**INDUSTRY GRADE PROJECT 1: (ABC TECHNOLOGIES)**

**Student Name: Krishna Chaitanya**

**E-mail:** [**nkrishnachaitanya81@gmail.com**](mailto:nkrishnachaitanya81@gmail.com)

**Github Repository:**

[**https://github.com/krischai81/EdurekaProject1**](https://github.com/krischai81/EdurekaProject1)

**Task 1:**

*Clone the project from the GitHub link shared in resources to your local machine. Build the code using Maven commands.*

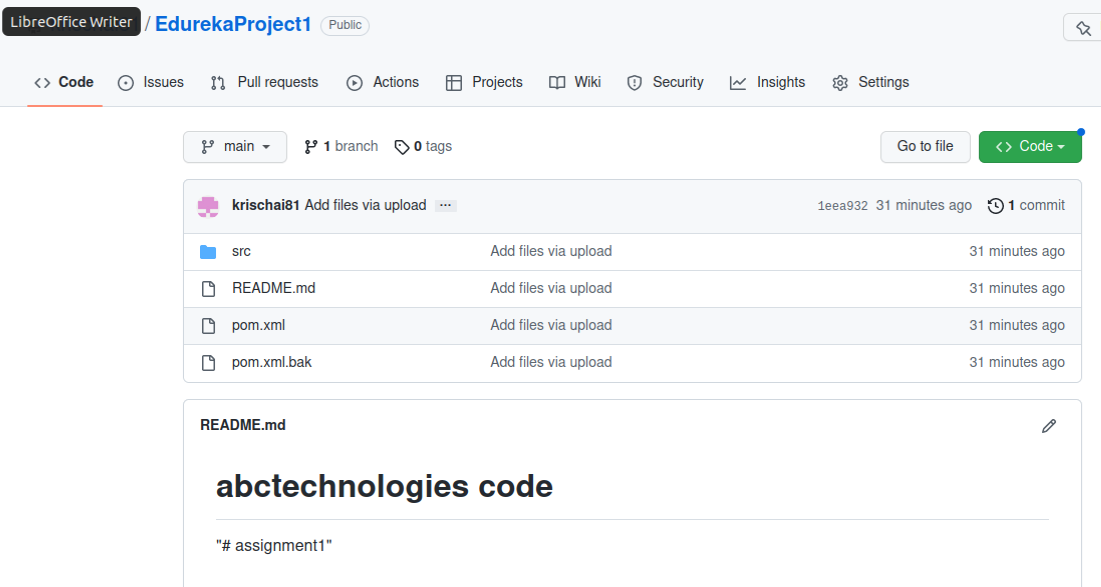
Step 1:

Installed all the below software on a Linux ubuntu VM.

1. Java
2. Maven
3. Git
4. Jenkins
5. Docker
6. Ansible
7. Kubernetes
8. Grafana
9. Prometheus

Step 2:

Configure the Github repository with the provided code.



**Task 2**

*Set up the Git repository and push the source code. Then, log in to Jenkins.*

*1. Create a build pipeline containing a job for each*

*• One for compiling source code*

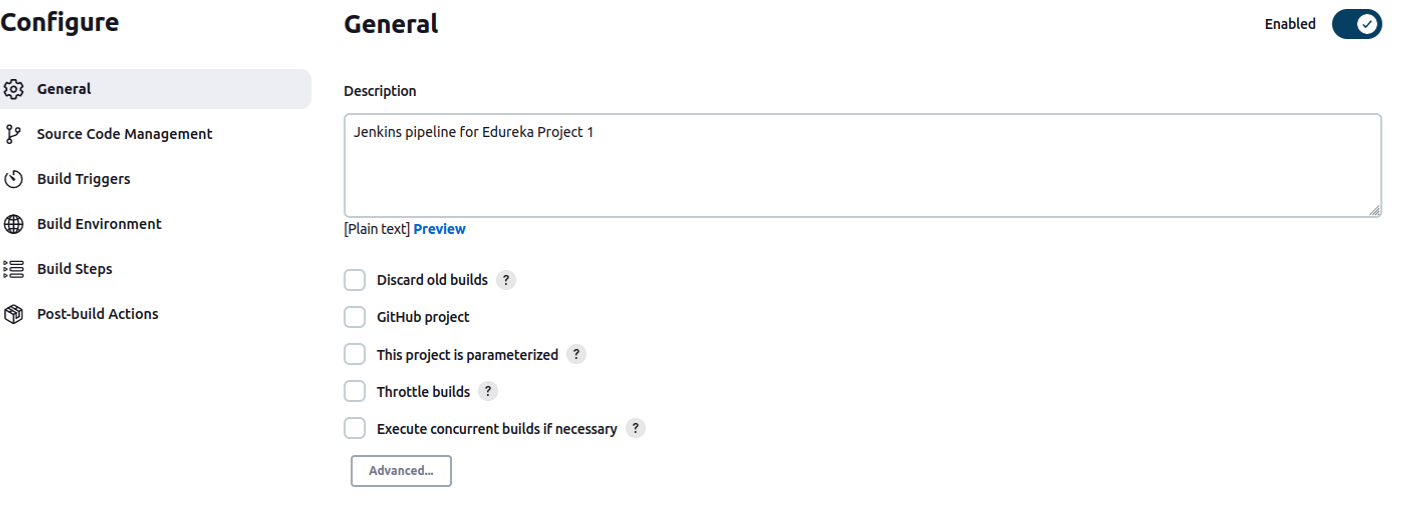
*• Second for testing source code*

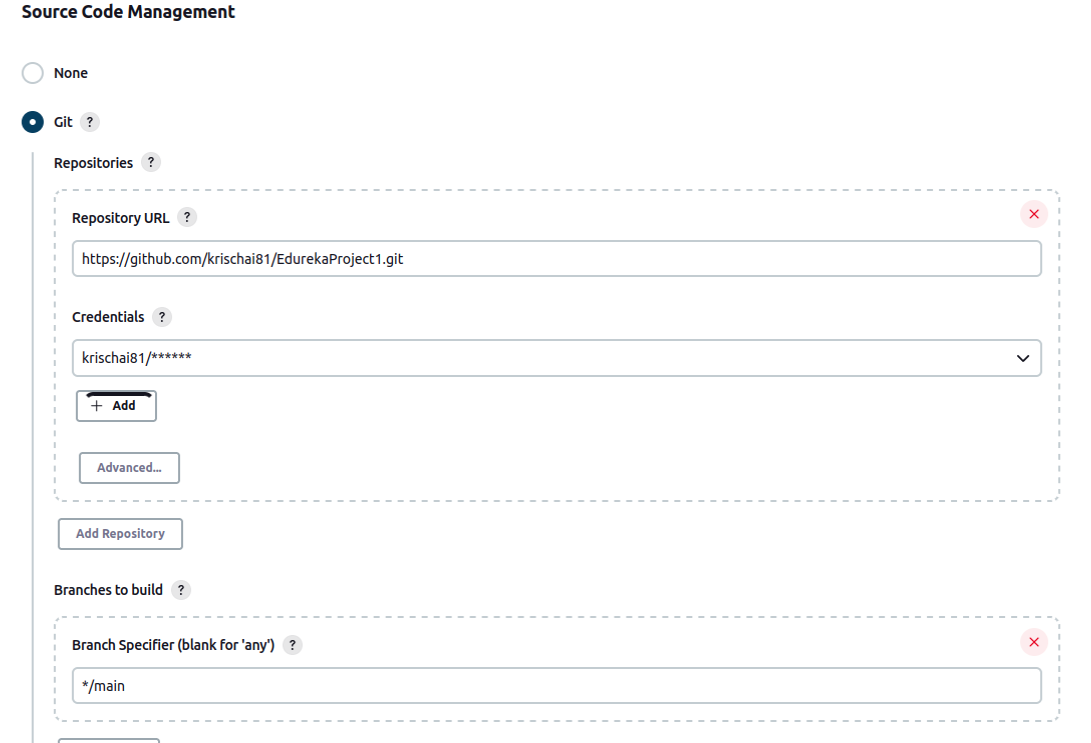
*• Third for packing the code*

*2. Execute the CI/CD pipeline to execute the jobs created in step 1*

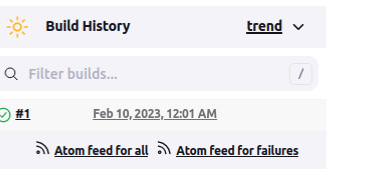
*3. Set up a master-slave node to distribute the tasks in the pipeline.*

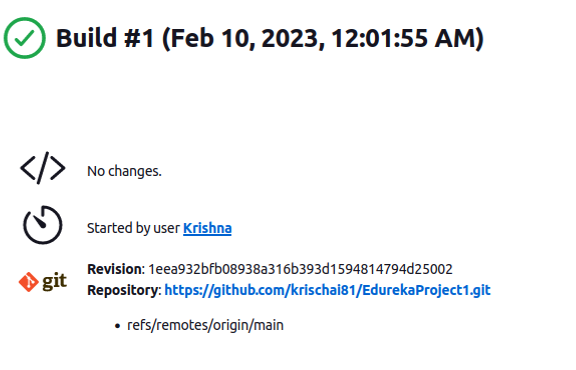
Step 3: Configure Jenkins to build the code using Maven commands



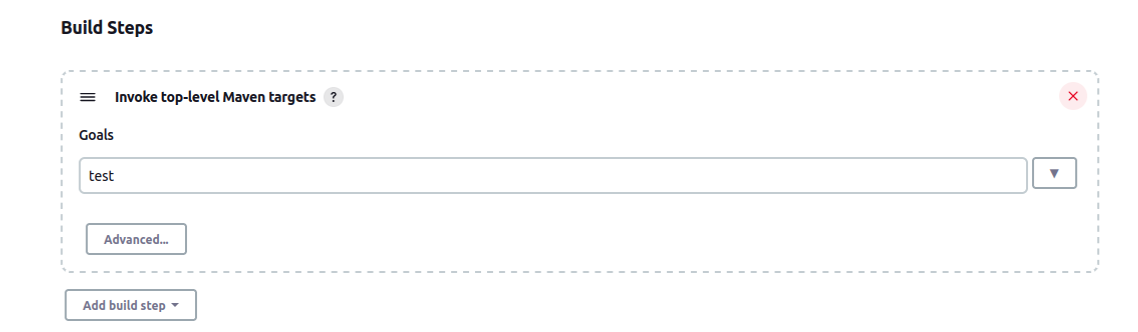
****



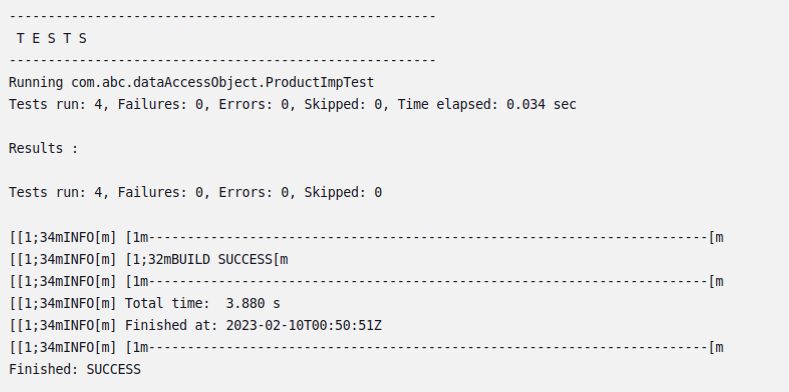
Step 4: Build the code in Jenkins



Step 5: Repeat the previous steps 2-4, but create a test job in Jenkins instead of compile job

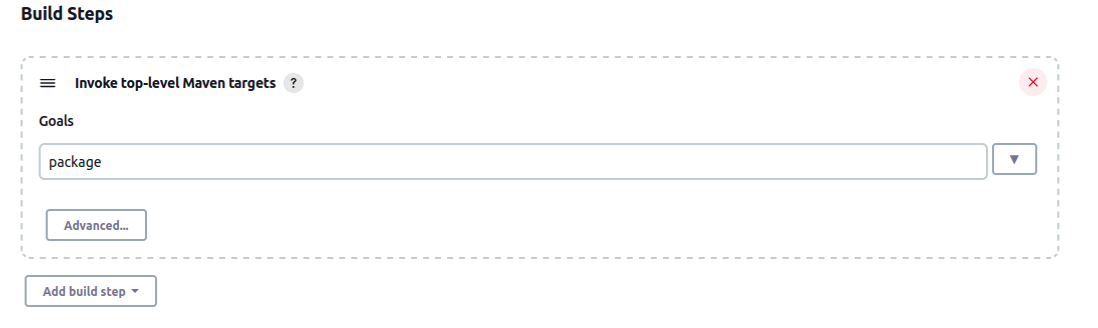


Step 6: Build the test job created in step 5 and verify that the tests ran successfully



Step 7:

Repeat the steps above but create a job for creating the package



Step 8:

Build the package job created in the previous step and verify that the task was SUCCESS as shown below:



Step 9: Execute the CI/CD pipeline to execute the jobs created in previous steps



**Step 10: Set up a master-slave node to distribute the tasks in the pipeline**

1. Create a second linux virtual machine in order to use it as a slave node to distribute the tasks in the pipeline.

In this case the Master machine name is “UbuntuTOM” and the Slave machine name is “UbuntuJERRY”

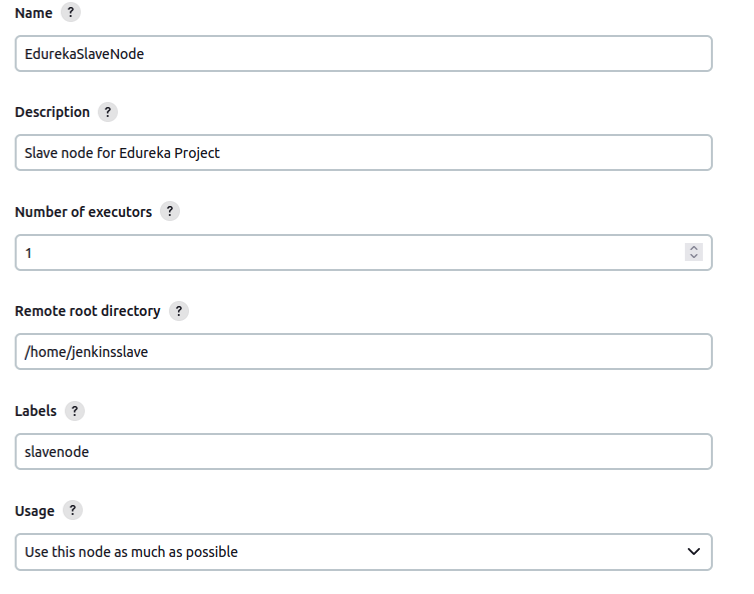
1. Install Jenkins on the slave machine.

Create a Jenkins user and a password on the slave machine. E.g. sudo adduser jenkinsslave --shell /bin/bash

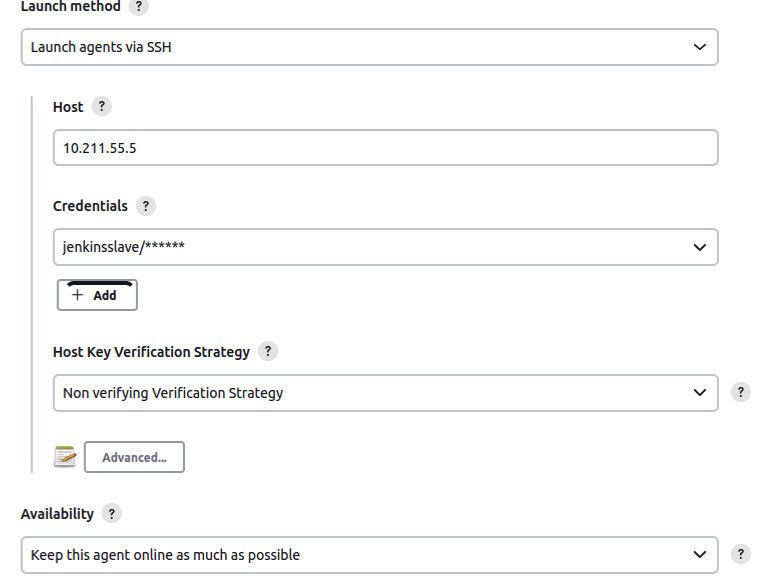
1. Make a directory called jenkins\_slave in the director /home/jenkinsslave

“mkdir /home/jenkinsslave/jenkins\_slave”

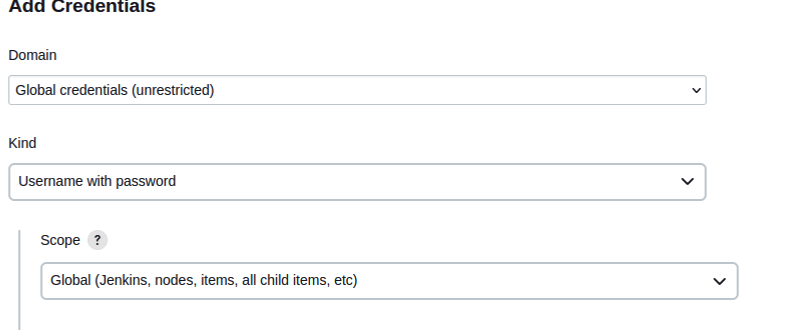
1. Now go to your Master machine and setup your Jenkins agent/slave using username and password
2. Go to Jenkins dashboard->Manage Jenkins -> Manage Nodes
3. Select new node option and provide an agent name, select it as “Permanent Agent” and click “OK”
4. Go to configure and add the directory path you created in “step c” in the Remote root directory section



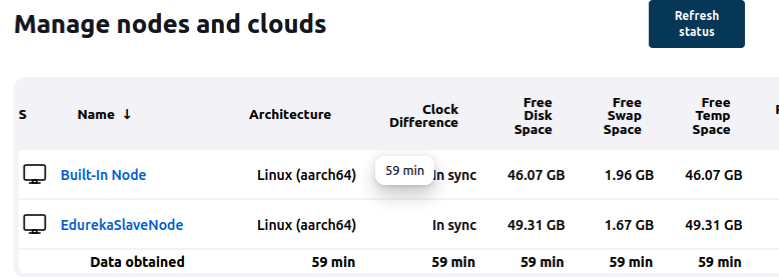
1. Select the launch method as Launch agents via SSH as shown below:

****

1. For the credentials, create a login with the option as “Username with password” and provide the slave node username and password.

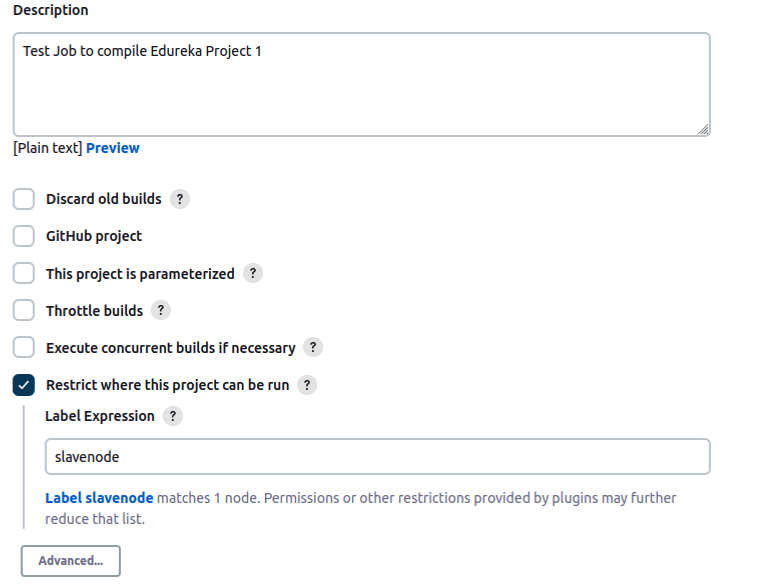


1. Ensure that the slave node is connected and both the Master and the Slave nodes are in Sync as shown below:

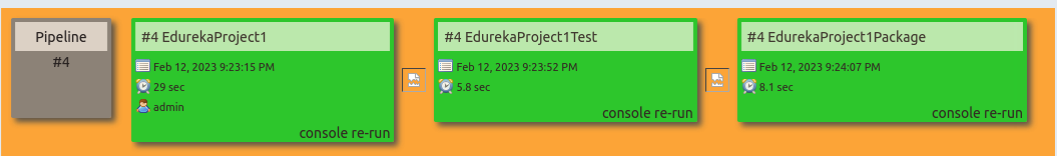
****

1. Select the job that you would want Jenkins to run on the slave node. (In this case we are just selecting the Test job)

Go to Dashboard-> EdurkaProject1Test->Configure and select the option “Restrict where this project can be run” and give the label name for the slave node as shown below:



1. Now select the pipeline and run the pipeline.



1. Select EdurekaProject1Test and view its ConsoleOutput. Confirm that the build was successful on the SlaveNode as indicated below:



**Task 3**

*Write a Docker file. Create an Image and container on the Docker host. Integrate docker host with Jenkins. Create CI/CD job on Jenkins to build and deploy on a container.*

*1. Enhance the package job created in step 1 of task 2 to create a docker image.*

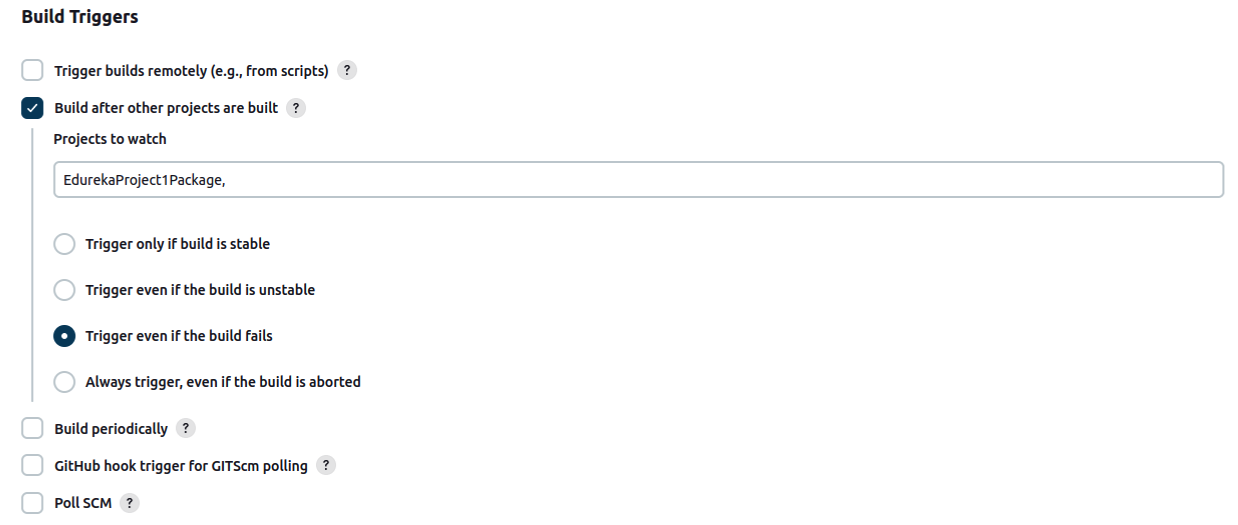
*2. In the Docker image, add code to move the war file to the Tomcat server and build the image.*

Step 11: create a freestyle job in Jenkins called “DeployWarToTomcatInContainer” with the below settings

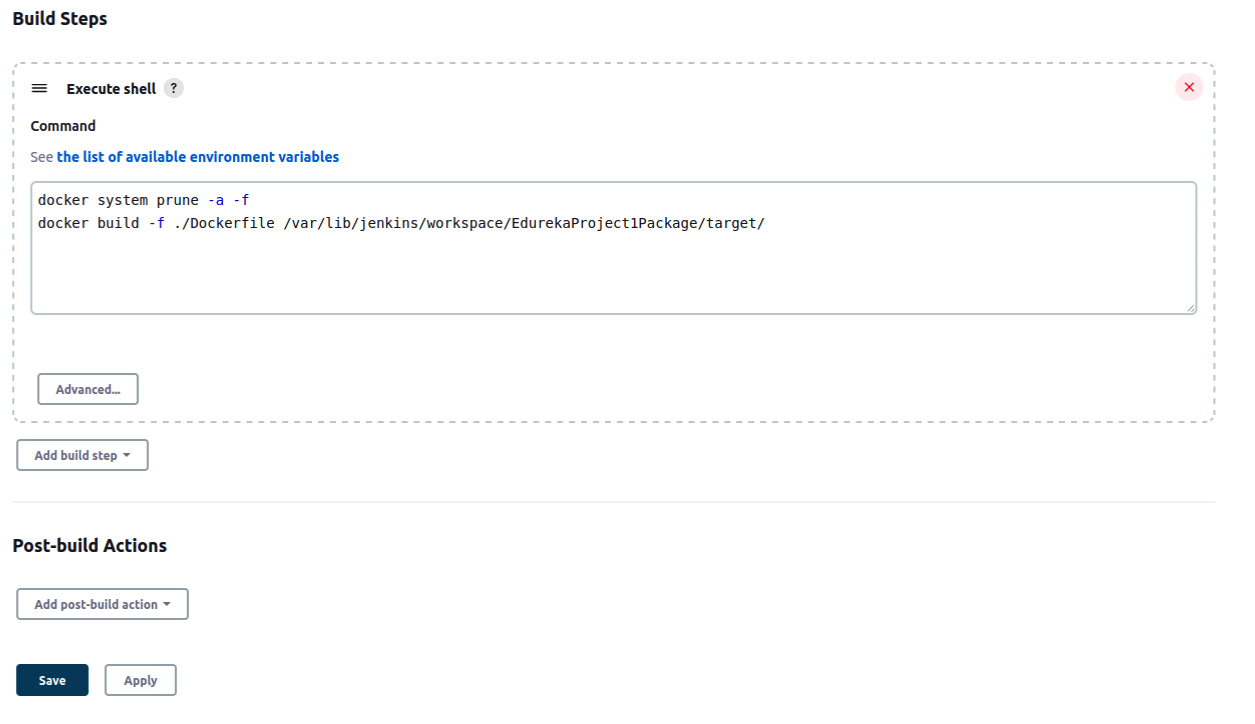




*Make sure the build trigger is set after the EdurekaProject1Package is finished as shown below:*

**

Step 13: In the build steps write the below commands in the shells

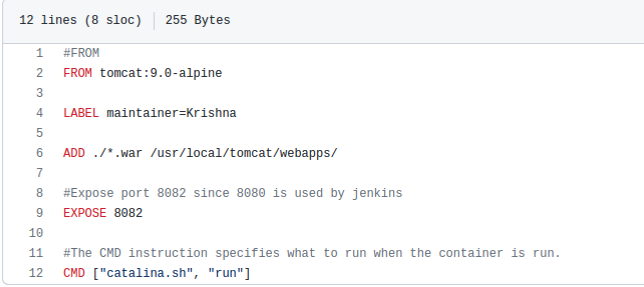


1. docker system prune -a -f 🡪 This will prune any old containers that are running
2. docker build -f ./Dockerfile /var/lib/jenkins/workspace/EdurekaProject1Package/target/

The above command builds the image from the docker file which is in the current directory. The second parameter “/var/lib/jenkins/workspace/EdurekaProject1Package/target/” is the context where the war file is located. Note: It is mandatory to set the right context, so that all the files in the context is sent to docker daemon for building the image as per the commands in Dockerfile.

Step 14:

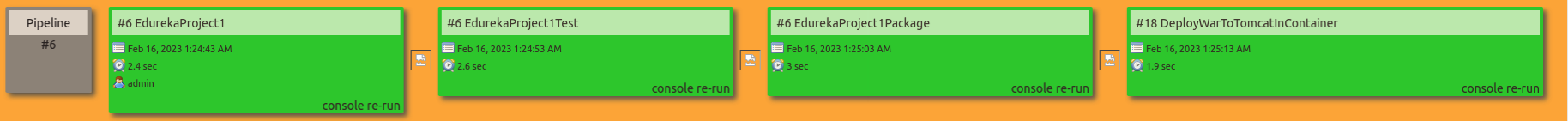
Create a docker file as shown below and upload it to the git repository.

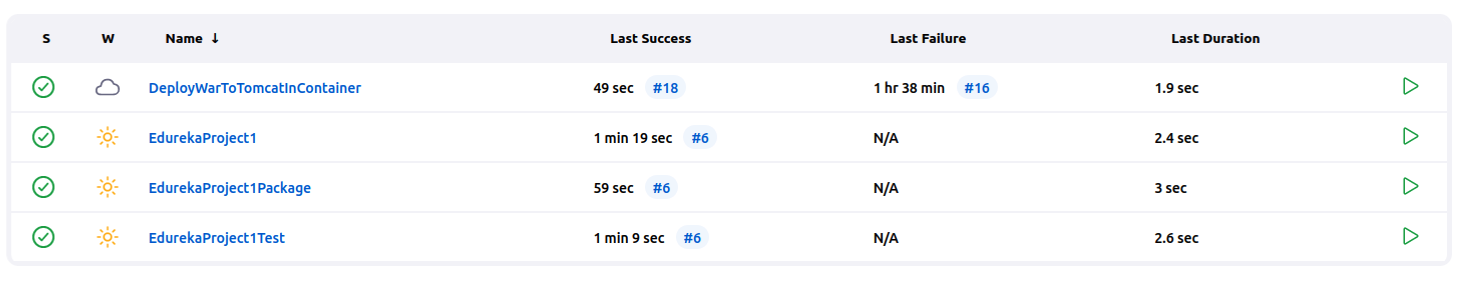


1. The FROM instruction specifies the parent image from which the custom Tomcat server is being built.
2. In the image, we are adding the war file created in Task 2 to the webapps folder in the Tomcat server.
3. The default port on Tomcat server is 8080. We need to expose port 8082 in case we need to use 8082 to access tomcat.
4. Run the Catalina.sh script when the container is run.

Step 15:

Add this newly built job in Jenkins to the build pipeline and run. Make sure the pipeline executes successfully.





Note: If you see any error message like “permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock:”, then add the user jenkins to the group docker as shown below:

$sudo usermod -a -G docker jenkins

And then restart Jenkins by using https://localhost:8081/restart

Step 16:

On your host machine (or docker host), run “docker image ls” to verify that the image is available



Step 17:

Now run the container in a detached mode by mapping the port 8080 in tomcat server to the port 8082 on the host machine as shown below



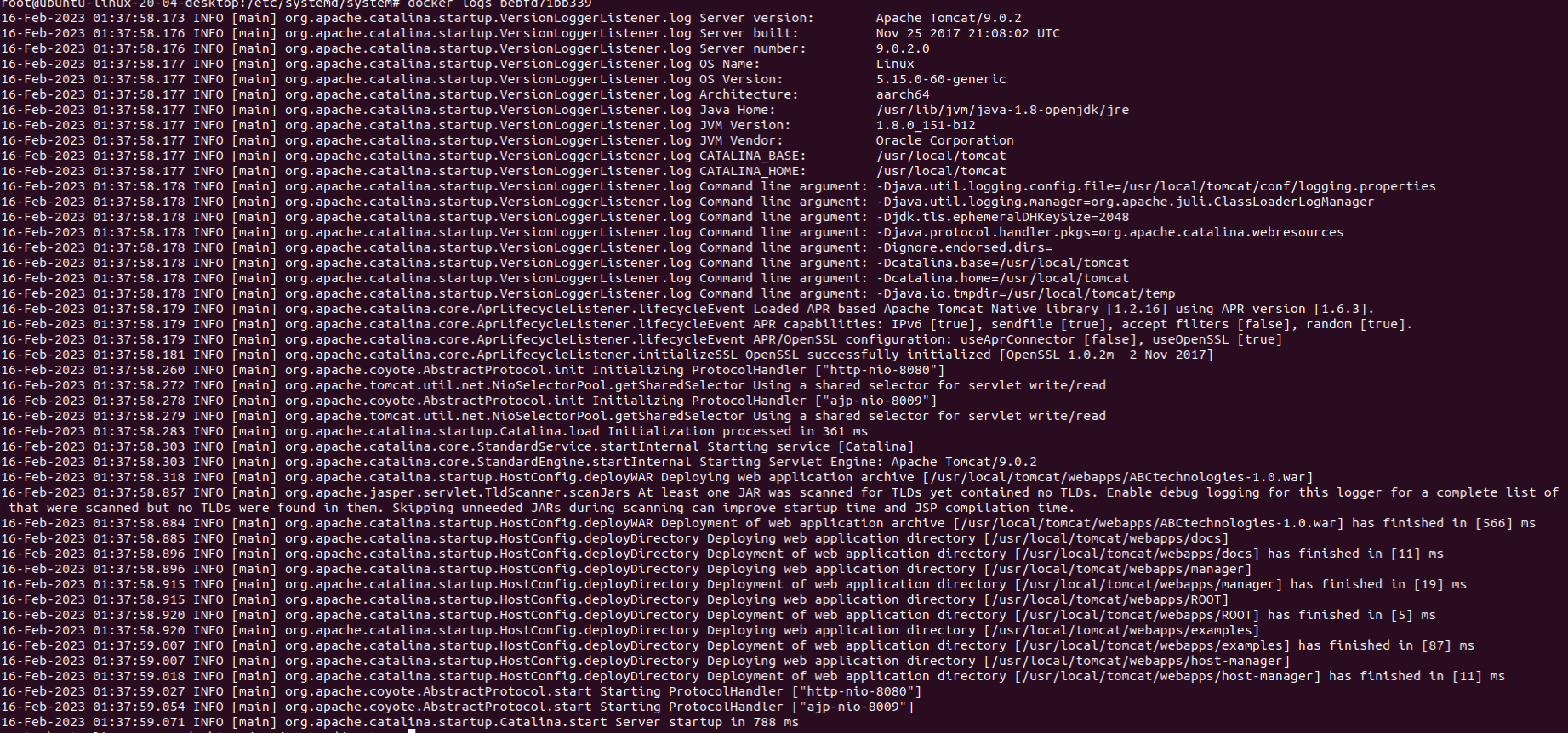
Step 18:

Run the command “docker ps” to make sure that the container is running.



Step 19:

Check the logs of the running container to make sure that the tomcat server is successfully running



Step 20: Now launch the application on localhost port 8082 and confirm that the application is successfully deployed.



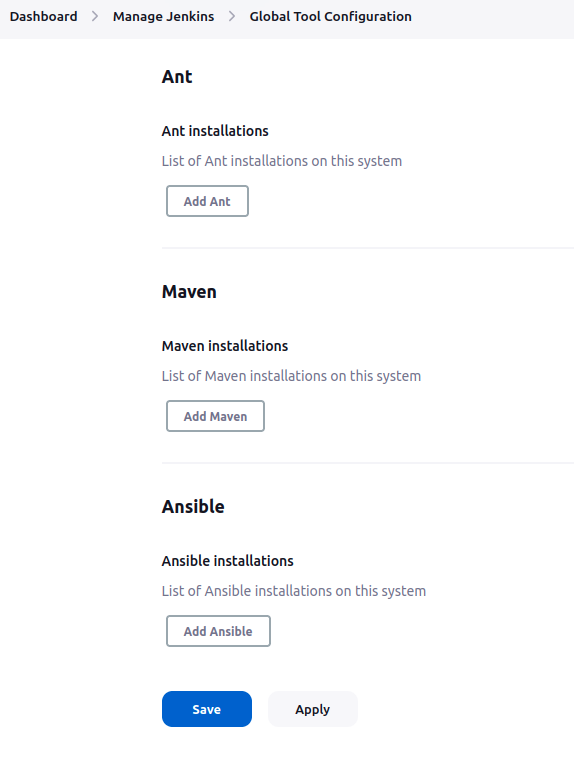
**Task 4**

**Task Description:**  
Integrate the Docker host with Ansible. Write an Ansible playbook to create an image and create a container. Integrate Ansible with Jenkins. Deploy Ansible-playbook. CI/CD job to build code on ansible and deploy it on docker container

1. Deploy Artifacts on Kubernetes
2. Write pod, service, and deployment manifest file
3. Integrate Kubernetes with Ansible
4. Ansible playbook to create deployment and service

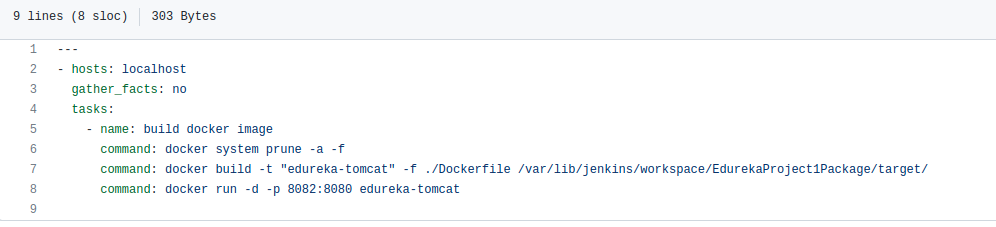
Step 21:

Install Ansible plugin in Jenkins and check if its successfully installed by checking in Dashboard-> Manage Jenkins -> Global Tool Configuration



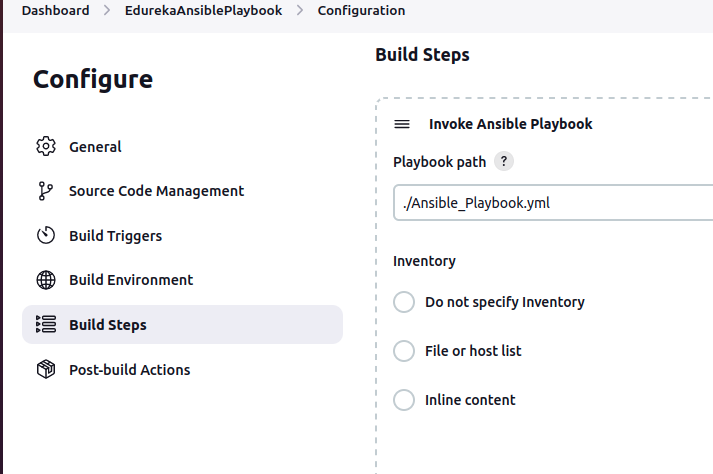
Step 22:

Create an Ansible\_playbook.yml file and upload to the github repository. This Ansible playbook should be able to create the docker image and container and also run the container.



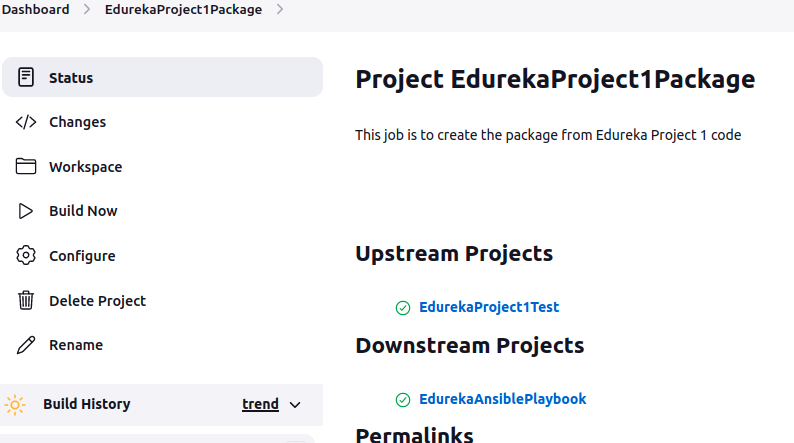
Step 23:

Create a free style project in Jenkins and invoke the Ansible Playbook in the “Build steps” as shown below.



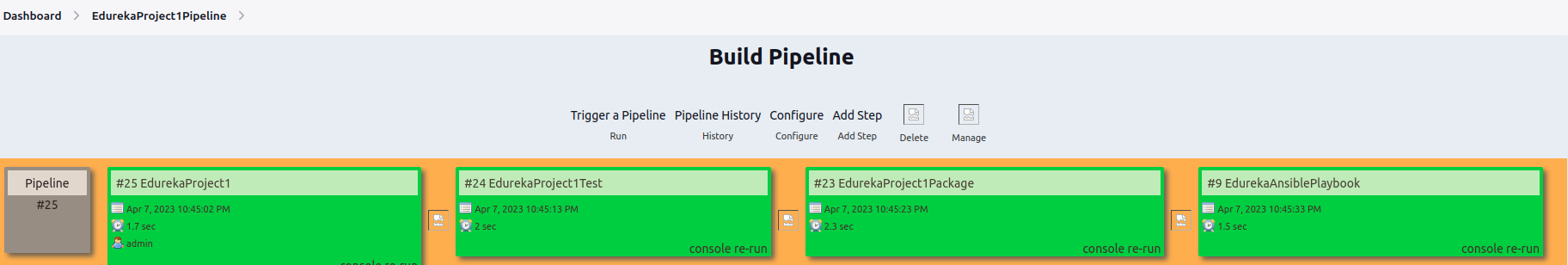
Step 24:

Make this a downstream project after “EdurekaProject1Package”



Step 25:

Execute the build pipeline and ensure that the pipeline is run successfully as shown below:

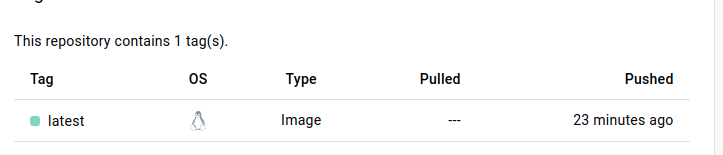


Step 26:

Create a pipeline script in Jenkins to Push the docker image to Docker Hub and make sure it runs after the Ansible playbook job is executed.

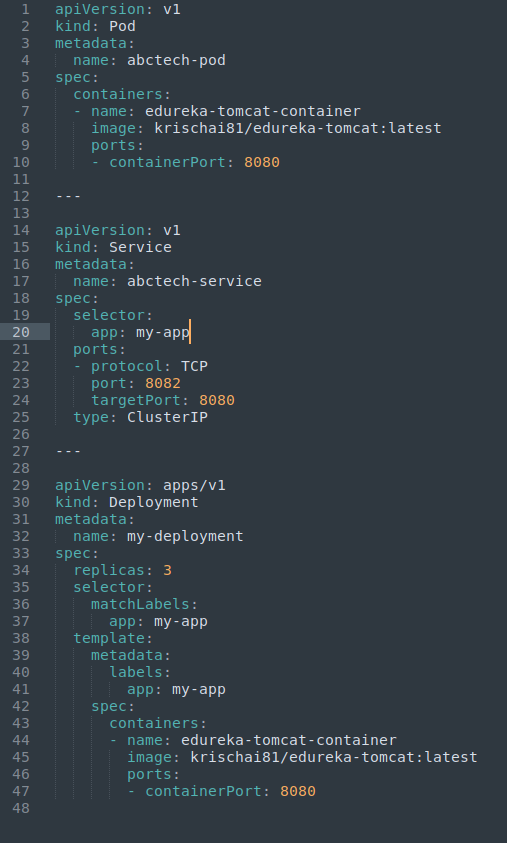






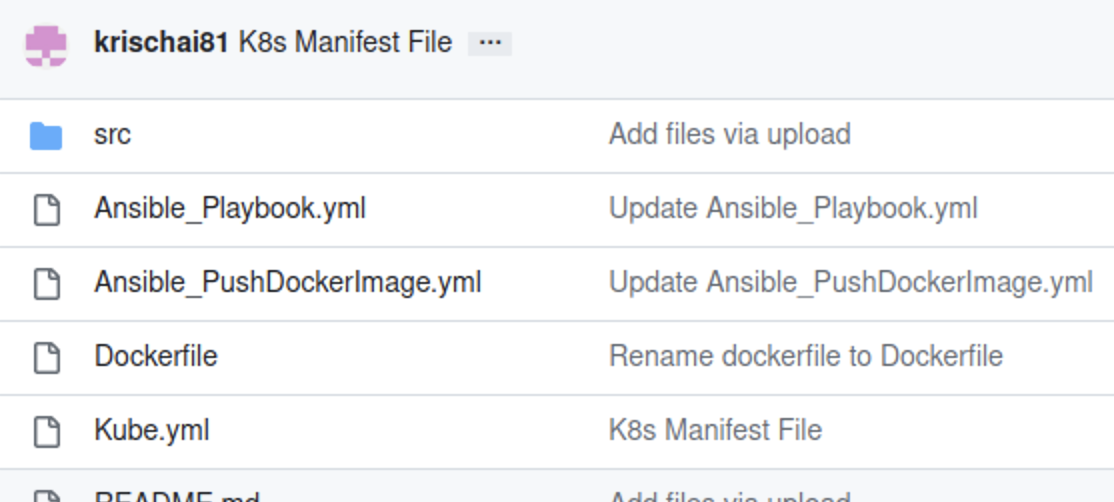
Step 27:

Create a manifest file to create pods, service and deployment and save it as Kube.yml



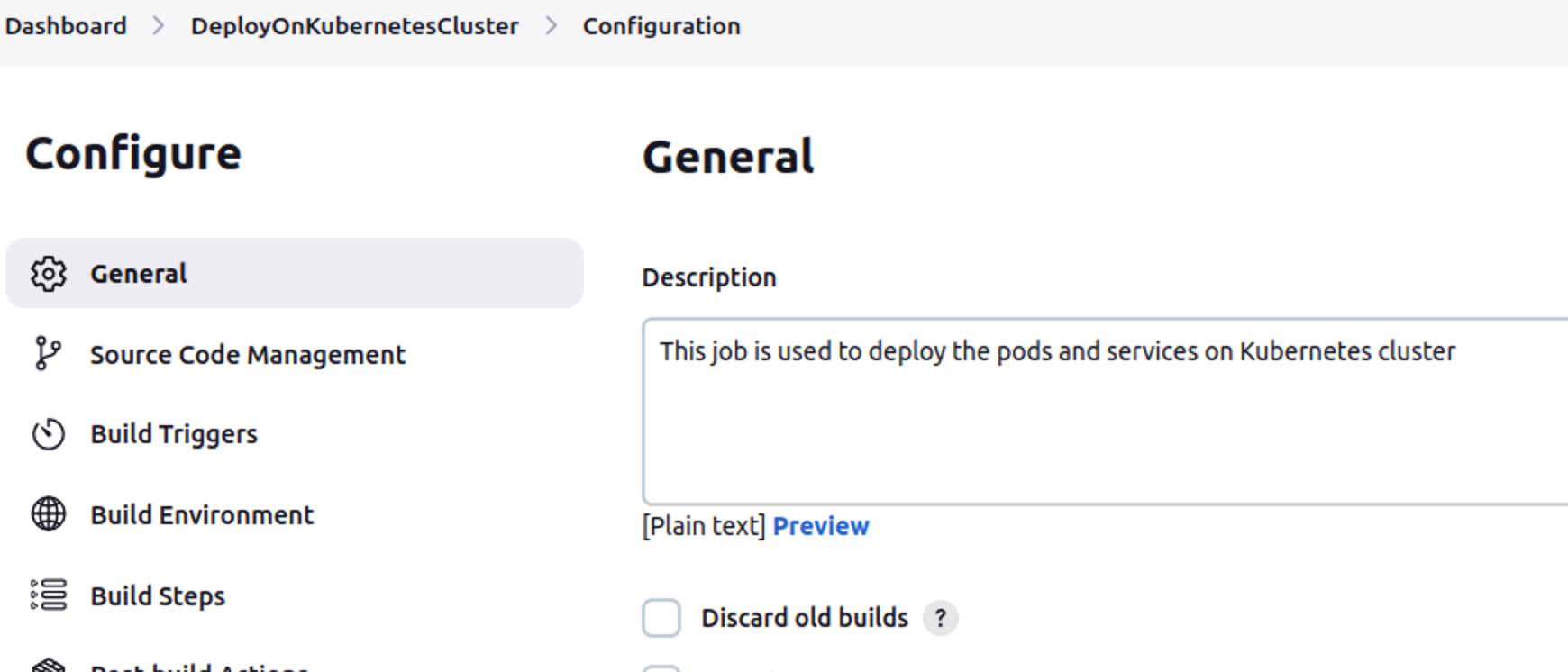
Step 28:

Upload the Kube.yml to the github repo



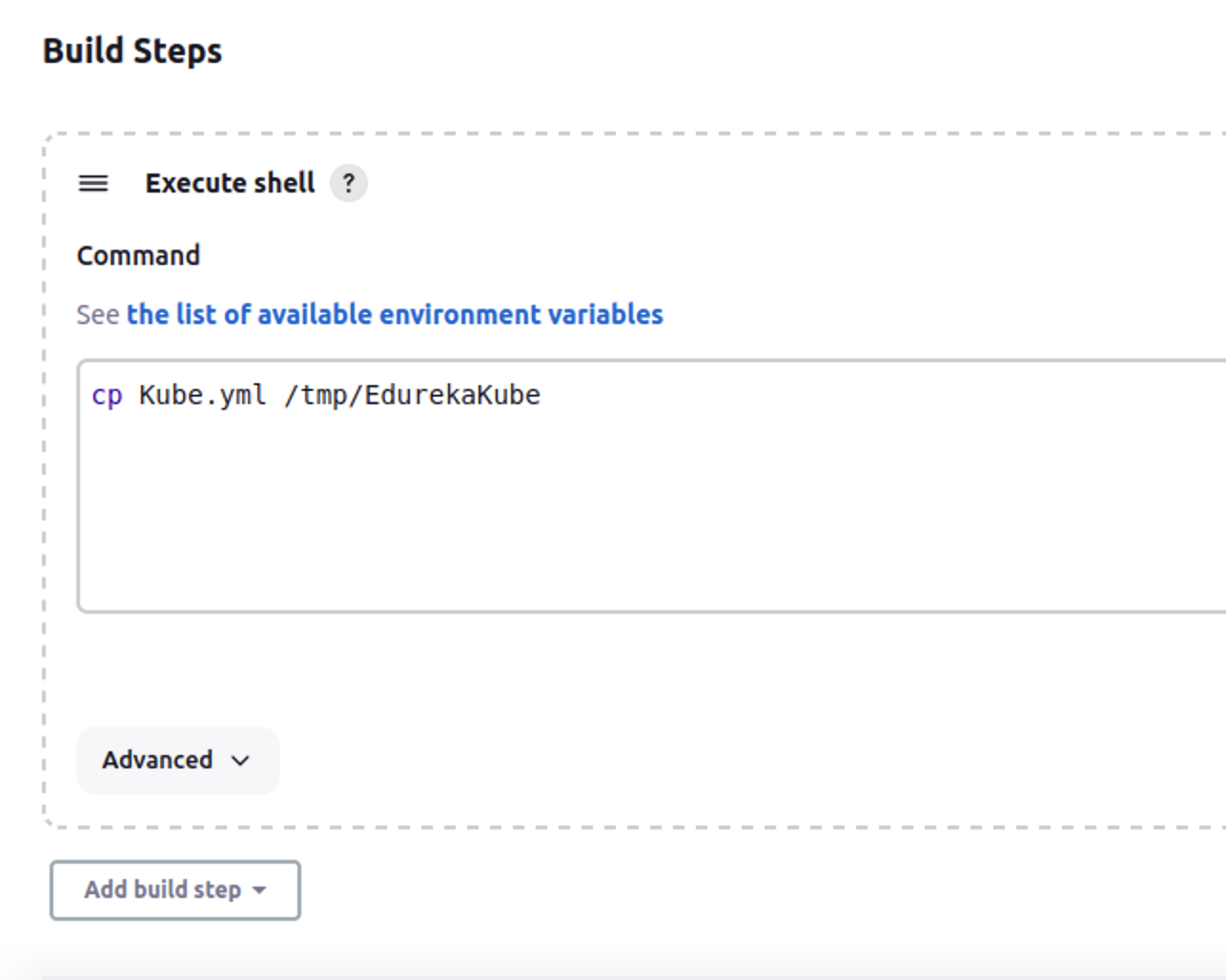
Step 29:

Create a Jenkins job to deploy the pods and services to the Kubernetes cluster as shown below



Step 30:

Configure the job to get the Kube.yml from the github repo and move the file to a local directory /tmp/EdurekaKube/ as shown below



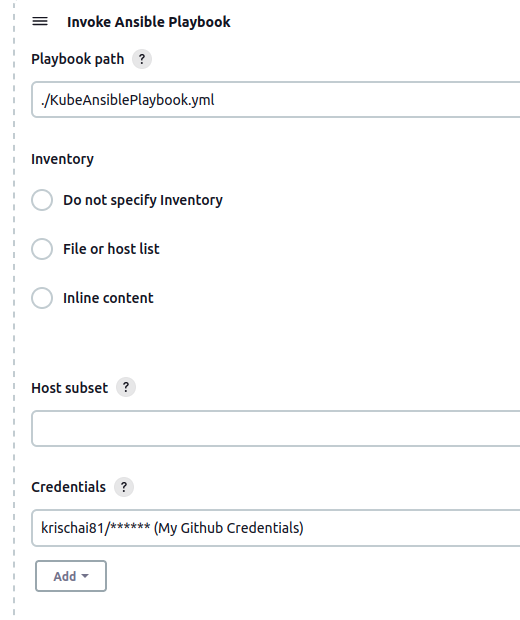
Step 31:

Create an ansible playbook to integrate Kubernetes with Ansible as shown below.



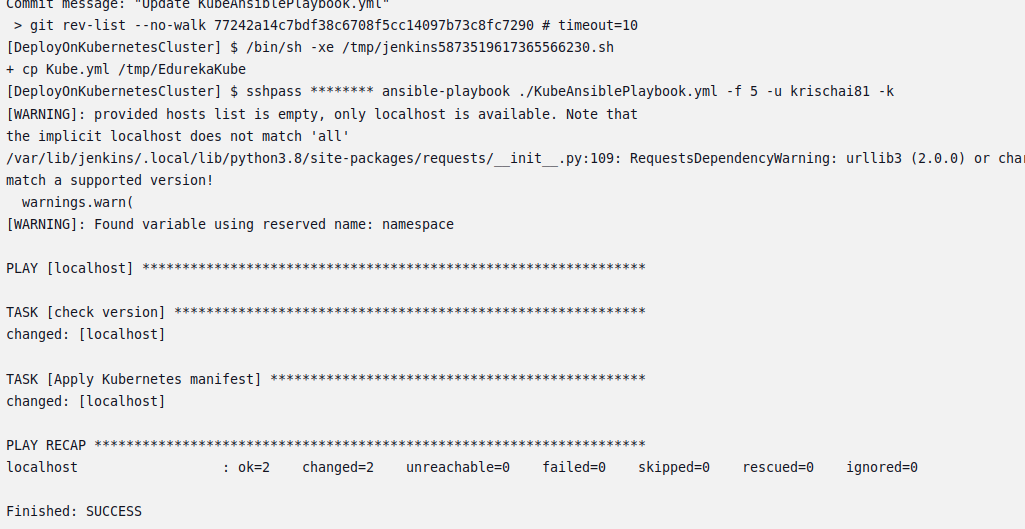
Step 32:

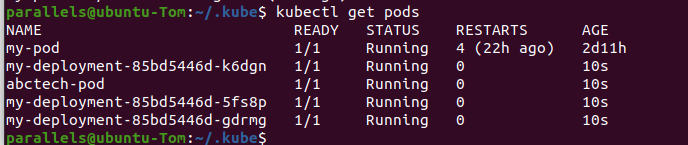
Upload the file in github repo as KubeAnsiblePlaybook.yml and invoke the file from the Jenkins job as shown below:



Step 33:

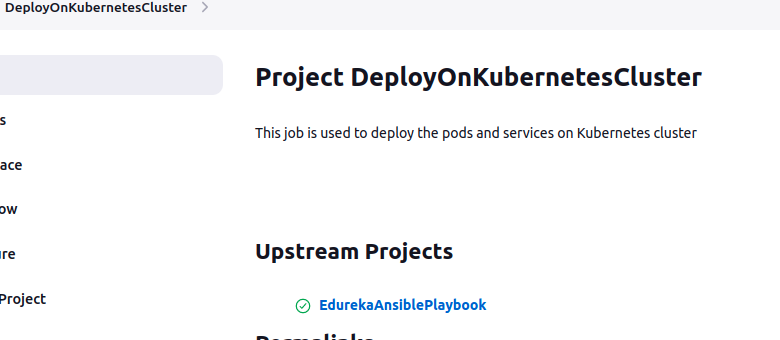
Build this job and Confirm that the pods and deployments are created.





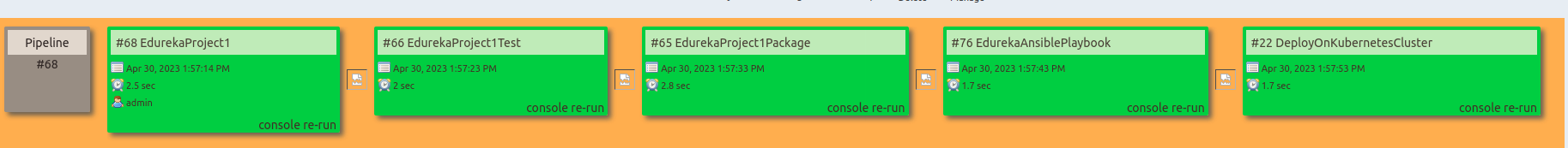
Step 34:

In the Jenkins job, make sure that DeployOnKubernetesCluster is executed only after the package job “EdurekaAnsiblePlaybook” is completed (i.e it is an upstream project)



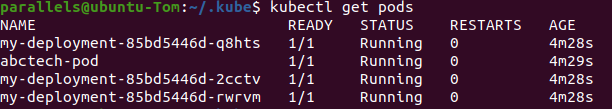
Step 35:

Execute the Jenkins Pipeline to run all the jobs in a pipeline



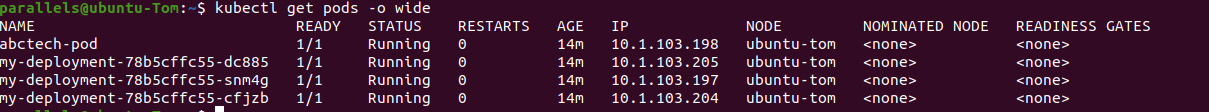
Step 36:

Confirm that the pods are running



Step 37:

Get the deployments ip addresses using the below command as shown in pic



Step 38:

Now access the deployment using the ip addresses shown above

**Deployment 1**



**Deployment 2**



**Deployment 3**



**Task 5**

**Task Description:**  
Using Prometheus, monitor the resources like CPU utilization: Total Usage, Usage per core, usage breakdown, memory, and network on the instance by providing the endpoints on the local host. Install the node exporter and add the URL to the target in Prometheus. Using this data, log in to Grafana and create a dashboard to show the metrics.

Step 39: Install node\_exporter by following the steps below

1. Download the node\_exporter-1.5.0.linux-arm64.tar.gz to your local directory in Ubuntu. Note: arm64 architecture is used since this is running on an M1 Mac machine.
2. Extract the tar file using the command “tar xzf node\_exporter-1.5.0.linux-arm64.tar.gz”
3. Copy the node\_exporter binary file from the extracted files to /usr/local/bin

Step 40: Install Prometheus by following the steps below

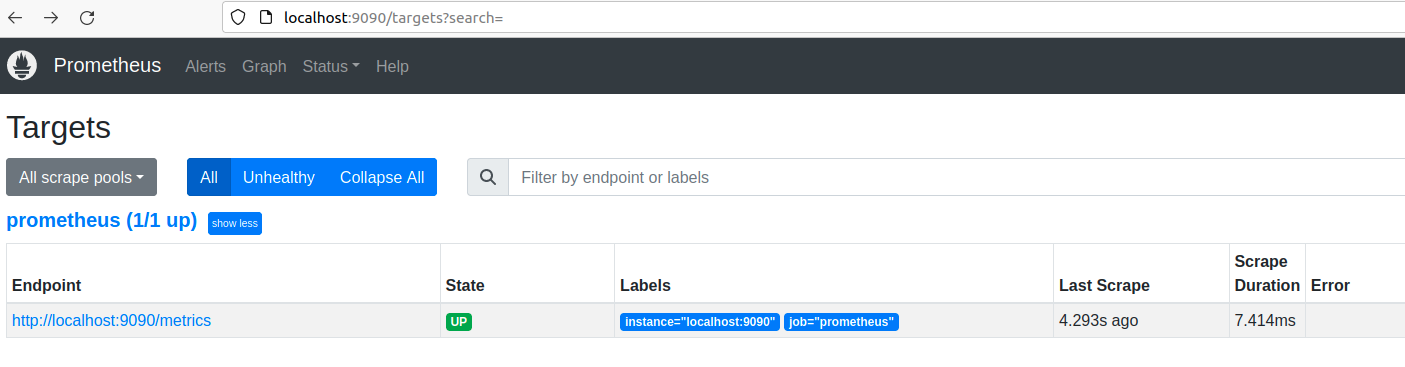
1. Download the prometheus-2.44.0-rc.0.linux-arm64.tar.gz to your local directory. In Ubuntu
2. Extract the tar file using the command “tar xzf prometheus-2.44.0-rc.0.linux-arm64.tar.gz”
3. Copy the “Prometheus” binary file from the extracted files to /usr/local/bin
4. Make a director “Prometheus” under /etc and copy the prometheus.yml file from the extracted files to /etc/prometheus directory
5. Make sure the .yml file has 9090 mapped in the scrape\_configs



Step 41: Execute both node\_exporter and prometheus by following the below steps

1. Navigate to /usr/local/bin and execute “./node\_exporter”
2. Open a different terminal and navigate to /usr/local/bin and execute “./prometheus”

Step 42: Launch Prometheus using the command <http://localhost:9090/>

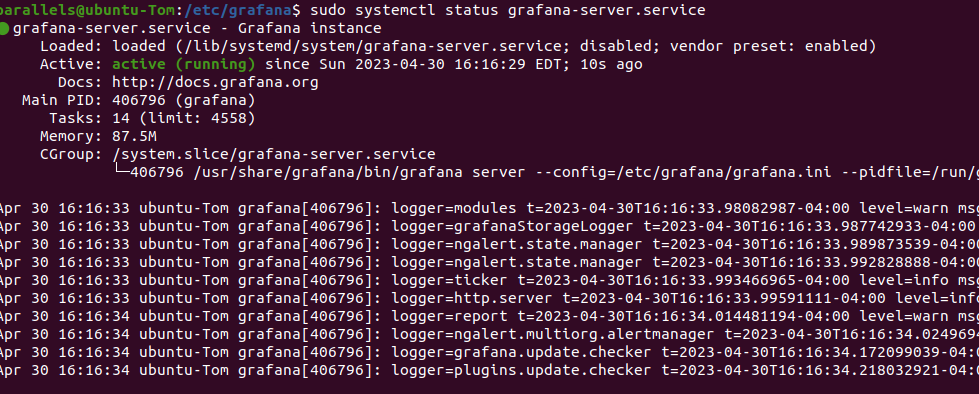


Step 43: Install Grafana on Ubuntu using the below steps (as shown in Grafana website)

1. sudo apt-get install -y adduser libfontconfig1
2. wget <https://dl.grafana.com/enterprise/release/grafana-enterprise_9.5.1_arm64.deb>
3. sudo dpkg -i grafana-enterprise\_9.5.1\_arm64.deb

Step 44: Start Grafana server and check the status

1. Start the Grafana server by running the command “sudo systemctl start grafana-server”
2. Check the status of the server by executinh the command “sudo systemctl status grafana-server.service”
3. Check if the status is running for the grafana service

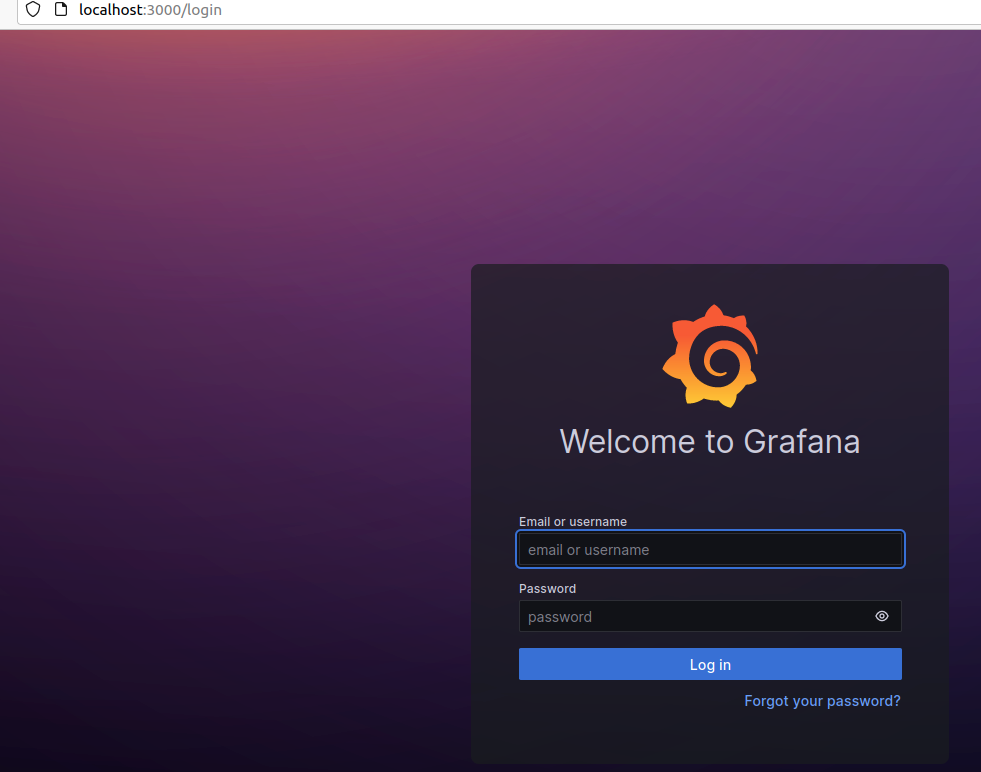


Step 45: Access the Grafana service through the web browser

1. Check the port Grafana service is running using the command “sudo netstat -plnt | grep grafana”



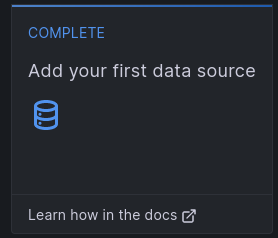
1. Access Grafana service from the web browser using the port 3000 displayed above

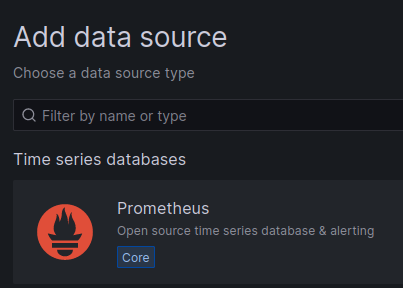


1. Login with default username and password admin/admin

Step 46: Create a dashboard and visualize

1. Add your first data source from the home page and select Prometheus as shown below





1. Enter the name for the data source in the Name field as shown below and provide the url for Prometheus (here it is local host)



1. Import the node\_exporter JSON config file for the Grafana Dashboard and select the data source as Prometheus.
2. View the DashboardGraphical user interface, application

   Description automatically generated