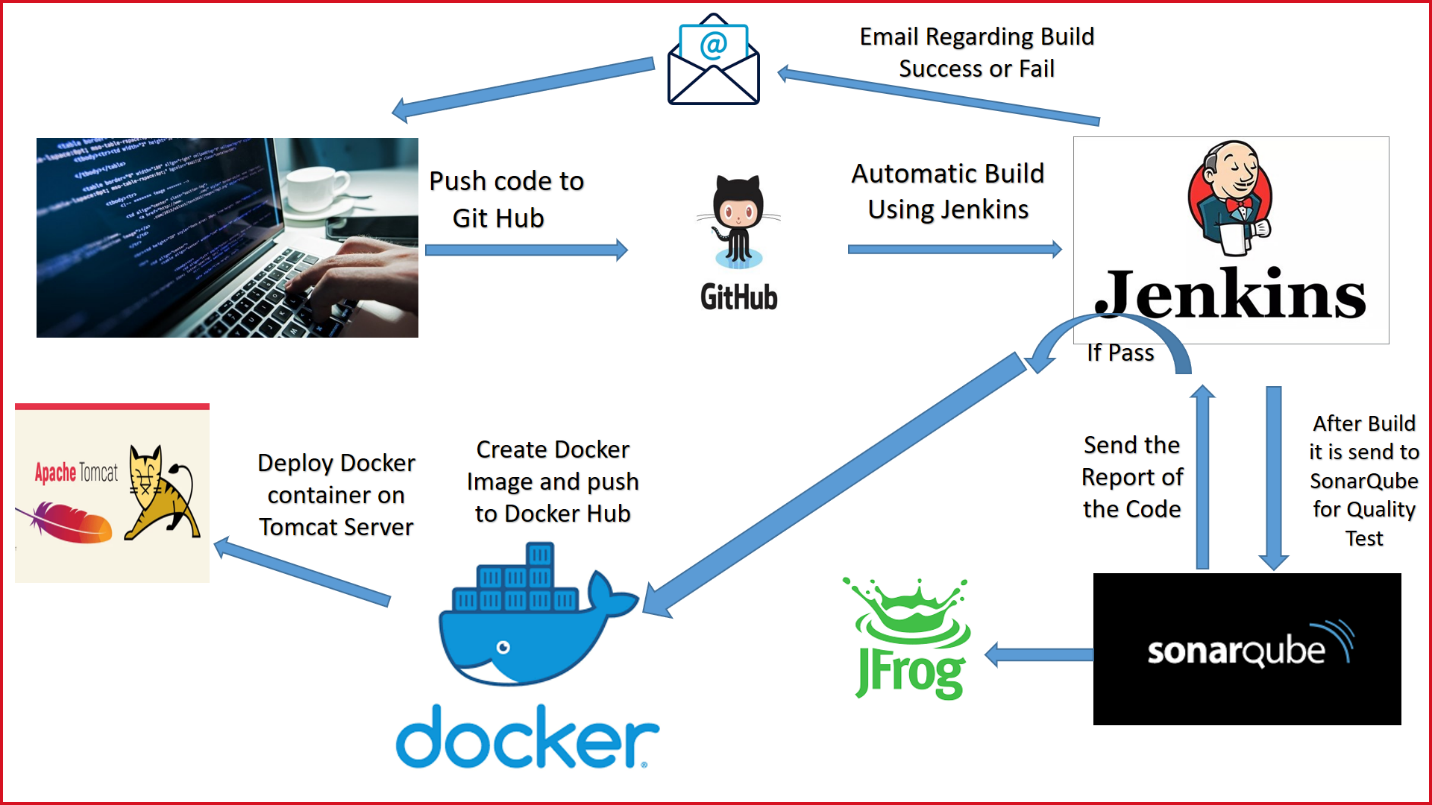
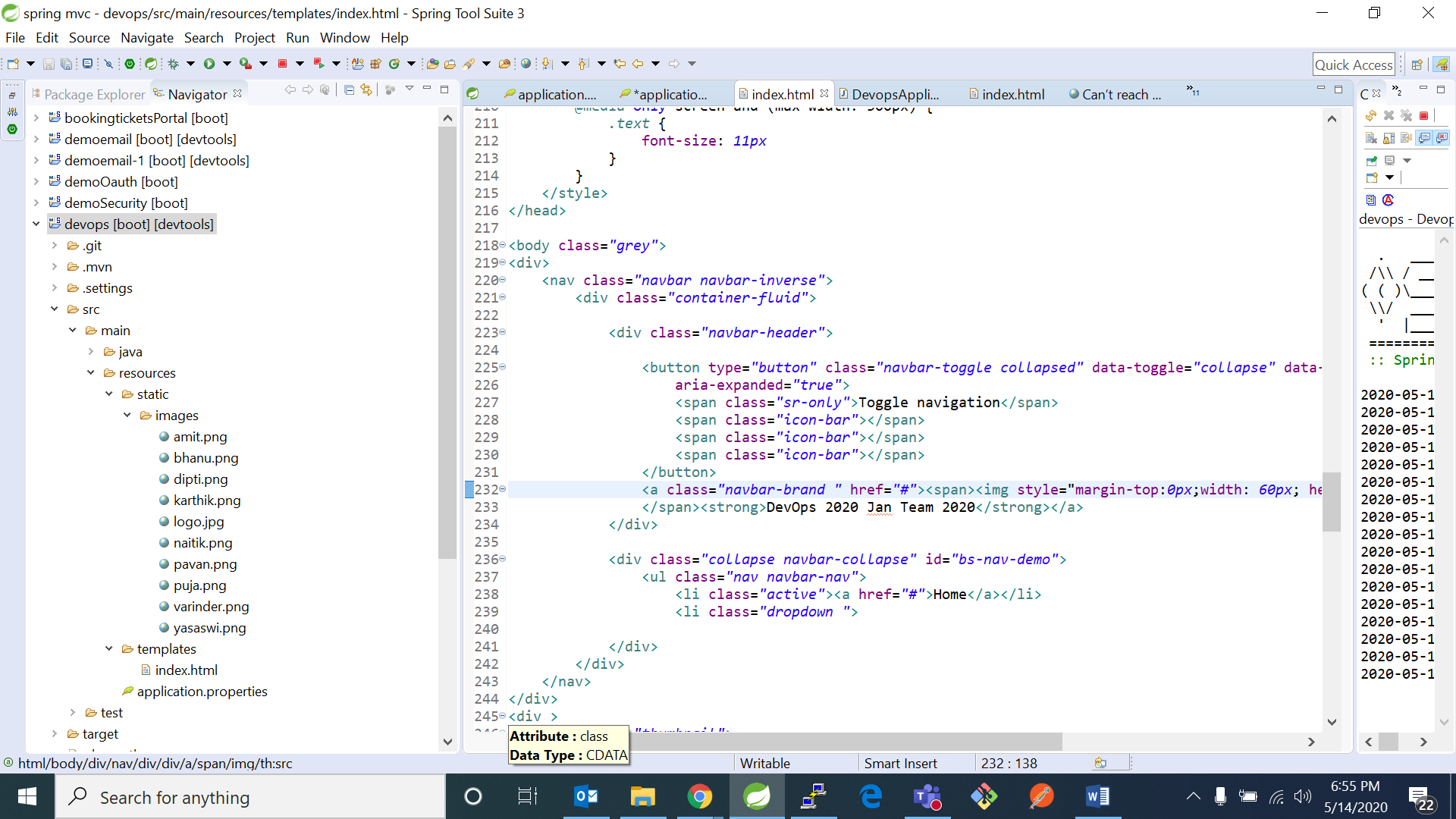
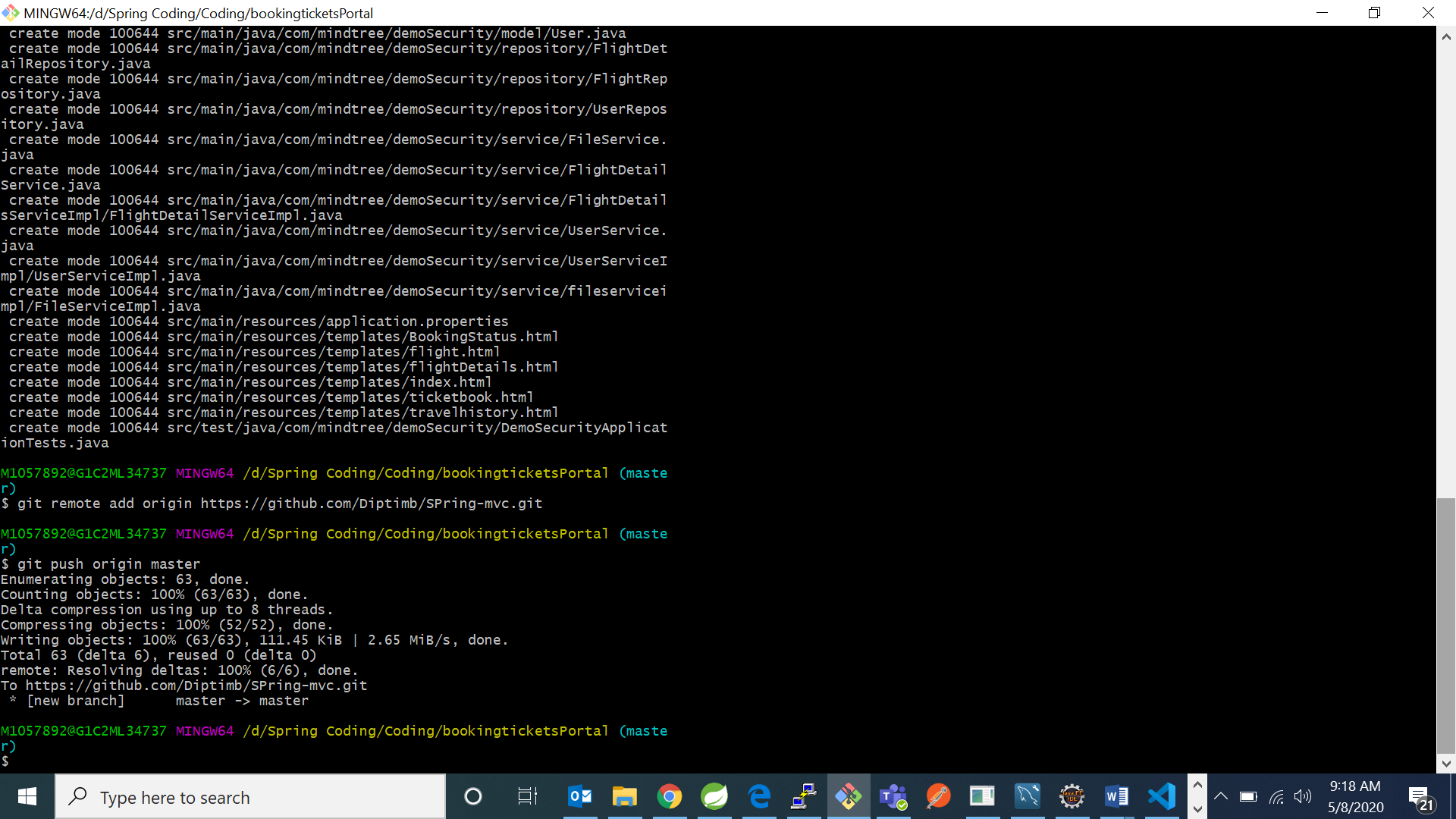
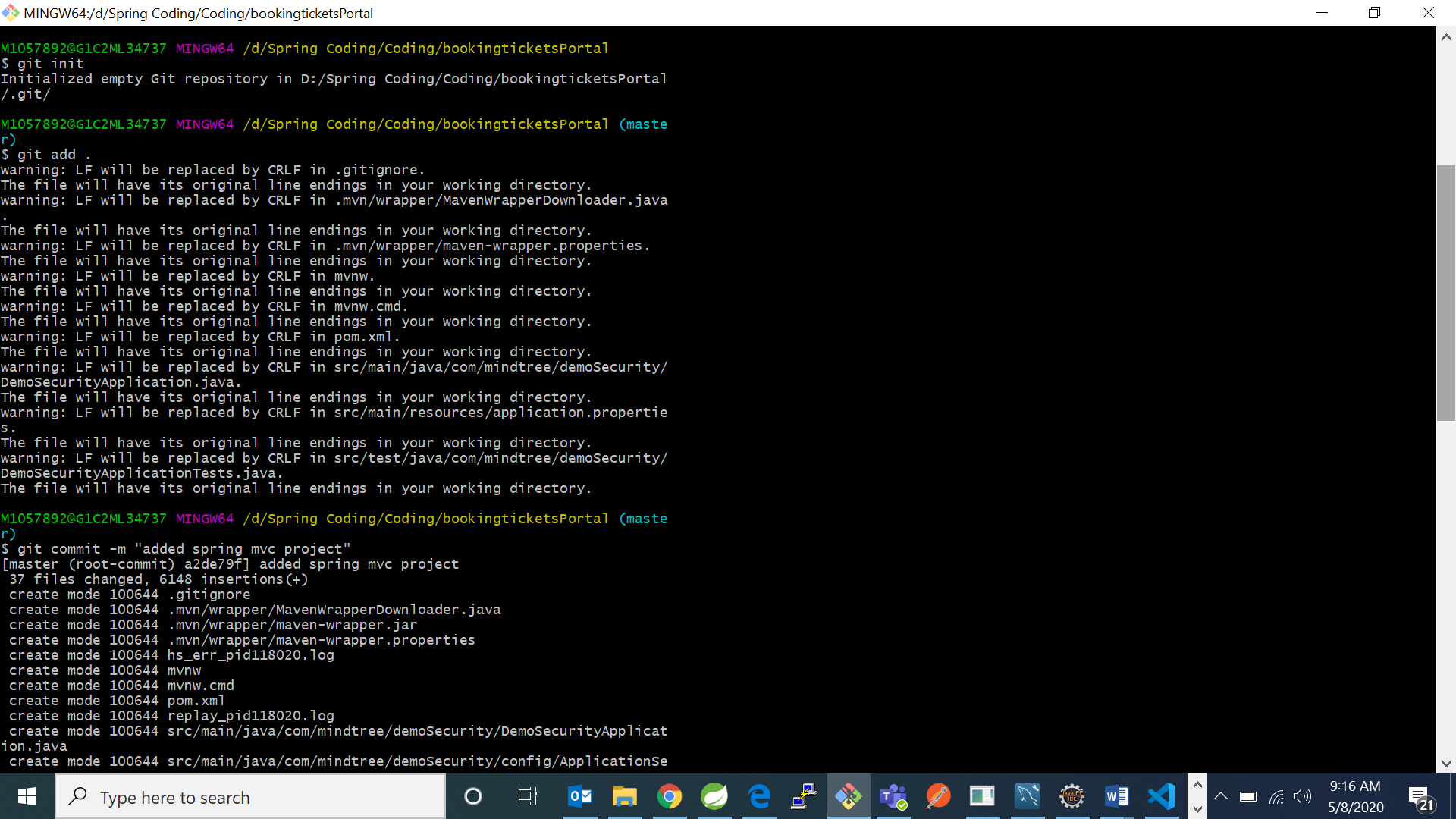
FINAL PROJECT OF DEVOPS



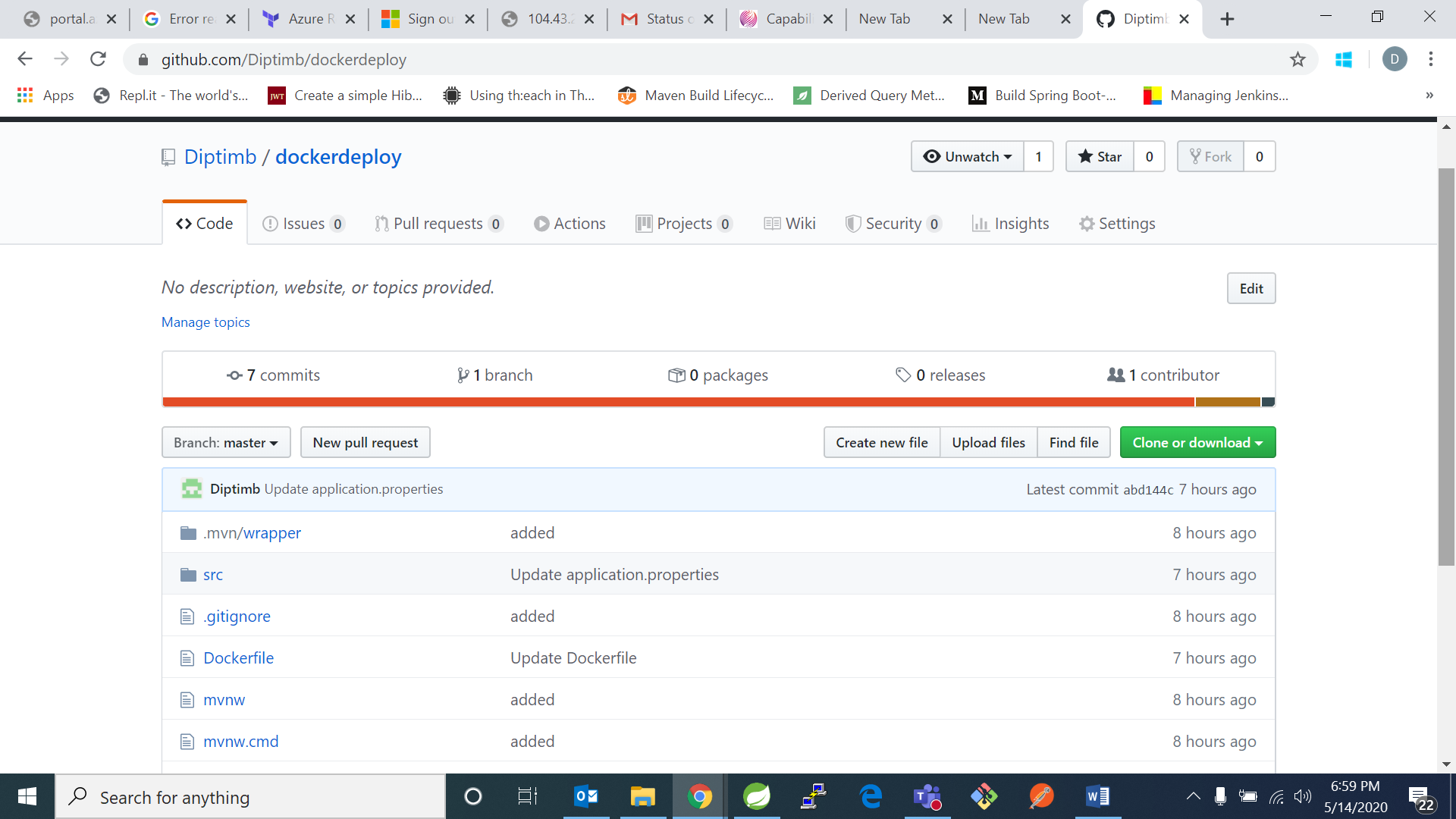
1. Create your spring project and push to git(your local repository).



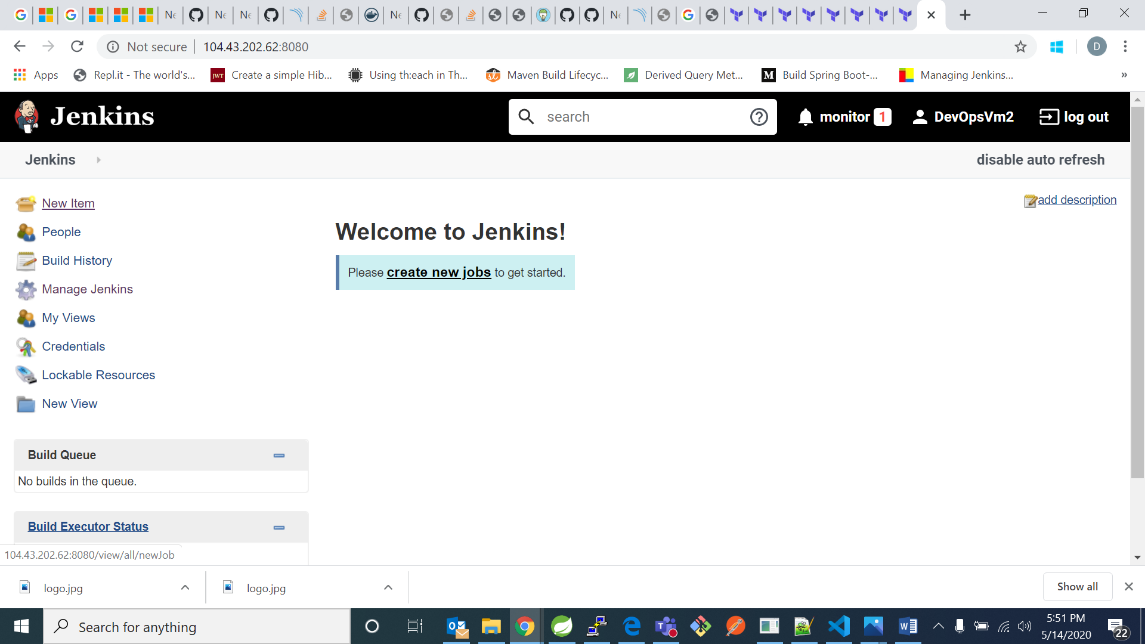
1. Push it from your local repository to your github repository 

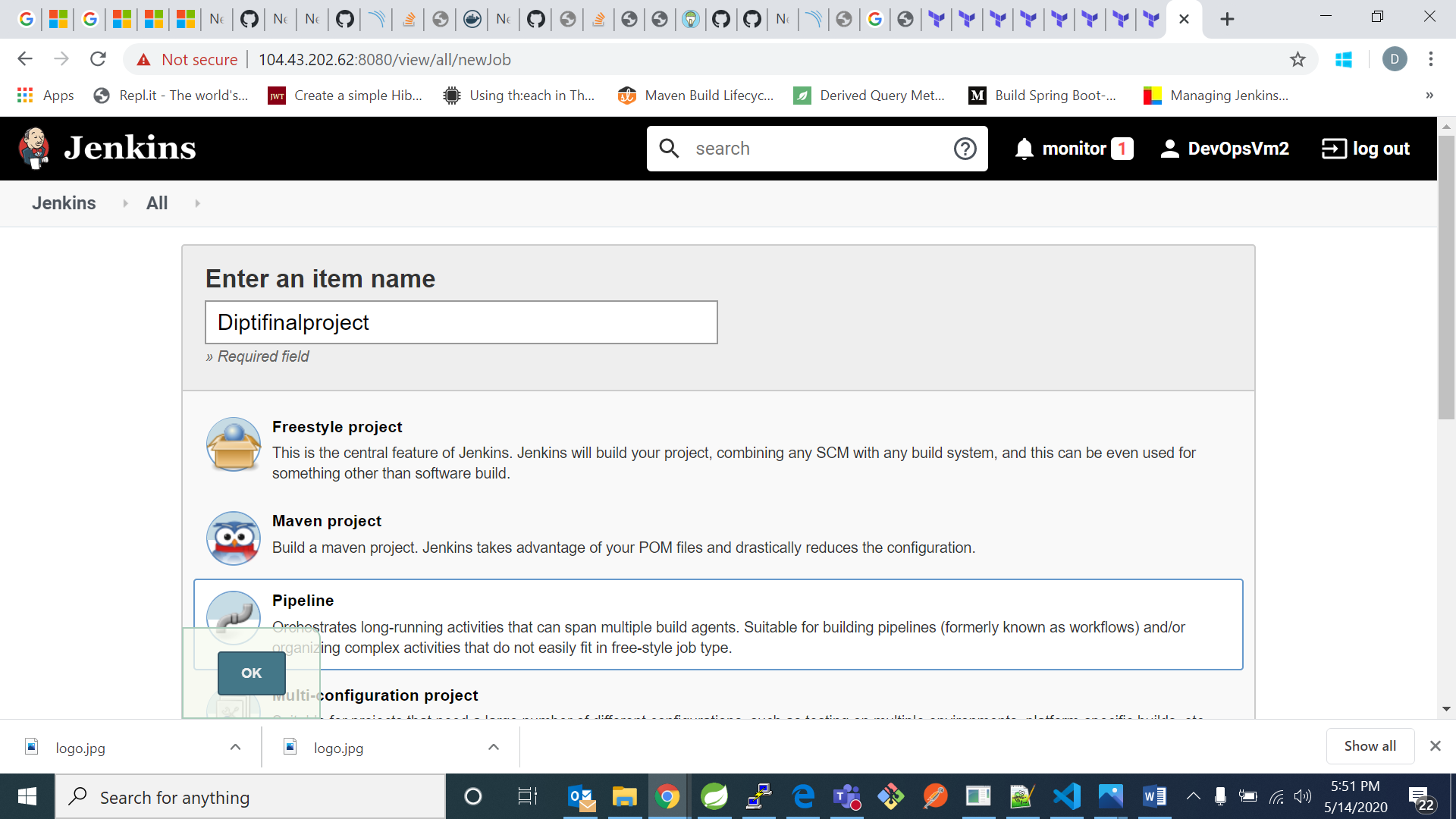
For this use commands:

1. “git init” –initializes git repository
2. “git add .” adds all the files to staging
3. “git commit –m “added project” – saves to local repository and adds comments while pushing to github repository
4. “git remote add origin [https://github.com/Diptimb/dockerdeploy.git”-adds](https://github.com/Diptimb/dockerdeploy.git) your GitHub repository url to git.
5. “git push origin master” – pushes all the files from local repository to your GitHub repository.
6. Go to git and check your project is pushed or not.



1. Now go to Jenkins and create a new pipeline project





1. Now go to pipeline section and add your pipeline:

pipeline{

agent any

tools{

maven 'Maven'

}

environment {

registry = "diptib/diptifinalproject"

registryCredential = 'dockerhubdipti'

dockerImage=''

}

stages{

stage('Source'){

steps{

git 'https://github.com/Diptimb/dockerdeploy.git'}

}

stage('Clean-Package-create-war-file'){

steps{

sh 'mvn clean package -Dmaven.test.skip=true'

}

}

stage('Sonar Analysis'){

steps{

script {

Pipeline\_Message=env.STAGE\_NAME

echo "Sonar Analysis"

}

dir("/var/lib/jenkins/workspace/Diptifinalproject")

{

withSonarQubeEnv('sonarqube')

{

sh 'mvn sonar:sonar'

}

}

}

}

stage('Quality Gate')

{

steps

{

script

{

timeout(time: 1, unit: 'HOURS')

{

def qg = waitForQualityGate()

if (qg.status != 'OK')

{

error "Pipeline aborted due to quality gate failure: ${qg.status}"

mail bcc: '', body: 'quality gate pass failed', cc: '', from: '', replyTo: '',

subject: 'Gate passing Failed', to: 'deepubebarta@gmail.com'

currentBuild.status='FAILURE'

}

}

}

}}

stage ('upload') {

steps{

script {

def server = Artifactory.server "1"

def buildInfo = Artifactory.newBuildInfo()

buildInfo.env.capture = true

buildInfo.env.collect()

def uploadSpec = """{

"files": [ {

"pattern": "\*\*/target/\*.war",

"target": "libs-snapshot-local"}]

}"""

server.upload spec: uploadSpec, buildInfo: buildInfo

server.publishBuildInfo buildInfo

}}}

stage('Build image') {

steps {

echo 'Starting to build docker image'

script {

dockerImage= docker.build registry + ":$BUILD\_NUMBER"

}

}

}

stage('Push Image') {

steps{

script {

sh 'docker login -u "diptib" -p "123456789" docker.io && docker push diptib/diptifinalproject'

docker.withRegistry( '', registryCredential ) {

dockerImage.push()

}

}

}

}

stage ('Deploy')

{

steps{

script{

sh 'docker container run -i --publish 8081:8081 diptib/diptifinalproject:$BUILD\_NUMBER'

/\*sshagent(['dev-server']) {

sh "ssh -o StrictHostKeyChecking=no DevOpsVm2@104.43.202.62:$dockerRun"

}\*/

}

}}

-,

steps{

sh "docker rmi $registry:$BUILD\_NUMBER"

}

}

/\*stage('Deploy'){

steps{

sh "cp /var/lib/jenkins/workspace/Diptifinalproject/target/\*.war /var/lib/tomcat8/webapps"}

}\*/

}

post {

always {

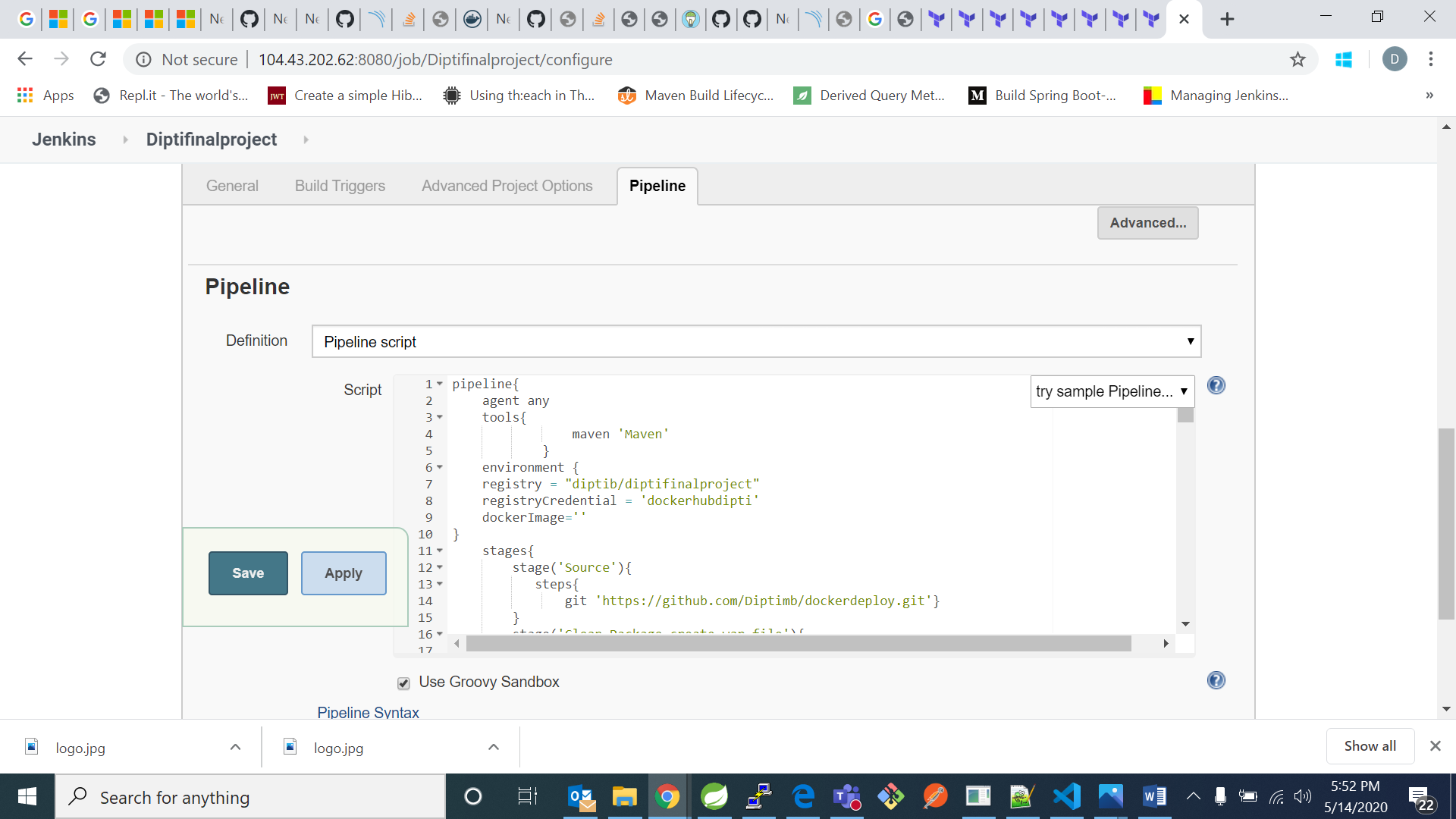
mail to: 'deepubebarta@gmail.com',

subject: "Status of pipeline: ${currentBuild.fullDisplayName}",

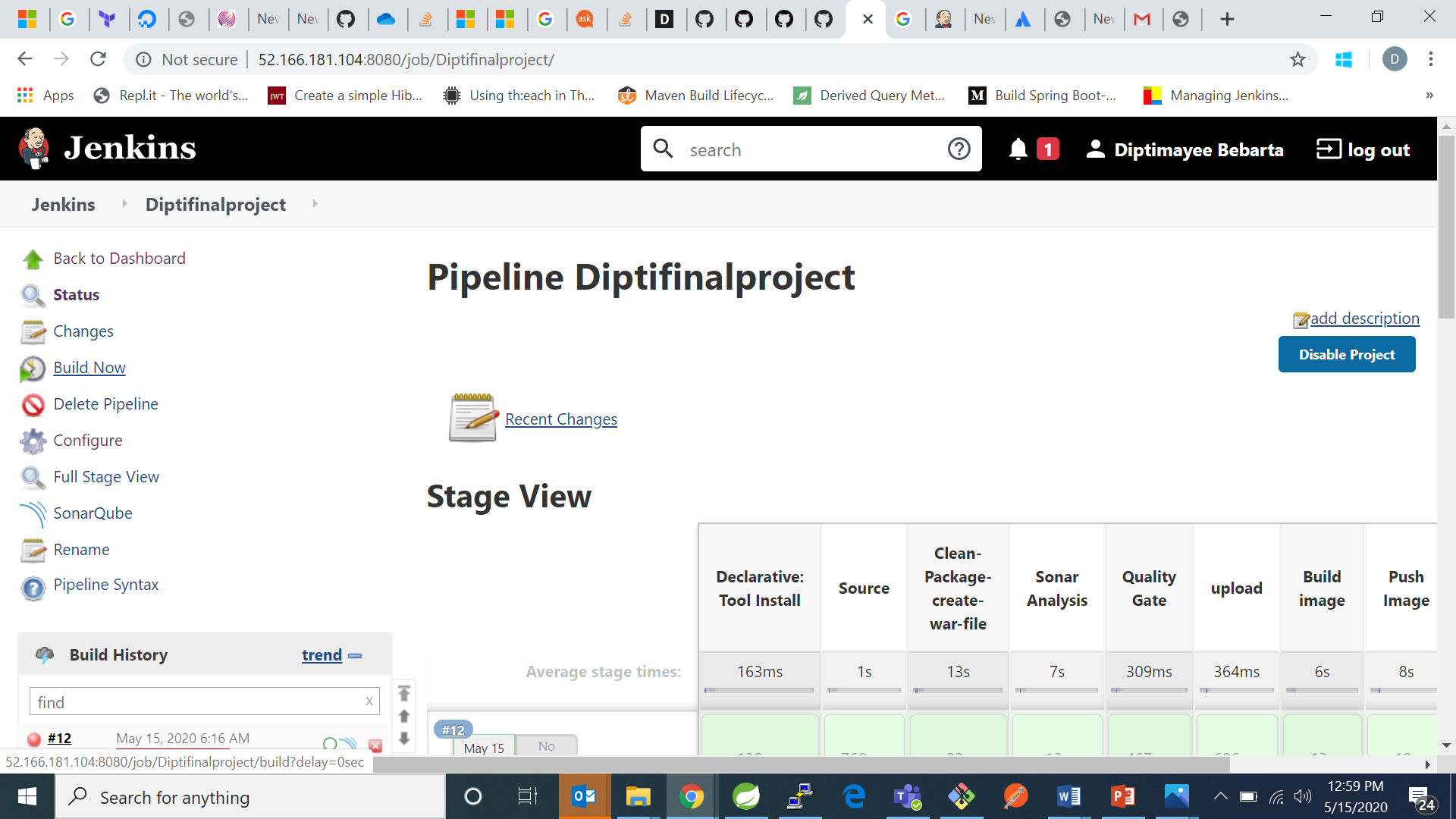
body: "${env.BUILD\_URL} has result ${currentBuild.result}"

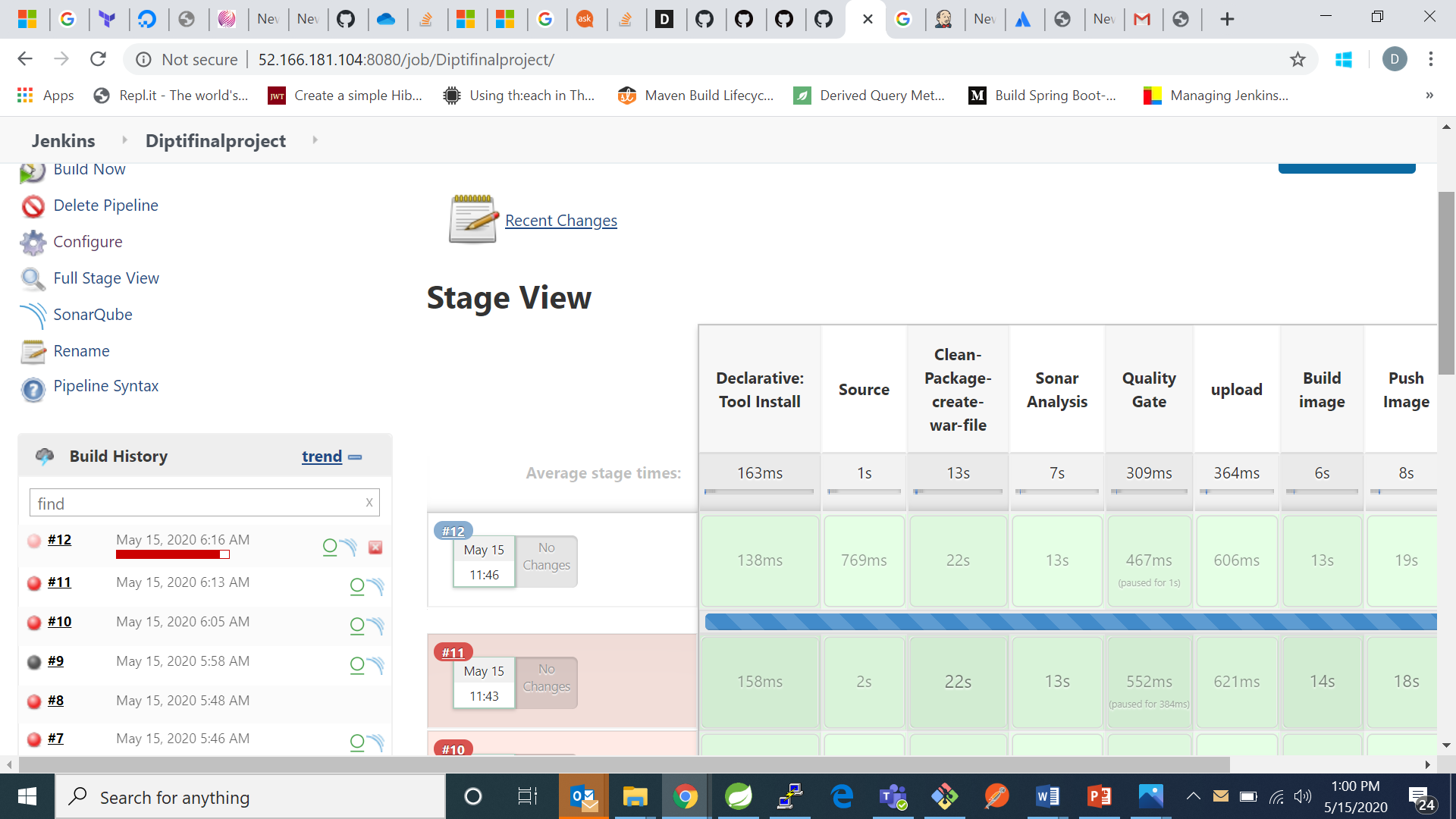
}

}

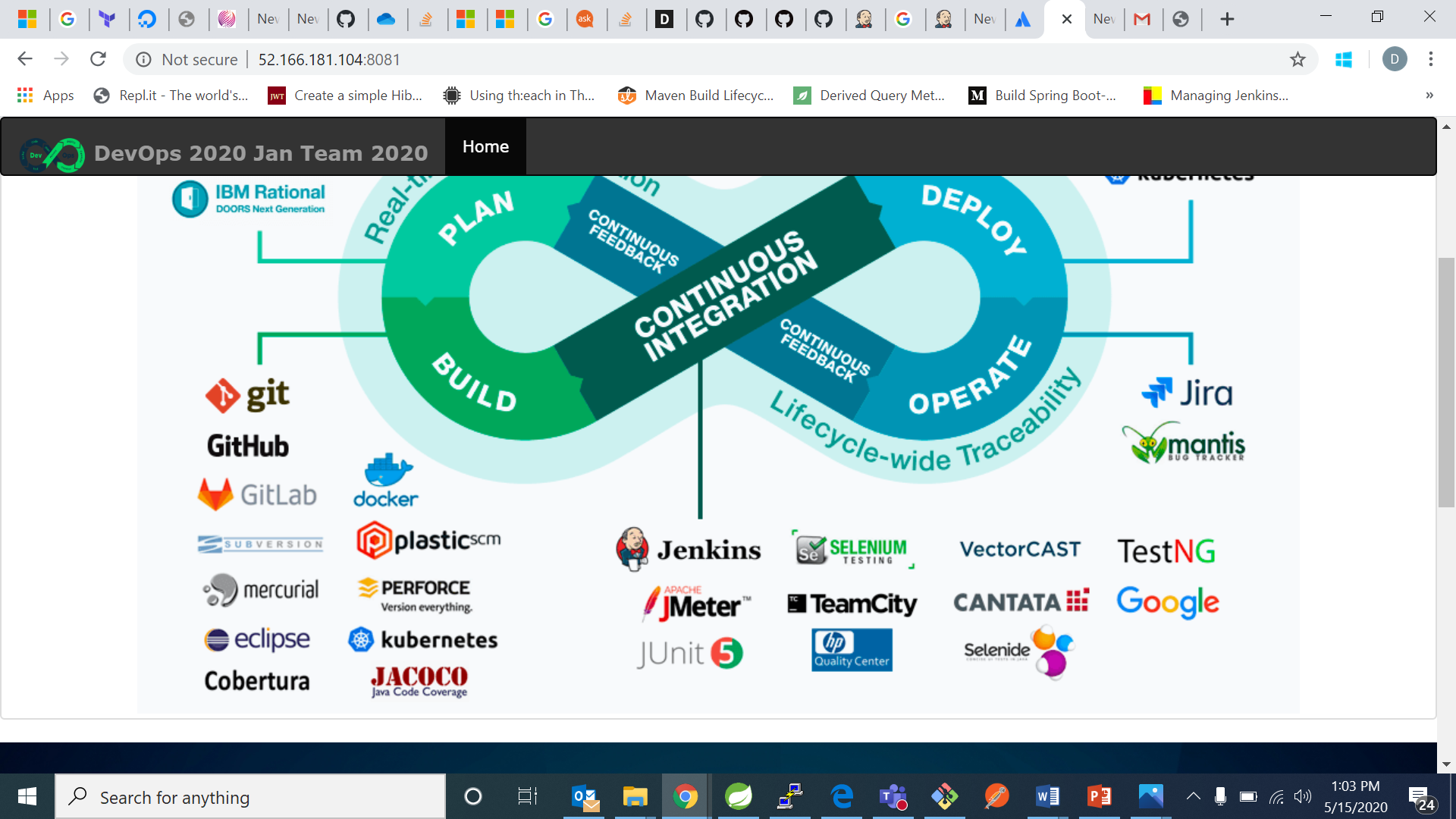


1. Click on save and apply.
2. Now in your project click on build now



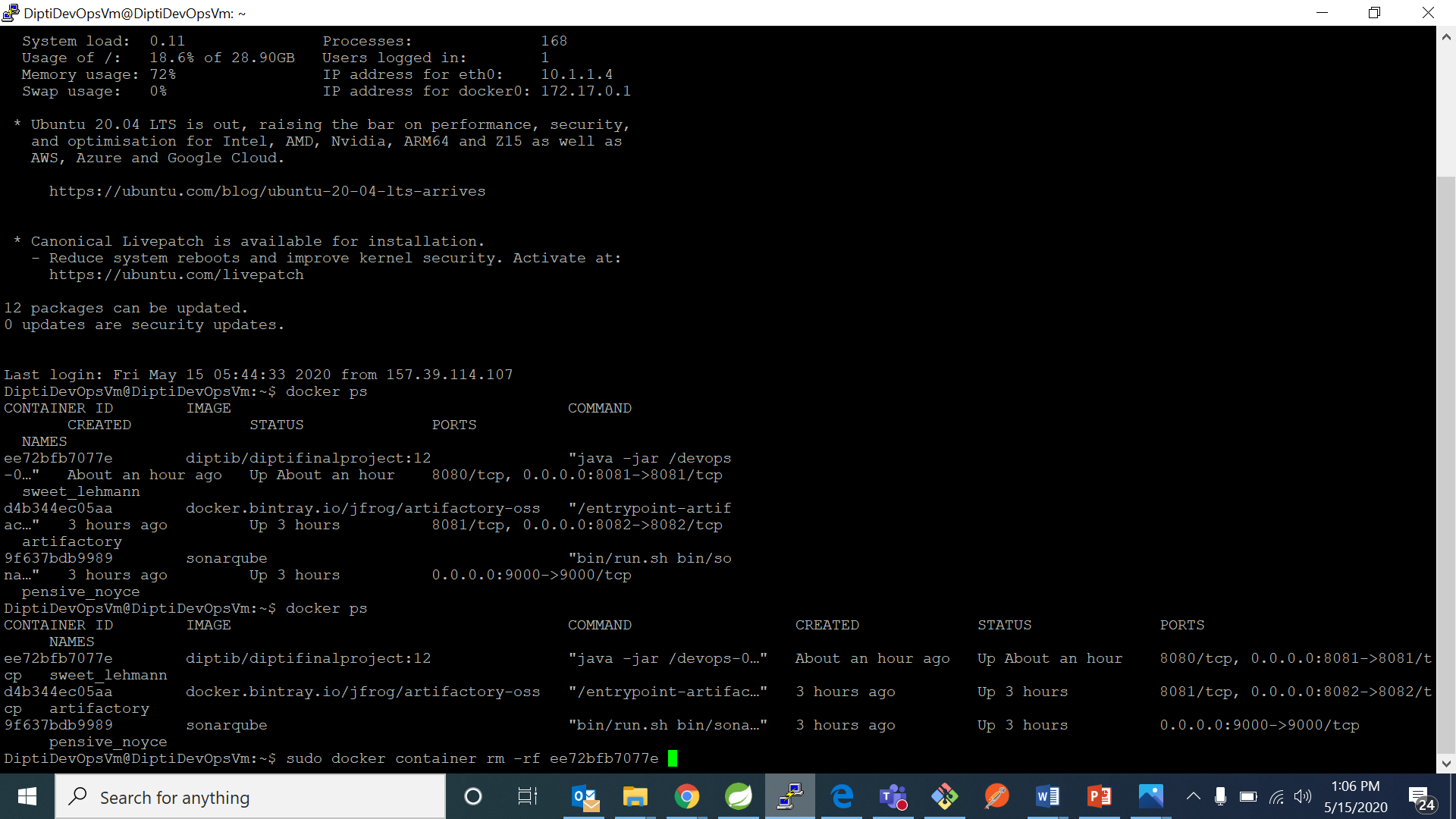


1. Now go to tomcat using your ip address and the port number you have mentioned in your pipeline.



My project has been successfully deployed in tomcat.

1. Now for stopping the project to run we will check for the container id in our vm using command “docker ps”



1. Now run the command “sudo docker container rm –f “container id” “ . It will stop your project container to run.

