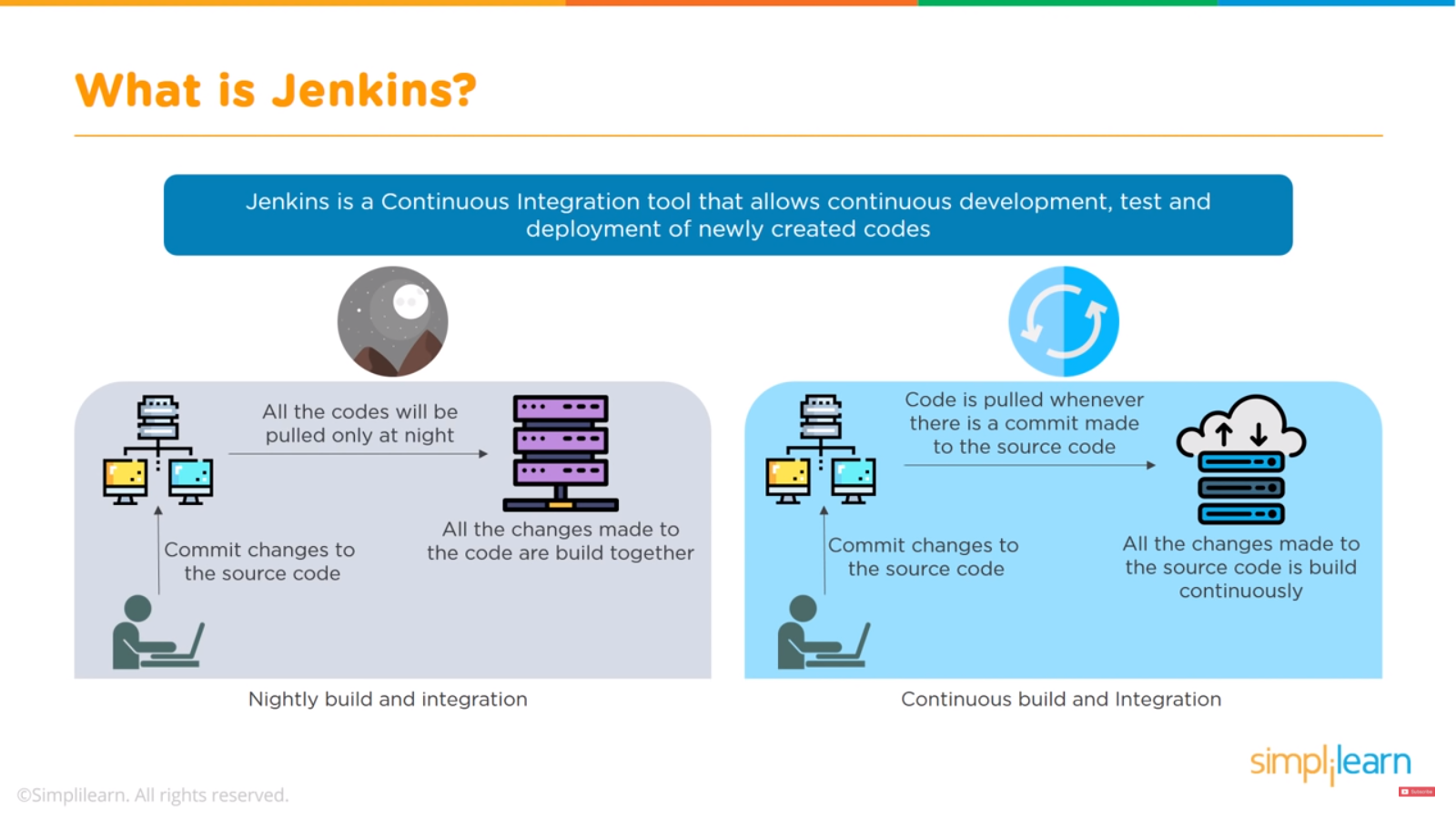


**JENKINS LIFECYCLE:**

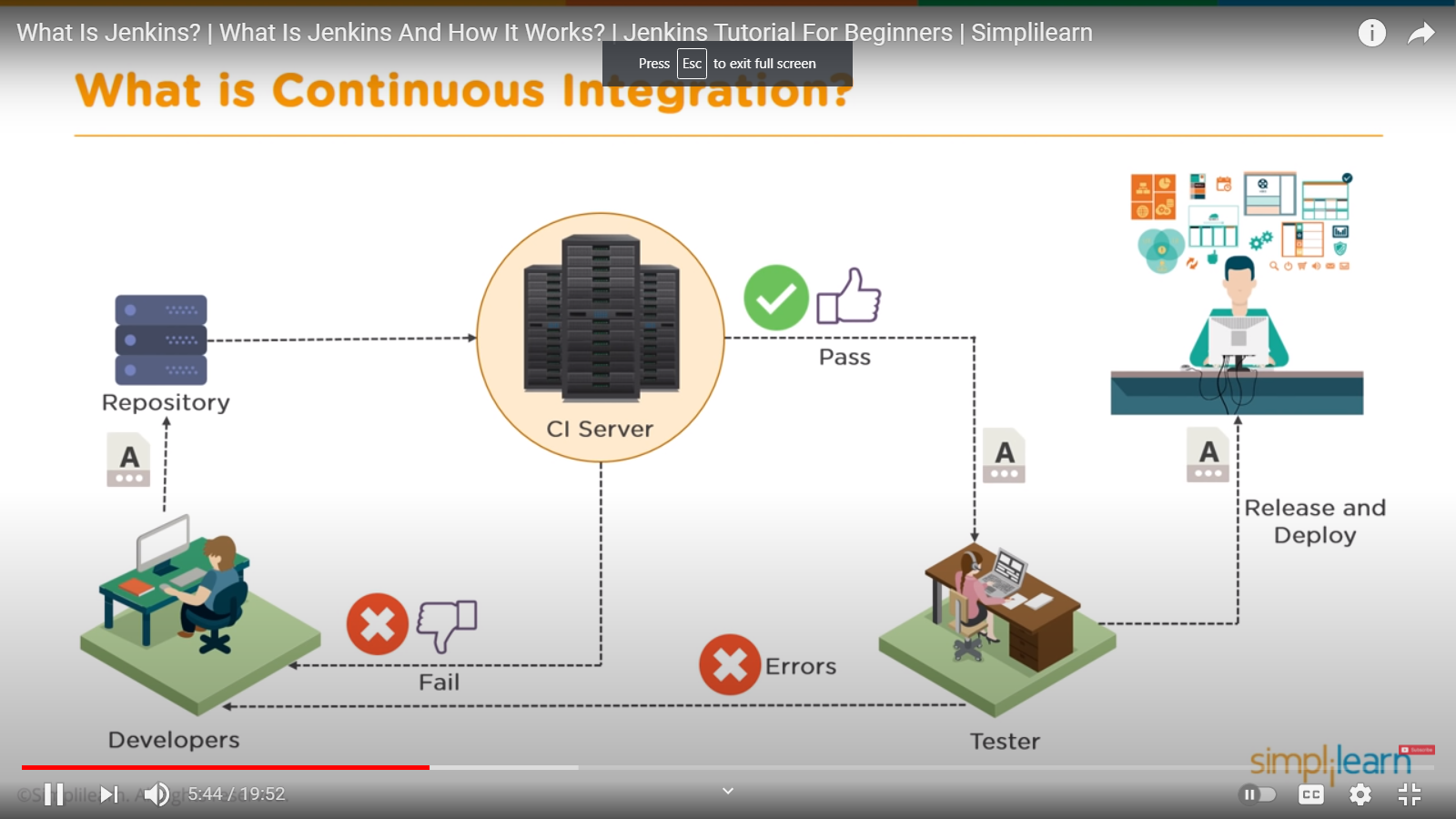




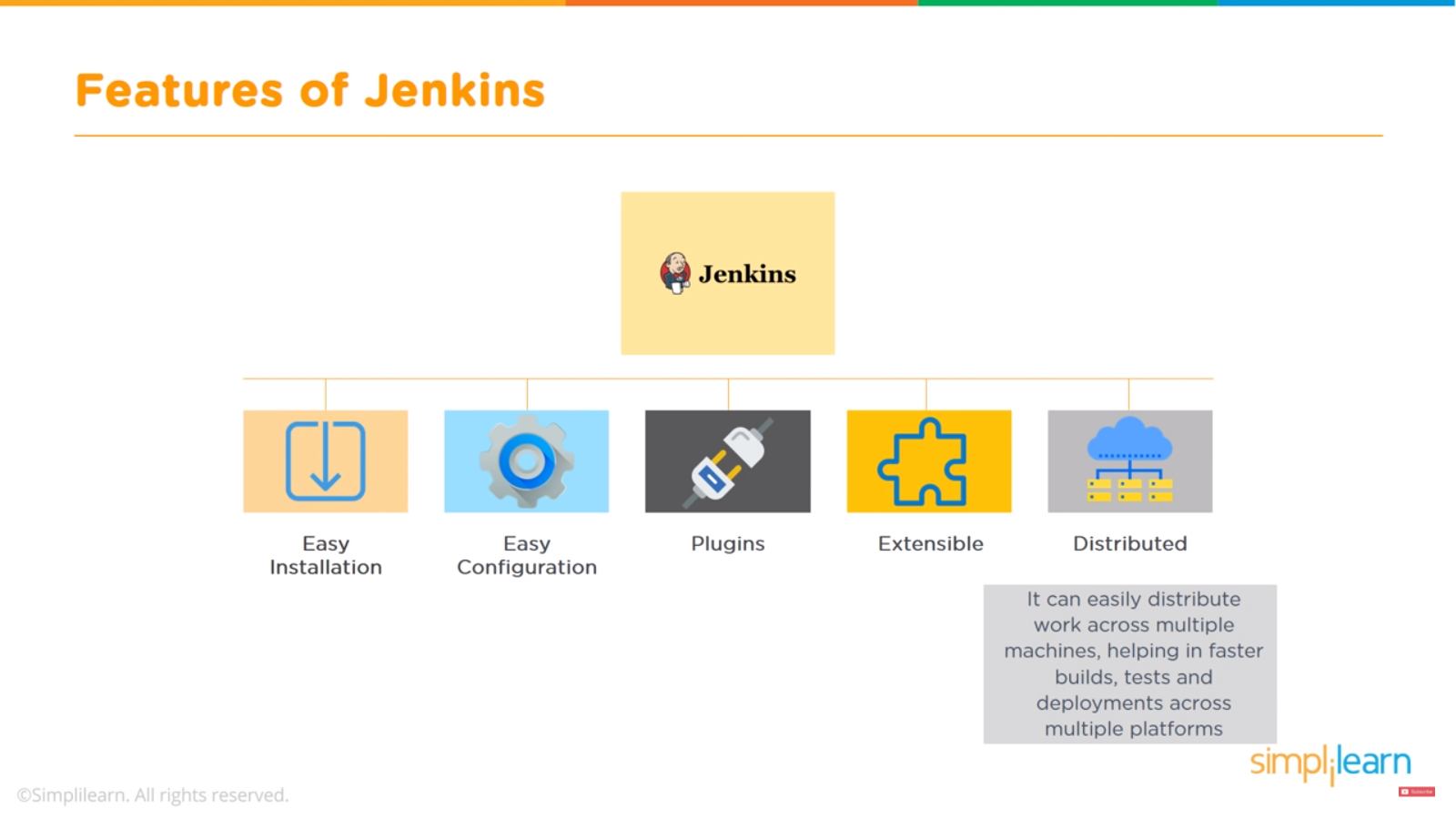
**NIGHTLY BUILD AND CONTINUOS BUILD:**



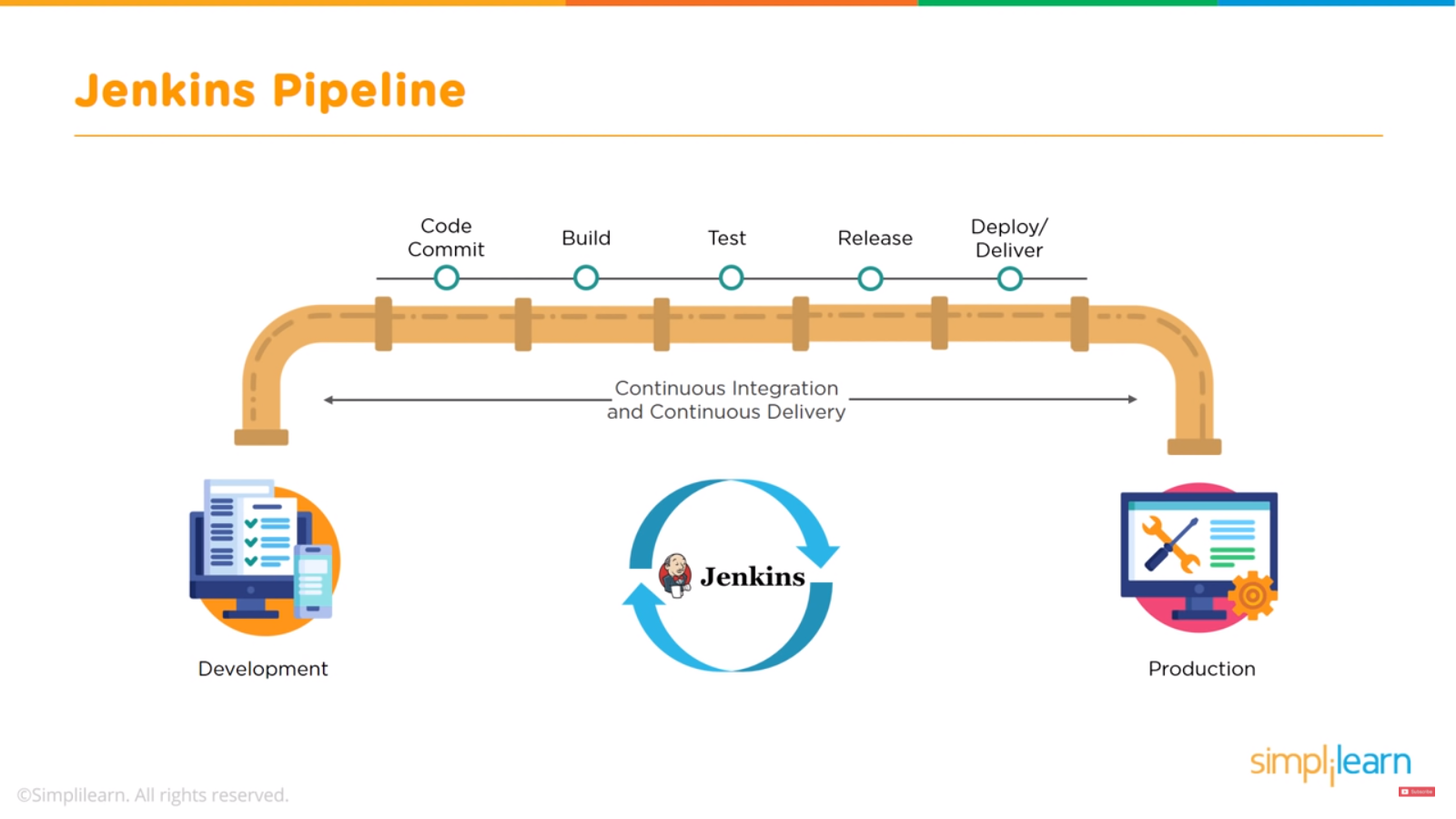
**CONTINUOS BUILD:**



**JENKINS FEATURES:**

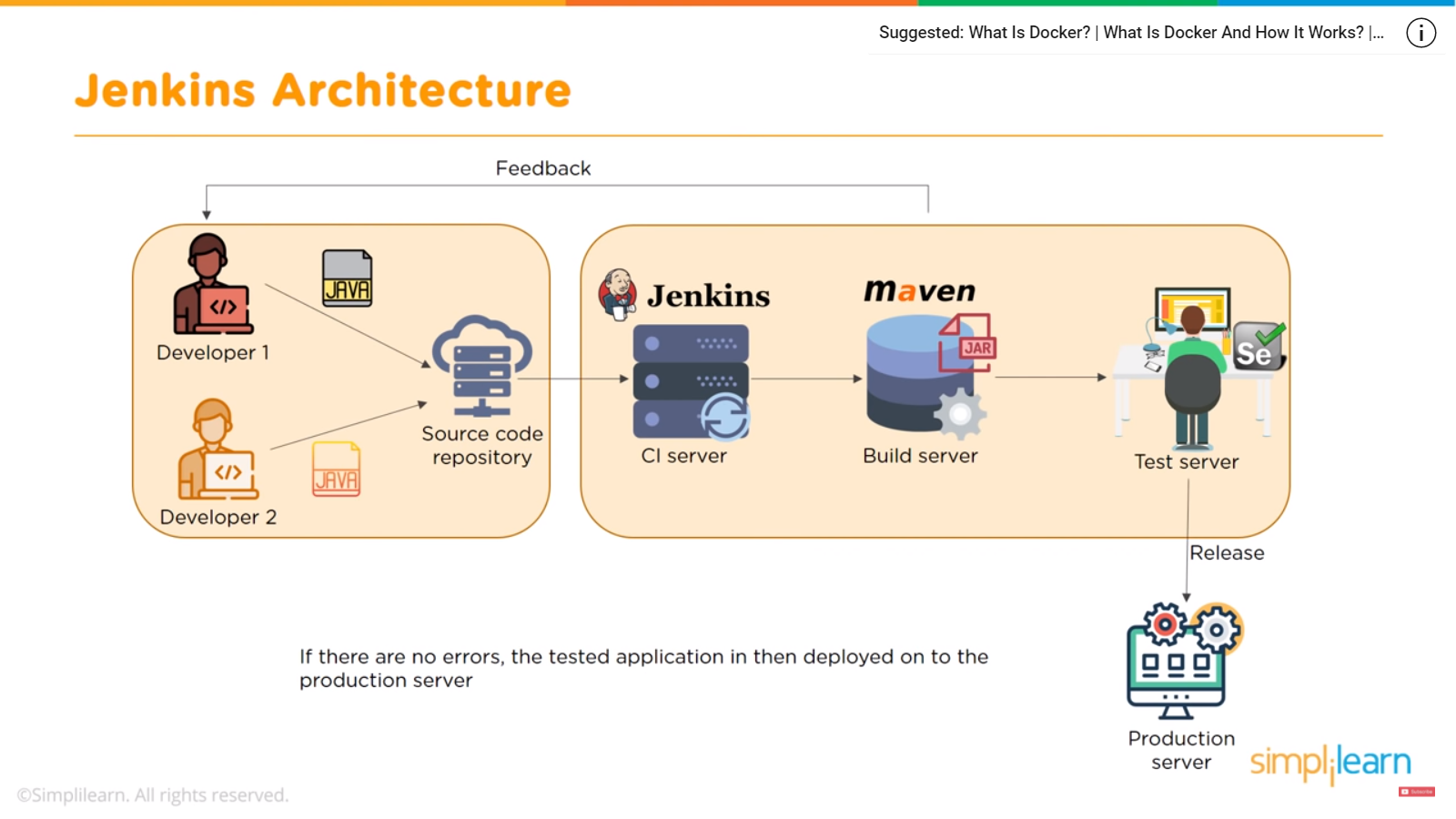


**JENKINS PIPELINE:**

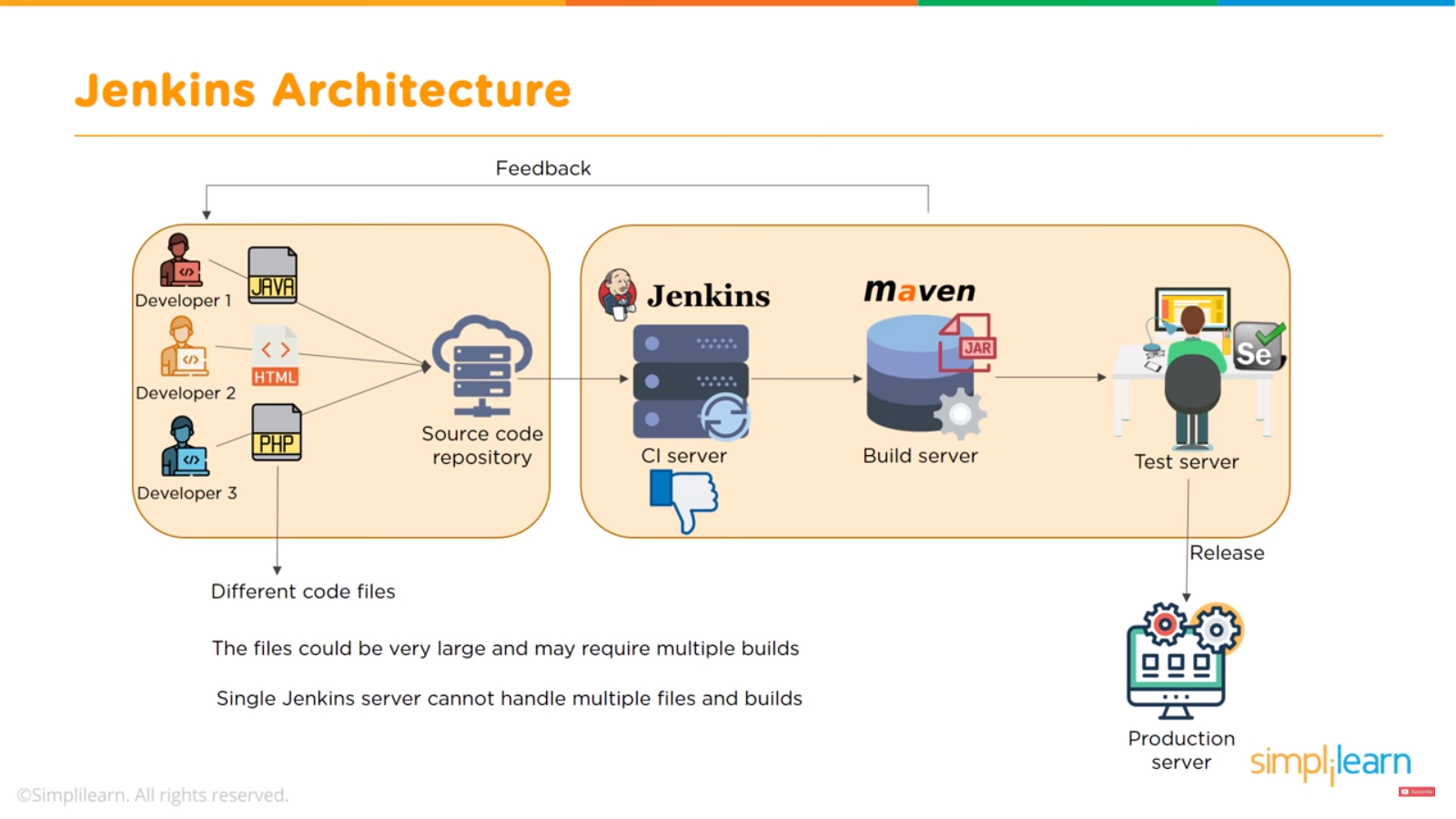


**JENKINS ARCHITECTURE:**

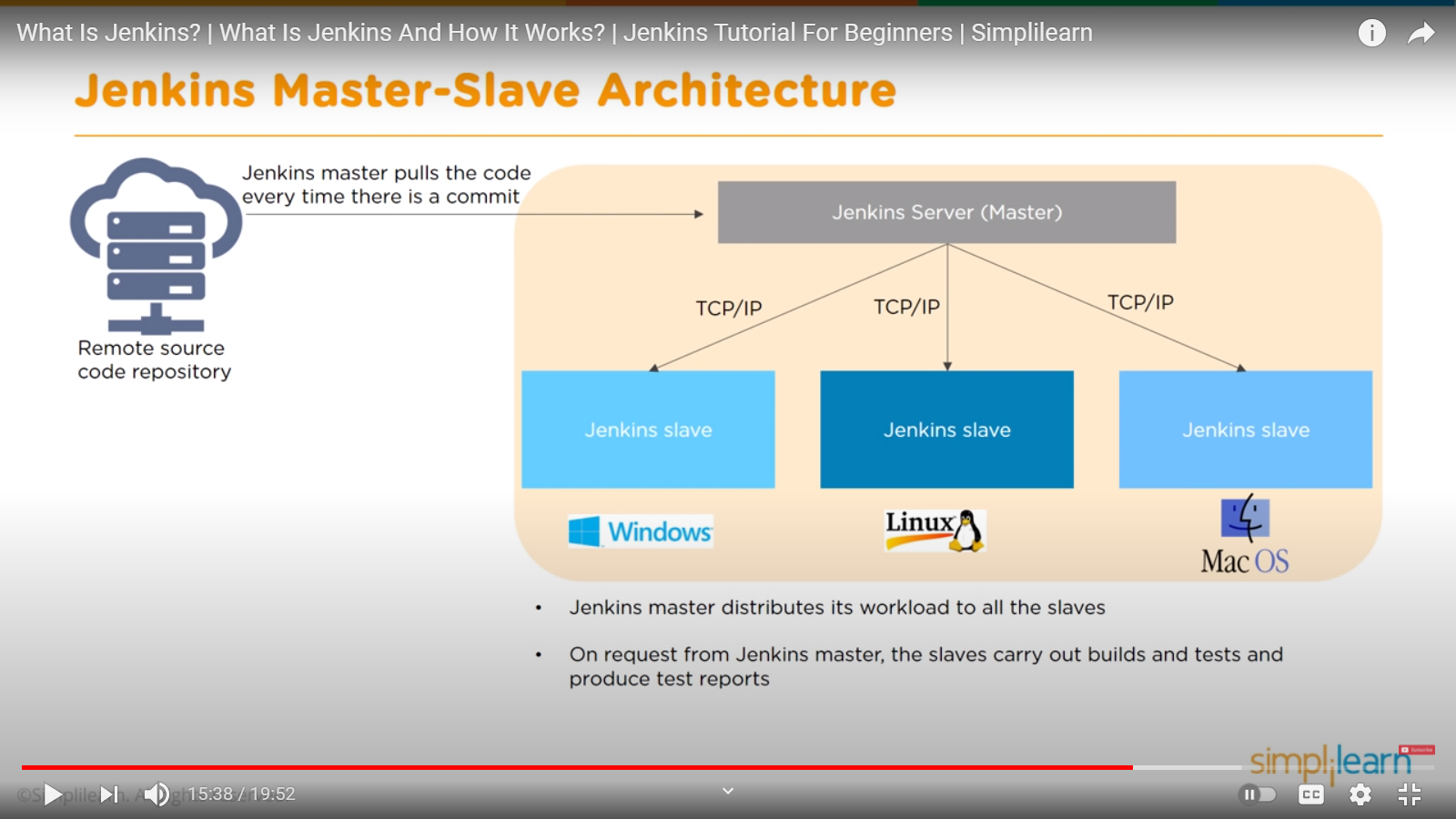
Same code files -



Different code files -



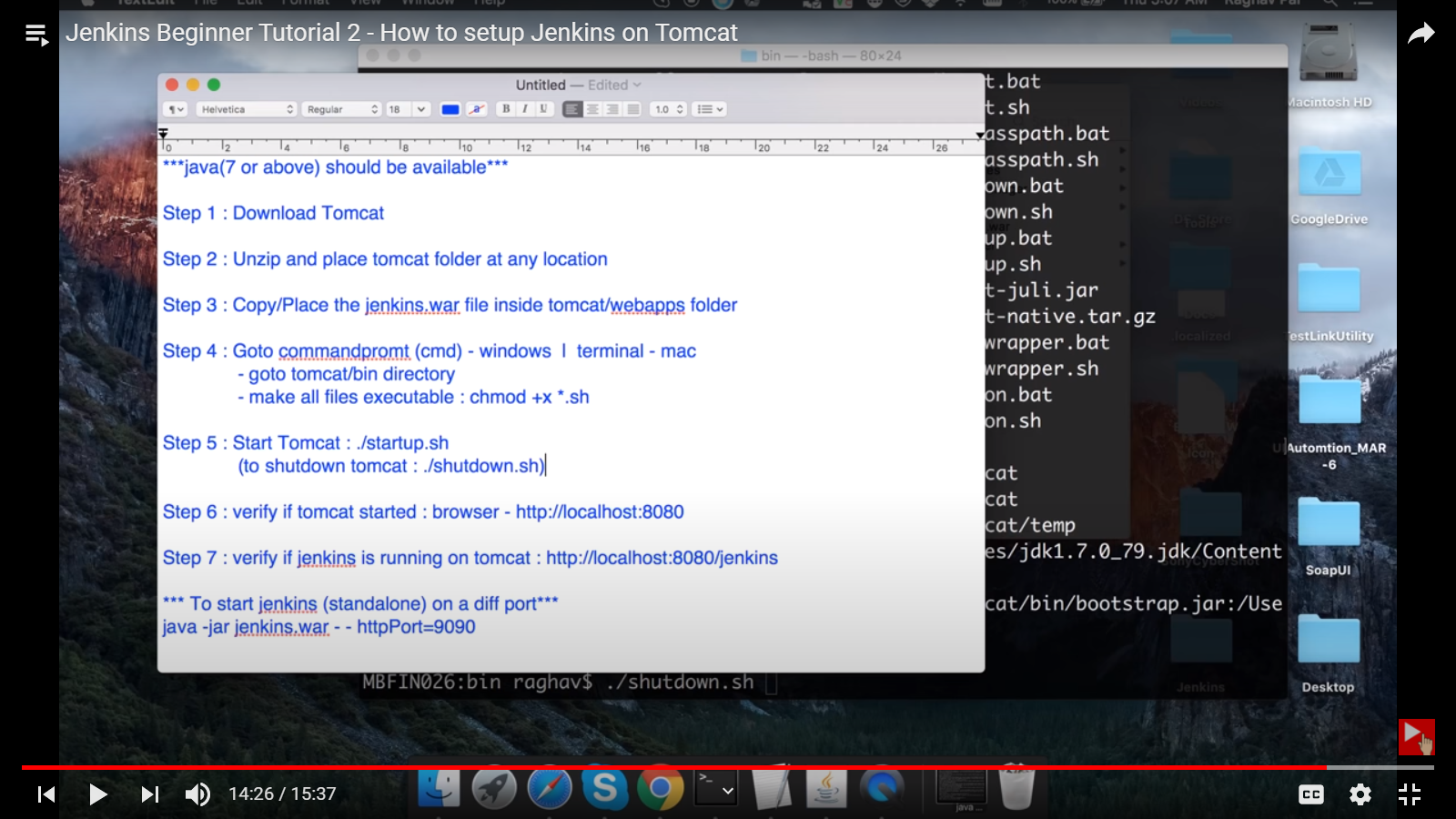
**JENKINS MASTER-SLAVE ARCHITECTURE:**



**RUNNING JENKINS:**

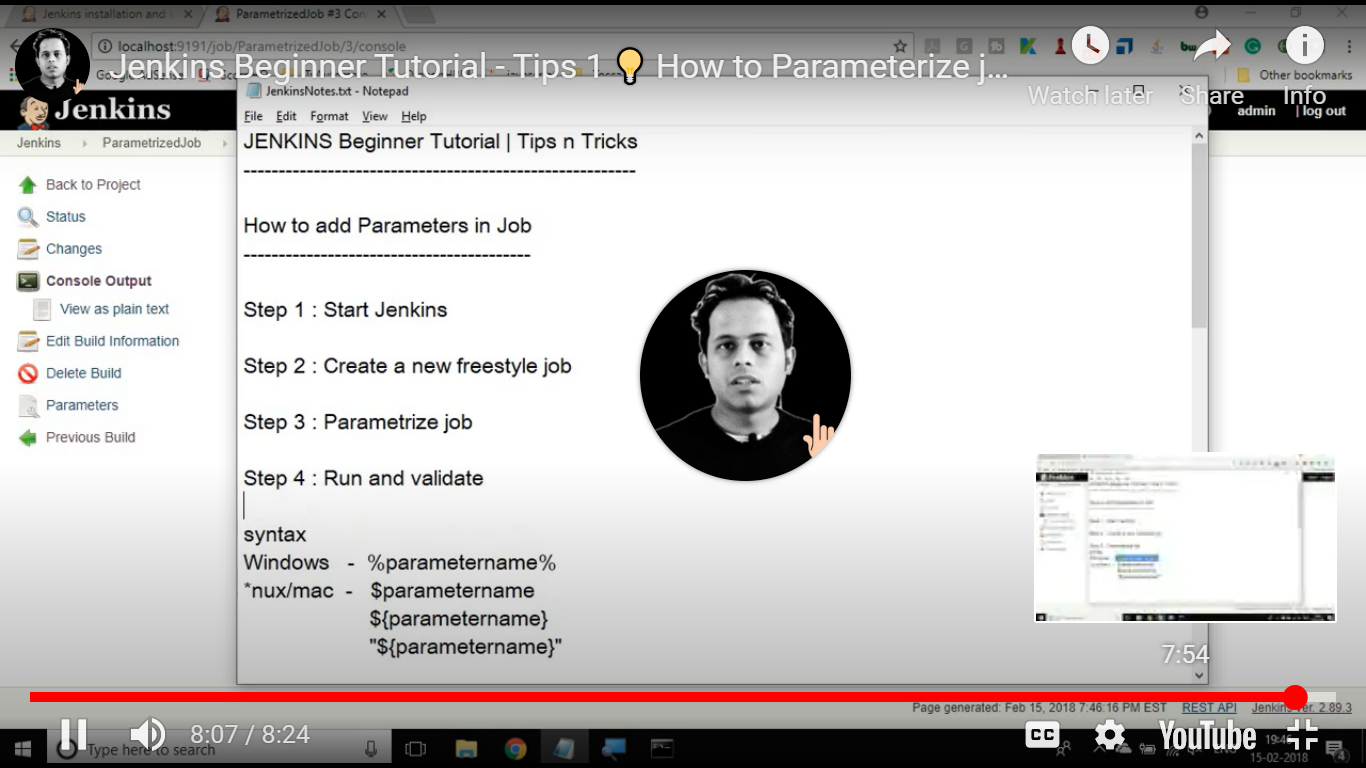
* Go to jenkins.war file directory through Command prompt 🡪 enter java -jar Jenkins.war 🡪 got to browser enter localhost:8080 and by default Jenkins will be use port 8080
* To launch it on another port enter localhost:8080 --httpPort=<your-port-no>

**JENKINS DEPLOYED ON TOMCAT:**



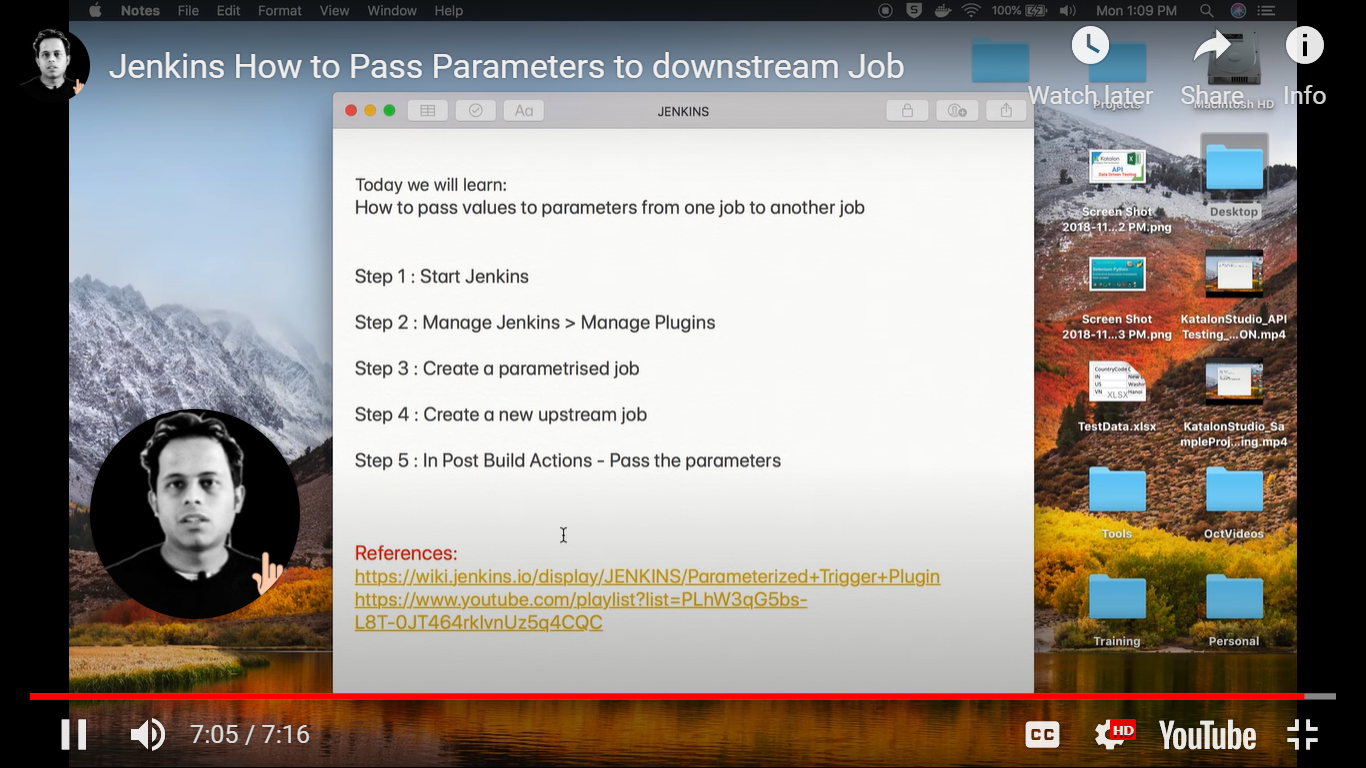
**PARAMETERIZE JOB:**

* **-p** flag on command line to enter parameter
* Extended choice parameter plugin



**PARAMETERIZE TRIGGER JOB (DOWNSTREAM):**

* Parameterized Trigger plugin



**DIFFERNCE BETWEEN FREESTYLE PROJECT AND PIPELINE PROJECT:**

* The main aim of the freestyle project is implementing, developing, or running simple jobs such as enabling you to specify the version control system from where you need to extract code and build it and call tests. Freestyle projects are for orchestration simple jobs for a project.
* Pipeline Project is better either to set up a CD pipeline or to define the deployment pipeline as code. The pipeline project is suitable to build pipelines for complex jobs whereas the freestyle project is suitable for simple jobs.

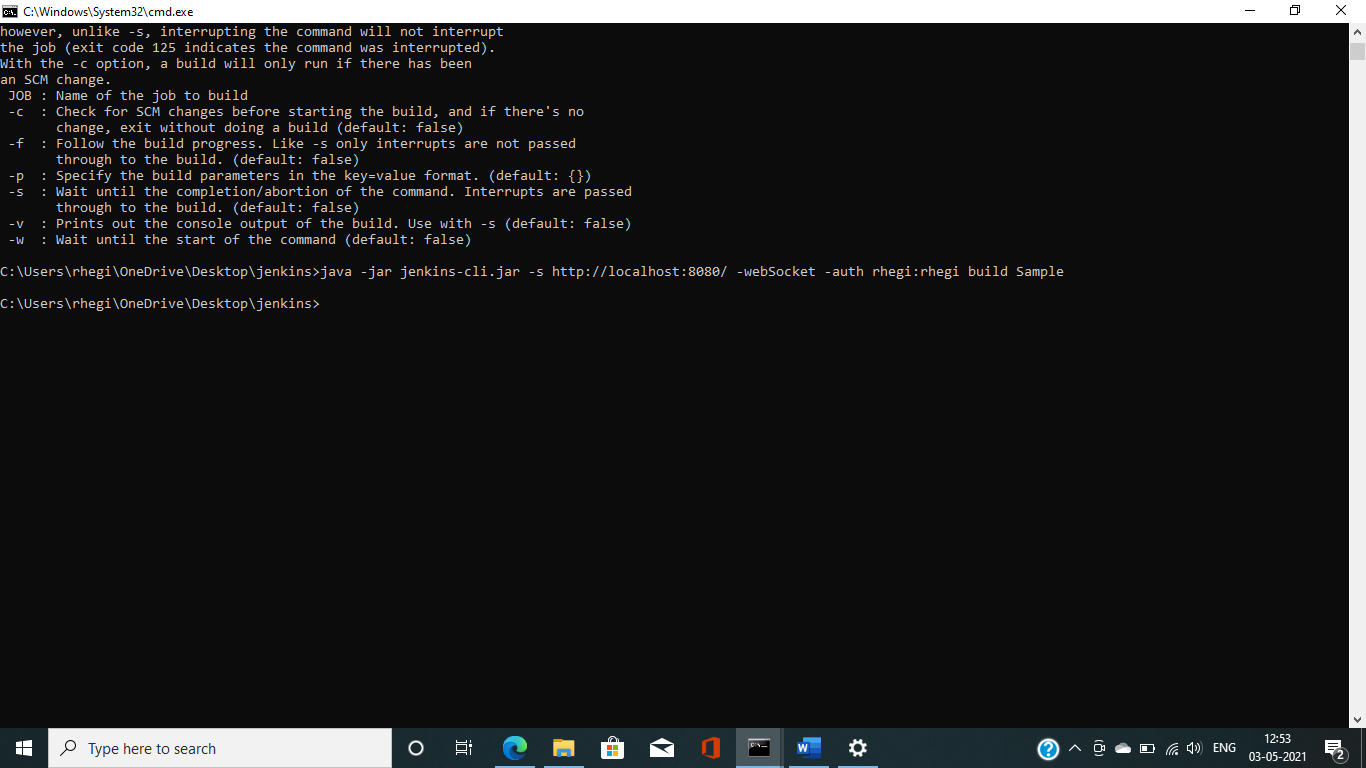
**WHY TO USE PIPELINE:**

If you hate typing things into CI systems and therefore want to use pipelines as code--where you put all of your CI configuration into a file in source control (Jenkinsfile) and let Jenkins read that file to figure out what to do--use [pipelines](https://jenkins.io/solutions/pipeline/). once you know pipelines, there won't be many cases where you'll prefer freestyle projects.

**JENKINS COMMAND LINE INTREFACE:**

* Go to localhost:8080/cli 🡪 download Jenkins-cli.jar file
* Go to Jenkins-cli.jar files location through terminal and start executing the commands

**BUILD A JOB THROUGH COMMANDLINE:**



**IF YOU ARE UNABLE TO RUN RESTART, THEN RUN JENKINS AS A SERVICE:**

* Go to Jenkins 🡪 manage Jenkins 🡪 install as window service 🡪 Install 🡪 click on Yes and allow it to restart 🡪 now run your restart command

**JENKINS DEPLOYEMENT – FREESTYLE PROJECT (project present inside GitHub):**

1. Create a new job 🡪 Freestyle project
2. From SCM select GIT 🡪 enter all the credentials
3. Select bash(for linux) and batch command(for windows)
4. Add build step 🡪 mvn compile
5. Add build step 🡪 mvn test
6. Add build step 🡪 mvn package
7. Appy and save
8. Build manually
9. Configure 🡪 add some post builds (create artifact , generate test reports etc…)
10. Apply save
11. Build manually ( now you can view artifact and test reports are generated)

**JENKINS PIPELINE:**

1. Create a new job 🡪 Pipeline project
2. Pipeline 🡪 script 🡪 hello world (sample syntax)
3. Apply and save
4. Build now 🡪 view result (all the stages are visible)
5. Configure 🡪 Add some more stages
6. Apply and save
7. Localhost:8080/pipeline-syntax (generates pipeline scripts) 🡪 click generate pipeline script.
8. Build now

**AUTOMATING JENKINS PIPELINE TO CREATE AN ARTIFACT AND PUBLISHING REPORTS:**

1. Create a new job 🡪 Pipeline project
2. Pipeline 🡪 script 🡪 hello world (sample syntax) 🡪 Enter this script (practice purpose)

pipeline {

agent any

stages {

stage('Git checkout') {

steps {

git ‘<Enter-your-repository-url>’

}

}

stage('Compile') {

steps {

sh ‘mvn compile’

}

}

stage('Test') {

steps {

sh ‘mvn test’

}

}

stage('Package') {

steps {

sh ‘mvn package’

}

post {

always {

junit ‘<reports-generated-directory’

}

Success {

archiveArtifacts artifacts: <.jar-file-directory> , followsmlinks: false

}

}

}

}

**DEPLOYING APPLICATION ON THE MACHINE:**

pipeline {

agent any

stages {

stage('Git checkout') {

steps {

git ‘<Enter-your-repository-url>’

}

}

stage('Compile') {

steps {

sh ‘mvn compile’

}

}

stage('Test') {

steps {

sh ‘mvn test’

}

}

stage('Package') {

steps {

sh ‘mvn package’

}

post {

always {

junit ‘<reports-generated-directory’>

}

Success {

archiveArtifacts artifacts: <.jar-file-directory> , followsmlinks: false

}

}

}

stage('Deploy') {

steps {

withEnv ([‘JENKINS\_NODE\_COOKIE=dontKillMe’]) {

sh ‘nohup java -jar Dserver.port=8080 <artifacts-name>

} }

}

}

}

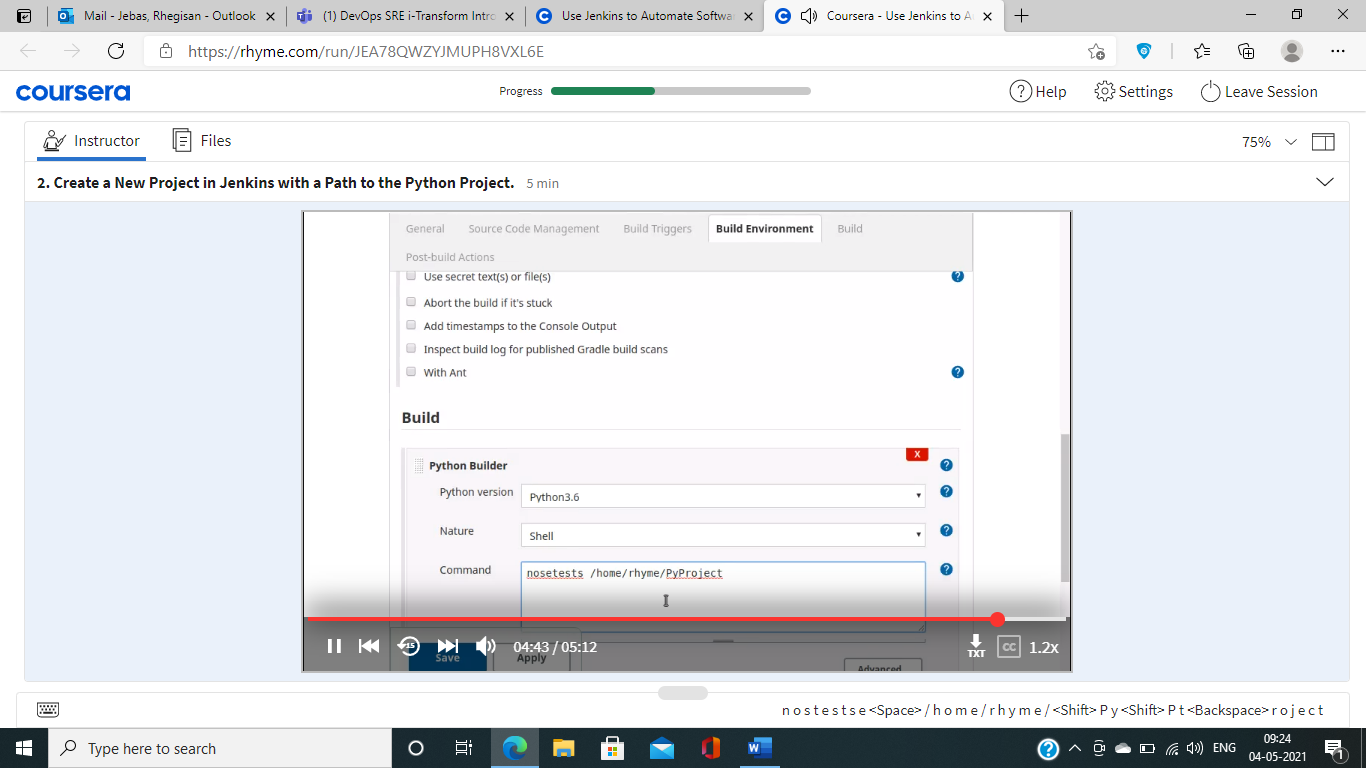
ps -aux | grep -I java (to check the info) and kill -9 <process-id> (to kill the process)

JENKINS\_FILE:

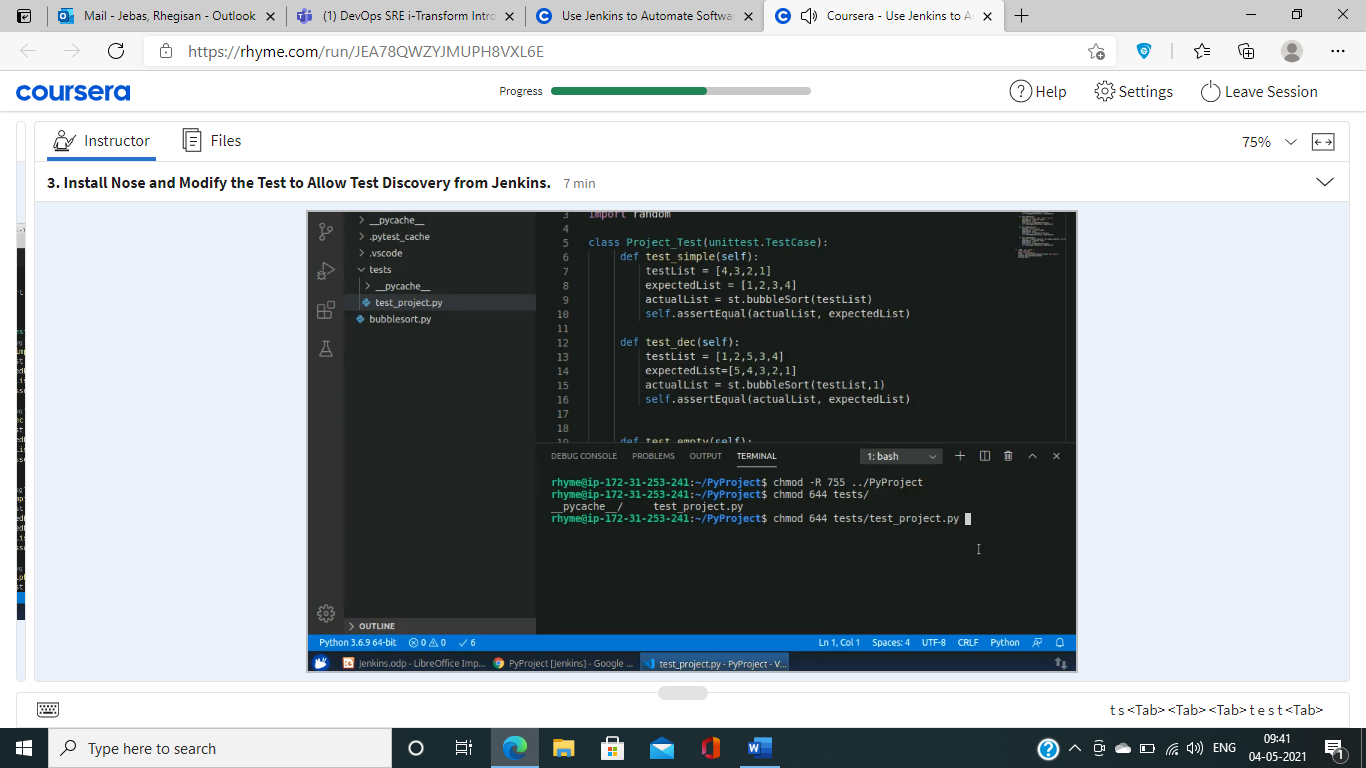
1. Copy paste the pipeline script into a Jenkinsfile
2. Push the Jenkinsfile to GitHub
3. Pipeline tab 🡪 change to pipeline with SCM 🡪 Select Git 🡪 enter details 🡪 enter script path 🡪 apply and save
4. Build now

**JENKINS WORKING WITH PYTHON PROJECT:**

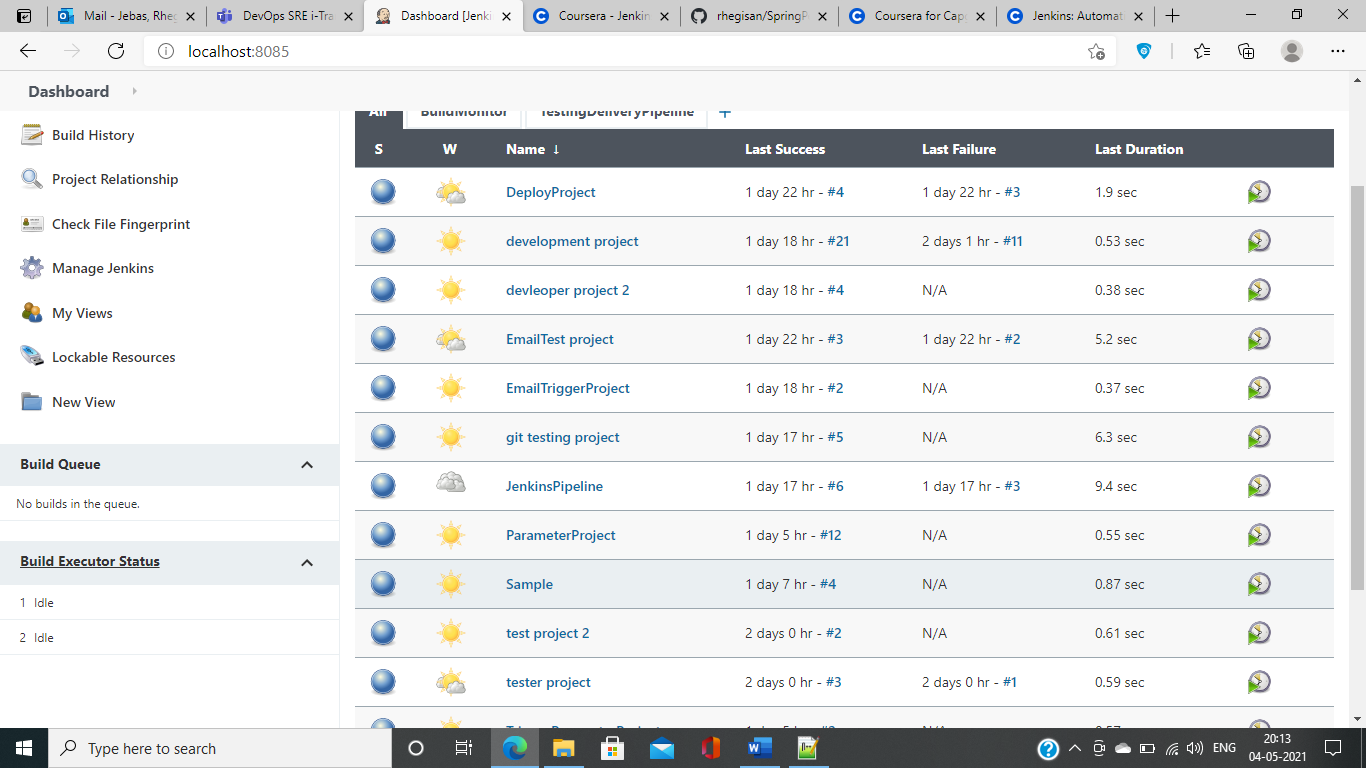
1. Manage plugins 🡪 install ShiningPanda plugin
2. Global Tool configuration 🡪 Add python installation setup
3. Create a new job 🡪 build 🡪 apply and save



1. Set permissions 🡪For files ‘rwx for owner’ and ‘rx for users’ 🡪 for tests ‘rw for owner’ and ‘r for users’ \*Don’t give x to tests\*



1. Check python version 🡪 install nose ‘sudo pip3 install nose’ 🡪 run nosetests
2. Go to Jenkins 🡪 Build project
3. Initialize the local directory with git init 🡪 add this SCM Directory in the Jenkins
4. Enable git polling 🡪 do changes and commit 🡪 see the output



**USER CREATION:**

1. Create dev1 dev2 test1 test2
2. Create users
3. Check Role based strategy 🡪 Assign and manage roles 🡪 give all read rights

**HOW TO TRIGGER BUILDS:**

1. Post build triggers DEV1 triggers Test1 (downstream)
2. Build based on previous job Test2 is build after DEV2 (upstream)
3. Trigger using scm

**AUTOMATED DEPLOYMENT:**

1. New job DeployProject 🡪 post build 🡪 deploy war ear to container
2. Select sample.war from .jenkins/workspace/DeployProject
3. Create a new user from apache/conf/users.xml 🡪 start tomcat on port 8080
4. Enter url as localhost:8080 🡪 apply and save
5. Now go to localhost:8080/sample.war 🡪 application is deployed

**HOW TO SEND AN EMAIL AFTER A JOB IS BUILD:**

1. Manage Jenkins 🡪 Global configuration settings
2. smtp.gmail.com 🡪 Use SMTP Authentication 🡪 senders email id and password
3. check use SSL 🡪 SMTP port will be 465 587 0r 25 🡪 Test email 🡪 incase not getting mail go to manage signin and allow less secure third party apps 🡪 apply and save
4. Go to job 🡪 post build action 🡪 Email notification 🡪 recipients email id 🡪 apply and save
5. Check your email
6. Plugins 🡪 Extreme Notification Plugin 🡪 Email Text plugin

**SETUP DELIVERY PIPELINE:**

1. Create 2 or 3 chained jobs
2. Install TestingDeliveryPipeline plugin 🡪 configure initial job 🡪 apply and save
3. View the result from dashboard

**SETUP BUILD PIPELINE:**

1. Create 2 or 3 chained jobs
2. Install BuildPipeline plugin 🡪 configure initial job 🡪 apply and save
3. View the result from dashboard

**PARAMETERIZED JOB:**

1. Create a freestyle project 🡪 This project is parameterized 🡪 Set parameter name and value
2. Build 🡪 for eg. echo %Name% 🡪 apply and save
3. Build with parameters

**PIPELINE PROJECT:**

1. Install Pipeline plugin
2. Create new job 🡪Pipeline project
3. Pipeline section 🡪 pipeline script 🡪 hello world 🡪 Apply and save
4. Build 🡪 View the Stage view

**HOW TO GET Jenkinsfile FROM GIT SCM:**

1. Create new job 🡪Pipeline project
2. Add Jenkinsfile to a GitHub repository
3. Pipeline section 🡪 pipeline script from SCM 🡪 GIT (add git plugin if its not available) 🡪 add url and credentials 🡪 select branch 🡪 Script path Jenkinsfile 🡪 apply and save
4. Build now 🡪 View the stage view

**HOW TO USE COMMAND LINE INTERFACE:**

1. How to build a job (with allow anyone and authenticated user)
2. How to build a parameterized job
3. How to delete a job