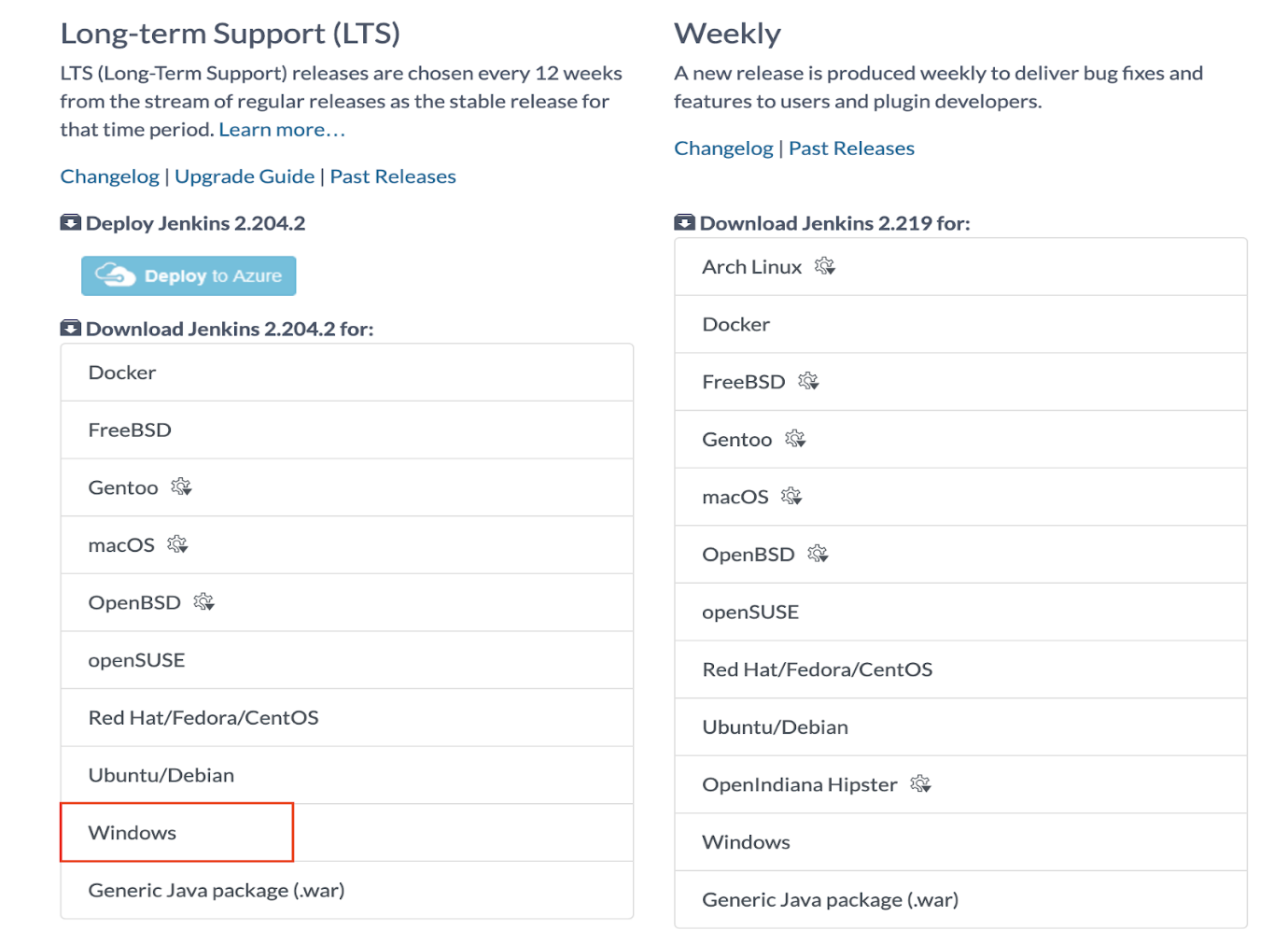
# **Food Delivery Application on DevOps**

# **1. JENKINS**

## 1.1 Jenkins Installation on Windows

**Step 1)** Go to <https://www.jenkins.io/download/> and download the Windows version.

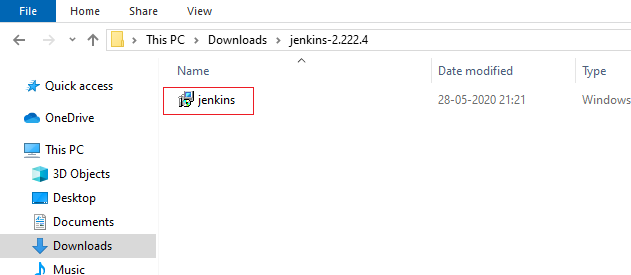


**Step 2)** Extract the ZIP archive

The Jenkins installer comes packed in a ZIP file. You need to extract the ZIP file before you can run the installer.

**Step 3)** Run the installer

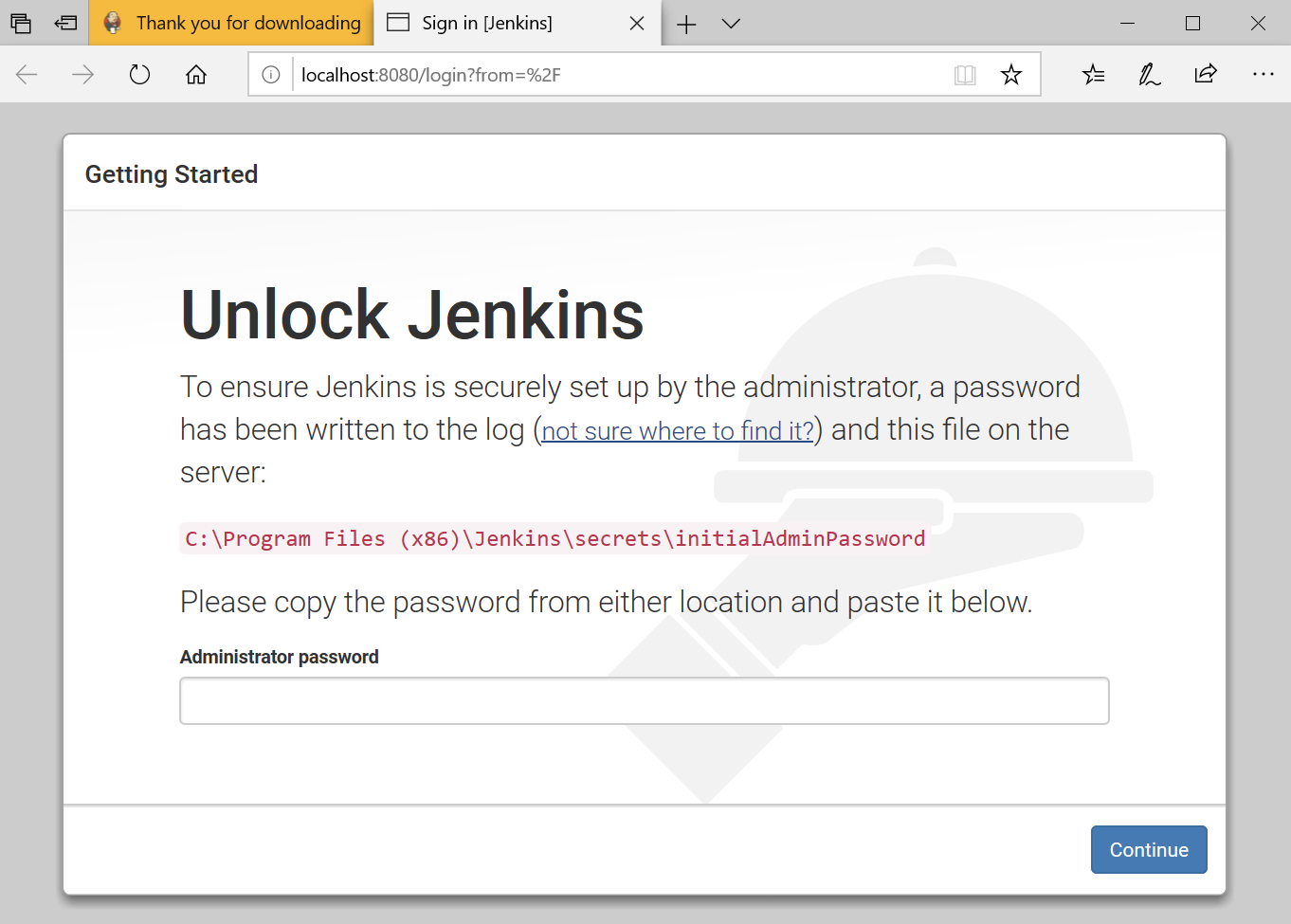
Double-click the installer to start the installation wizard.



The installation process is straightforward and you can just use the default settings.

**Step 4)** Get the installation password

As soon as the installation wizard is complete, it will open a new browser page pointing you to this URL: <http://localhost:8080/>

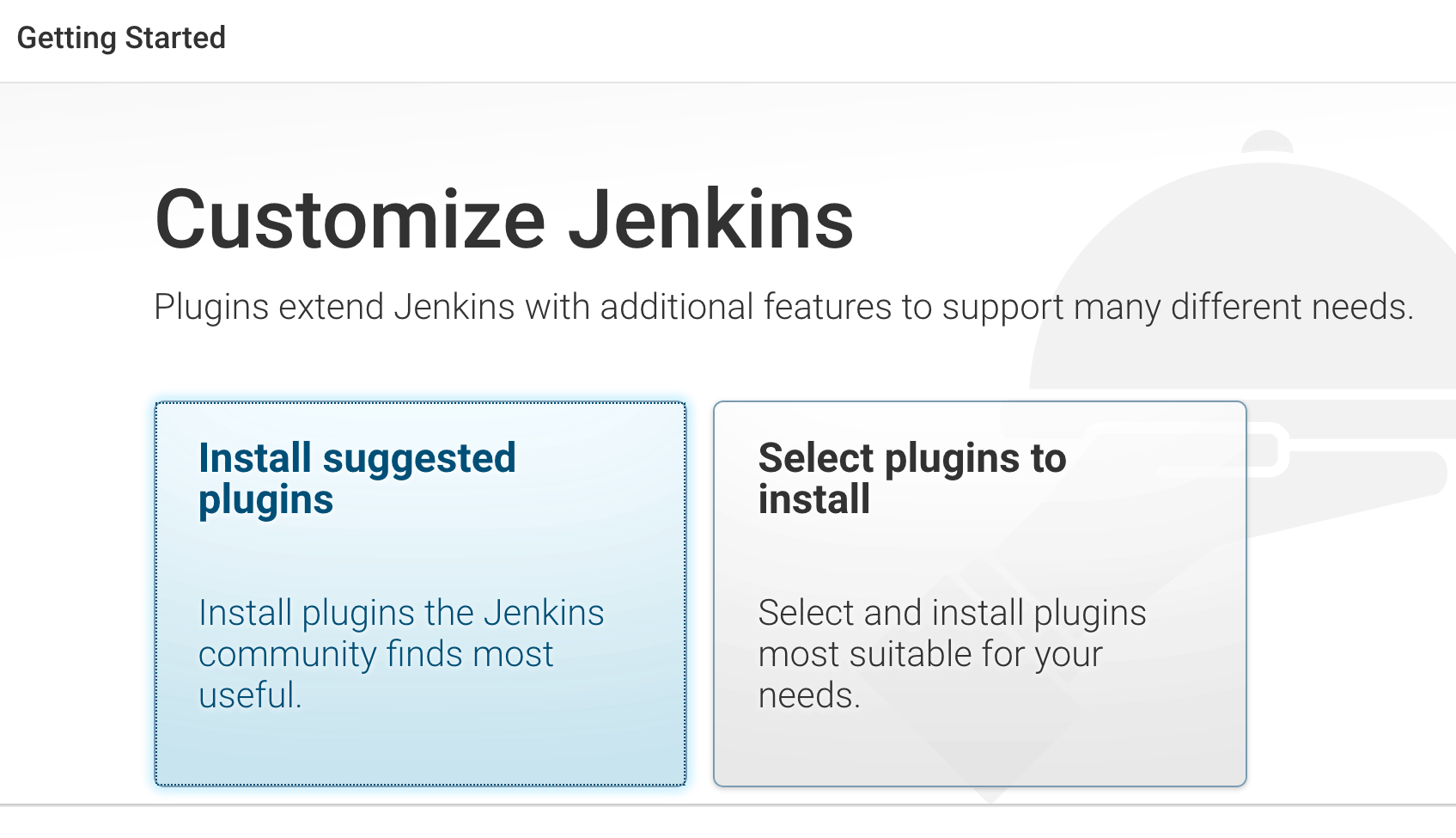


The initial Administrator password should be found under the Jenkins installation path. A file called **initial Admin Password** can be found under C:\Program Files (x86)\Jenkins\secrets. Open that file with Notepad and copy the password.

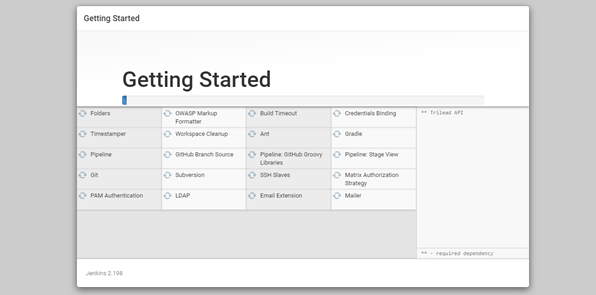
## 1.2 Customize Jenkins

You can also customize your Jenkins environment by below-given steps:

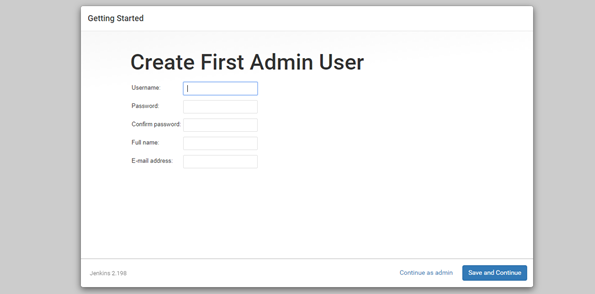
**Step 1)** Click on the "Install suggested plugins button" so Jenkins will retrieve and install the essential plugins.



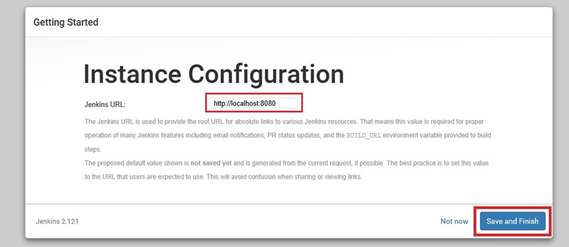
Jenkins will start to download and install all the necessary plugins needed to create new Jenkins Jobs.



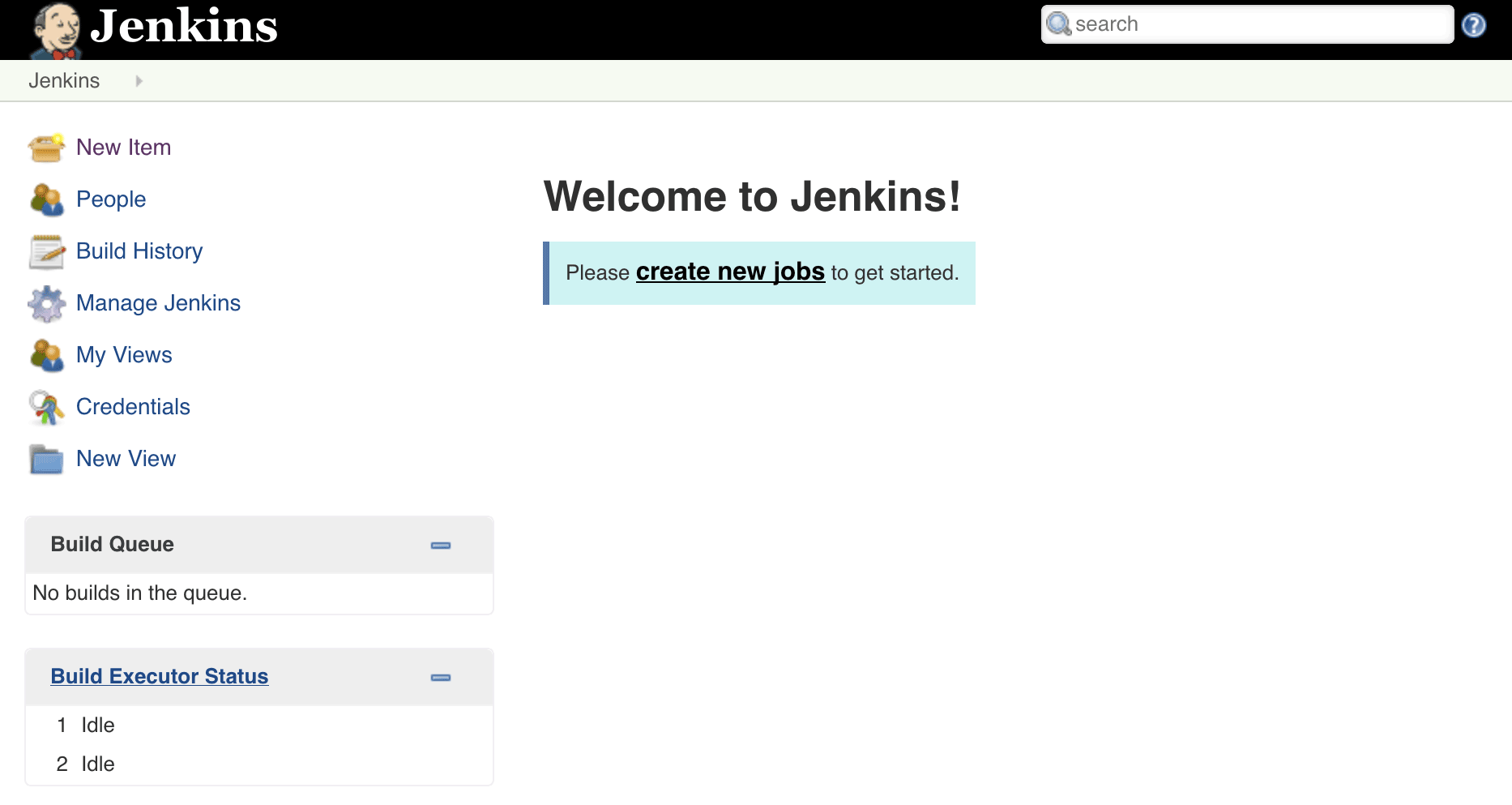
**Step 2)** After all suggested plugins were installed, the "Create First Admin User" panel will show up. Fill all the fields with desired account details and hit the "**Save and Finish**" button.



**Step 3)** Once you have filled the above data, finally it will ask for URL information where you can configure the default instance path for Jenkins. Leave it as it is to avoid any confusions later. However, if another application is already using 8080 port, you can use another port for Jenkins and finally save the settings, and you are done with installation of Jenkins. Hit the "**Save and Continue**" button:



We have successfully installed a new Jenkins Server. Below you can find the Jenkins instance up and run, ready to create first Jenkins Jobs:



## 1.3 Create Maven Project in Jenkins

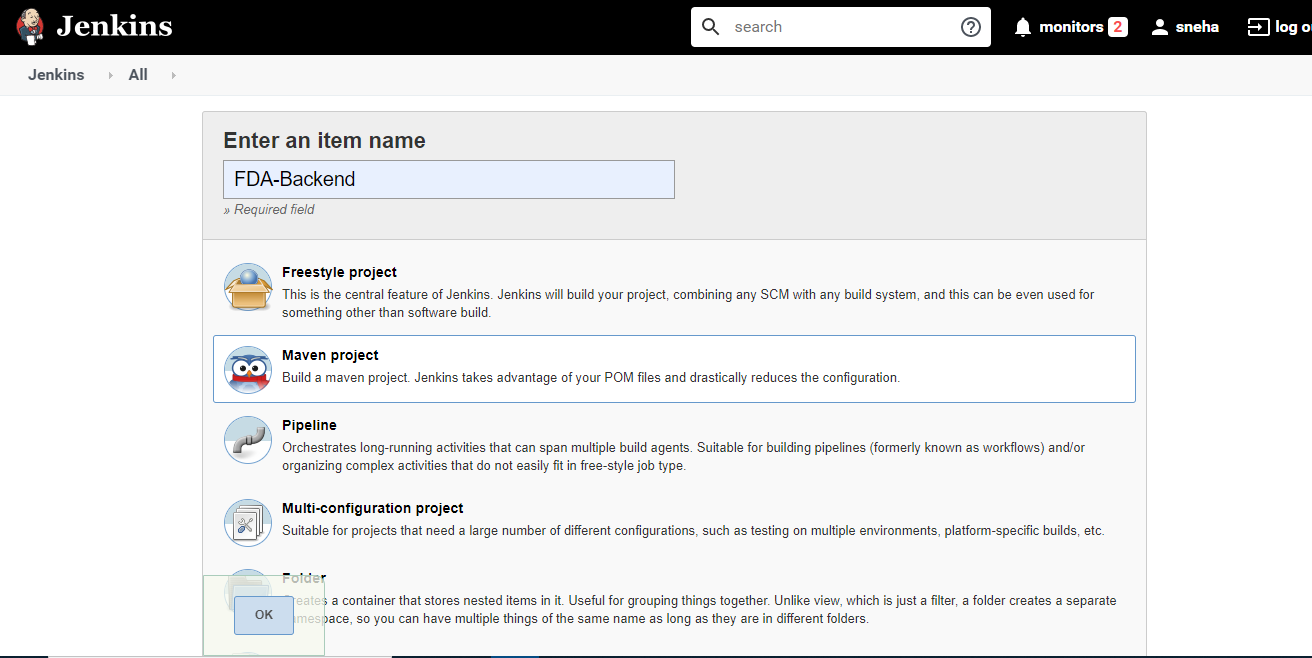
**Step 1)** Install Maven plugin in Jenkins

* Go to Jenkins Dashboard -> Manage Jenkins -> Manage Plugins -> Available -> Maven Integration -> Install.
* Go to Manage Jenkins -> Global Tool Configuration -> Maven -> Add Maven\_home variable value (i.e. path of the maven file on your system).
* Go to Manage Jenkins -> Global Tool Configuration -> JDK -> Add Java\_home variable value (i.e. path of the jdk file on your system).

**Step 2)** Create and Build a Maven Project

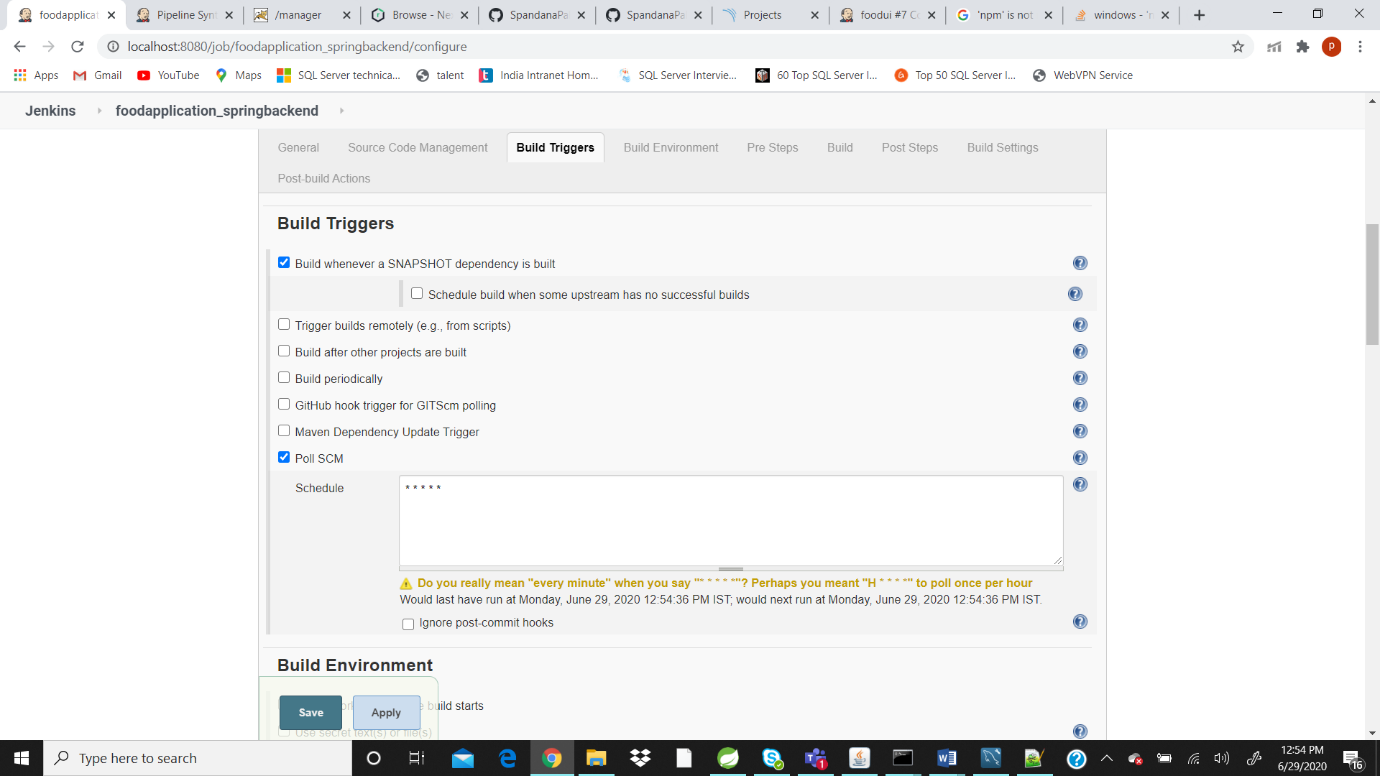
Jenkins makes it easy to automatically build, test and deploy software. It does this by allowing you to define “projects” and then specifying “steps” to build, test and deploy them. It also lets you automatically trigger builds on specific events (such as a repository change) or on a fixed schedule.

* To see how this works in practice, begin by clicking the “New Item” link in the Jenkins dashboard. You should be presented with a form to define a new project, as shown below. Enter a name for the new project (in this example, “FDA-Backend”) and select “Maven Project” as the project type. Click “ok” to proceed.

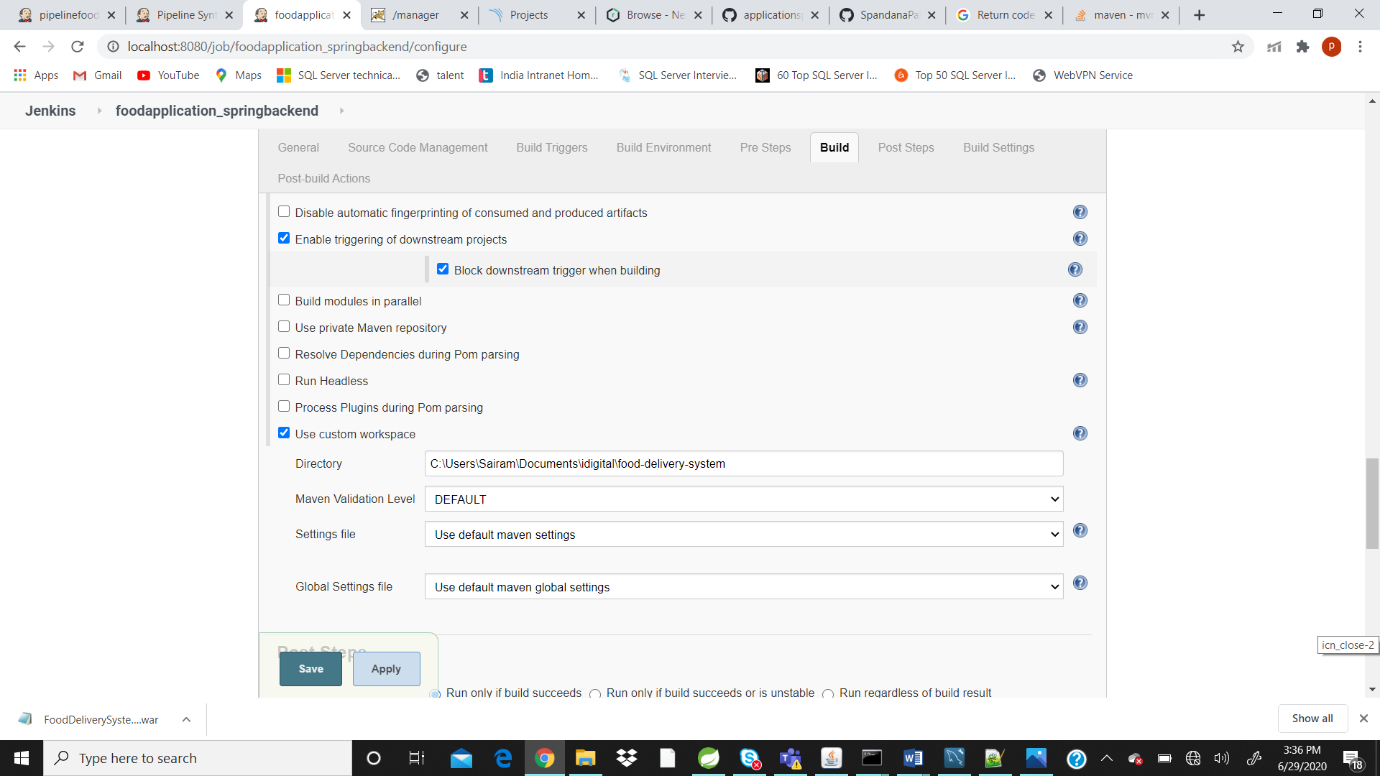


* Go to project (FDA-Backend) -> configure and set the following:

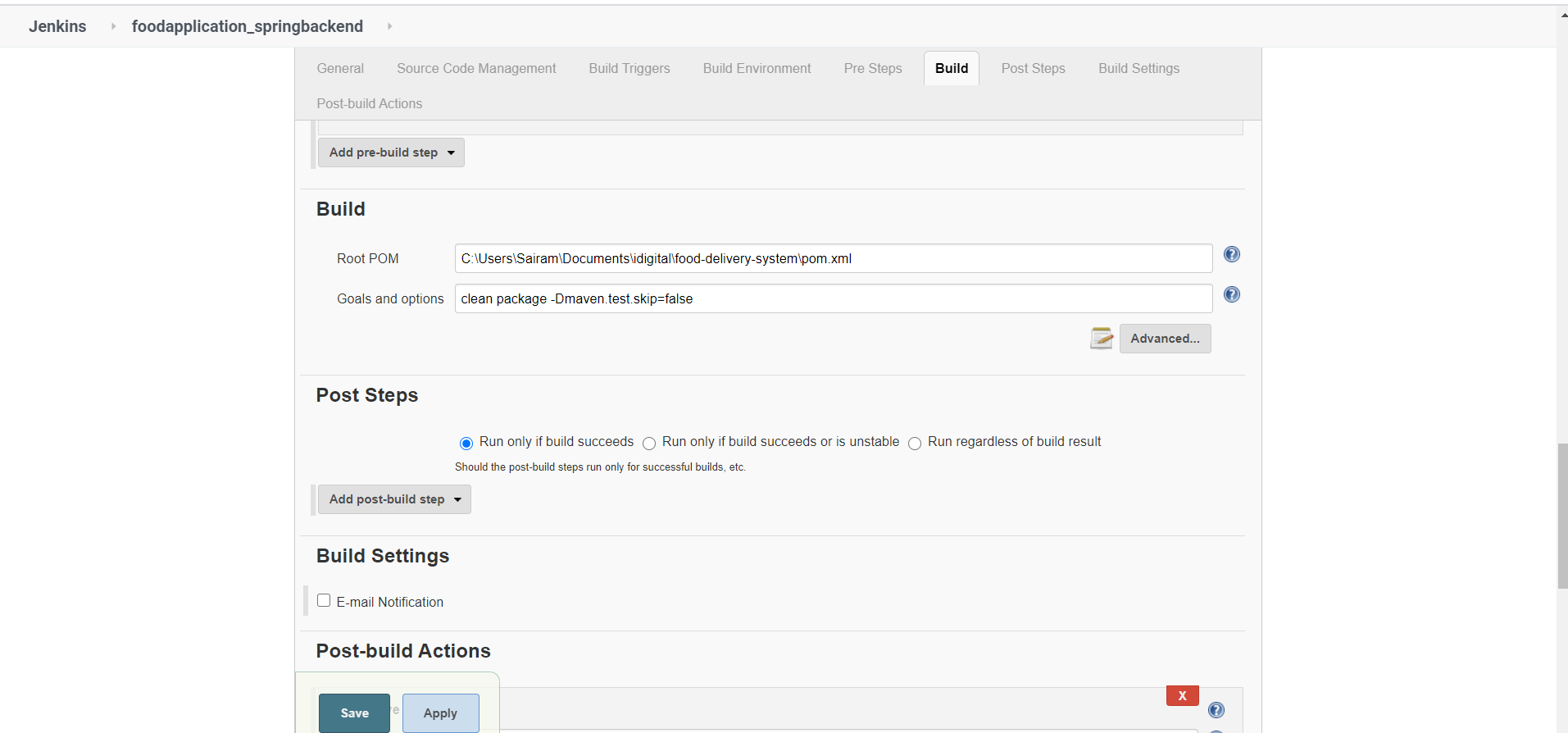
- For continuous build trigger, we have given in Poll SCM



- Use custom workspace (add path of the project folder) as below:



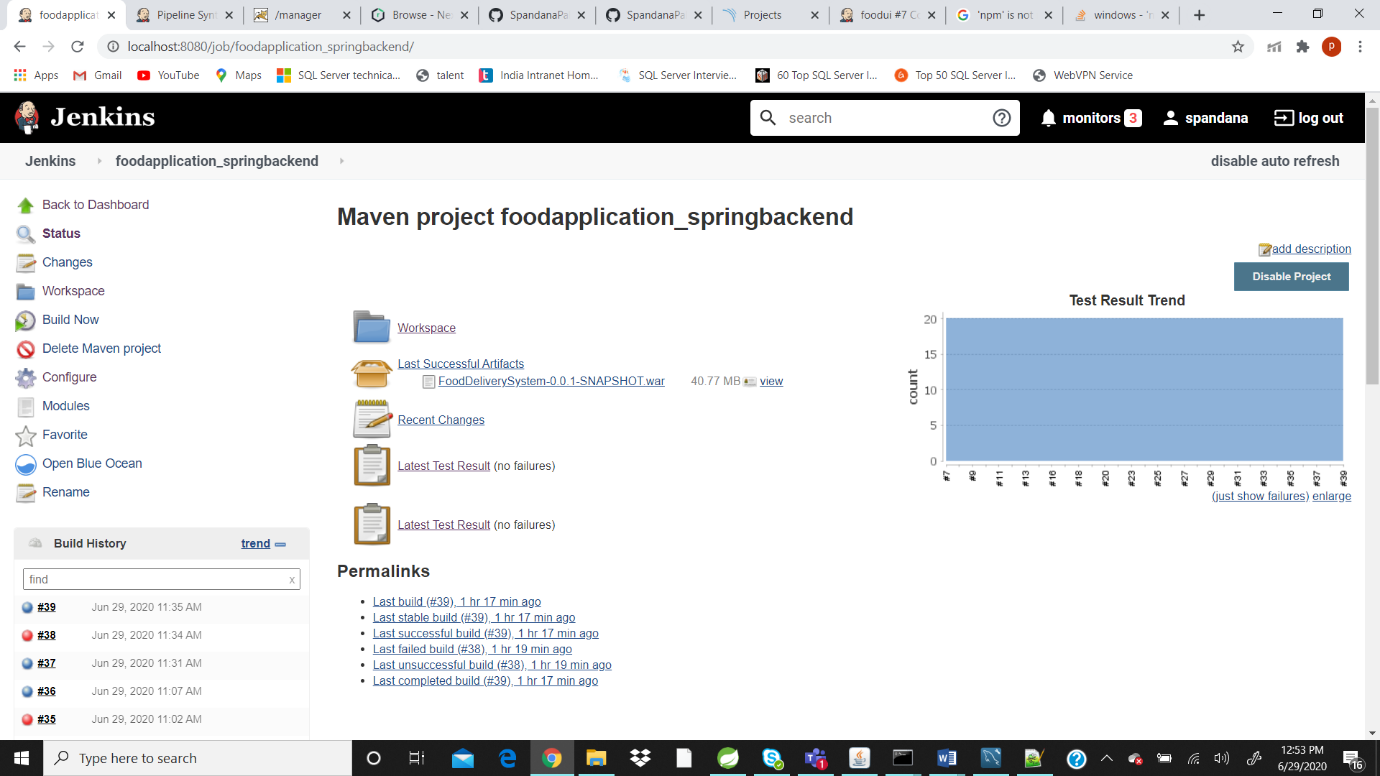
- Goals & Options = clean package -Dmaven.test.skip=false (this will include the testcases in build).



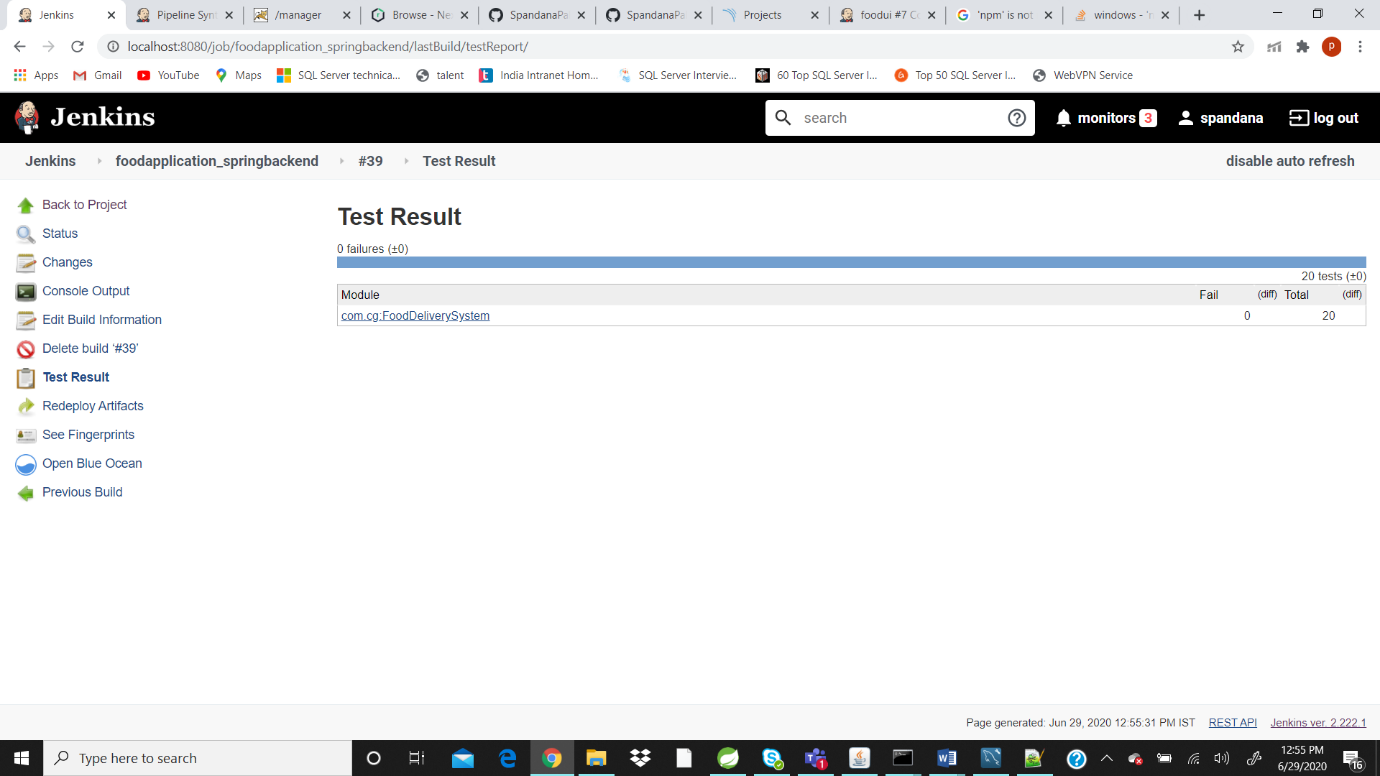
- Click on Apply and Save.

- Go to project window and click “Build Now”.

- Once the build is success, you can see the Jenkins Dashboard with all the testcases passed as below:



- Click on Latest Test Result, it will give the details as below:



## 1.4 Nexus Integration with Jenkins

**Step 1)** Install the Nexus Artifact Uploader plugin in Jenkins

* Go to Jenkins Dashboard -> Manage Jenkins -> Manage Plugins -> Available -> Nexus Artifact Uploader -> Install.

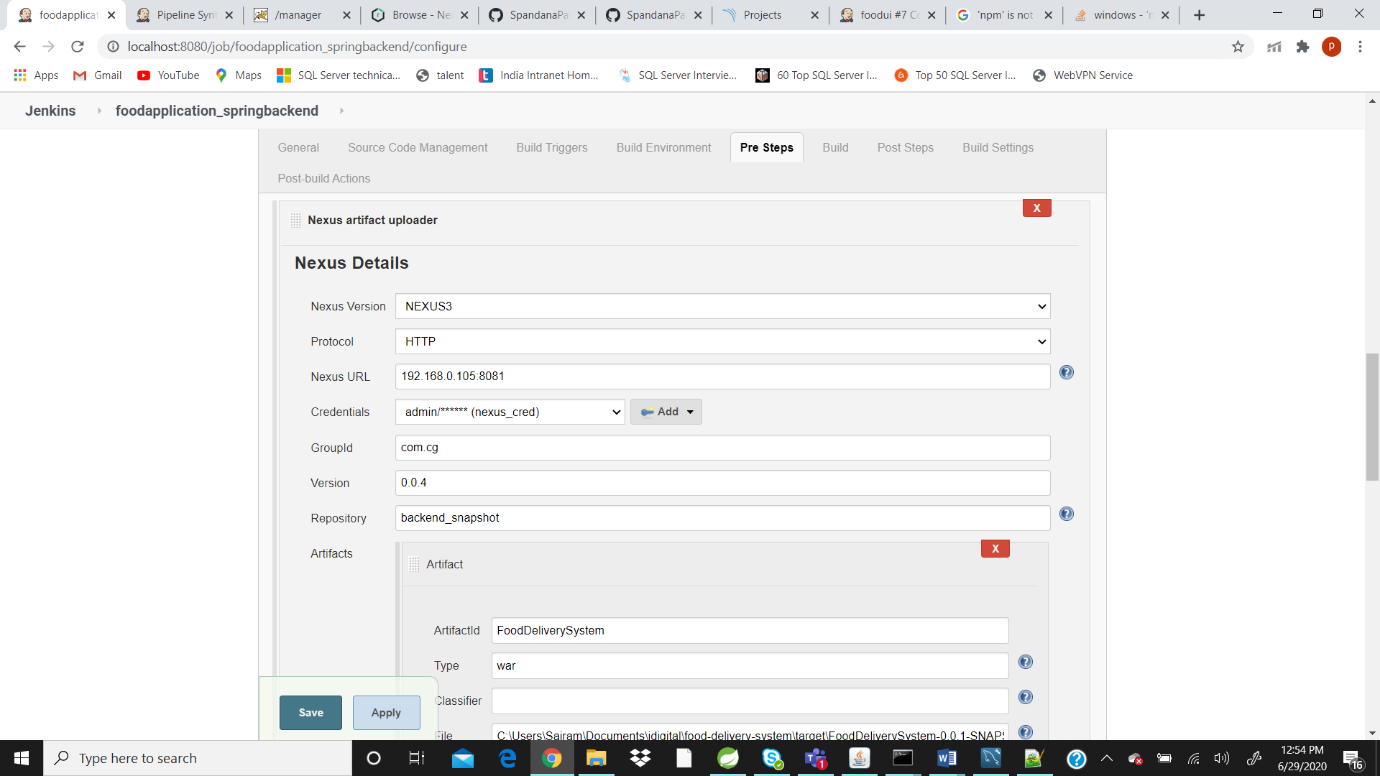
**Step 2)** Configure the Jenkins Nexus Plugin

* Go to project (FDA-Backend) -> configure and set the following:

- In Pre Steps, select Nexus artifact uploader and give the details as below.

**Step 3)** Go to nexus Server and create a new repository

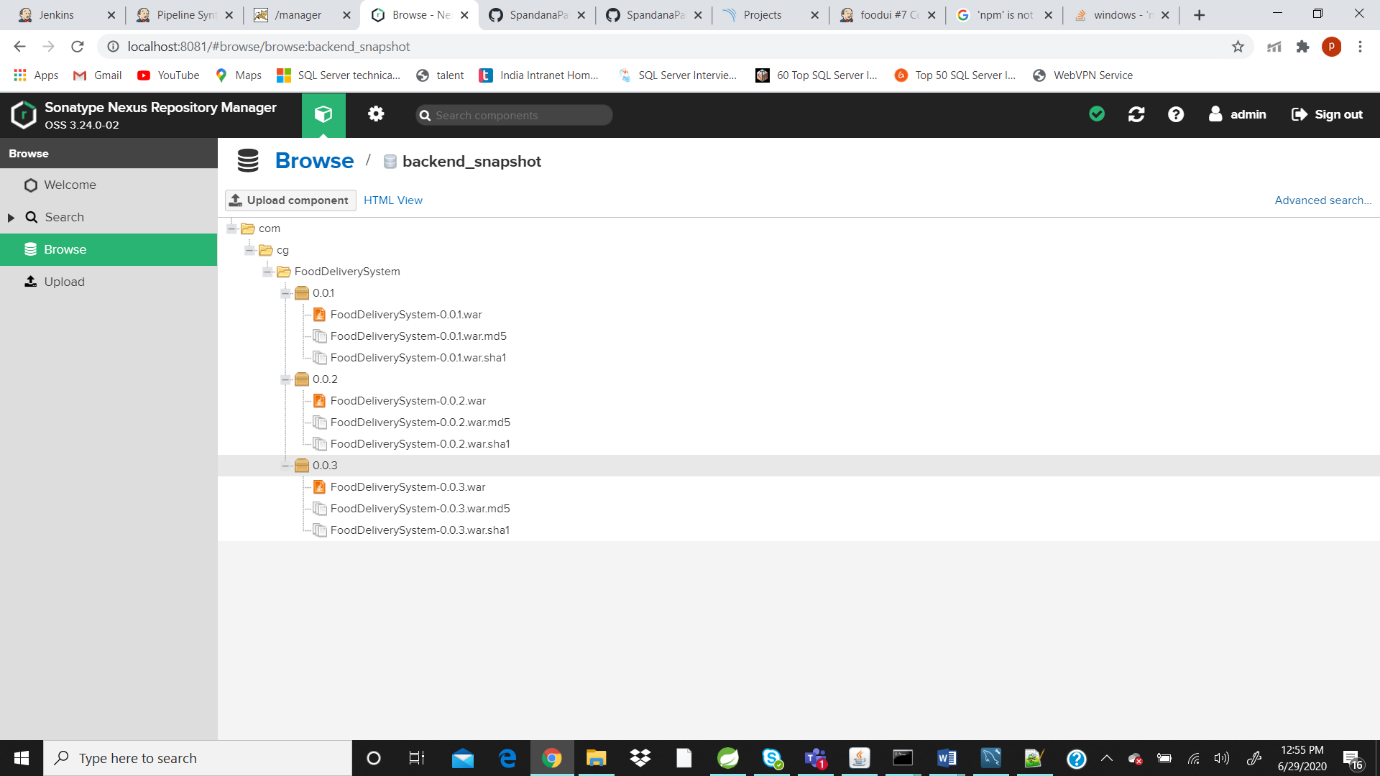
* Settings -> create repository 🡪m2 (hosted) 🡪 give name 🡪 click ok



- Click on Apply and Save.

- Go to project window and click “Build Now”

- Once the build is success, go to the Nexus server you can see the artifacts stored as below:

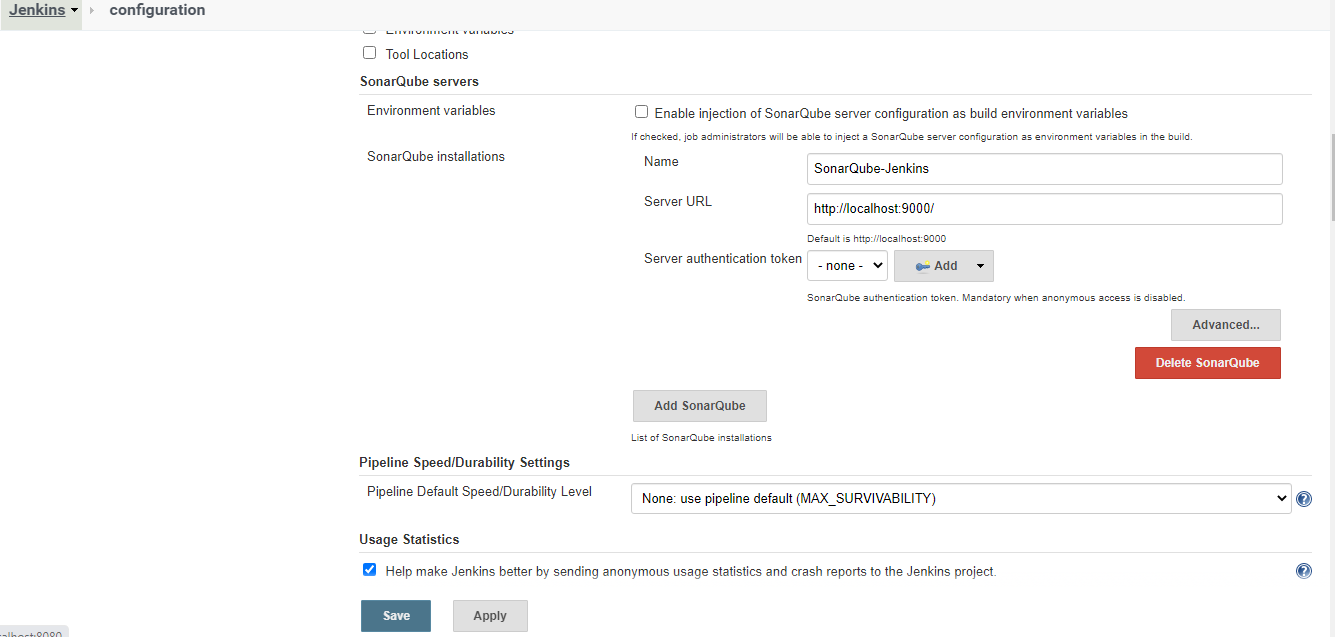


**Note:** We have to install the Nexus locally i.e. in system and login. After login, go to repositories and create one repository.

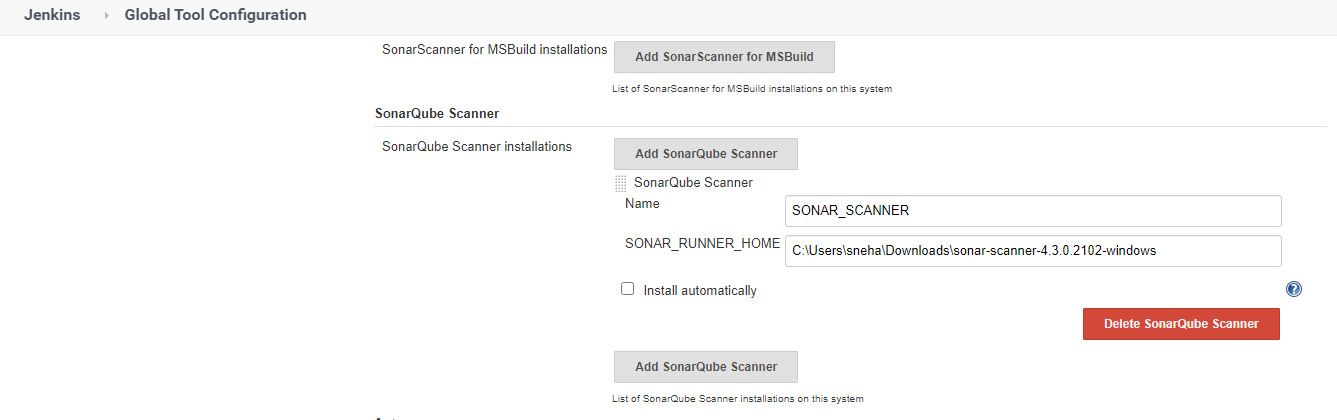
## 1.5 SonarQube Integration with Jenkins

**Step 1)** Install the SonarQube and SonarQube Scanner plugins in Jenkins

* Go to Jenkins Dashboard -> Manage Jenkins -> Manage Plugins -> Available -> SonarQube -> Install.
* Go to Jenkins Dashboard -> Manage Jenkins -> Manage Plugins -> Available -> Sonar Scanner -> Install.
* Go to Jenkins Dashboard -> Manage Jenkins -> Configure System -> Add SonarQube.



* Go to Jenkins Dashboard -> Manage Jenkins -> Global Tool Configuration -> Add SonarQube Scanner.

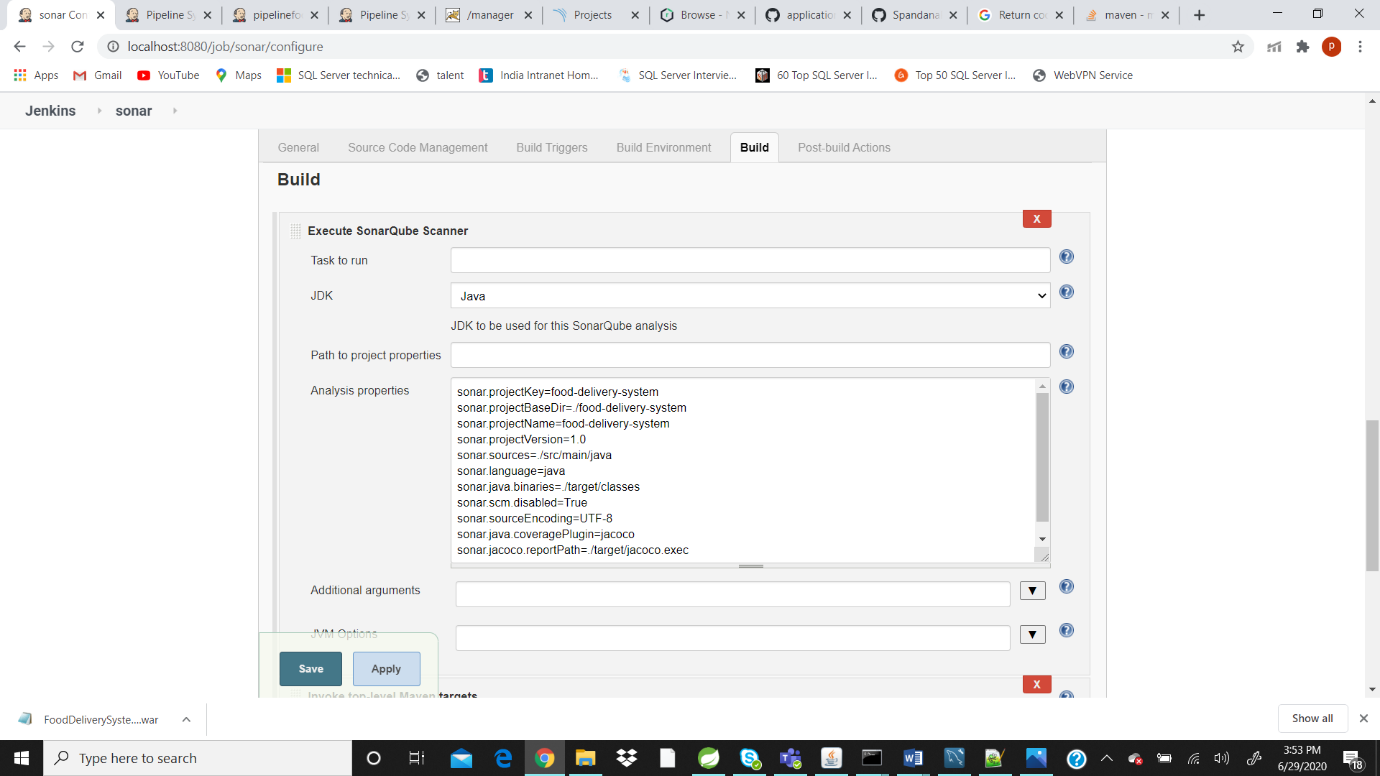


**Note:** We have to install SonarQube and SonarQube Scanner locally i.e. in system and login.

**Step 2)** Configure the SonarQube

* Go to project (FDA-Backend) -> configure and set the following:

- In Build, select Execute SonarQube Scanner and give the details as below

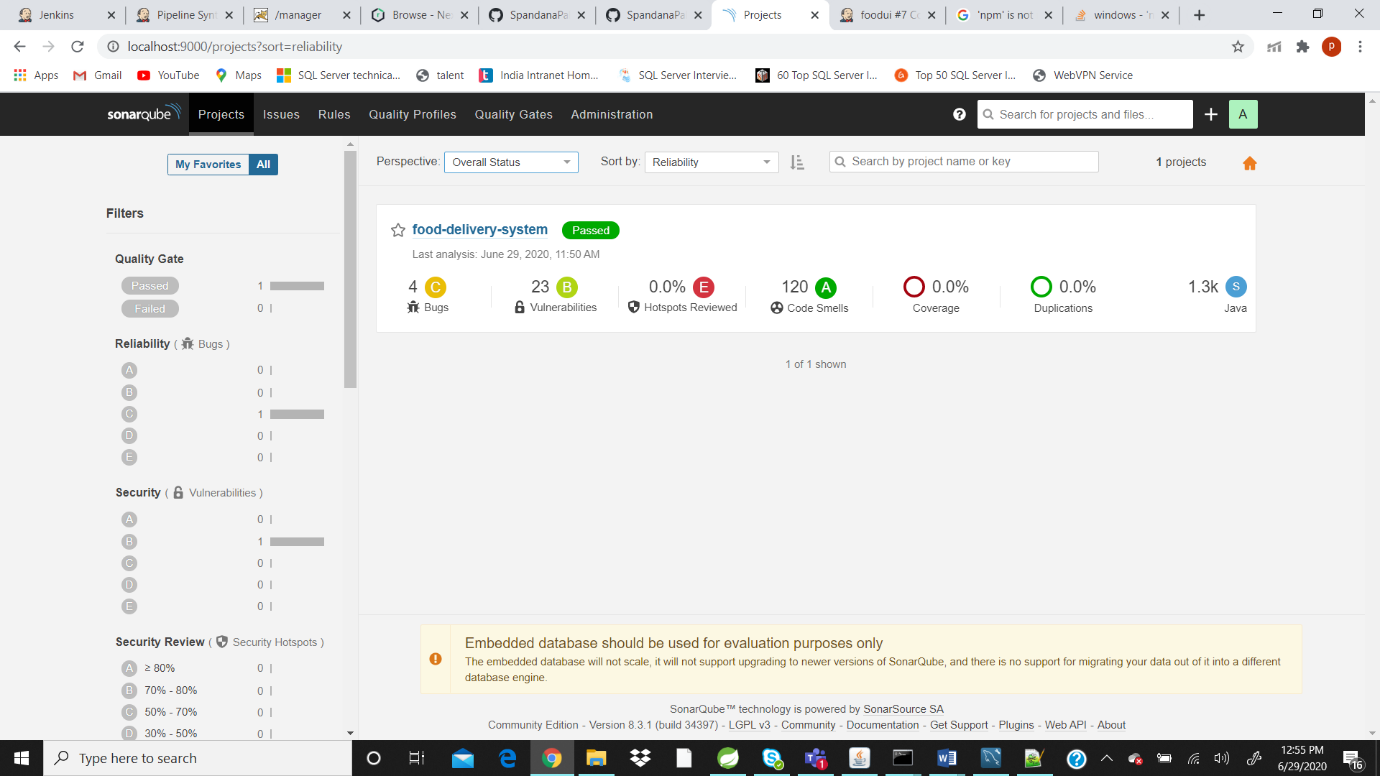


|  |
| --- |
| Analysis properties:  sonar.projectKey=food-delivery-system sonar.projectBaseDir=./food-delivery-system sonar.projectName=food-delivery-system sonar.projectVersion=1.0 sonar.sources=./src/main/java sonar.language=java sonar.java.binaries=./target/classes sonar.scm.disabled=True sonar.sourceEncoding=UTF-8 sonar.java.coveragePlugin=jacoco sonar.jacoco.reportPath=./target/jacoco.exec |

- Click on Apply and Save.

- Go to project window and click “Build Now”

- Once the build is success, go to the SonarQube you can see the Sonar details as below:



## 1.6 Deploying on Tomcat Server

<https://tomcat.apache.org/download-90.cgi> -- download tomcat for windows

Go to tomcat folder

C:\Program Files (x86)\Apache Software Foundation\Tomcat 9.0\conf\tomcat-users.xml file

Edit the file and give the values as below

<user username="deployuser" password="deployuser" roles="manager-script" />

<user username="admin" password="admin" roles="manager-gui" />

* Use this manager-script credentials In Jenkins while configuring.
* Go to tomcat folder

C:\Program Files (x86)\Apache Software Foundation\Tomcat 9.0\webapps\host-manager\META-INF\context.xml -🡪 Comment the value part

Do the same in C:\Program Files (x86)\Apache Software Foundation\Tomcat 9.0\webapps\manager\META-INF\context.xml-🡪 Comment the value part

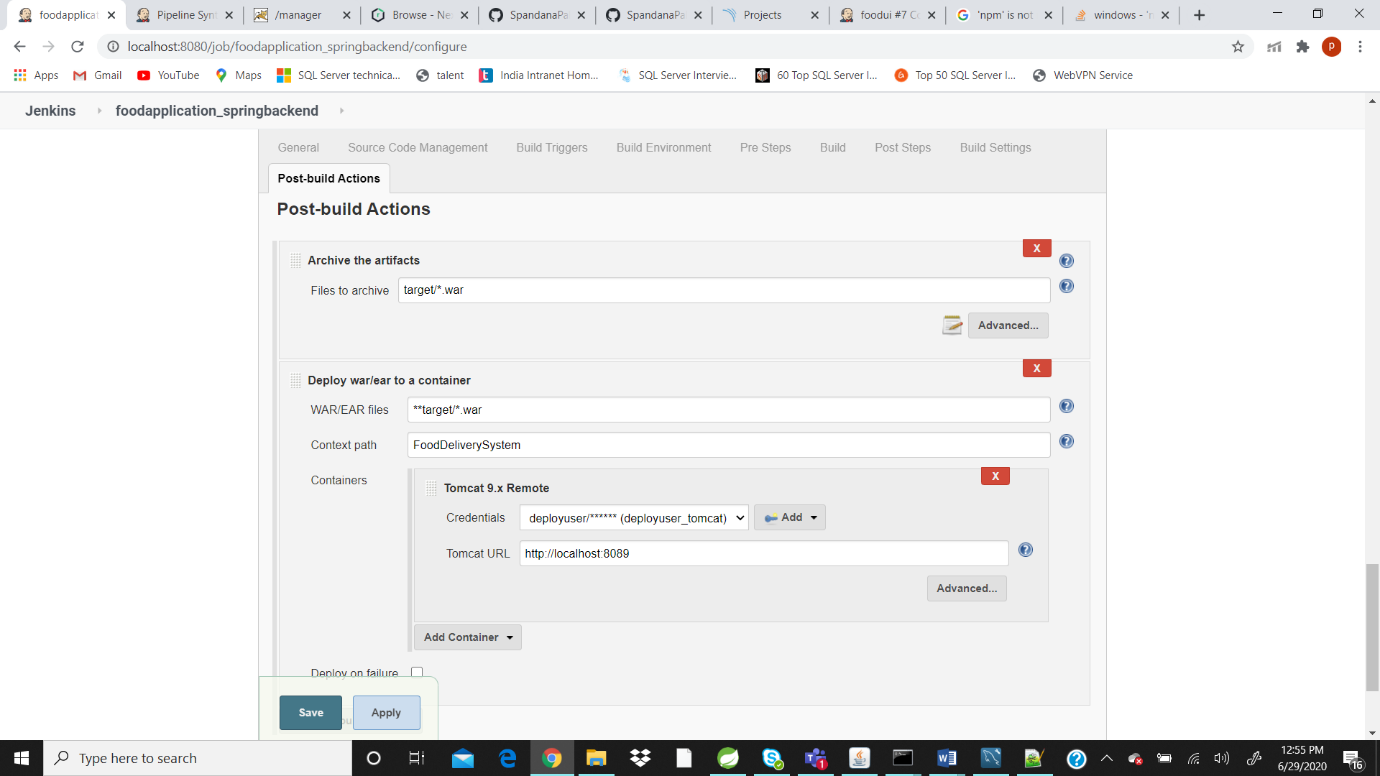
**Step 1)** Install the Tomcat plugins in Jenkins

* Go to Jenkins Dashboard -> Manage Jenkins -> Manage Plugins -> Available -> Deploy to Container -> Install

**Step 2)** Configure the Tomcat Deploy to Container Plugin

* Go to project (FDA-Backend) -> configure and set the following:

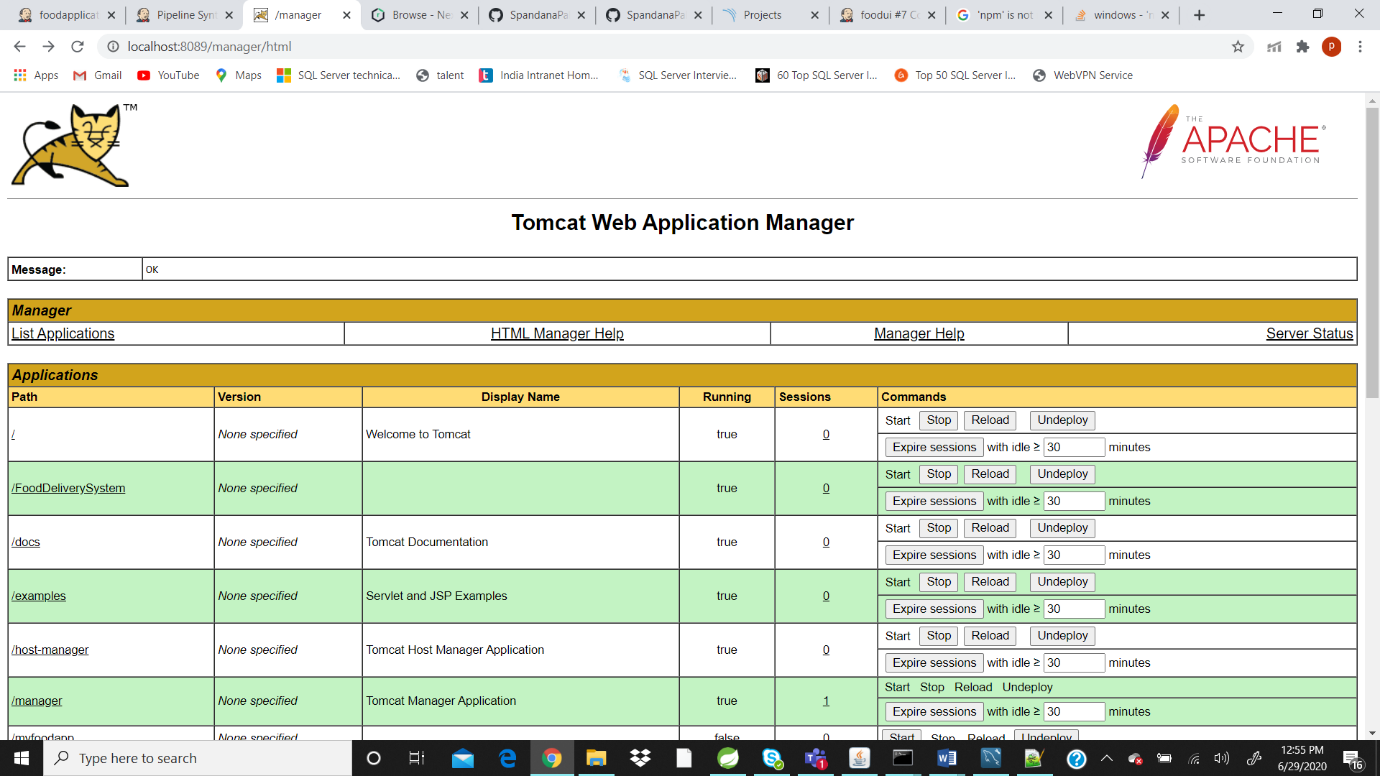
- In Post-build-actions, select Deploy war to a container and give the details as below:



- Click on Apply and Save

- Go to project window and click “Build Now”

- Once the build is success, go to the Tomcat server you can see the deployed project (FoodDeliverySystem) as below:



## 1.7 Jenkins Pipeline

**Step 1)** Create a Pipeline

* Go to Jenkins Dashboard -> New Item -> name as FDA-Pipeline -> Pipeline -> Click ok.

**Step 2)** Configure the Pipeline

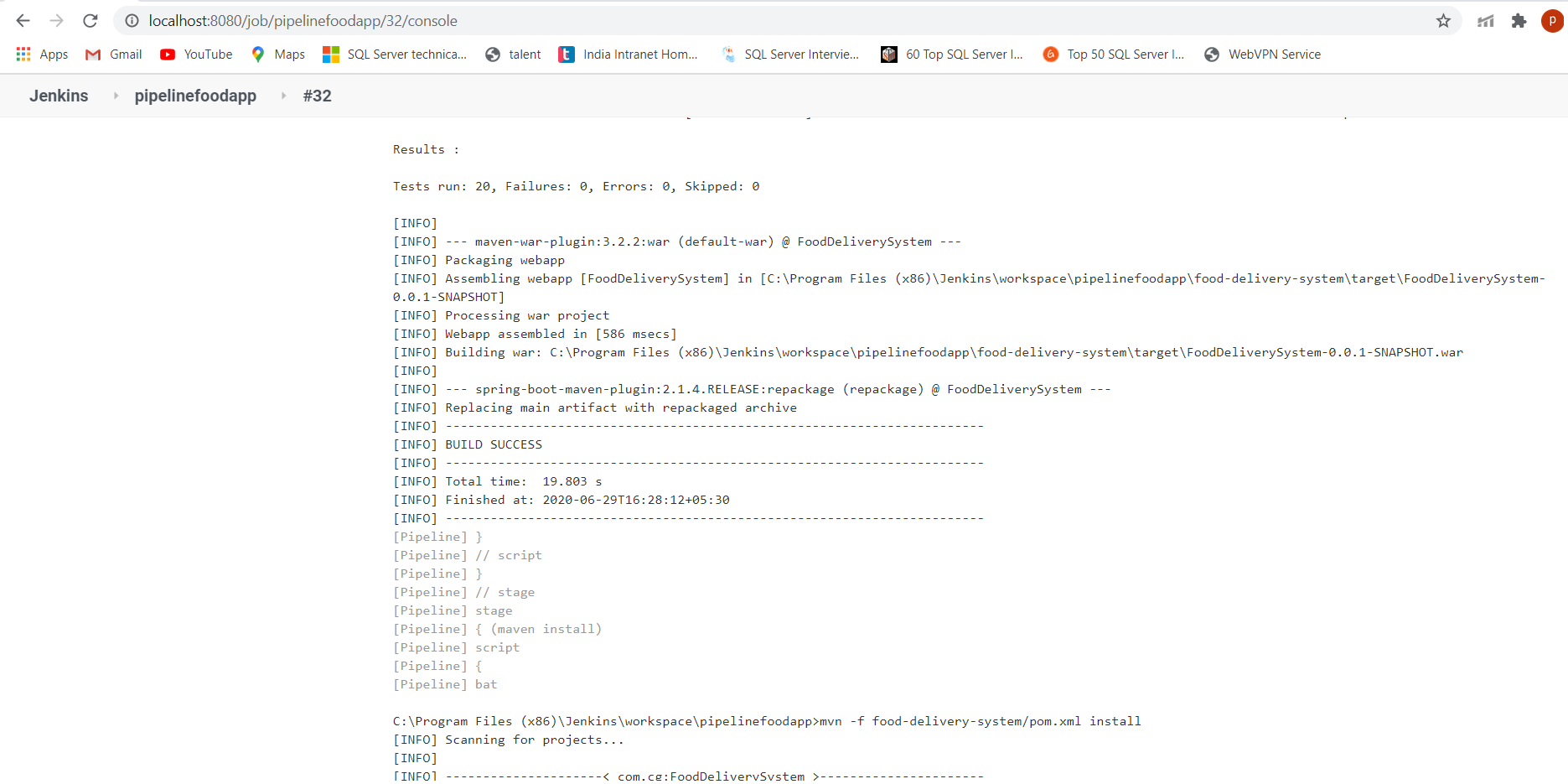
* Go to project (FDA-Pipeline) -> configure and set the following:

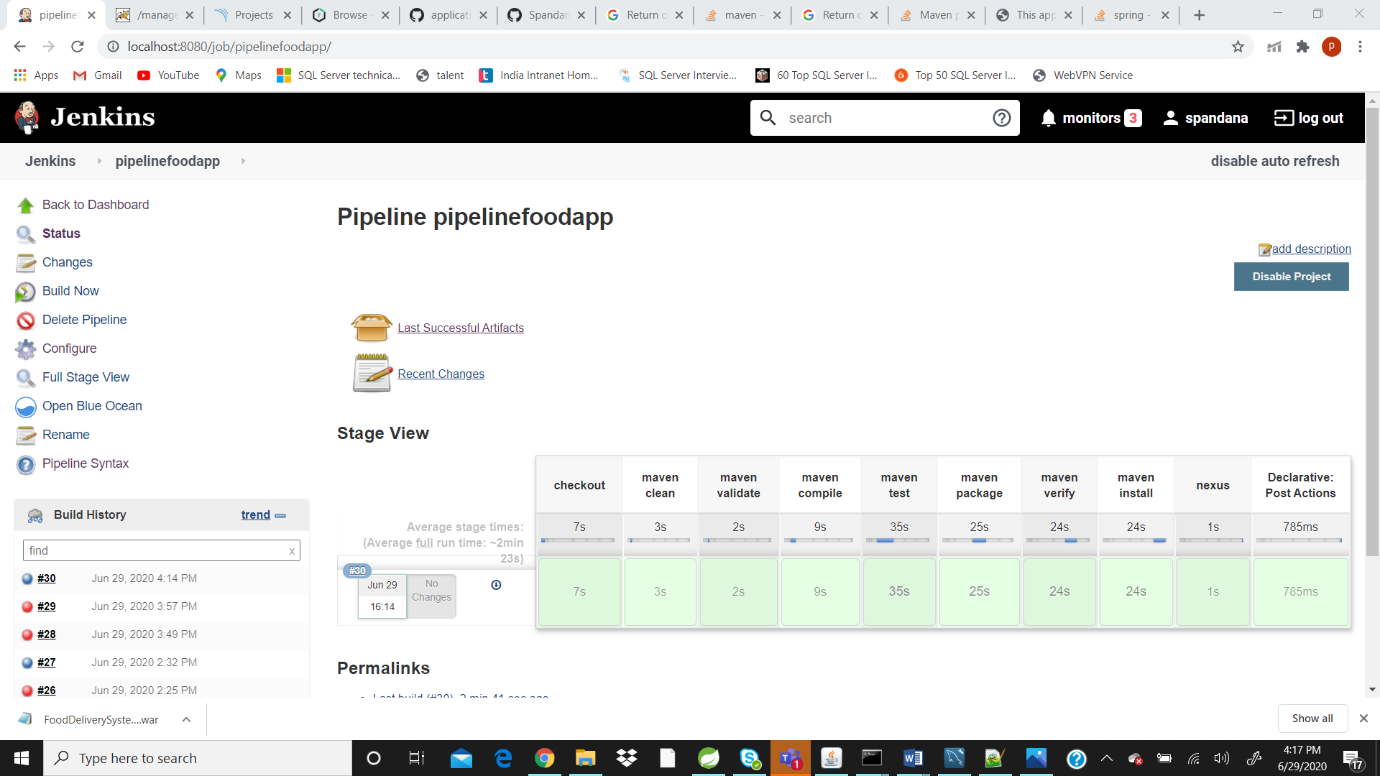
- In Pipeline, give the Pipeline Script as below:

|  |
| --- |
| pipeline {  options {  buildDiscarder(logRotator(numToKeepStr: '6'))  }  agent any  stages {  stage('checkout') {  steps {  git 'https://github.com/abc/applicationspring.git'  }  }  stage ('maven clean'){  steps {  script {  bat "mvn -f food-delivery-system/pom.xml clean"  }  }  }  stage ('maven validate'){  steps {  script {  bat "mvn -f food-delivery-system/pom.xml validate"  }  }  }  stage ('maven compile'){  steps {  script {  bat "mvn -f food-delivery-system/pom.xml compile"  }  }  }  stage ('maven test'){  steps {  script {  bat "mvn -f food-delivery-system/pom.xml test"  }  }  }  stage ('maven package'){  steps {  script {  bat "mvn -f food-delivery-system/pom.xml package"  }  }  }  stage ('maven verify'){  steps {  script {  bat "mvn -f food-delivery-system/pom.xml verify"  }  }  }  stage ('maven install'){  steps {  script {  bat "mvn -f food-delivery-system/pom.xml install"  }  }  }  stage ('nexus'){  steps {  script {  nexusArtifactUploader artifacts: [[artifactId: 'FoodDeliverySystem', classifier: '', file: 'C:\\Program Files (x86)\\Jenkins\\workspace\\pipelinefoodapp\\food-delivery-system\\target\\FoodDeliverySystem-0.0.1-SNAPSHOT.war', type: 'war']], credentialsId: 'new\_nexuscredentials', groupId: 'com.cg', nexusUrl: '192.168.1.9:8081', nexusVersion: 'nexus3', protocol: 'http', repository: 'fooddelivery\_spring', version: '0.0.2'  }  }  }    }  post {  always {  archiveArtifacts artifacts: 'food-delivery-system/target/\*\*'  cleanWs()  }  }} |

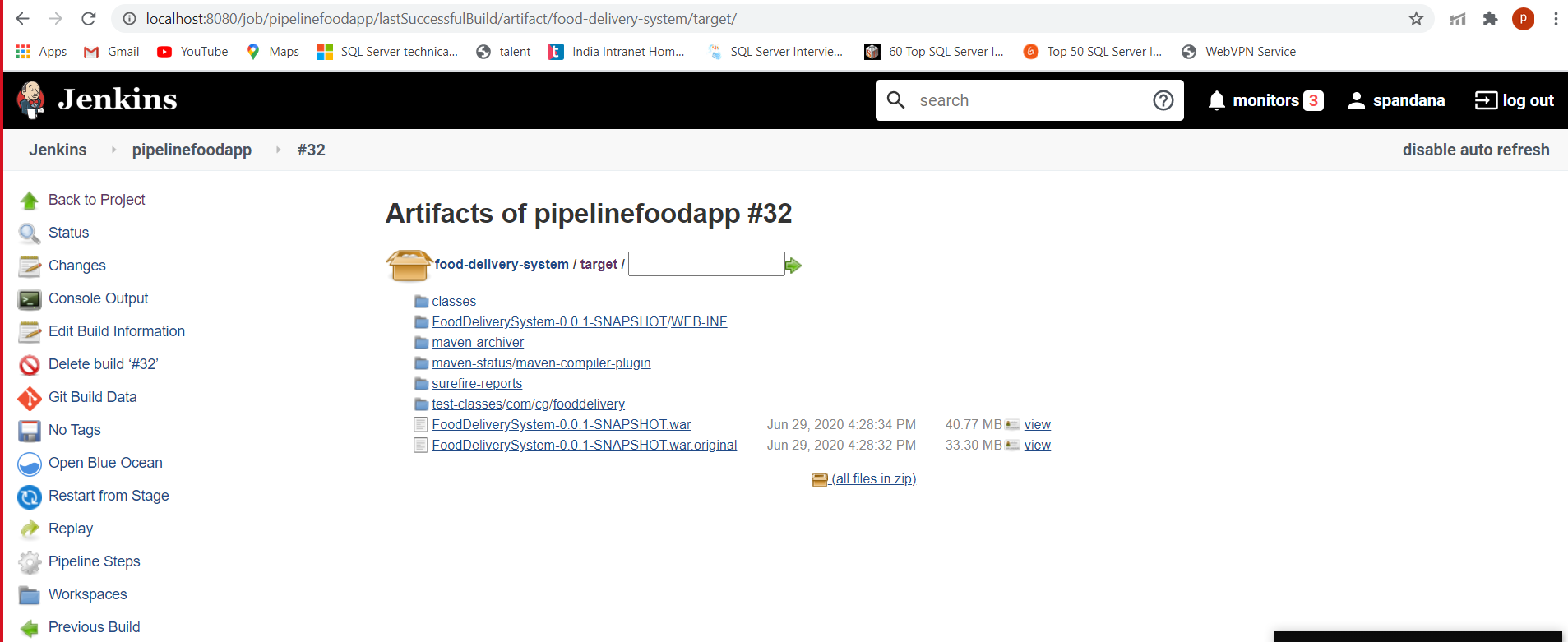
- Click on Apply and Save

- Once the build is success, you can see as below:

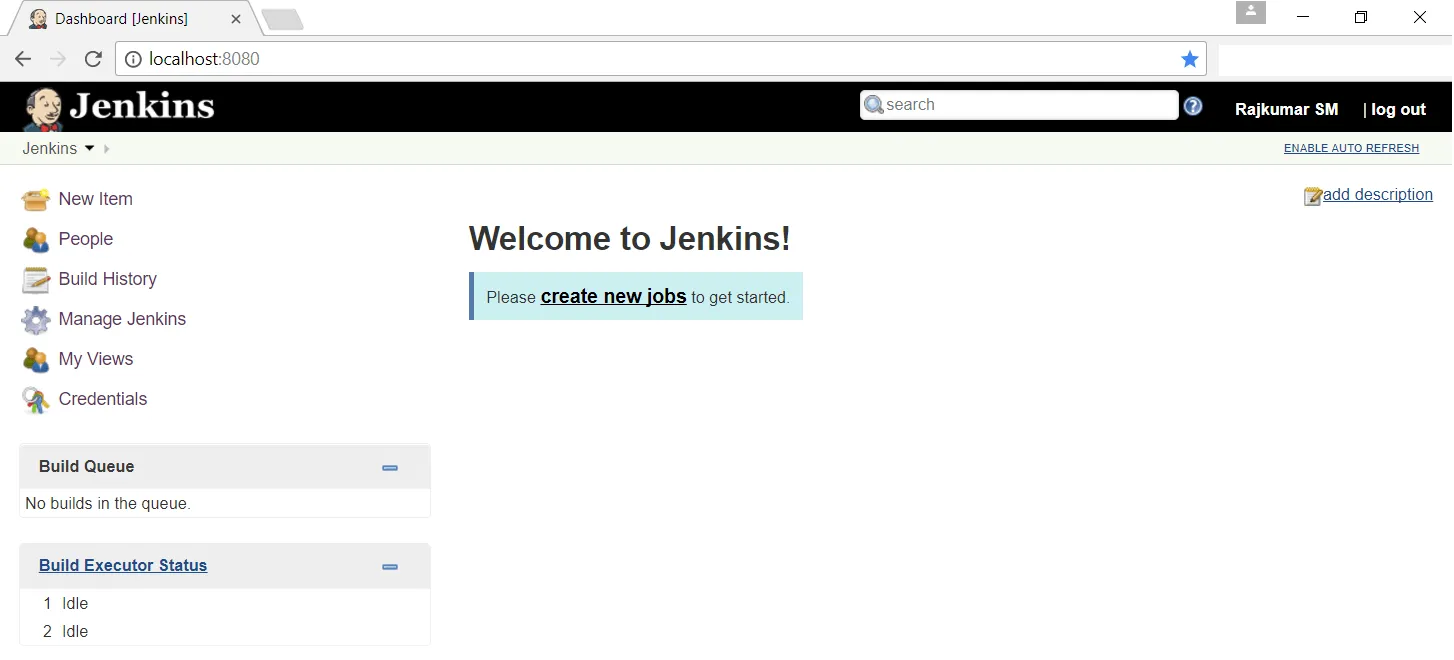




* Click on LastSuccessfulArtifacts for checking latest artifacts



# **2. Integration of Angular project with Jenkins**



* Click on **New Item** link to create a job on Jenkins
* Enter an item name (here I am adding a name ‘Angular-Local) and click on Freestyle Project and click on OK
* Open Configure
* Download File SCM plugin in manage plugins
* Click on Source Code Management and select File System radio button

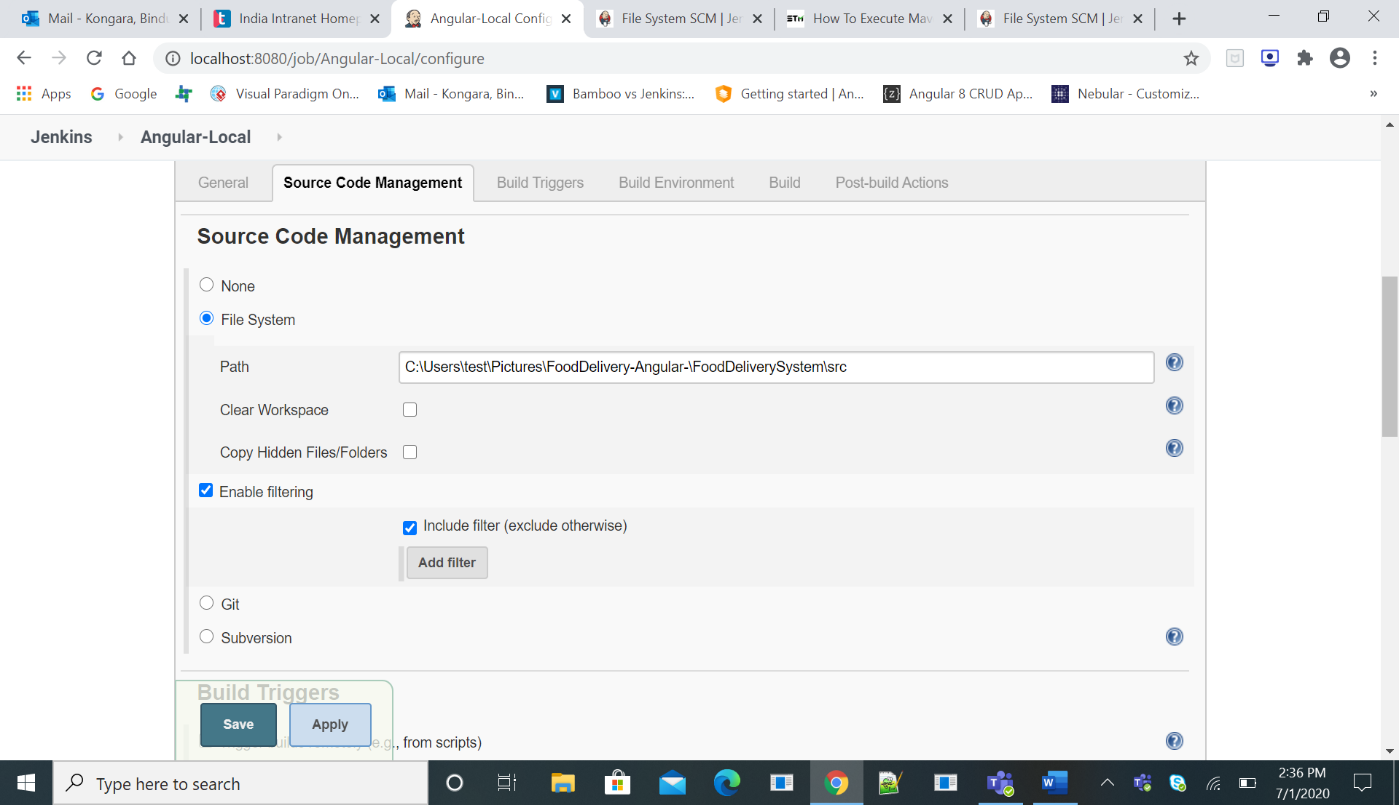
(Path: C:\Users\test\Pictures\FoodDelivery-Angular-\FoodDeliverySystem\src)

* Scroll down to **Build** option. Click on **Add Build Step** and choose the value **Execute windows batch command** from the drop-down list.
* Enter commands:

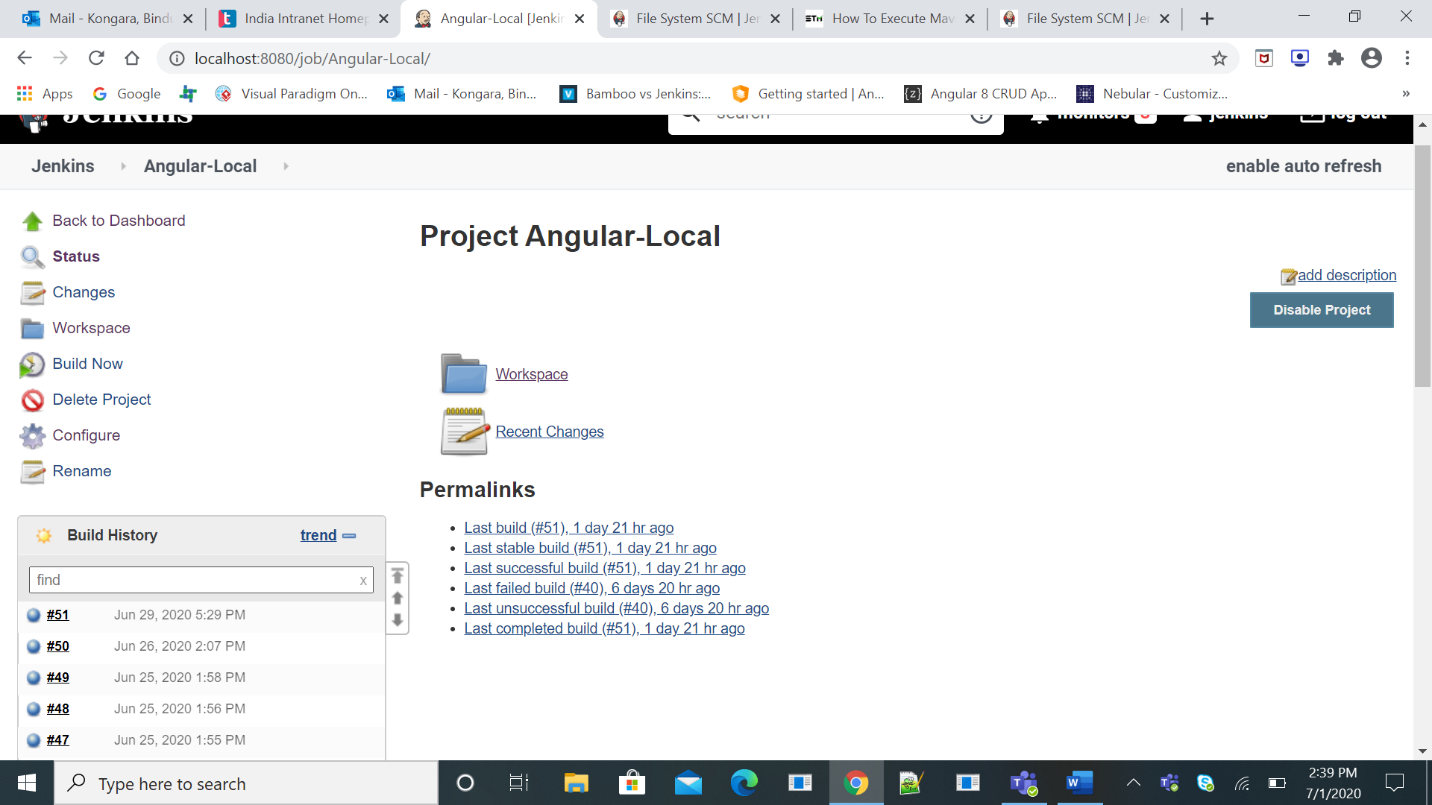
npm install

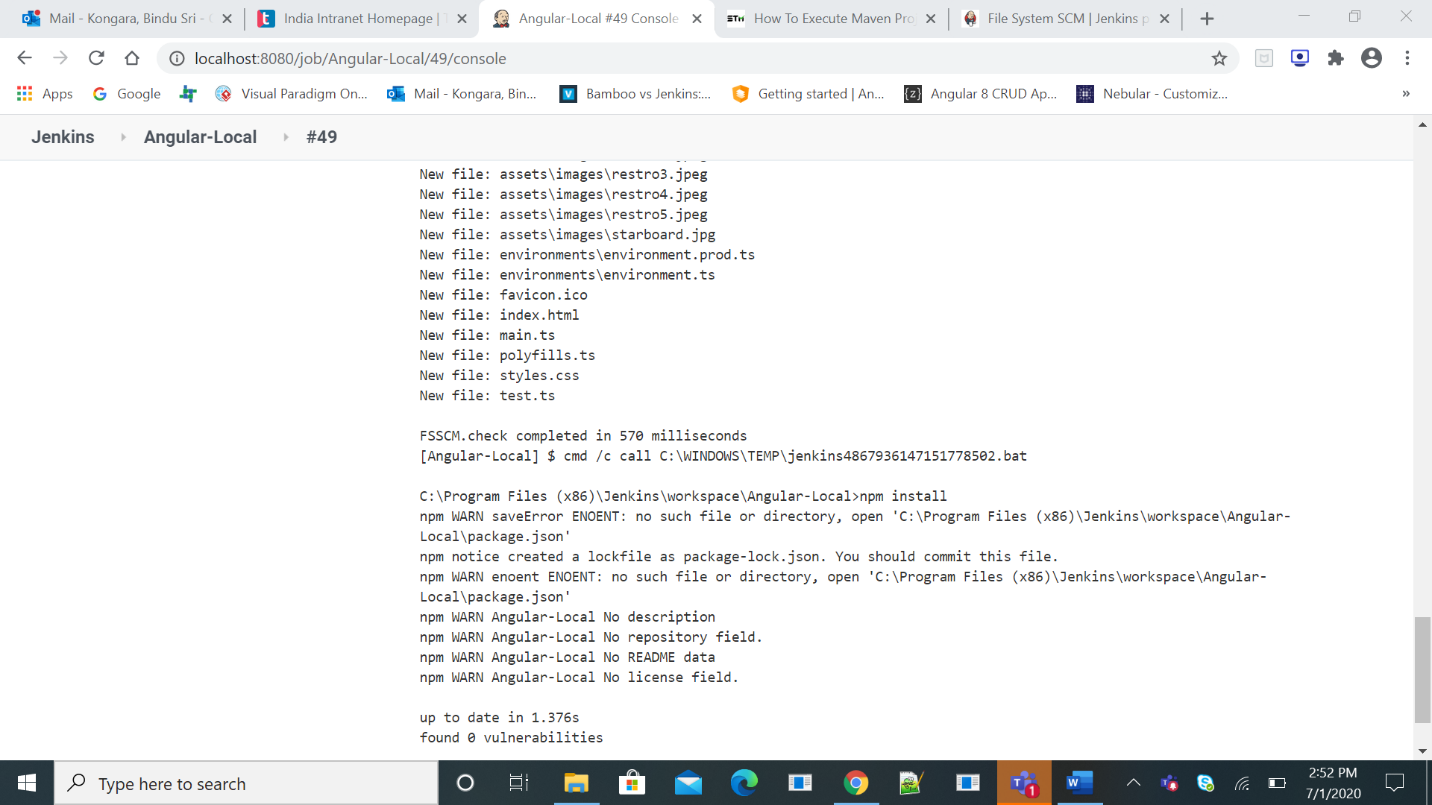
npm run build

npm start



* Click on **Apply** and **Save**
* Let’s execute it now. Click on **Build Now** button.

 Right click on Build Number (here in my case it is #51) and click on Console Output to see the result. You could see Build Status ‘Success’ on Console Output.

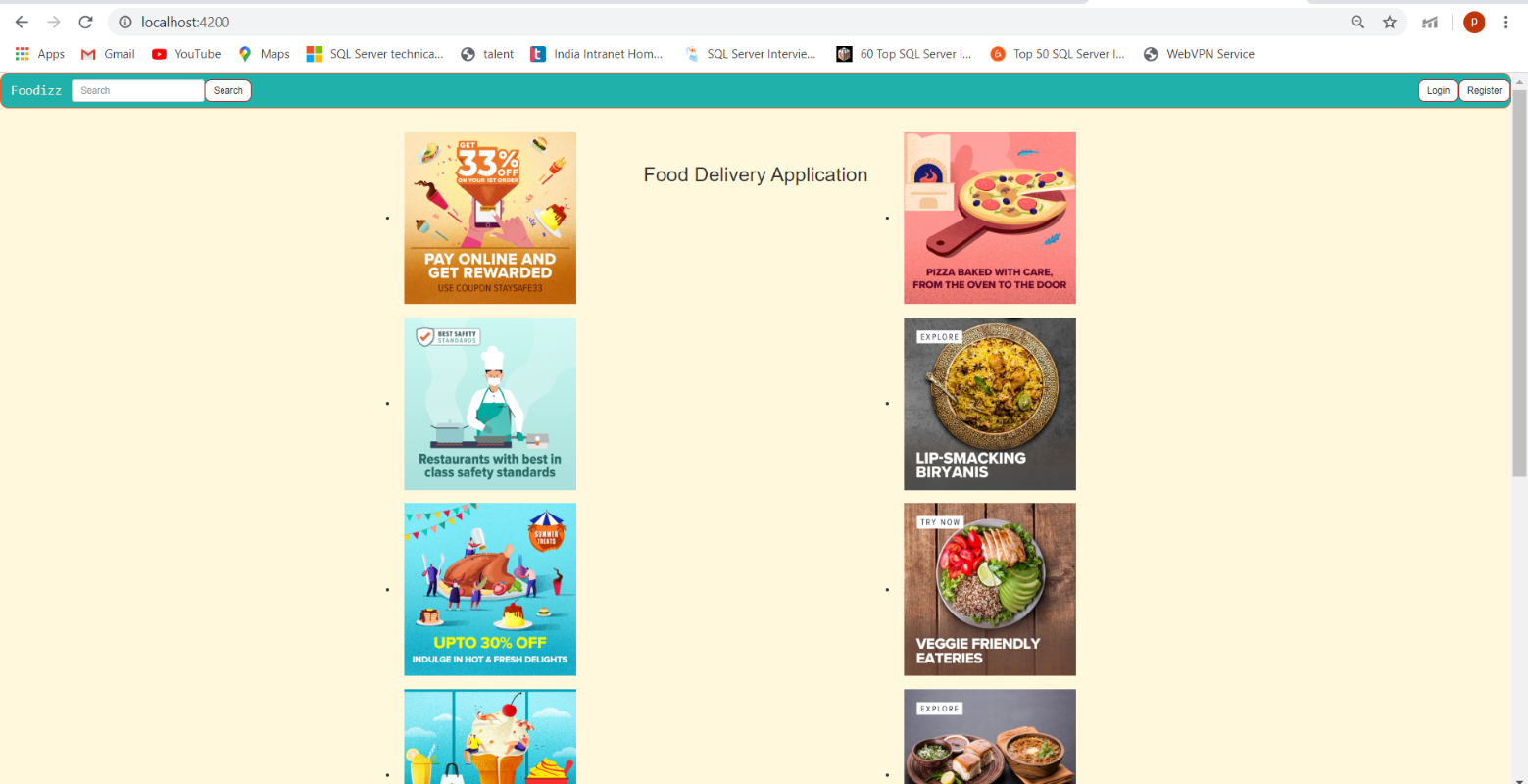


## 2.1 Deploying Project

* run ng build in Visual studio code terminal
* dist folder will be made in the project folder
* Go to AWS S3 bucket
* upload the dist file their
* S3 Bucket will give you the url

<http://fooddelivery5.s3-website-us-east-1.amazonaws.com/>

* run it in chrome clearing all the cache



# **3. Team Members**

* Palakurthy Spandana
* Sneha Manga
* Kongara BinduSri
* Kamble Aditya
* Pooja Tiwari
* Boreddy Swapna
* Amith kumar (MENTOR)
* Chetan Yadav (MENTOR)