**Building CI/CD Pipeline in Openshift Cluster using Jenkins**

**Step 1:** Created one spring boot application using maven build tool and pushed it to github repository. Also created one Jenkinsfile to setup the CI/CD pipeline.

Github URL: [*https://github.com/sibsankarb4/springboot-openshift-jenkins-CICD-app.git*](https://github.com/sibsankarb4/springboot-openshift-jenkins-CICD-app.git)

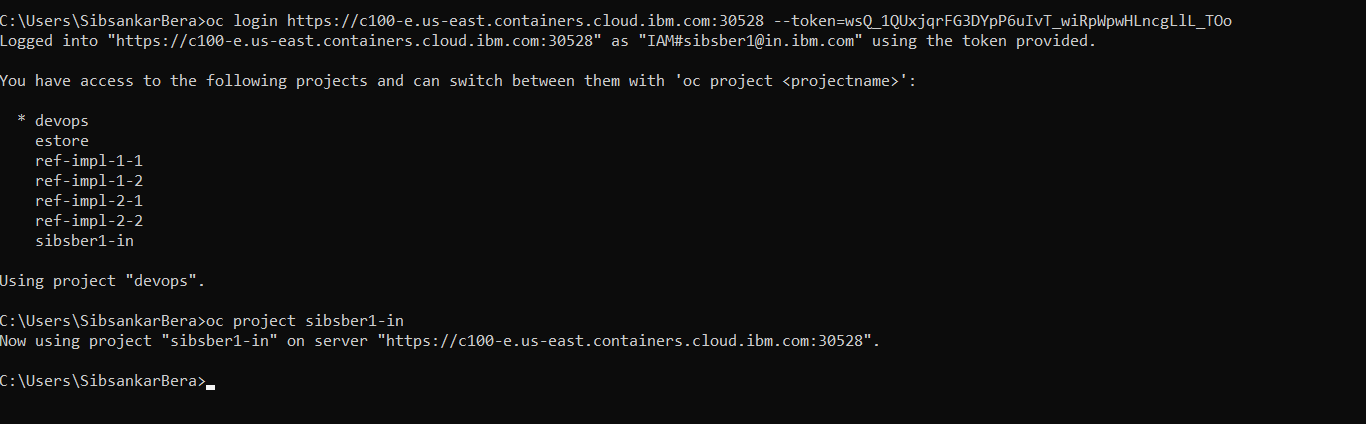
Jenkins file URL: [*https://github.com/sibsankarb4/springboot-openshift-jenkins-CICD-app/blob/master/Jenkinsfile*](https://github.com/sibsankarb4/springboot-openshift-jenkins-CICD-app/blob/master/Jenkinsfile)

**Step 2:** This Spring boot rest api application is to create Jenkins CI/CD pipeline using jenkins file and deploy the corresponding application in openshift cluster pod. We have created one Jenkins pipeline file inside this application. This pipeline will move/deploy the application from one environment/stages to another environment/stages. How the flow will go, that is given below,

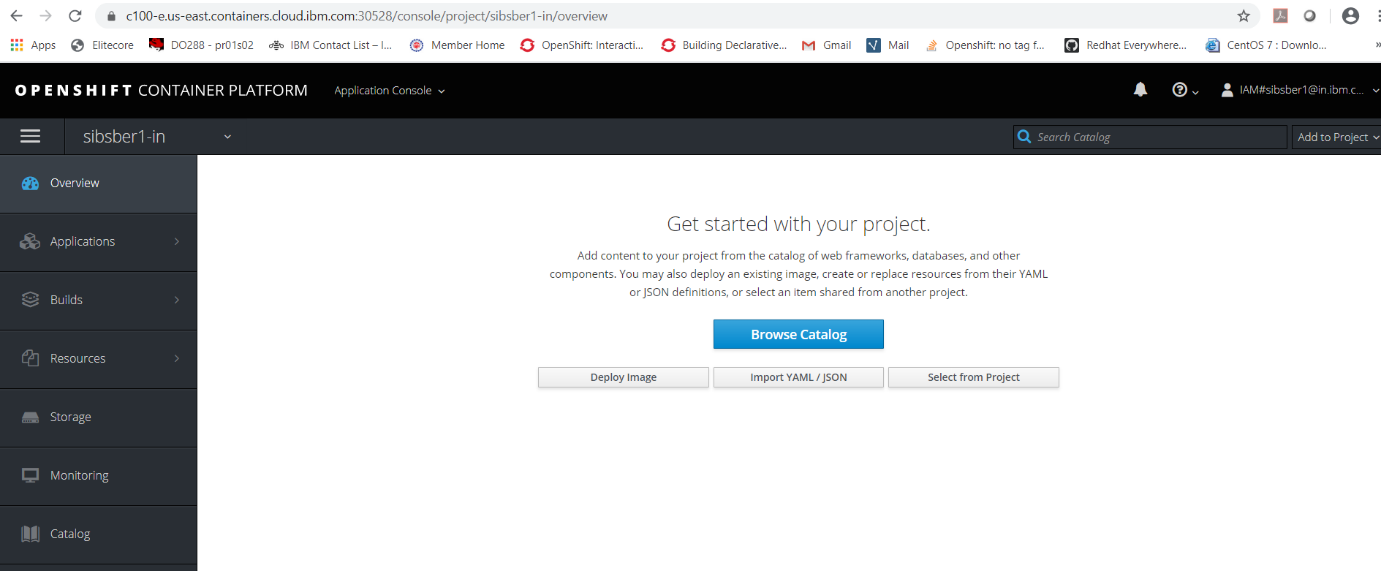
Stages: **Dev >> UAT >> PROD**

**Step 3:** We have exposed some api from this application, so that we can test it after deployment in corresponding environment. And we are going to deploy it is in WildFly, it can also be deployed in JBoss EAP or Redhat OpenJDK8 as well of openshift cluster.

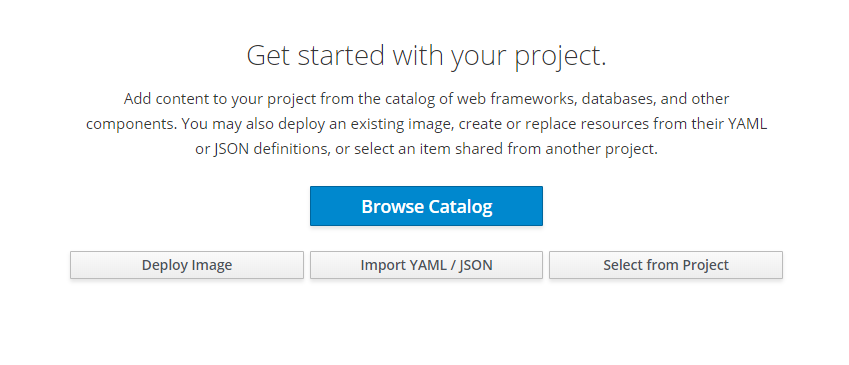
**Step 4:** Now we are logging into openshift cluster and using project “sibsber1-in”, as we have already created the project.



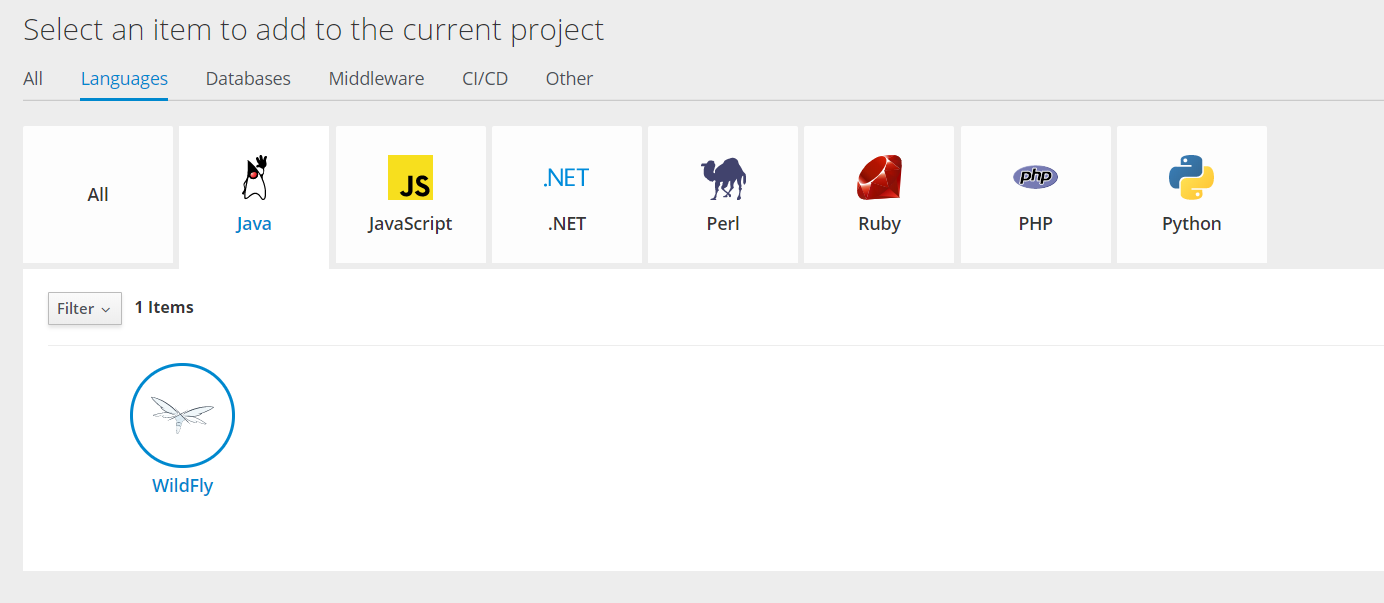
**Step 5:** Lets login into openshift GUI from browser and select the same project “sibsber1-in”. This project is empty now as we have just created this project.



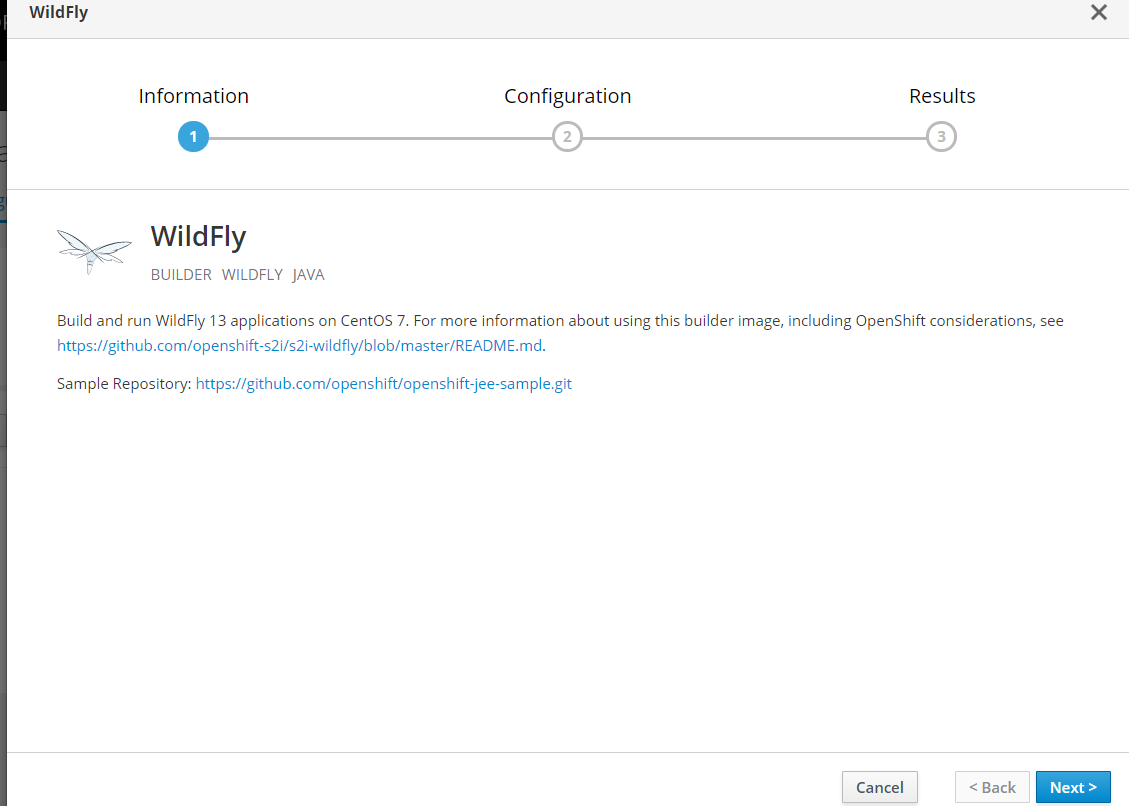
**Step 6:** Now click on the “Browse Catalog” button.



**Step 7:** Choose the WildFly option from Language >> Java catalogue option.



**Step 8:** Click on **WildFly** icon, it will launch one deployment wizard popup screen.

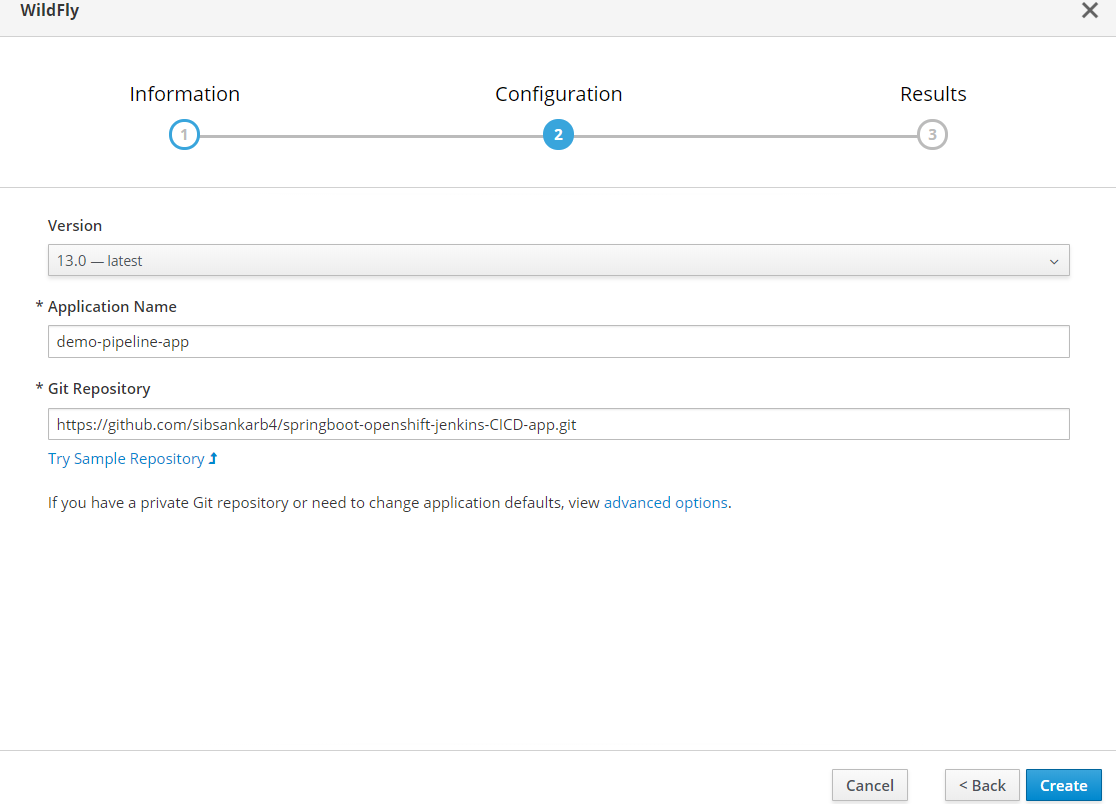


**Step 9:** Now click on **Next** button at bottom right corner of the page. It will open one application details form to fill up.

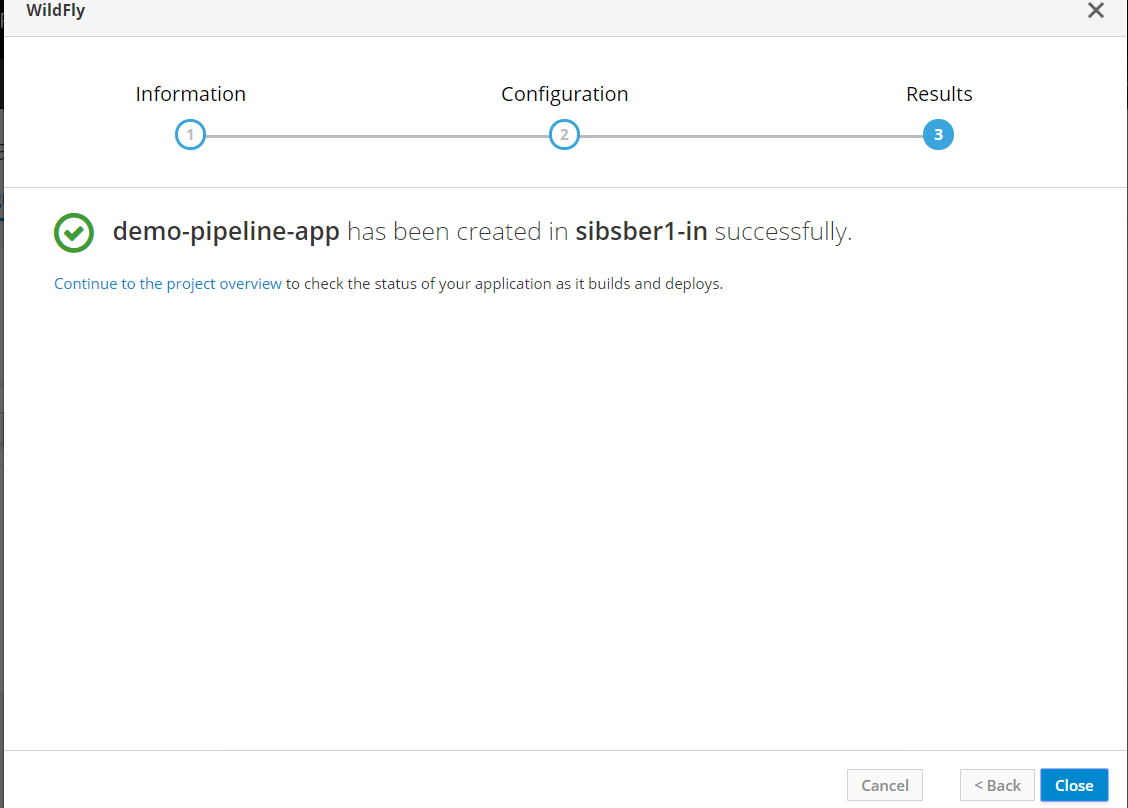
**Version**: 13.0 – latest (it’s a dropdown option, you can choose any other compatible older version as well if you want)

**Application Name:** demo-pipeline-app (you can give any other application name also)

**Git Repository**: https://github.com/sibsankarb4/springboot-openshift-jenkins-CICD-app.git (giving our previous git repository url)

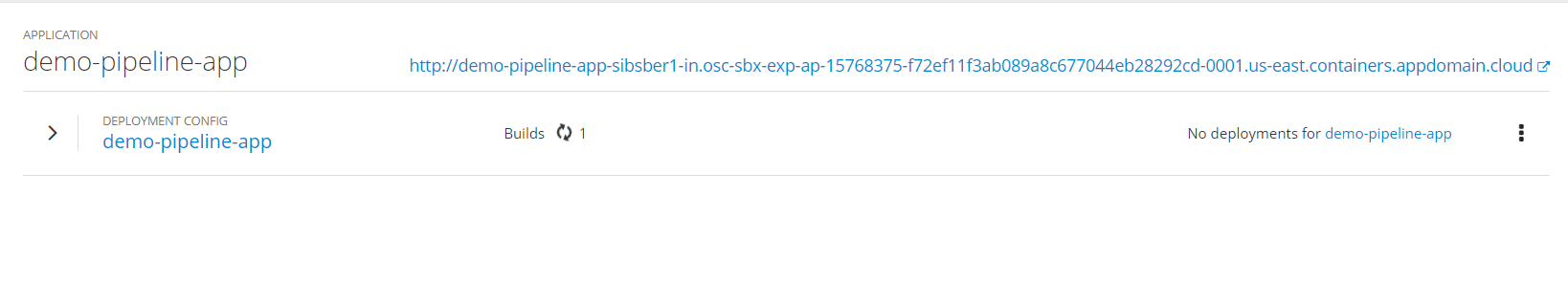


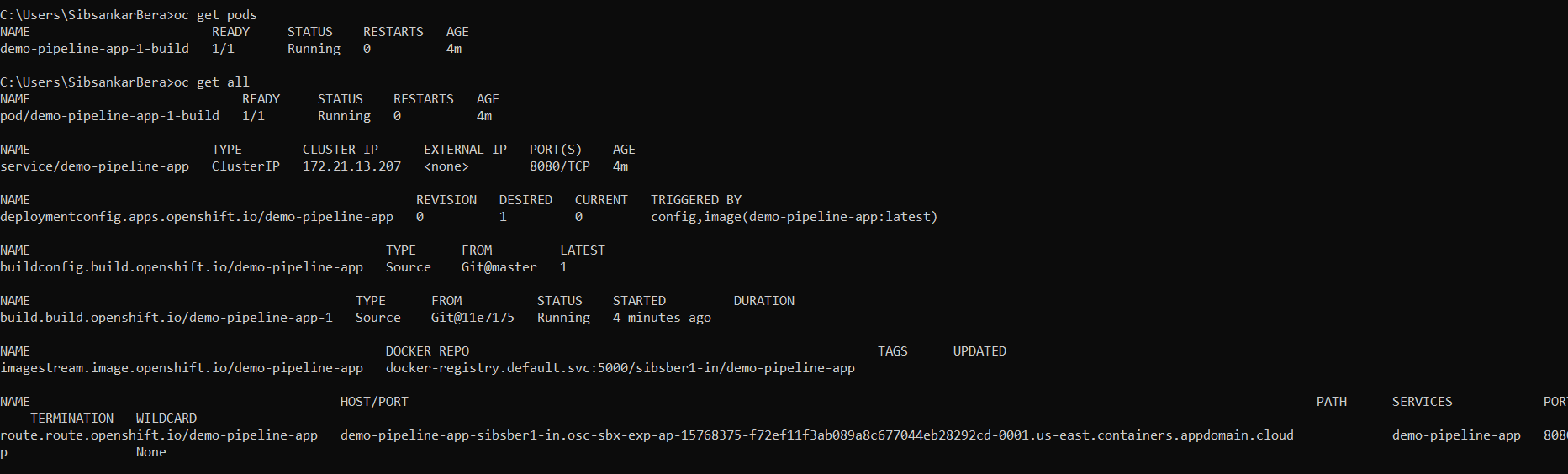
**Step 10:** After filling up all those above information click on the **Create** button at bottom left corner. It will show that our application is created successfully or not.



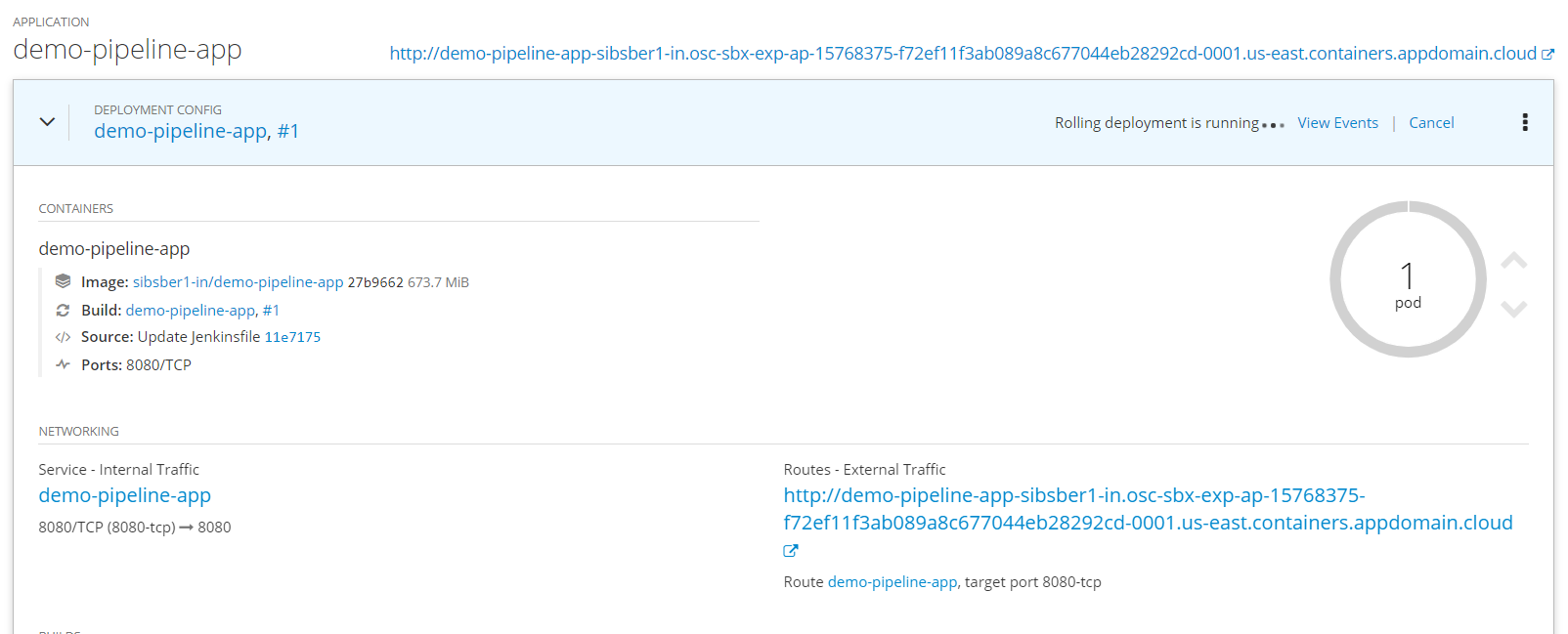
We are now able to create our application successfully in openshift cluster.

**Step 11:** Now click on the “[Continue to the project overview](https://c100-e.us-east.containers.cloud.ibm.com:30528/console/project/sibsber1-in/overview) ” link, now we can see that our build has started.

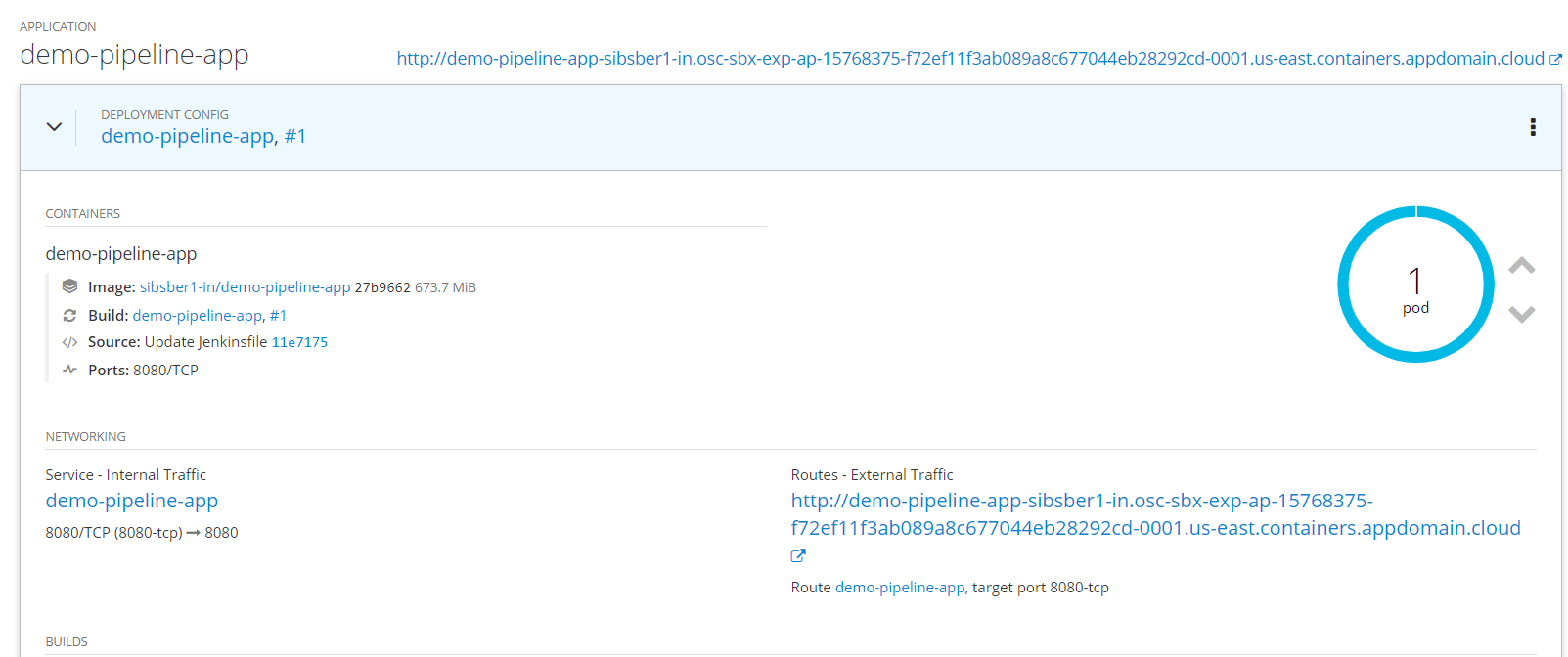




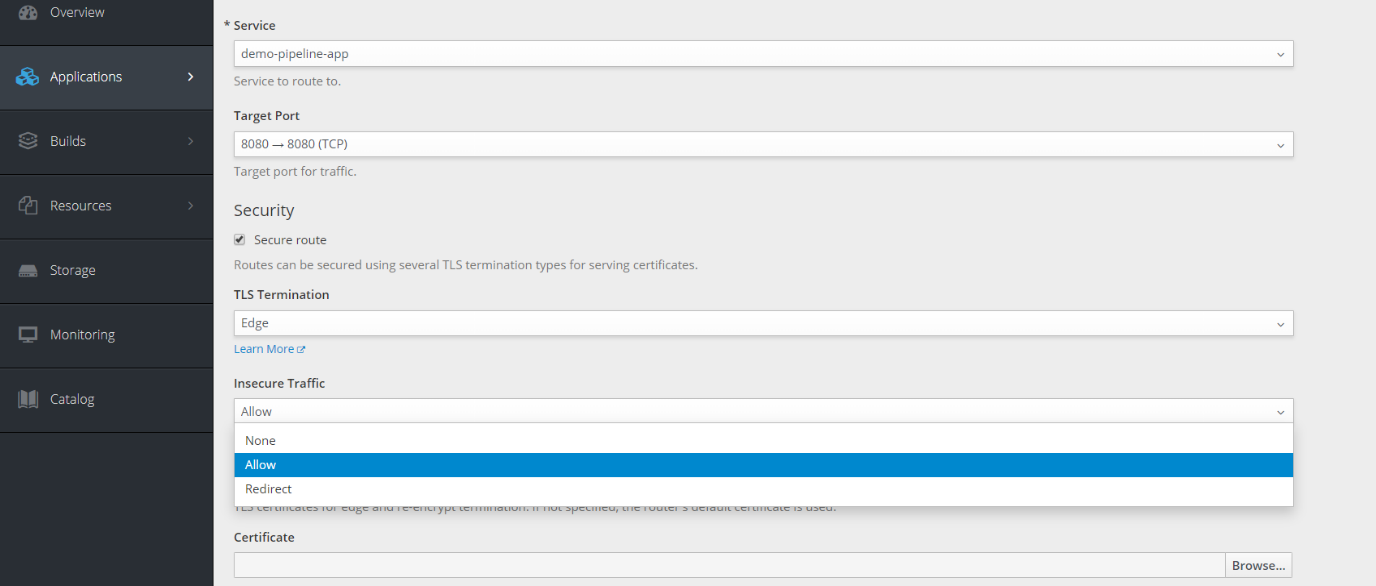
**Step 12:** Pod creation in process now

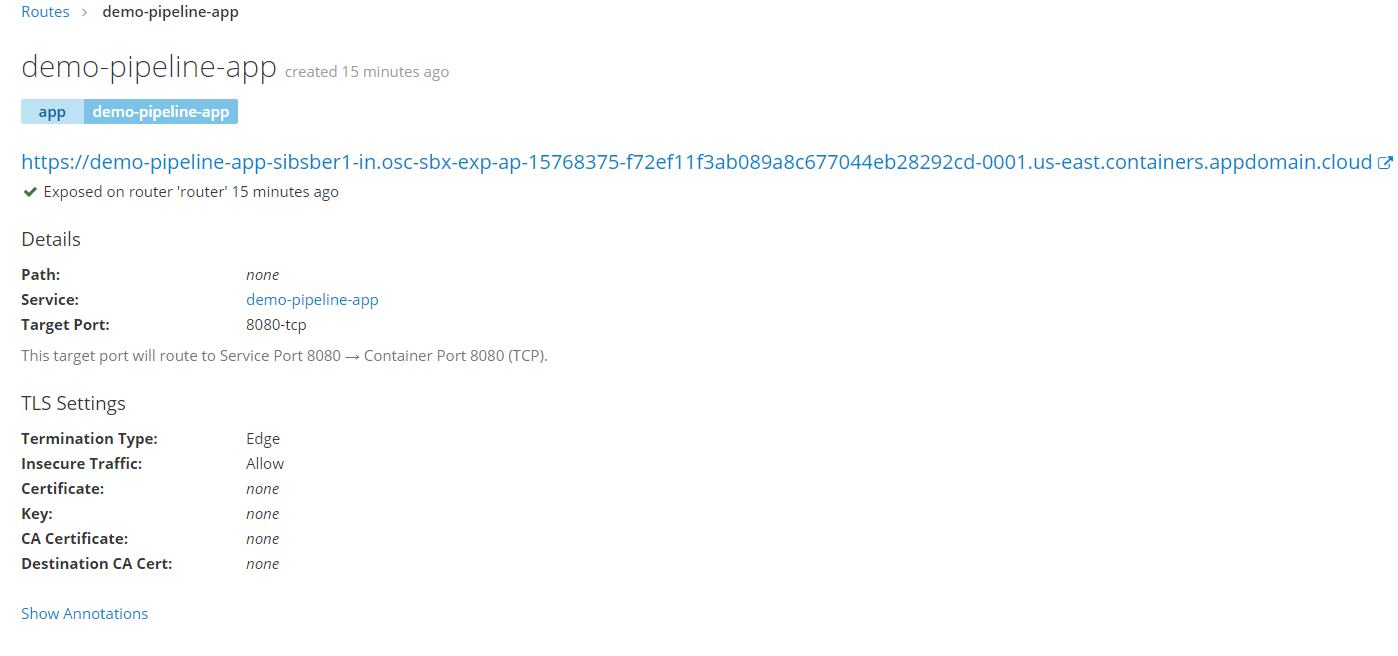


**Step 13:** Pod is up and running now



**Step 14:** If required make the route as secure route, by selecting Allow option from Insecure Traffic drop down. After that save the changes.

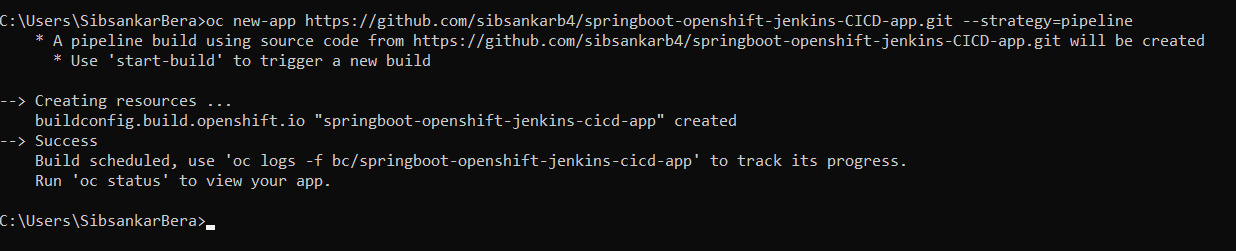




Now you can access the above url to see the output.

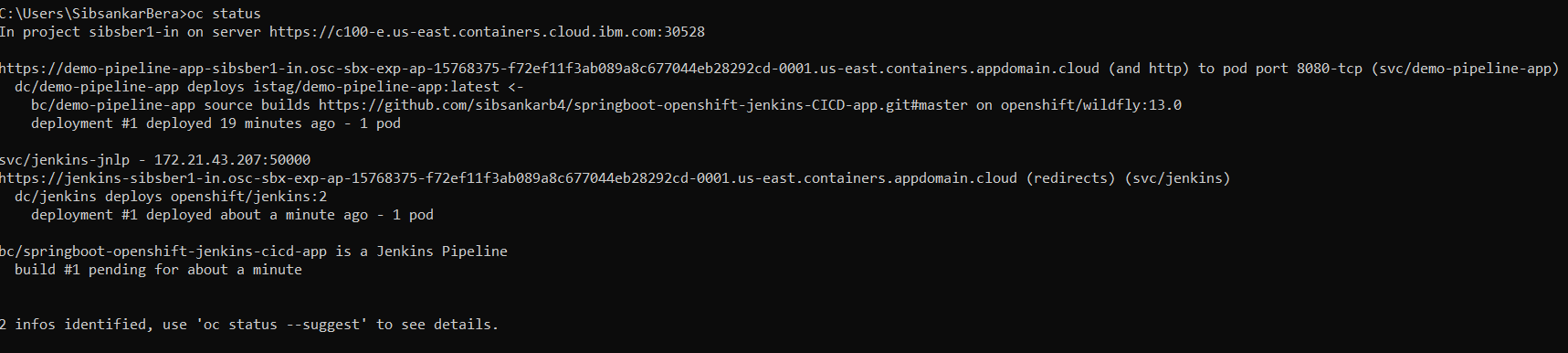
**Step 15:**  We can do the same deployment from openshift CLI (which is more convenient, more effective and recommended) by running below command.

*$ oc new-app https://github.com/sibsankarb4/springboot-openshift-jenkins-CICD-app.git -- strategy=pipeline*

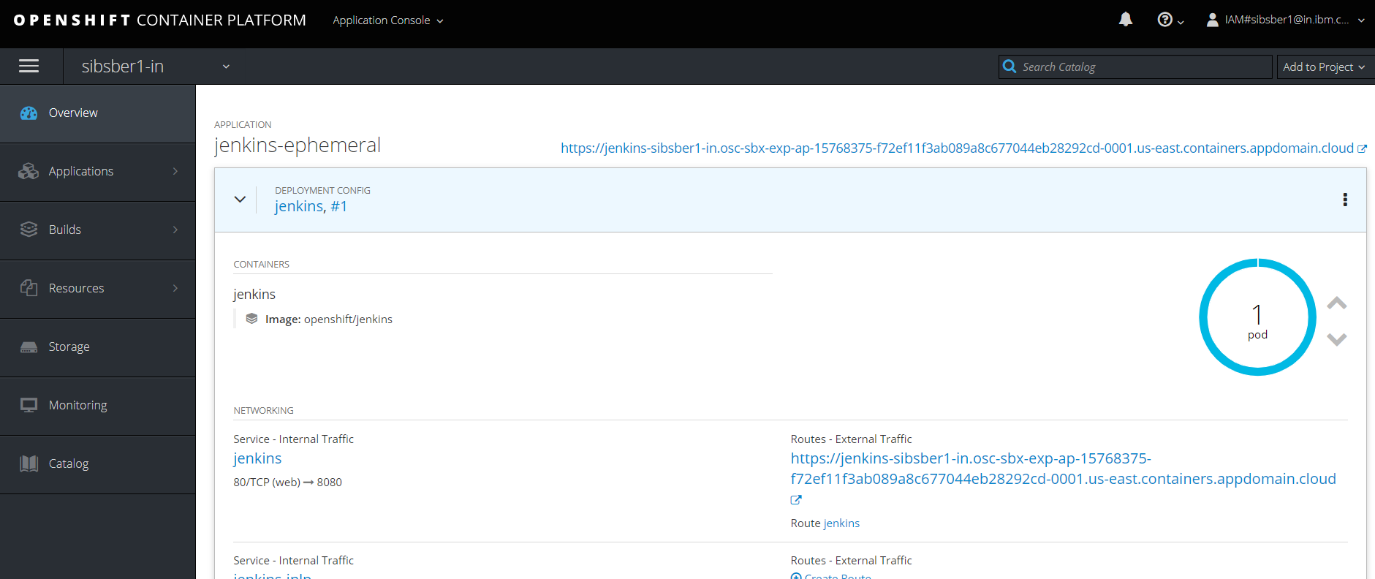


To check the deployment status use below command

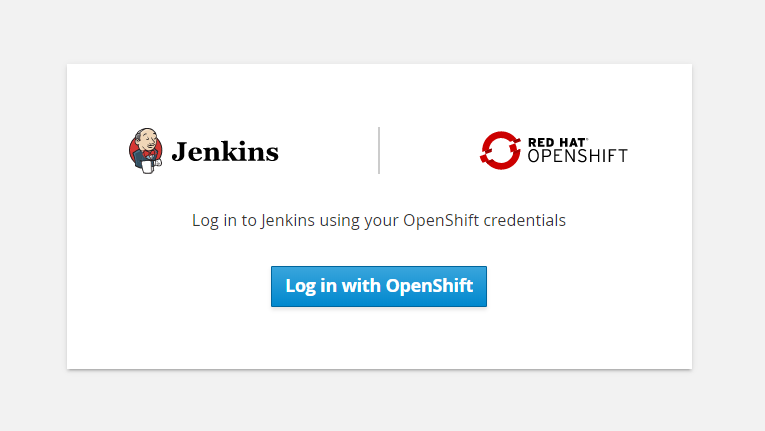
*oc status*



**Step 16:** Now you can see that one application created with name jenkins-ephemeral and pod is also up and running.



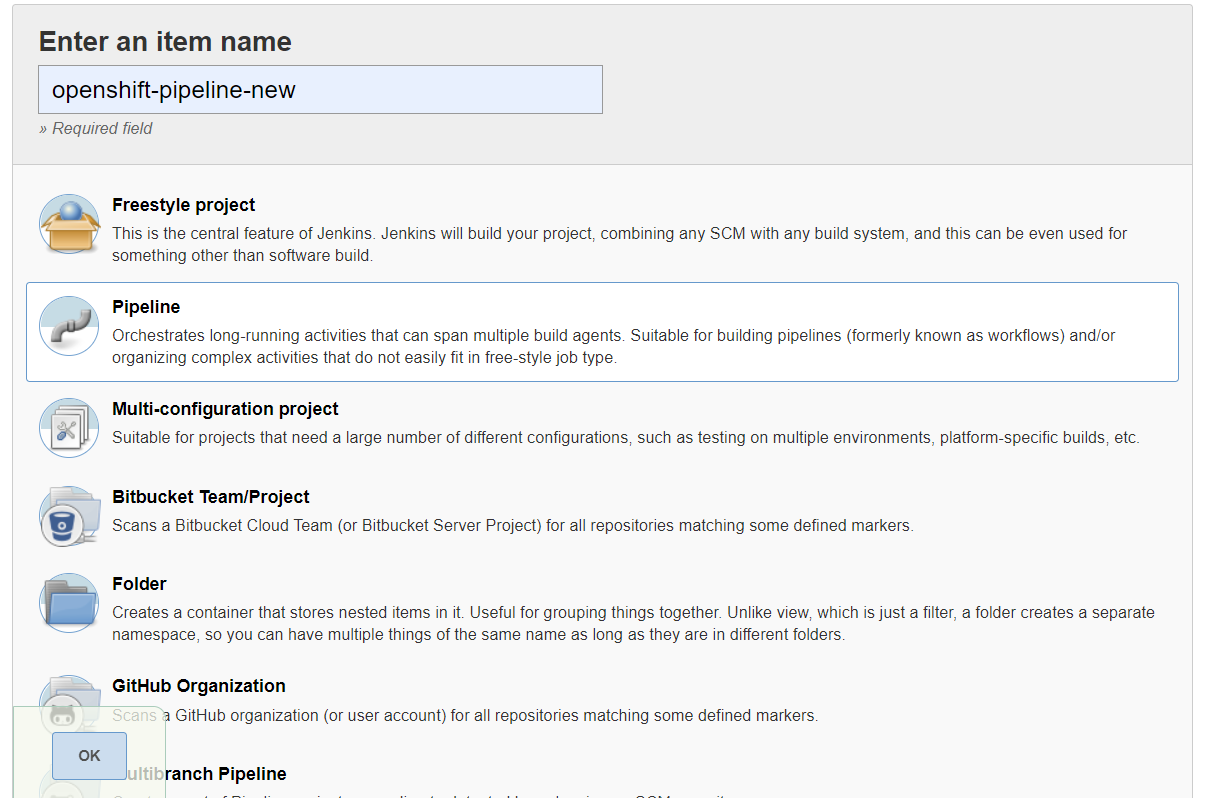
**Step 17:** Click on the above expose url, it will open below page



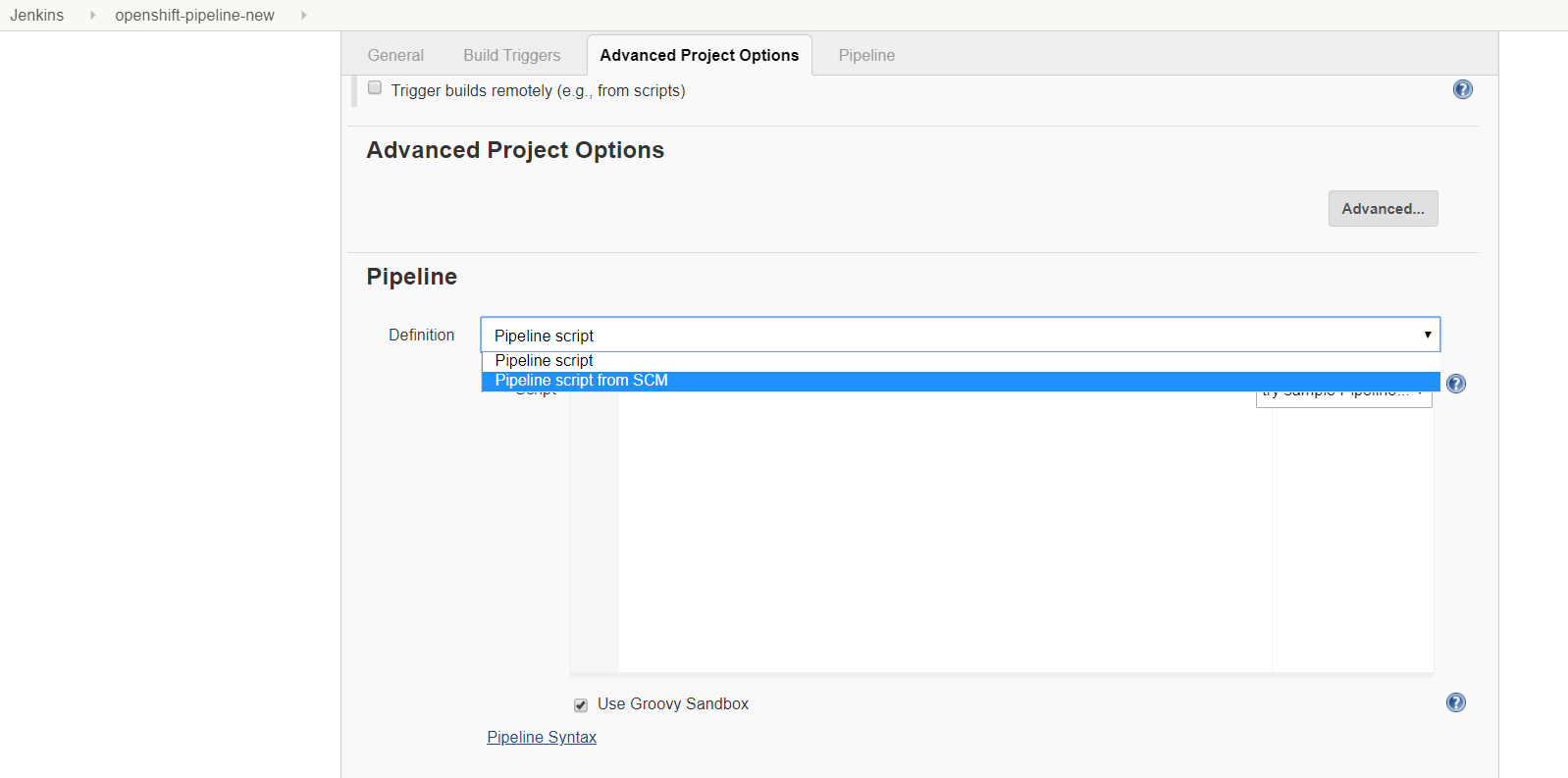
**Step 18:** Click on the “Log in with OpenShift” button. It will open Jenkins dashboard page.



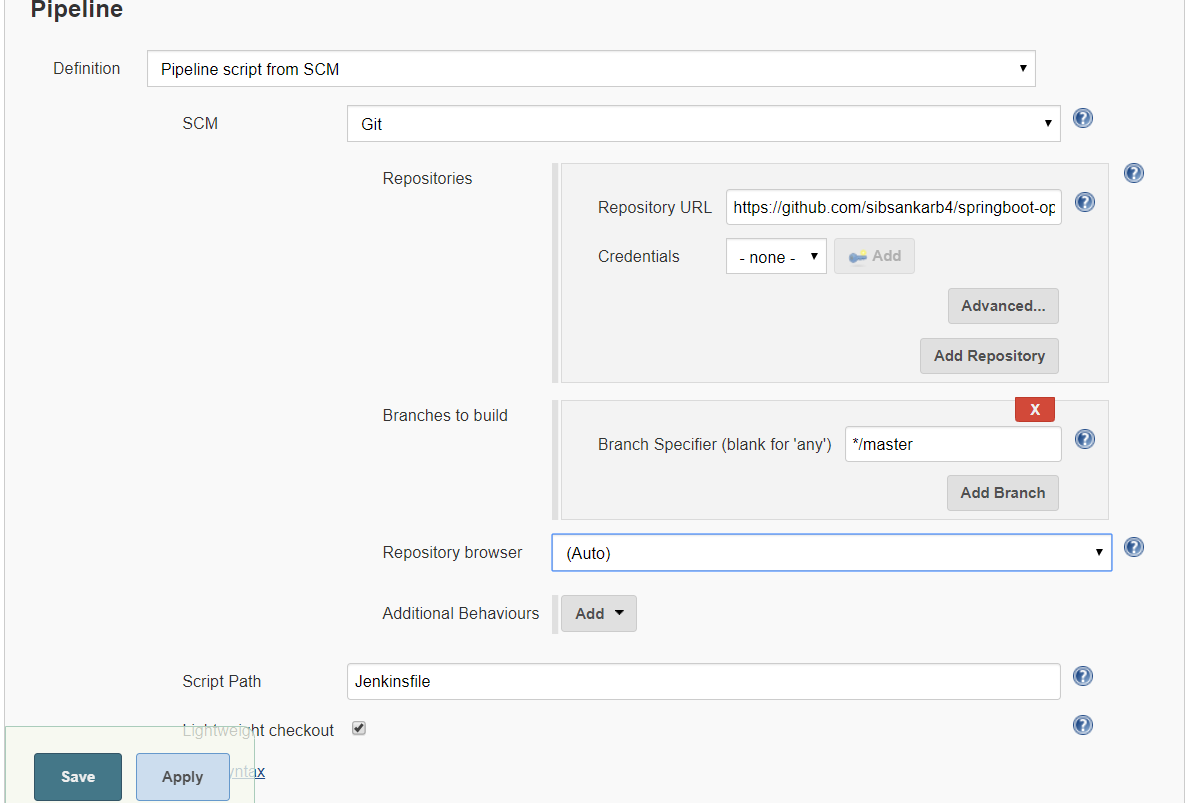
**Step 19:** Now create a New Item (button available on the top left corner of the Jenkins dashboard page) with name “openshift-pipeline-new” and select pipeline option, after that click on OK button.



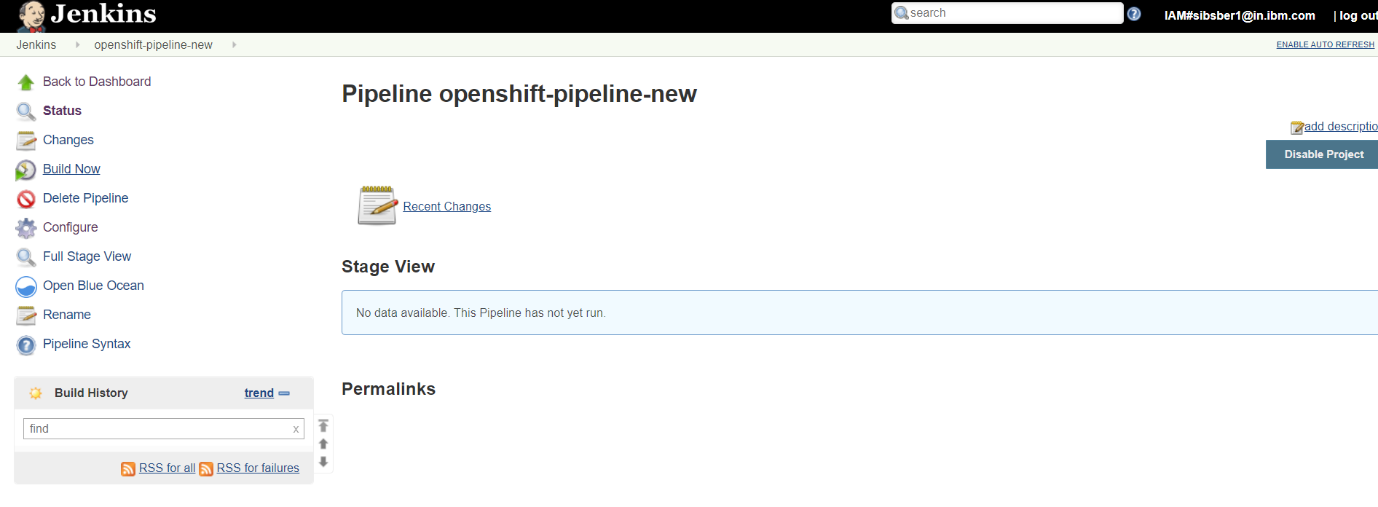
**Step 20:** In the next page select “Pipeline script from SCM” under Pipeline header.



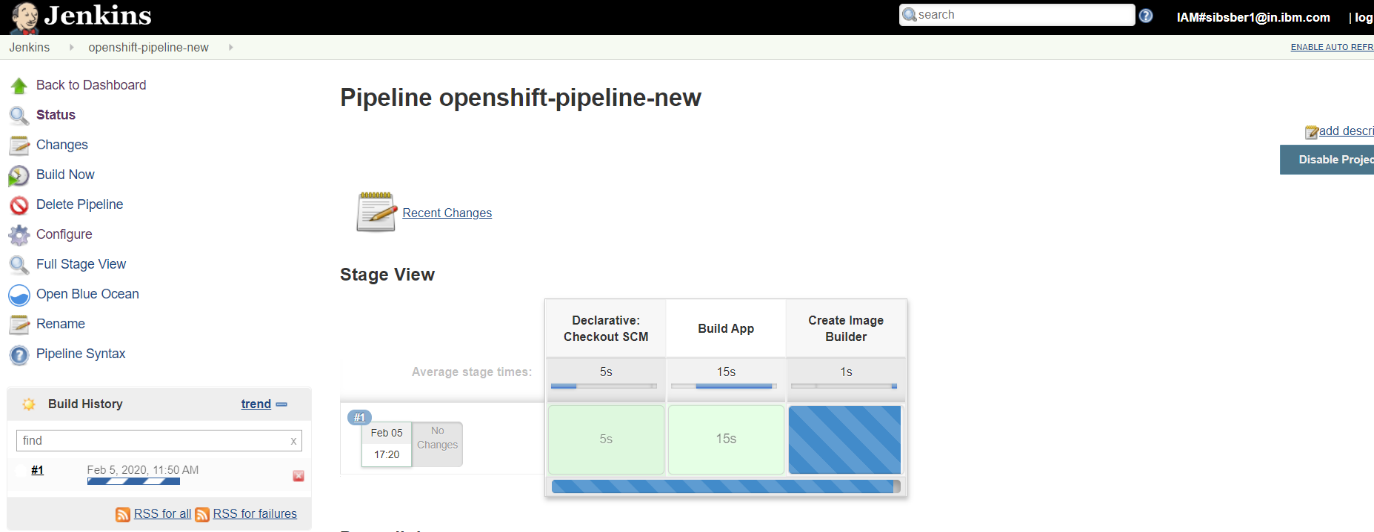
Then select SCM as Git and after that putting our application git url and git credentials if required. They click on Save button.



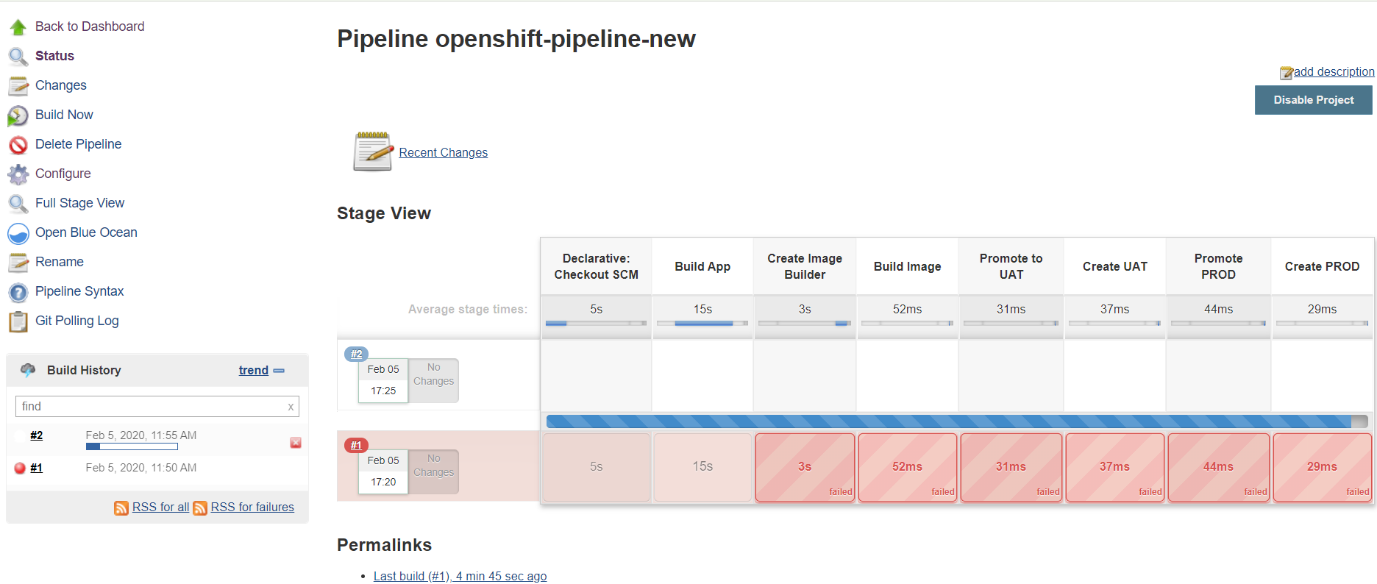
**Step 21:** Now click on Build Now option link from left panel. It will initiate the build and deployment pipeline.



**Step 22:** Pipeline has started now

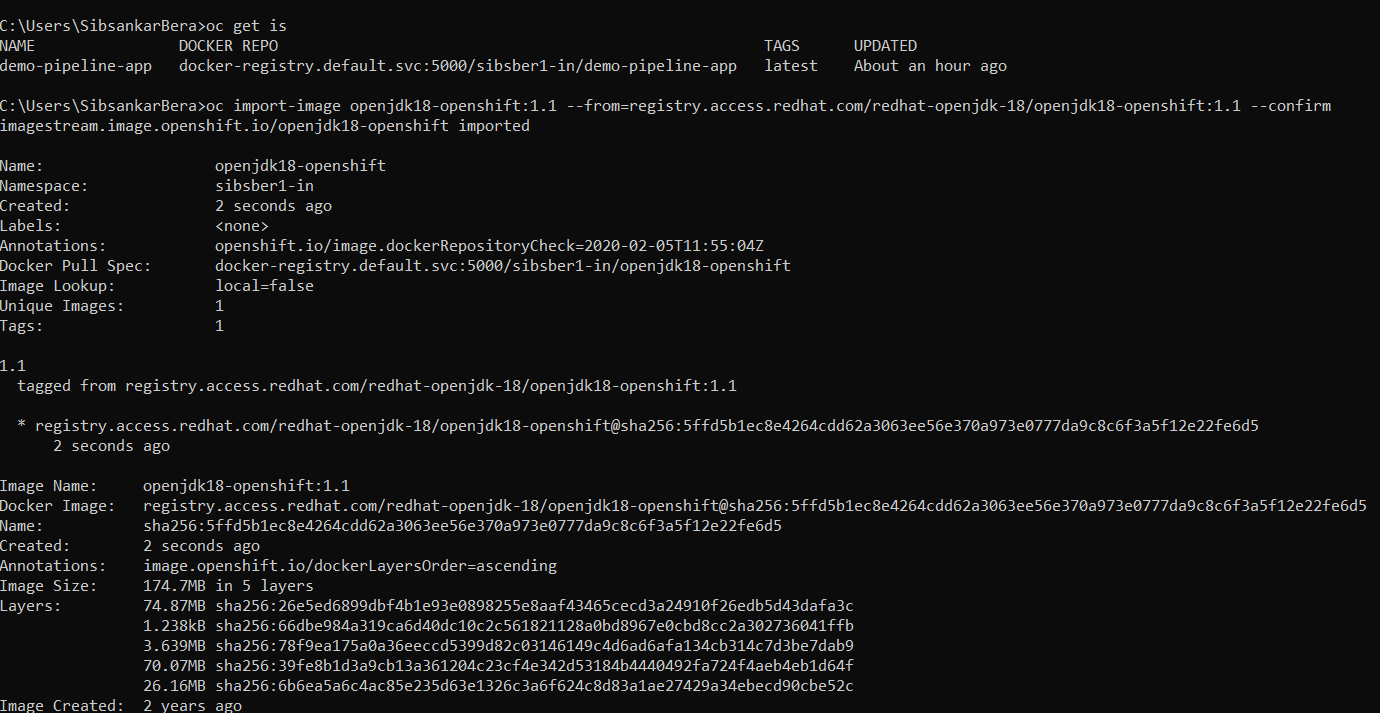


**Step 23:** First time, pipeline has been failed because we have not imported the images-stream (openjdk18-openshift:1.1) in our current project. And we are using this same image-stream in git Jenkinsfile to build/deploy the application source code.

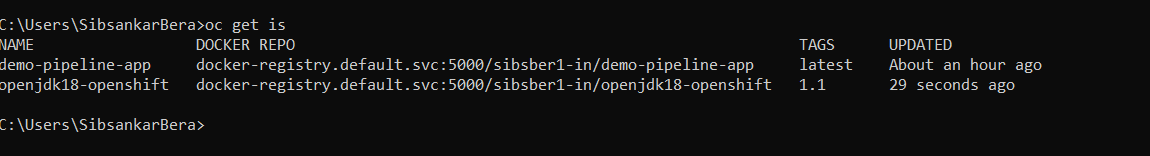
****

**Step 24:** To fix this issue, use below command to import the image from registry in our current project.

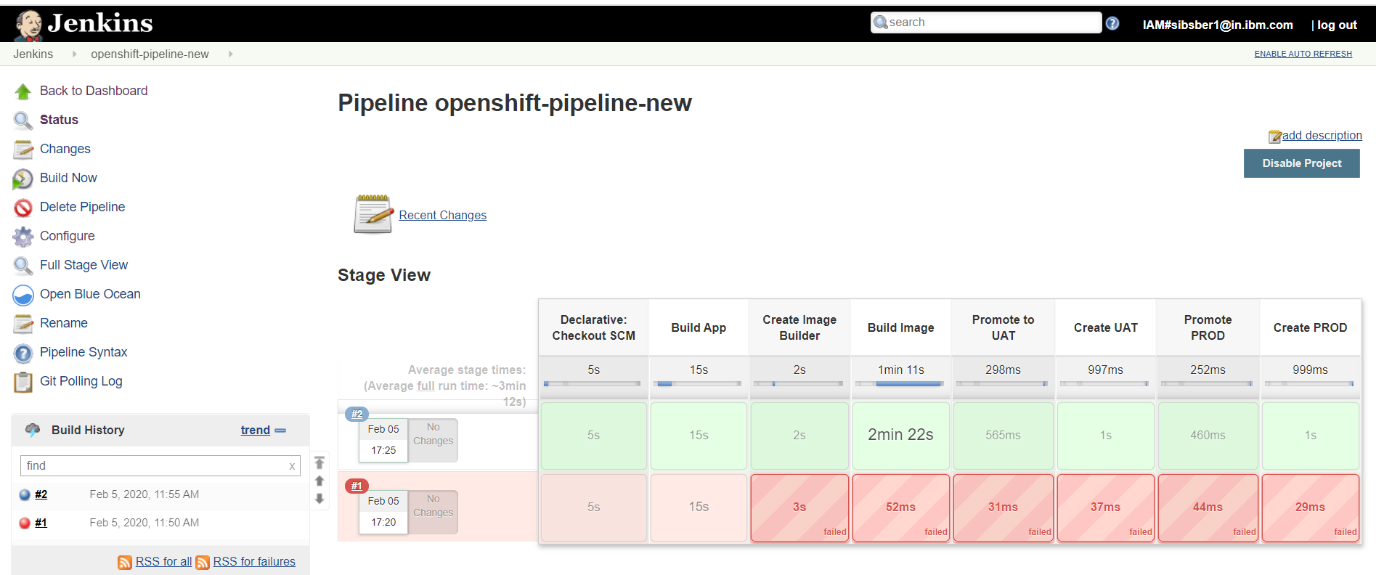
*oc import-image openjdk18-openshift:1.1 --from=registry.access.redhat.com/redhat-openjdk-18/openjdk18-openshift:1.1 --confirm*

**

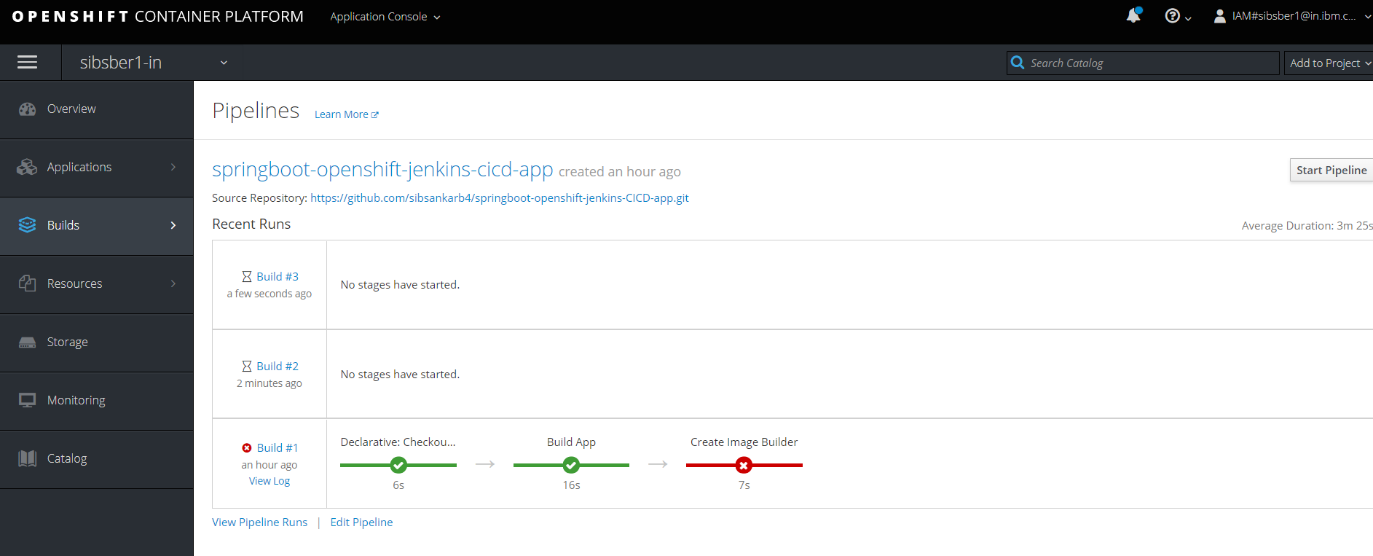
Now we can check that image has been imported successfully.



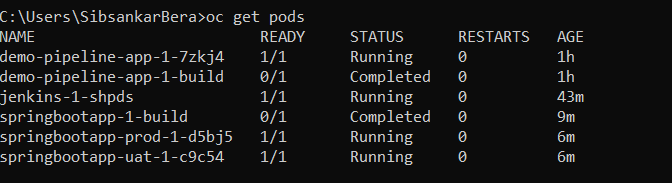
**Step 25:** Lets rerun our pipeline, now we can find the each and every stages of pipeline has ran successfully.

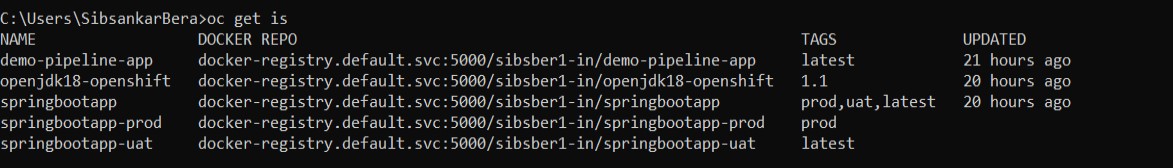


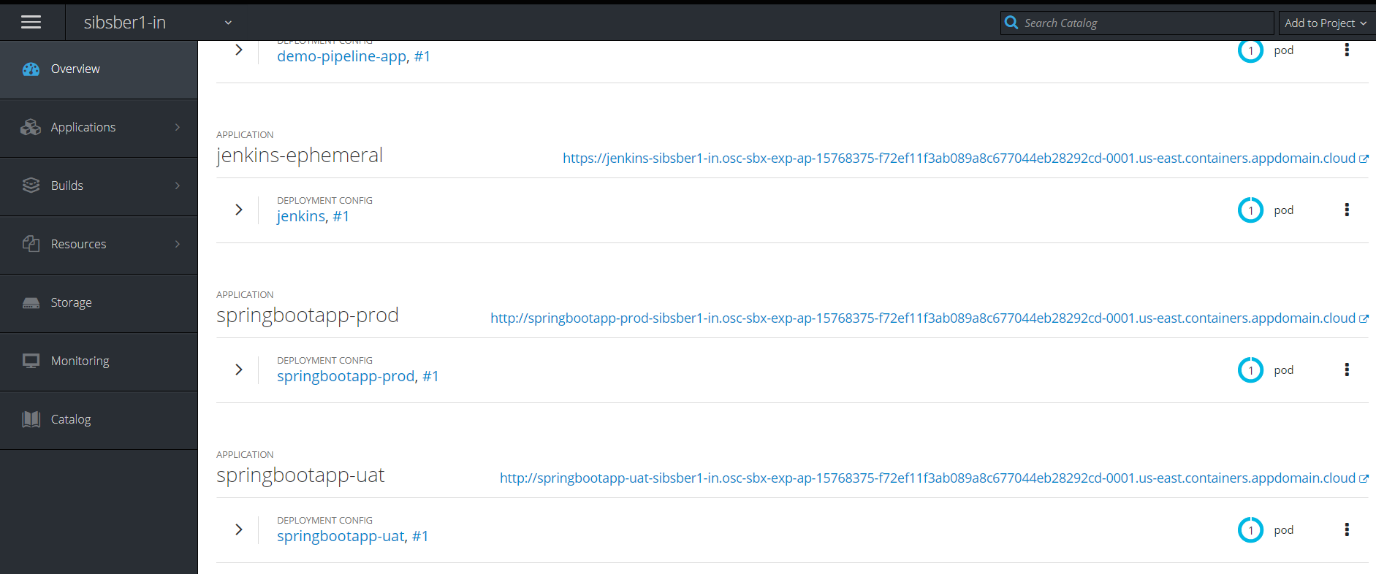
**Step 26:** We can also find the same output from openshift cluster GUI from browser(Builds >>Pipelines).



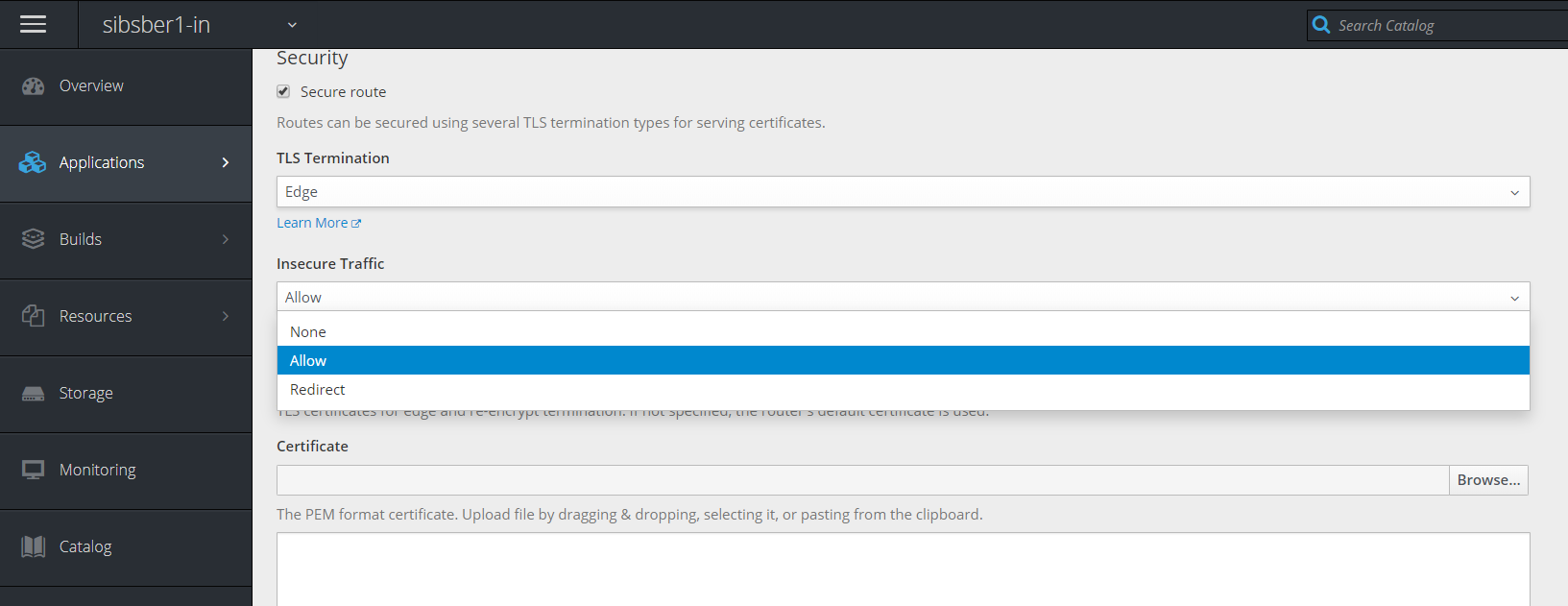
**Step 27:** We can also find that for every environment (uat, prod etc.) one new application pod has created automatically after successfully ran the pipeline.



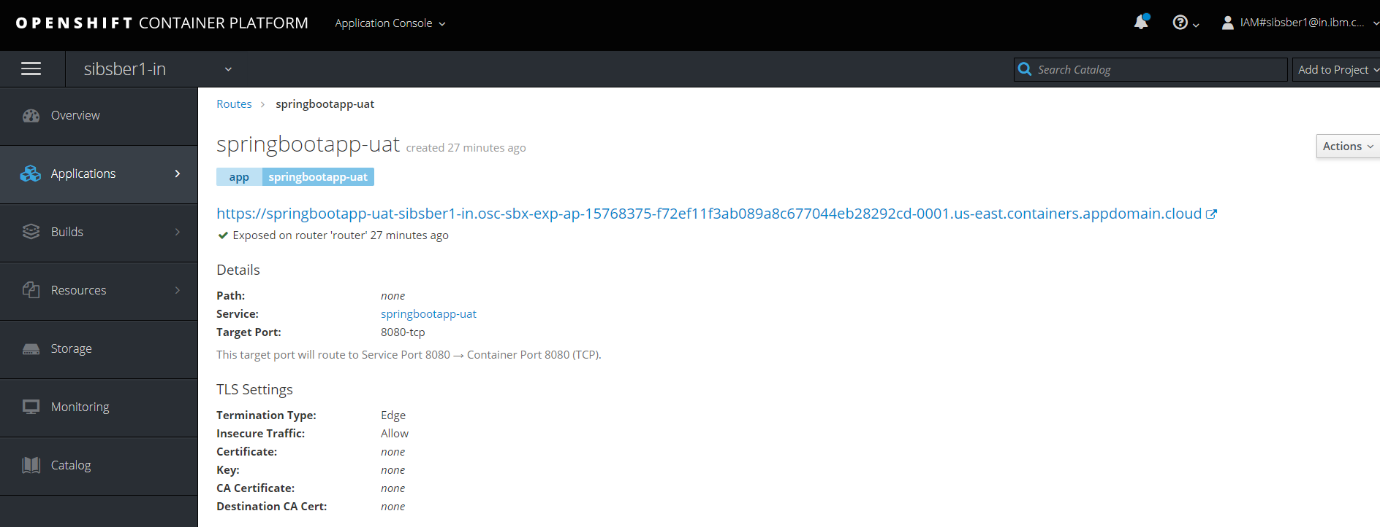




**Step 28:** Now test the application for each environment, before that make all the exposed url as secure if required.

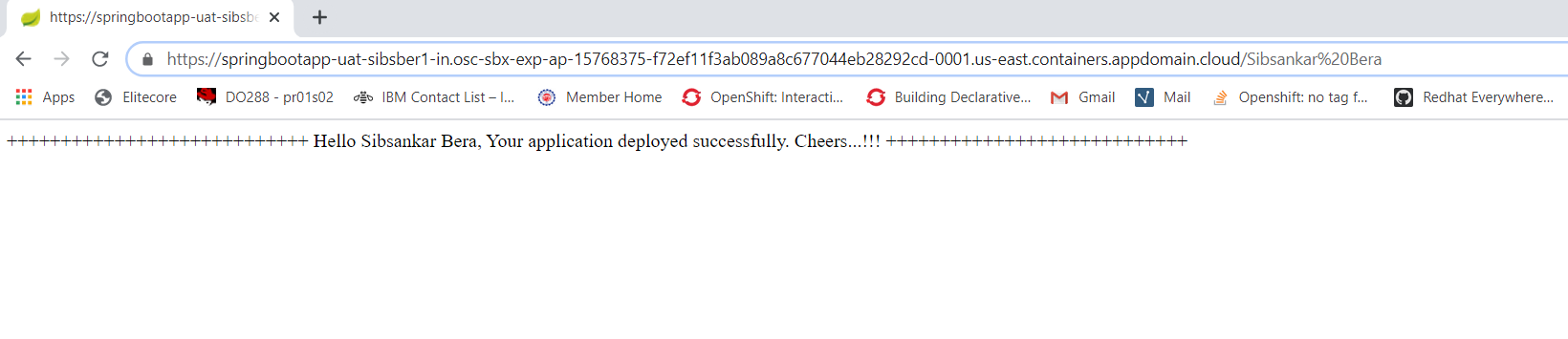


**Step 29:** Now let’s test the application in UAT environment first, by clicking below exposed url.

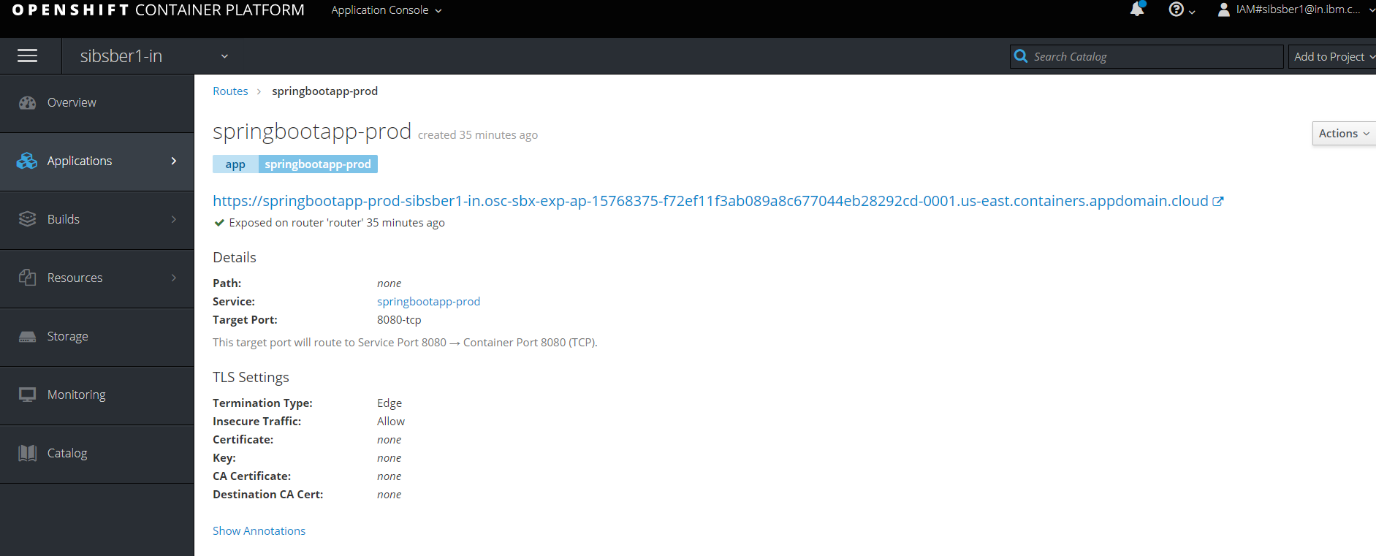


uat url : <https://springbootapp-uat-sibsber1-in.osc-sbx-exp-ap-15768375-f72ef11f3ab089a8c677044eb28292cd-0001.us-east.containers.appdomain.cloud/info>

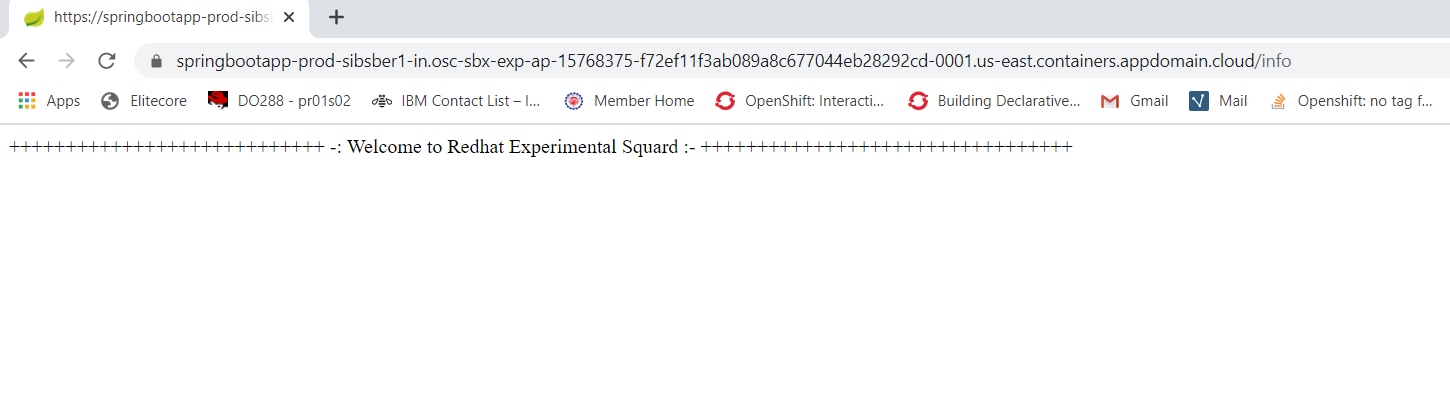
uat url : [https://springbootapp-uat-sibsber1-in.osc-sbx-exp-ap-15768375-f72ef11f3ab089a8c677044eb28292cd-0001.us-east.containers.appdomain.cloud/Sibsankar Bera](https://springbootapp-uat-sibsber1-in.osc-sbx-exp-ap-15768375-f72ef11f3ab089a8c677044eb28292cd-0001.us-east.containers.appdomain.cloud/Sibsankar%20Bera)



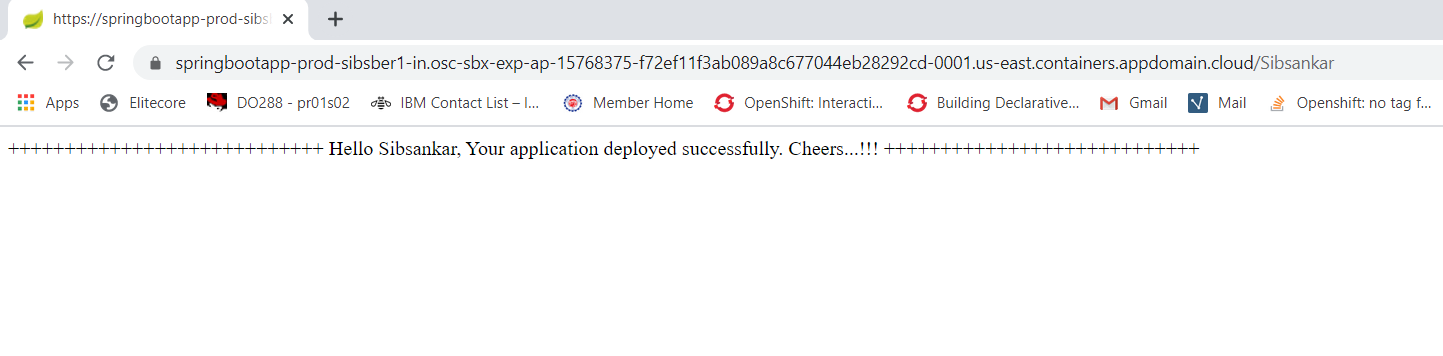
**Step 30:** Now test the application in Prod environment, by clicking below exposed url.



Prod url: <https://springbootapp-prod-sibsber1-in.osc-sbx-exp-ap-15768375-f72ef11f3ab089a8c677044eb28292cd-0001.us-east.containers.appdomain.cloud/info>



Prod url: <https://springbootapp-prod-sibsber1-in.osc-sbx-exp-ap-15768375-f72ef11f3ab089a8c677044eb28292cd-0001.us-east.containers.appdomain.cloud/Sibsankar>



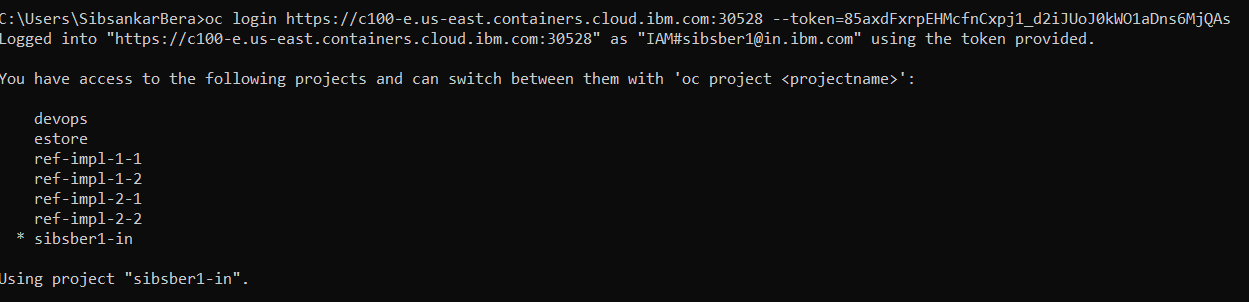
**Sonarqube JaCoCo Integration in Openshift**

**Step1:** To integrate sonarqube and JaCoCo and deploy the same in openshift cluster, we have created one sample spring boot project and one sonarqube-ephemeral-template.yaml file. Please find below git url for reference.

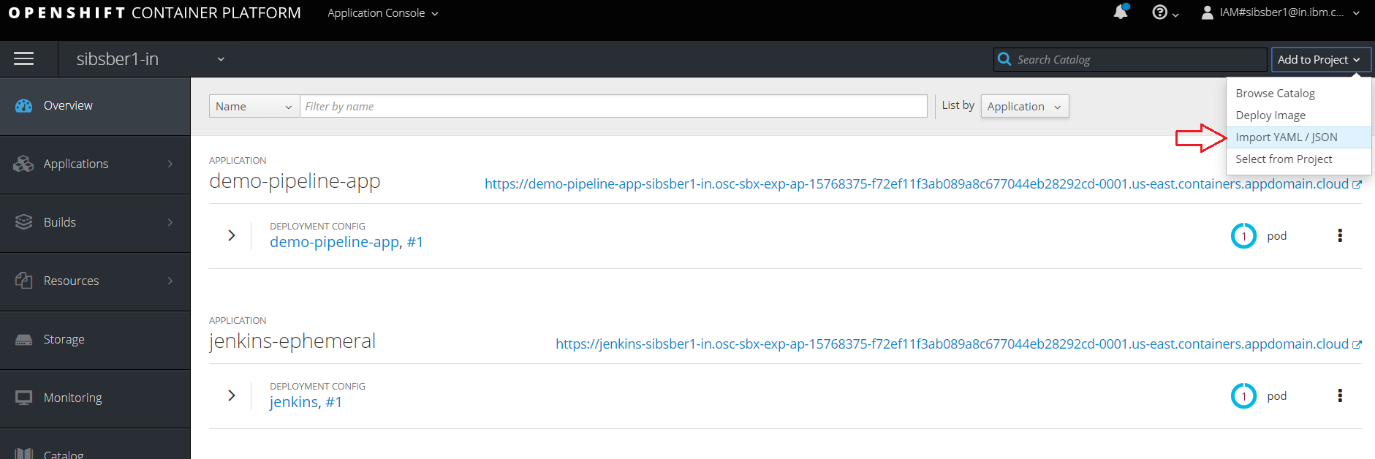
Git url: https://github.com/sibsankarb4/Sonarqube-JaCoCo-Openshift-Demo.git

Sonarqube template location url: <https://github.com/sibsankarb4/Sonarqube-JaCoCo-Openshift-Demo/blob/master/sonarqube-ephemeral-template.yaml>

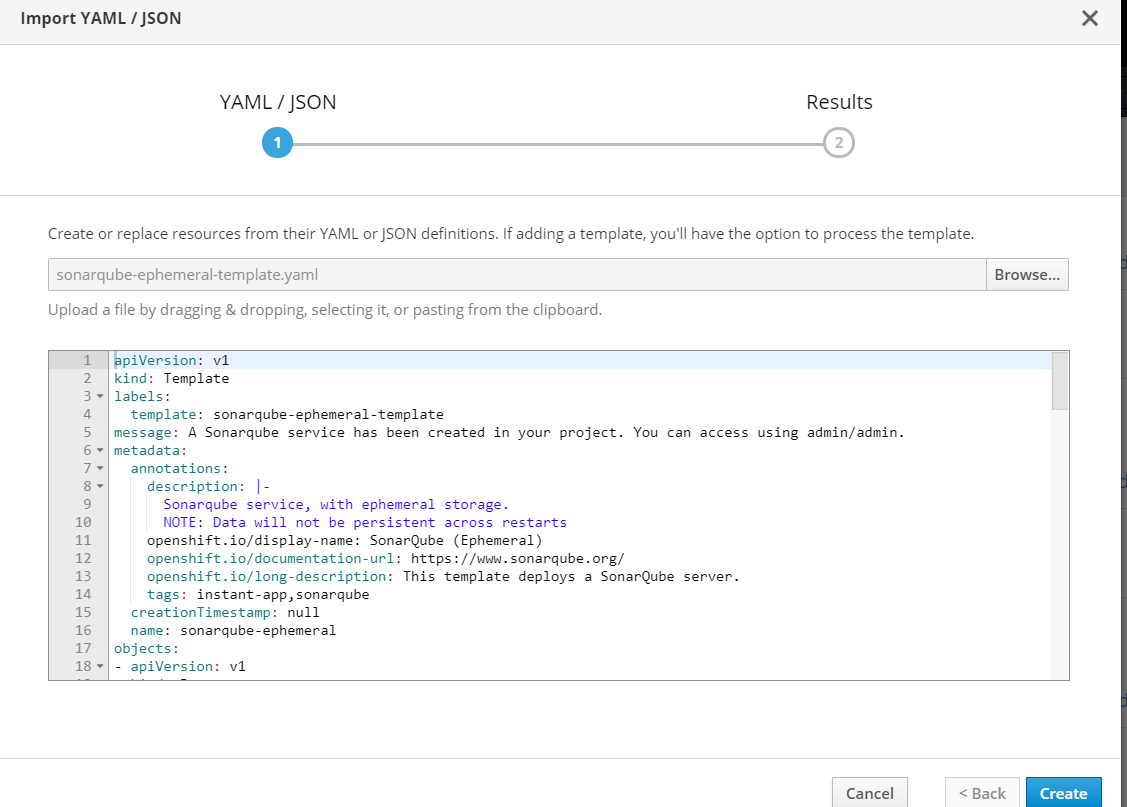
**Step 2:** Login into openshift cluster and navigate to your corresponding project.



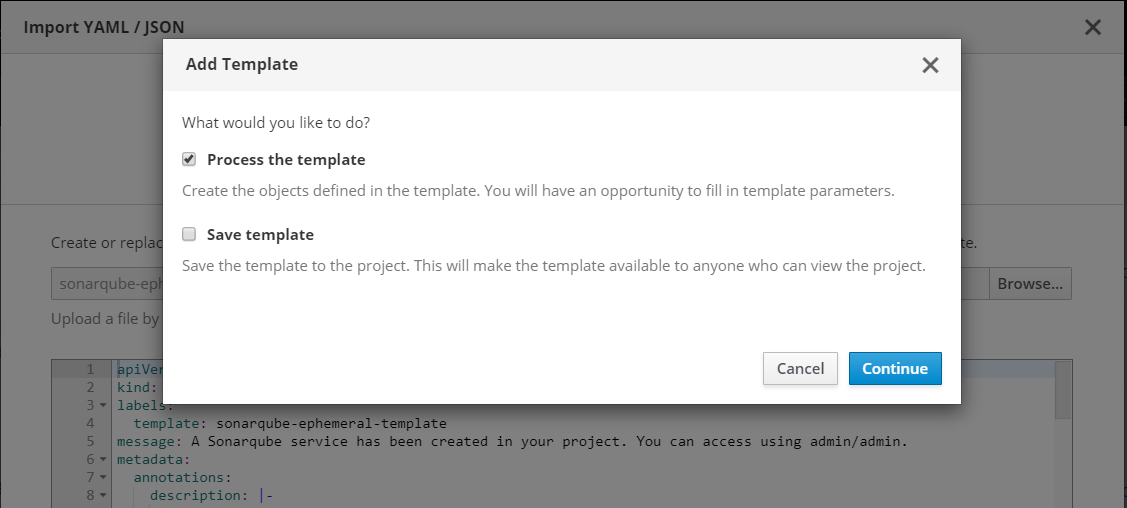
**Step 3:** Login into openshift web console as well, select the desire project and click on **Import YAML/JSON** option.



**Step 4:** Now browse your sonarqube-ephemeral-template.yaml file from your system.

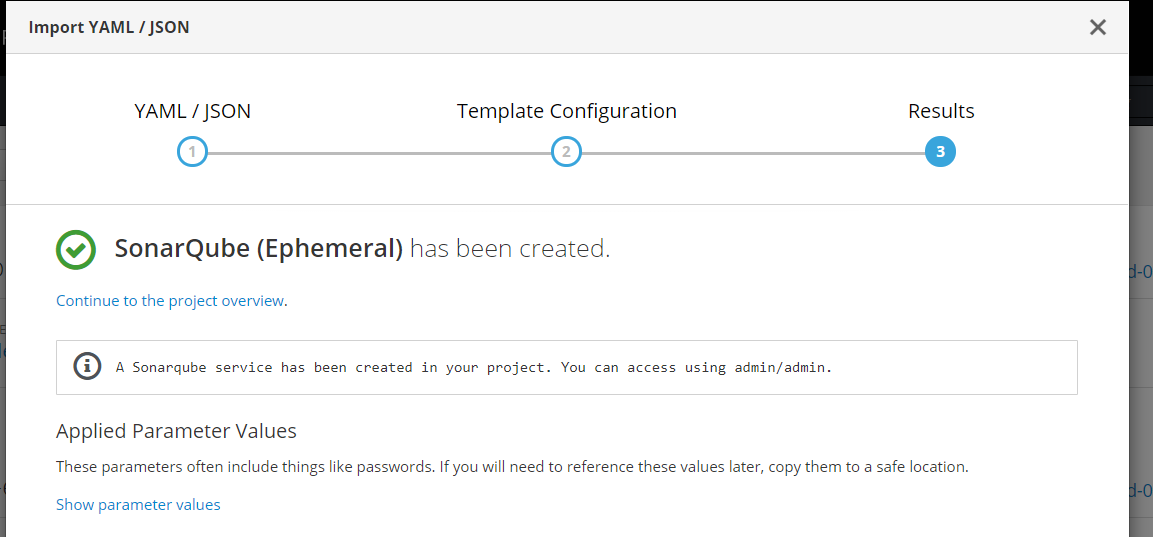


**Step 5:** Click on **Create** button and then click on **Continue** with default checked options.

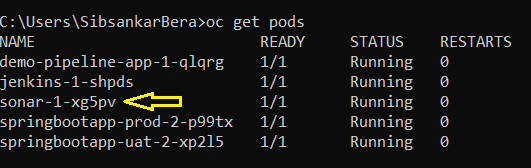


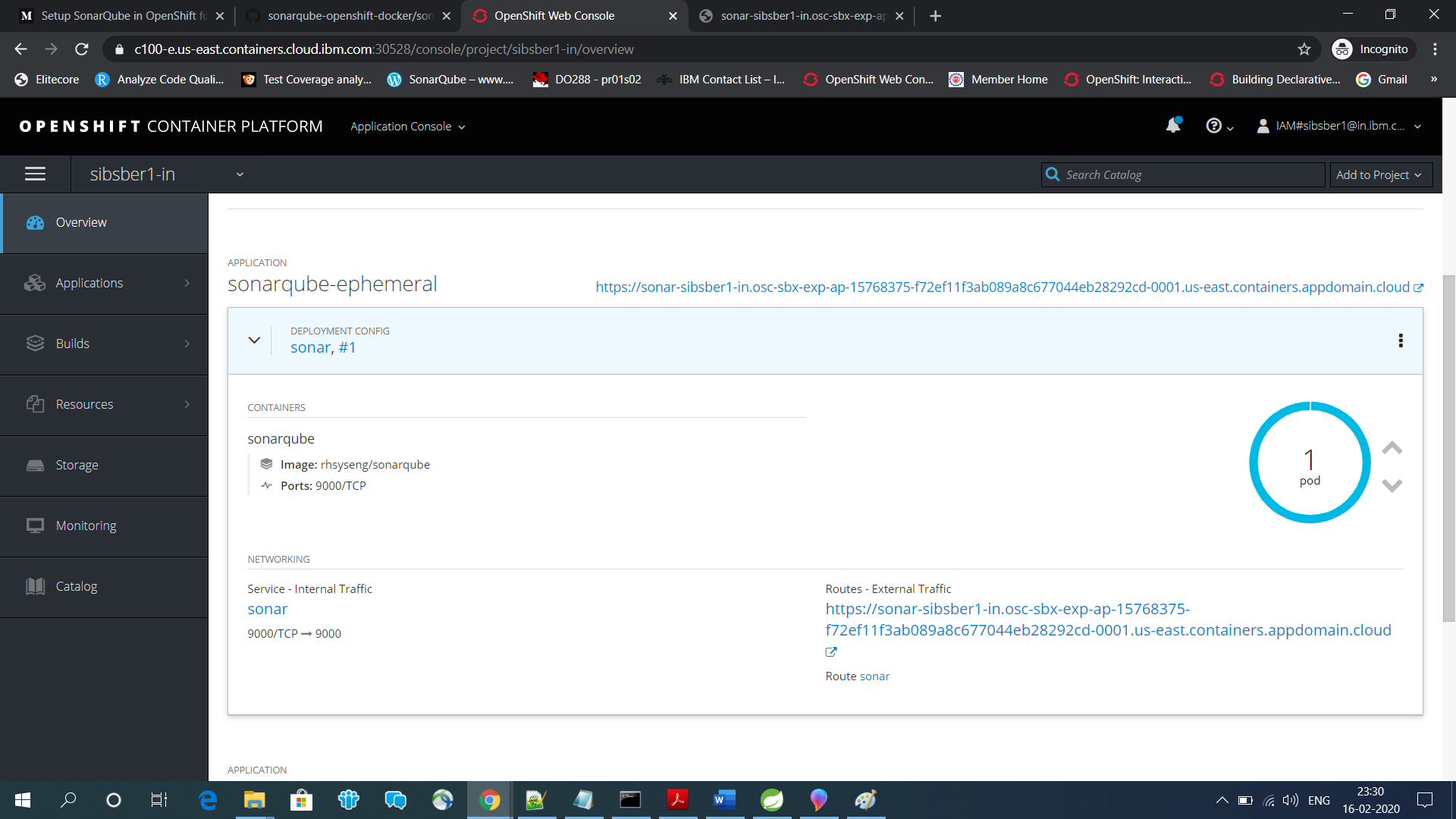
**Step 6:** At the final stage just click on **Create** button with all default configurations.





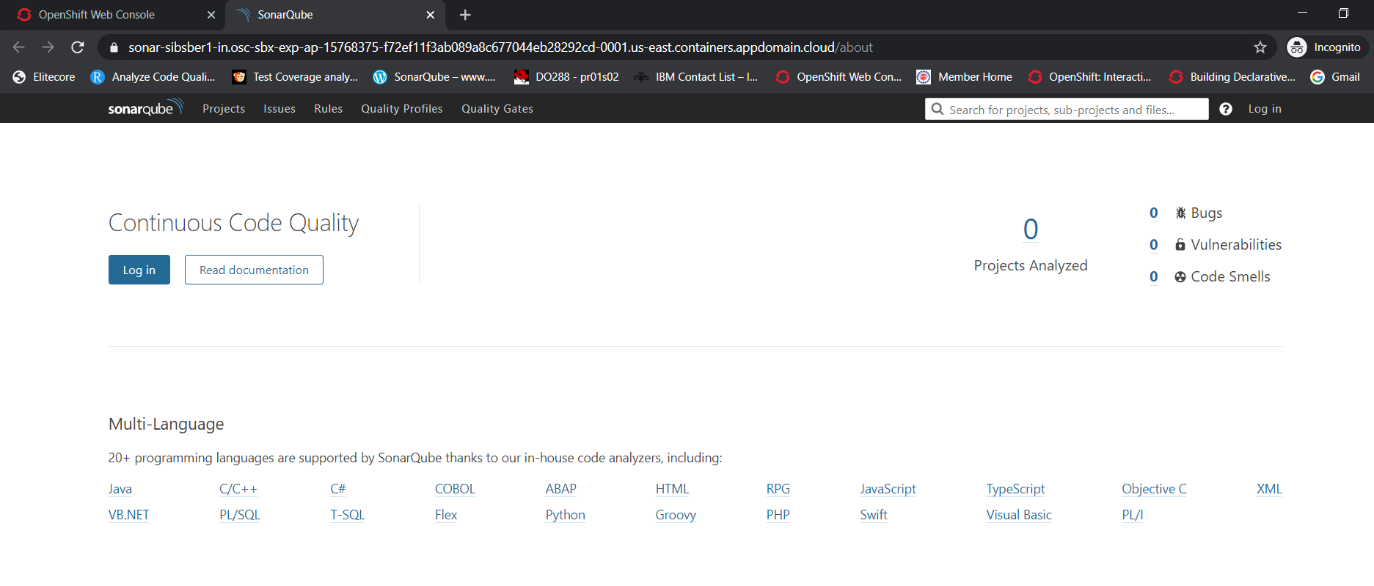
**Step 7:** Now you can see that sonarqube ephemeral pod has been created and it is up and running.





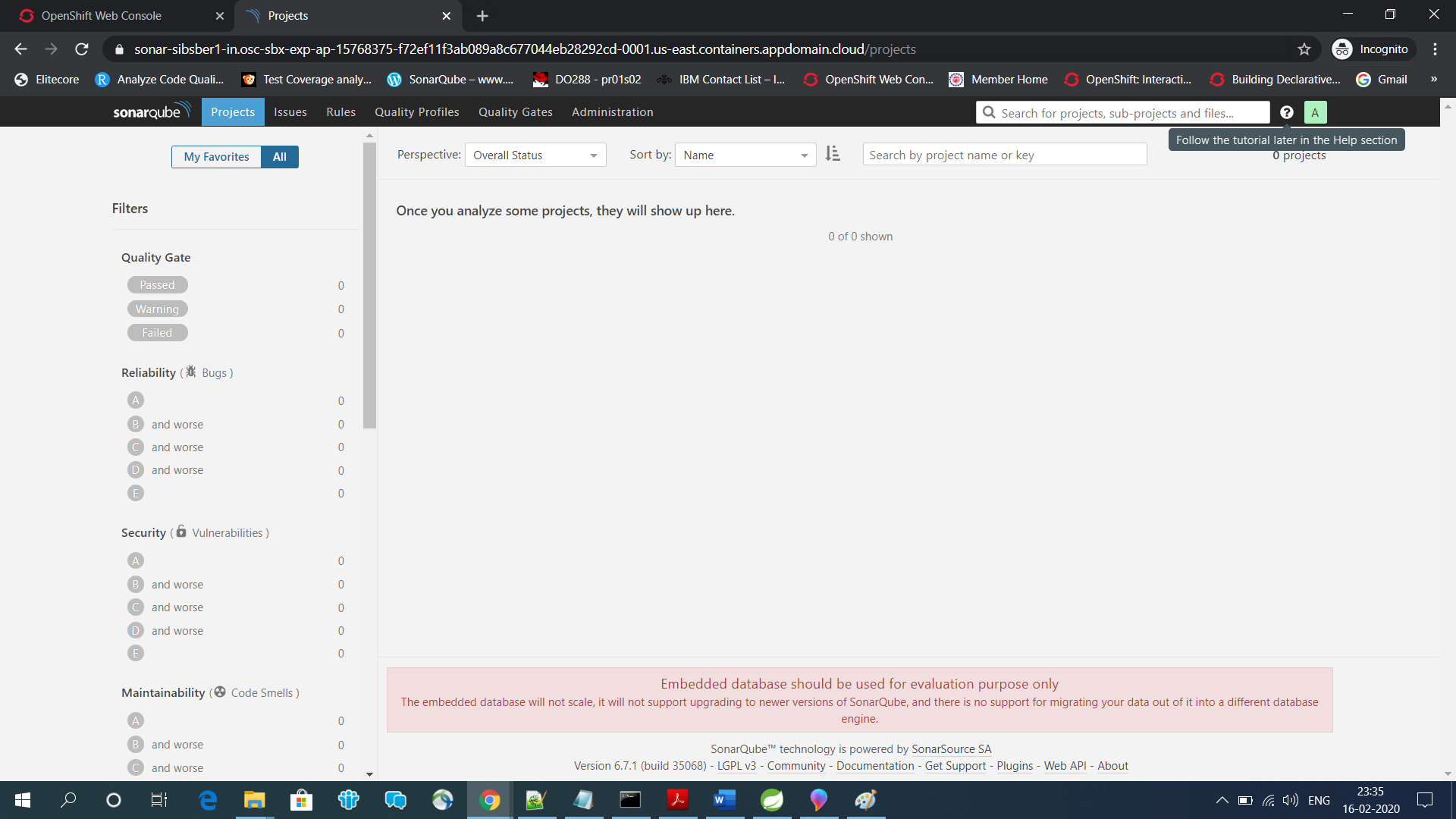
If required change the insecure route to secure route.

**Step 8:** Now open the sonarqube routes url.



Let’s login into sonar with default username and password (admin/admin).

As we have not created/pushed/integrated any project in sonar, that’s why nothing is showing in project page.



**Step 9:** In our spring boot project we have added latest JaCoCo maven plugin in pom.xml file to integrate with sonar server in openshift cluster.

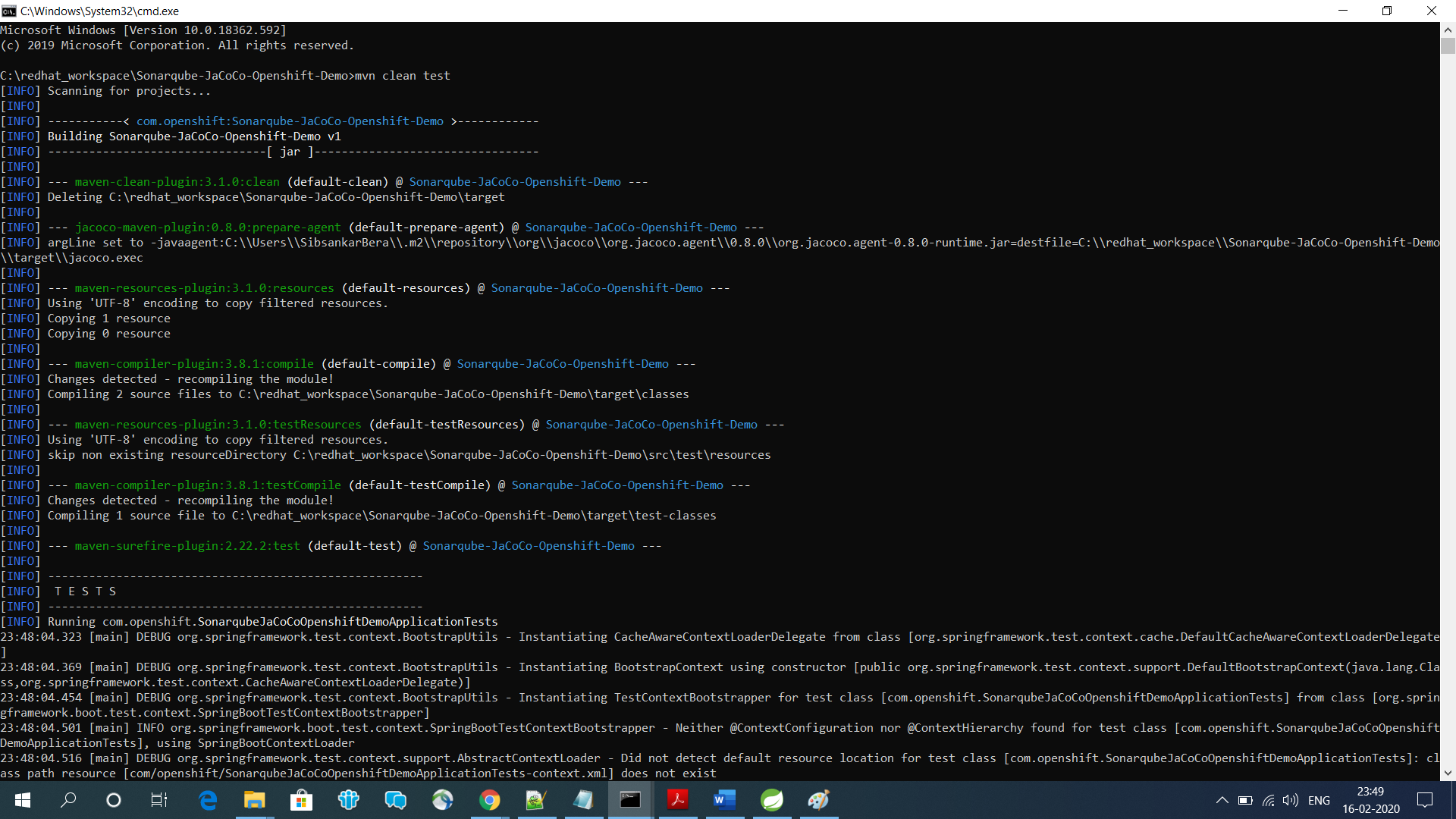
Pom.xml url: <https://github.com/sibsankarb4/Sonarqube-JaCoCo-Openshift-Demo/blob/master/pom.xml>

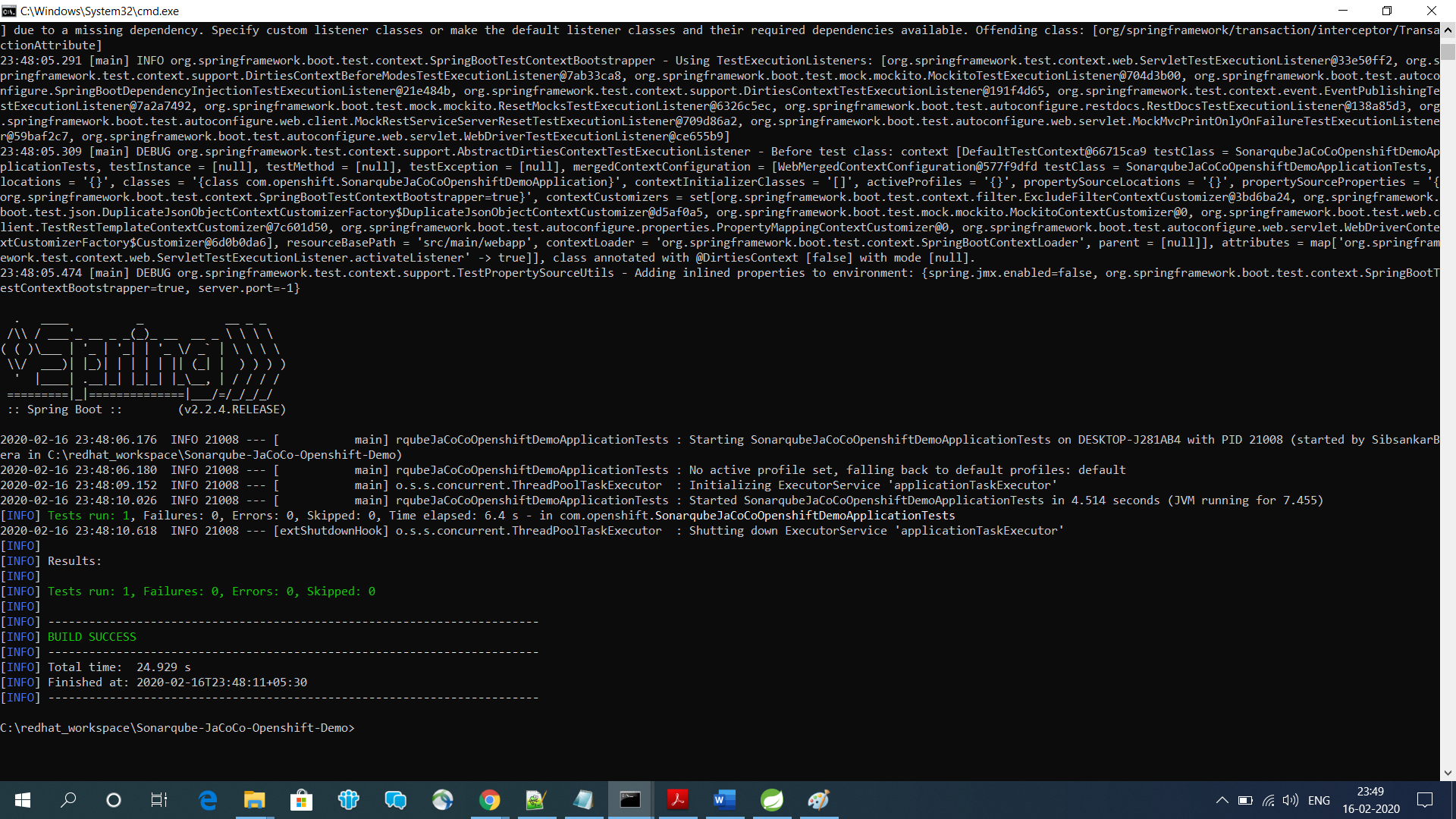
**Step 10:** Now go to your project folder directory where pom.xml file exits. For my case location is

***C:\redhat\_workspace\Sonarqube-JaCoCo-Openshift-Demo***

Open the same path in command prompt and run the below maven command to build/test our spring boot application.

***mvn clean test***





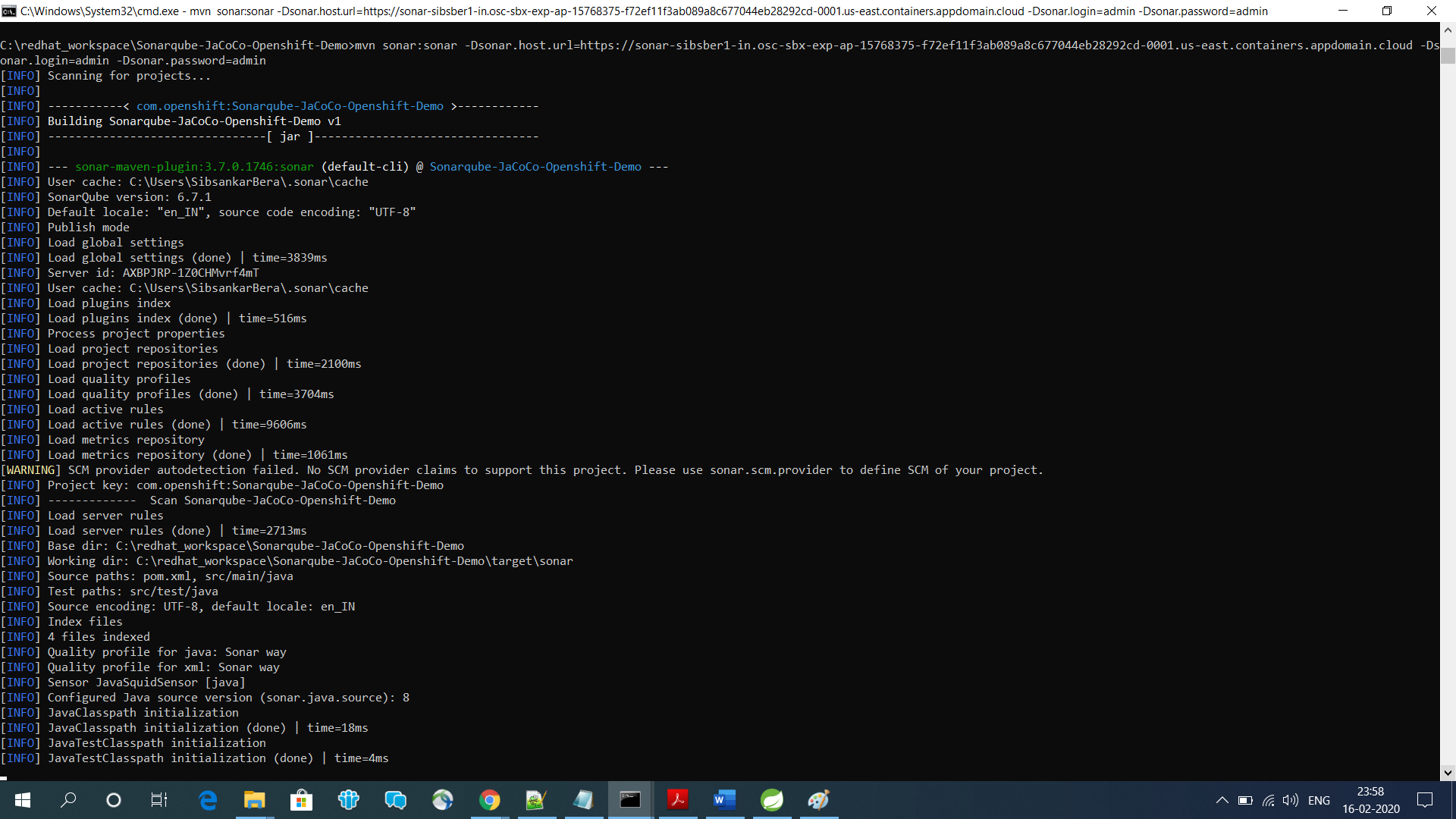
Application build was successful.

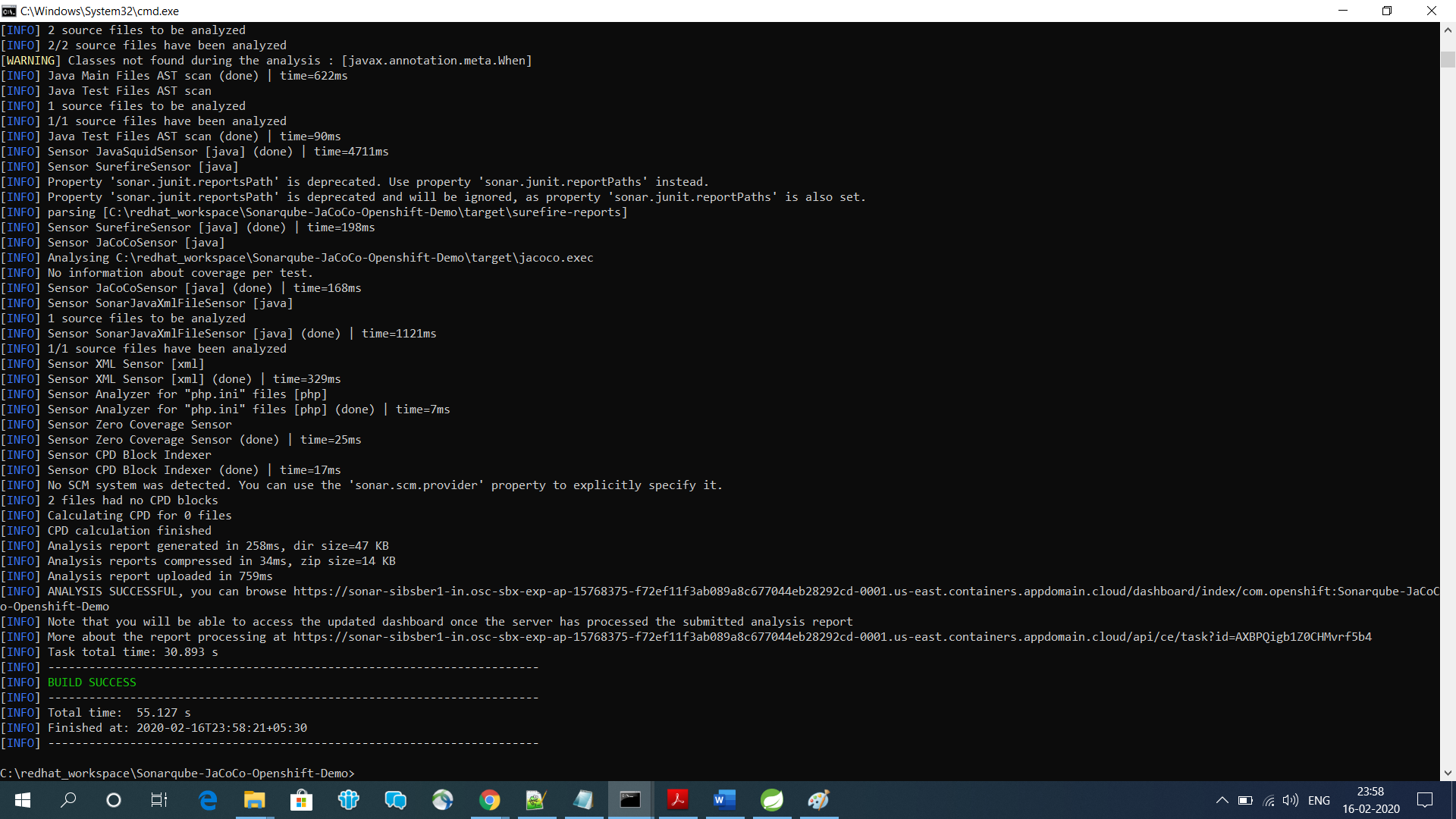
**Step 11:** Now run this below command to push our project code in sonar server, so that we can initiate code analysis of our current project.

***mvn sonar:sonar -Dsonar.host.url=https://sonar-sibsber1-in.osc-sbx-exp-ap-15768375-f72ef11f3ab089a8c677044eb28292cd-0001.us-east.containers.appdomain.cloud -Dsonar.login=admin -Dsonar.password=admin***

*Sonar server remote url:* [*https://sonar-sibsber1-in.osc-sbx-exp-ap-15768375-f72ef11f3ab089a8c677044eb28292cd-0001.us-east.containers.appdomain.cloud*](https://sonar-sibsber1-in.osc-sbx-exp-ap-15768375-f72ef11f3ab089a8c677044eb28292cd-0001.us-east.containers.appdomain.cloud)

*Sonar server username/password:* admin/admin





**Step 12:** Now let’s check the sonar server to check whether our project has created or not with all other details.



No, it is not crated yet, now let’s refresh the page.

After refresh, we are getting below page with our project details, code coverage percentage details, analysis details etc. **Quality Check: Passed**

