Project 1

Building a CI/CD Pipeline for a Retail Company

I would like to thank Edureka for the Devops Course, it was a great learning experience.

I have chosen project1.

Business Challenge/Requirement:

ABC Technologies is a leading online retail store, and it has recently acquired a large retail offline business store. The business store has a large number of stores across the globe but is following the conventional pattern of development and deployment. As a result, it has landed at a great loss and is facing the following challenges.

● Low available

● Low scalable

● Low performance

● Hard to built and maintain

● Developing and deploying are time-consuming

ABC will acquire the data from all these storage systems and plans to use it for analytics and prediction of the firm's growth and sales prospects. In the first phase, ABC has to create the servlets to add a product and display product details. Add servlet dependencies required to compile the servlets. Create an HTML page that will be used to add a product. The team is using Git to keep all the source code.

ABC has decided to use the DevOps model. Once source code is available in GitHub, we need to integrate it with Jenkins and provide continuous build generation for continuous delivery as well as integrate with Ansible and Kubernetes for deployment. Use Docker Hub to pull and push images between Ansible and Kubernetes.

Problem Statements/Tasks:

We need to develop a CI/CD pipeline to automate the software development, testing, packaging, and deployment, reducing the time to market the app and ensuring good quality service is experienced by end users. In this project, we need to—

● push the code to our GitHub repository.

● create a continuous integration pipeline using Jenkins to compile, test, and package the code present in GitHub.

● Write Dockerfile to push the war file to the Tomcat server.

● Integrate Docker with Ansible and write the playbook.

● Deploy artifacts to the Kubernetes cluster

• Monitor resources using Grafana.

Approach to Solve:

Task 1: Clone the project from the GitHub link shared in resources to your local machine. Build the code using Maven commands.

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

Task 3: Write a Docket 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.

Task 4: Integrate the Docker host with Ansible. Write an Ansible playbook to create an image and create a continuer. Integrate Ansible with Jenkins. Deploy Ansible-playbook. CI/CD job to build code on ansible and deploy it on docker container

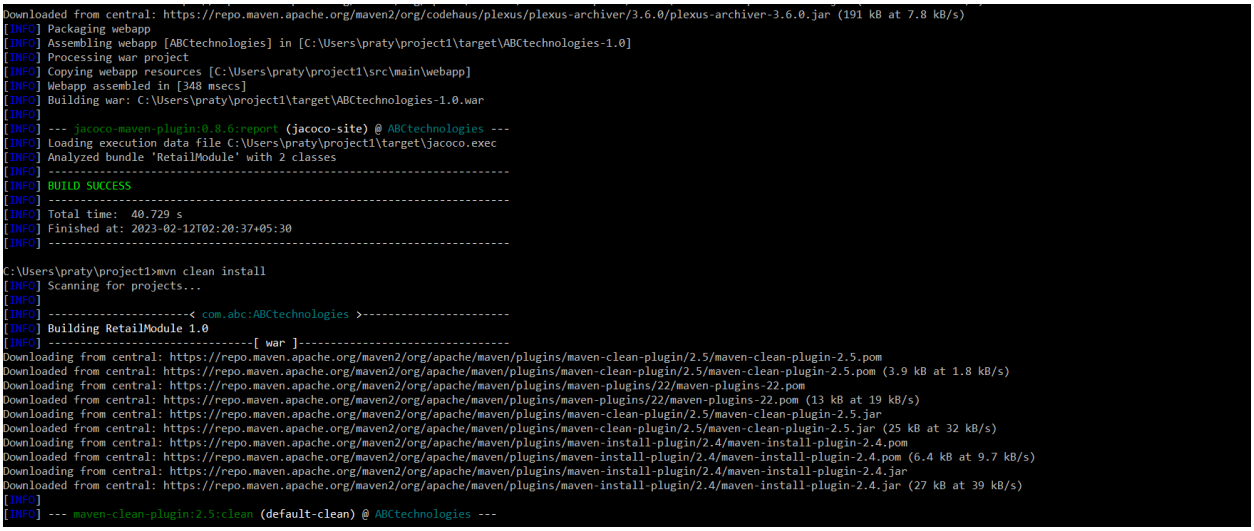
1. Deploy Artifacts on Kubernetes
2. Write pod, service, and deployment manifest file
3. Integrate Kubernetes with Ansible
4. Ansible playbook to create deployment and service

Task 5: Using Prometheus, monitor the resources like CPU utilisation: 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.

Below are the step by step process that I followed to complete each task :-

Task 1: Clone the project from the GitHub link shared in resources to your local machine. Build the code using Maven commands.

Mvn Clean Install Screenshot



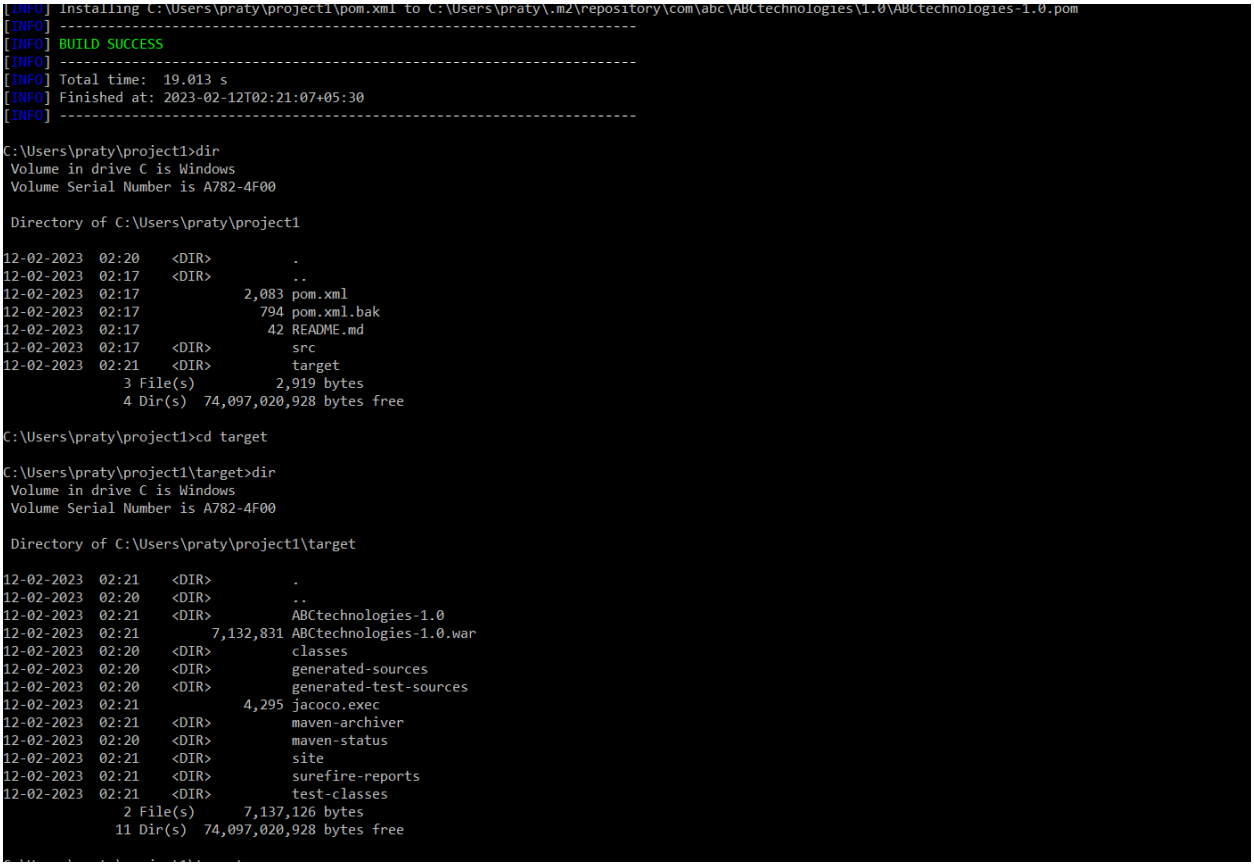
Mvn Test job screenshot

```
Downloaded from central: https://repo.maven.apache.org/maven2/com/google/collections/google-collections/1.0/google-collections-1.0.jar (640 kB at 148 kB/s)
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 3 source files to C:\Users\praty\project1\target\classes
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  01:21 min
[INFO] Finished at: 2023-02-12T02:19:12+05:30
[INFO] ------------------------------------------------------------------------

C:\Users\praty\project1>mvn test
[INFO] Scanning for projects...
[INFO]
[INFO] ----------------------< com.abc:ABCtechnologies >-----------------------
[INFO] Building RetailModule 1.0
[INFO] --------------------------------[ war ]---------------------------------
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-surefire-plugin/2.12.4/maven-surefire-plugin-2.12.4.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-surefire-plugin/2.12.4/maven-surefire-plugin-2.12.4.pom (10 kB at 5.2 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/surefire/surefire/2.12.4/surefire-2.12.4.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/surefire/surefire/2.12.4/surefire-2.12.4.pom (14 kB at 20 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-surefire-plugin/2.12.4/maven-surefire-plugin-2.12.4.jar
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-surefire-plugin/2.12.4/maven-surefire-plugin-2.12.4.jar (30 kB at 43 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/junit/junit/4.4/junit-4.4.jar
Downloaded from central: https://repo.maven.apache.org/maven2/junit/junit/4.4/junit-4.4.jar (161 kB at 154 kB/s)
[INFO]
[INFO] --- jacoco-maven-plugin:0.8.6:prepare-agent (jacoco-initialize) @ ABCtechnologies ---
[INFO] argLine set to -javaagent:C:\\Users\\praty\\.m2\\repository\\org\\jacoco\\org.jacoco.agent\\0.8.6\\org.jacoco.agent-0.8.6-runtime.jar=destfile=C:\\Users\\praty\\project1\\target\\jacoco.exec
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ ABCtechnologies ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory C:\Users\praty\project1\src\main\resources
```

Mvn Package screeshsot

```
-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running com.abc.dataAccessObject.Product1ImpTest
Tests run: 4, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.063 sec

Results :

Tests run: 4, Failures: 0, Errors: 0, Skipped: 0

[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  22.620 s
[INFO] Finished at: 2023-02-12T02:19:46+05:30
[INFO] ------------------------------------------------------------------------

C:\Users\praty\project1>mvn package
[INFO] Scanning for projects...
[INFO]
[INFO] ----------------------< com.abc:ABCtechnologies >-----------------------
[INFO] Building RetailModule 1.0
[INFO] --------------------------------[ war ]---------------------------------
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-war-plugin/3.2.2/maven-war-plugin-3.2.2.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-war-plugin/3.2.2/maven-war-plugin-3.2.2.pom (9.7 kB at 4.9 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-war-plugin/3.2.2/maven-war-plugin-3.2.2.jar
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-war-plugin/3.2.2/maven-war-plugin-3.2.2.jar (91 kB at 97 kB/s)
[INFO]
[INFO] --- jacoco-maven-plugin:0.8.6:prepare-agent (jacoco-initialize) @ ABCtechnologies ---
[INFO] argLine set to -javaagent:C:\\Users\\praty\\.m2\\repository\\org\\jacoco\\org.jacoco.agent\\0.8.6\\org.jacoco.agent-0.8.6-runtime.jar=destfile=C:\\Users\\praty\\project1\\target\\jacoco.exec
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ ABCtechnologies ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory C:\Users\praty\project1\src\main\resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ ABCtechnologies ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ ABCtechnologies ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory C:\Users\praty\project1\src\test\resources
[INFO]
```

Mvn output War screenshot



Git Clone Screeshot

```
C:\Users\praty>git clone https://github.com/vermakeshav1994/project1.git
Cloning into 'project1'...
remote: Enumerating objects: 24, done.
remote: Counting objects: 100% (24/24), done.
remote: Compressing objects: 100% (15/15), done.
remote: Total 24 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (24/24), 3.88 KiB | 39.00 KiB/s, done.

C:\Users\praty>cd project1

C:\Users\praty\project1>ls -lrth
'ls' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\praty\project1>dir
 Volume in drive C is Windows
 Volume Serial Number is A782-4F00

 Directory of C:\Users\praty\project1

12-02-2023  02:17    <DIR>          .
12-02-2023  02:17    <DIR>          ..
12-02-2023  02:17             2,083 pom.xml
12-02-2023  02:17               794 pom.xml.bak
12-02-2023  02:17                42 README.md
12-02-2023  02:17    <DIR>          src
               3 File(s)          2,919 bytes
               3 Dir(s)  74,137,915,392 bytes free
```

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

Approach I have followed :-

- Jenkins has been installed in master server already so i have created the 3 jobs for compile,test,package in jenkins and created pipeline with these 3 jobs and set up the agent machine(slave machine) and shared the load to agent as well.

- As a given project is based on java i have used maven to build the code and Jenkins is a build automation server that helps to automate these things so i have set up the java,maven paths of master in global tool configuration in jenkins and set up the jenkins goals and left git path as default.

- Tools location :

/opt/maven

/usr/lib/jvm/java-8-oracle

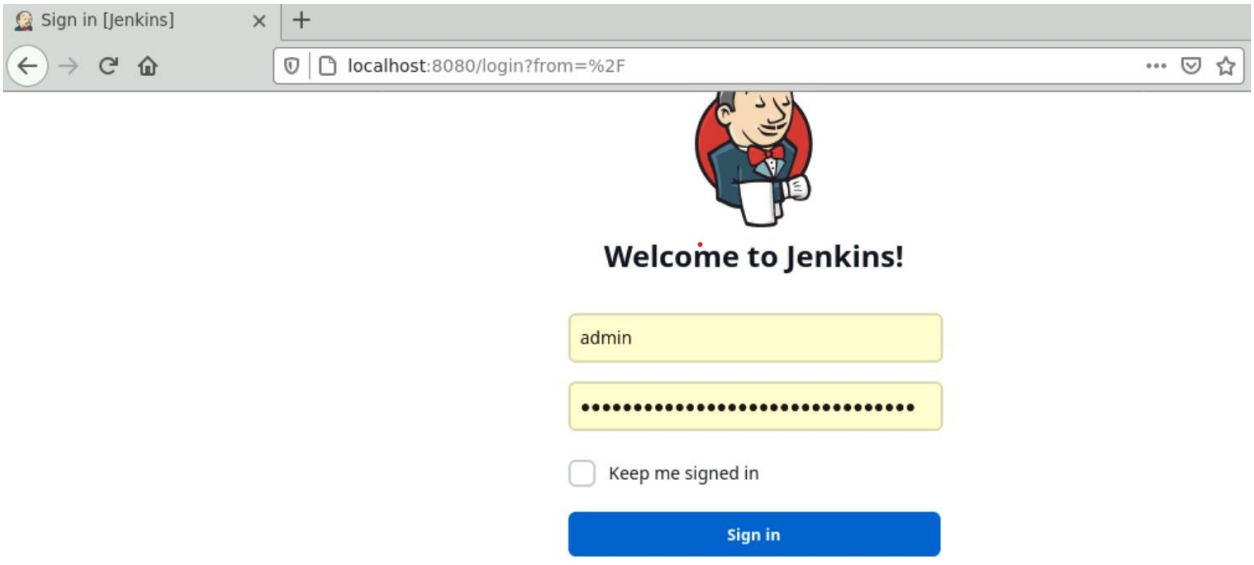
- Goals: Compile

Test

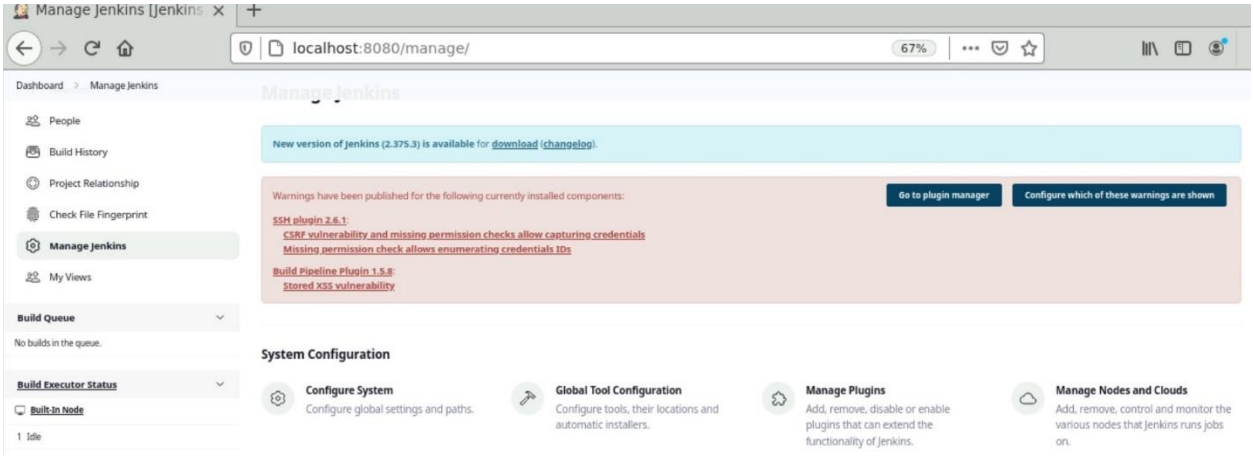
Package

Screenshots of the above task :-

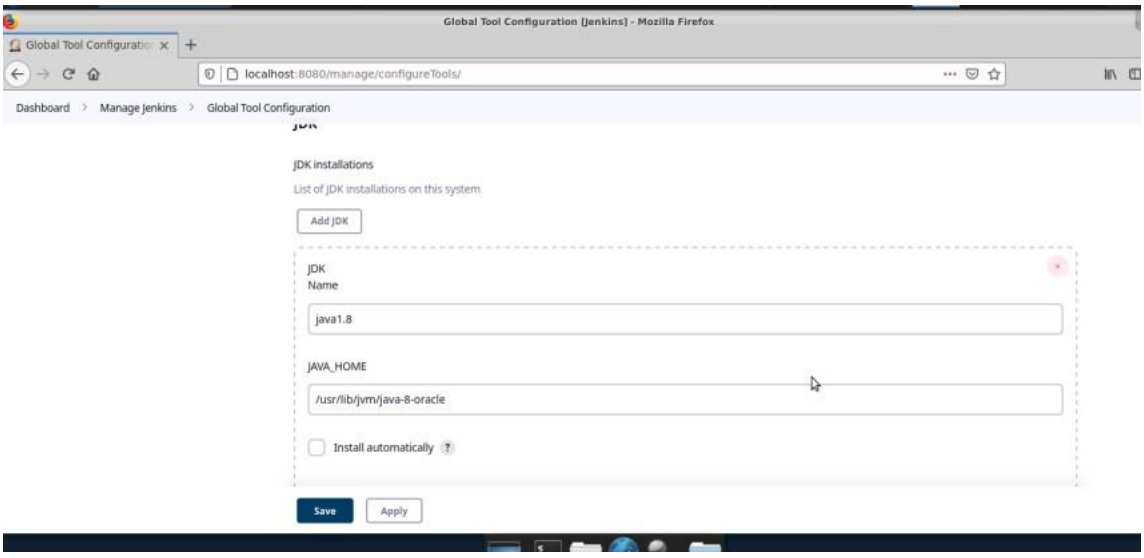
Jenkins Login:



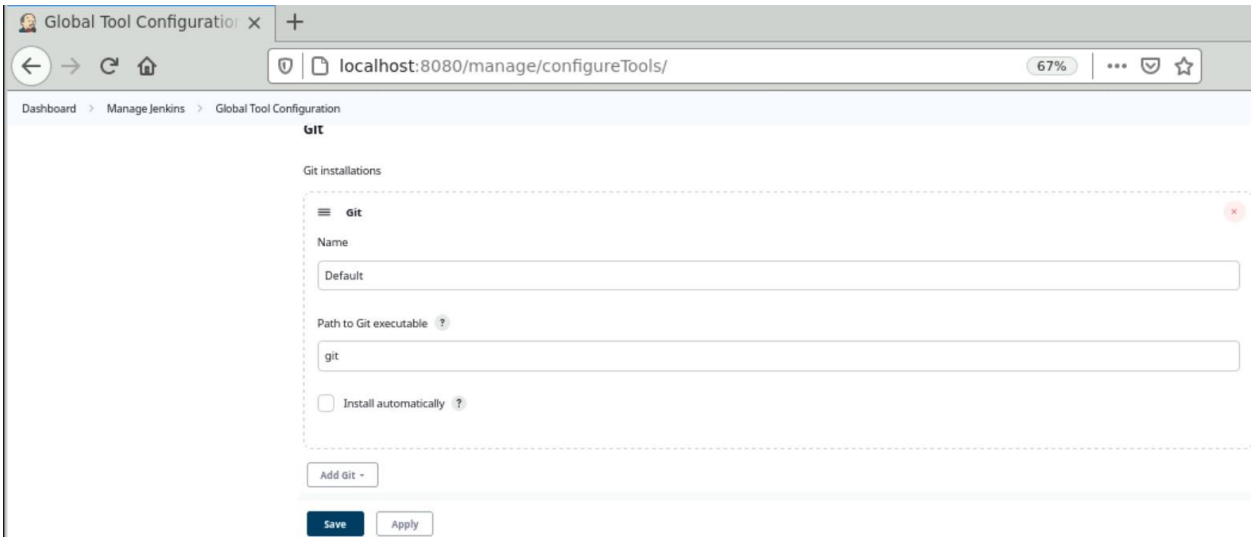
Global Tool Configuration:



Configuration for Java:

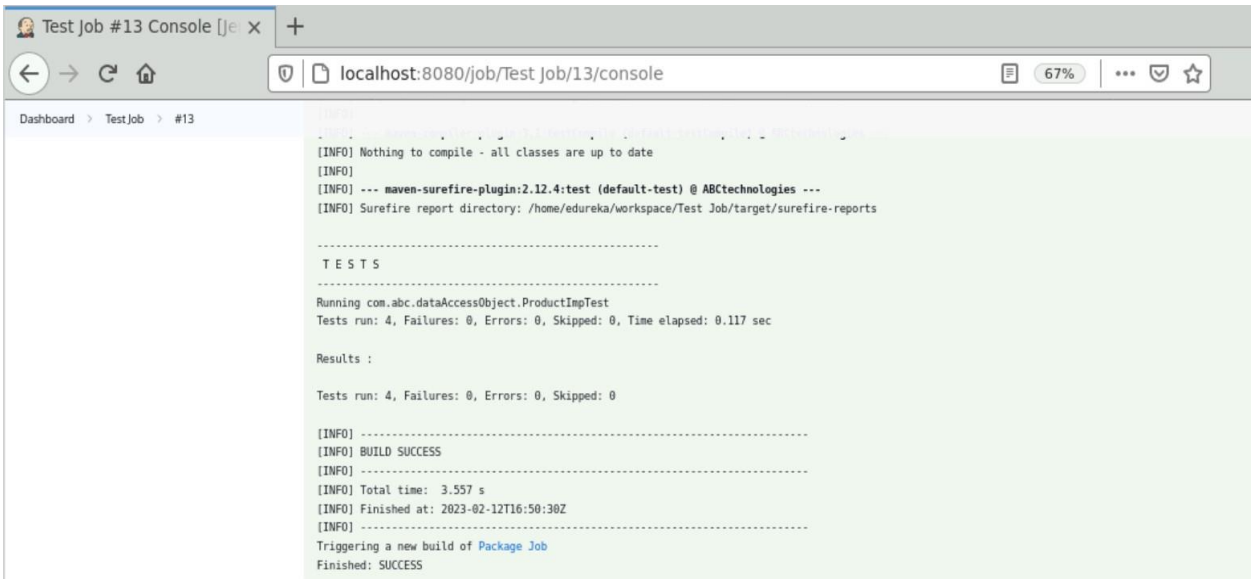


Confirguration for Git:

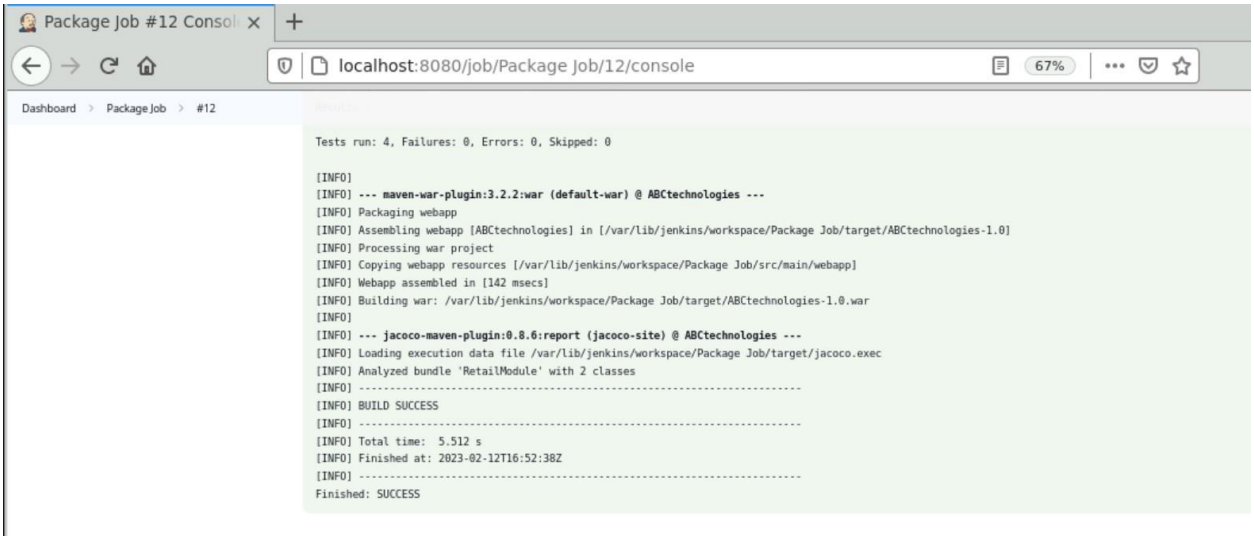


Compile job result in Jenkins

Test Job result in Jenkins :

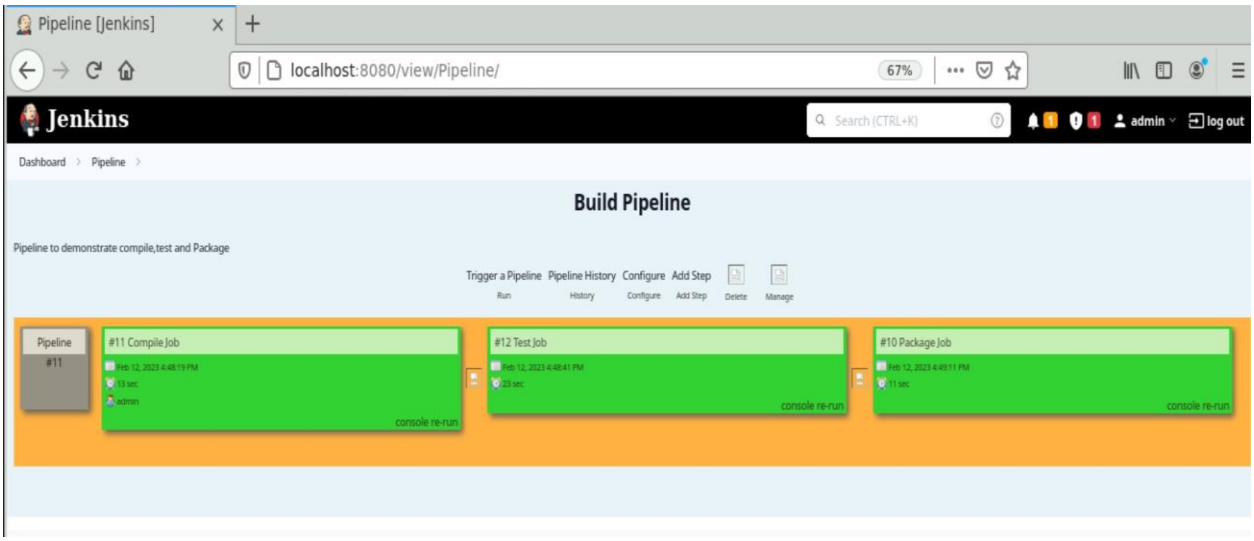


Package Job result in Jenkins:



Step 2: Execute the CI/CD pipeline to execute the jobs created in step 1

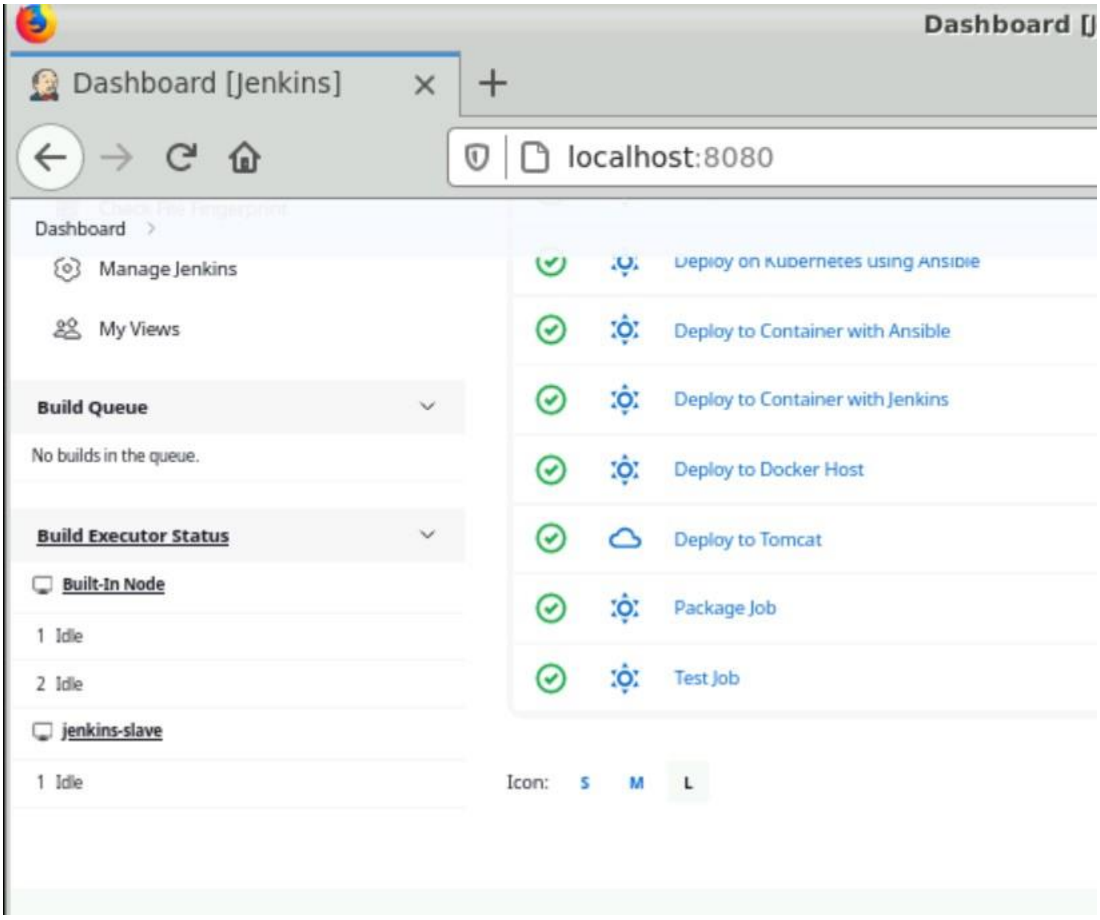
PipeLine Screenshot



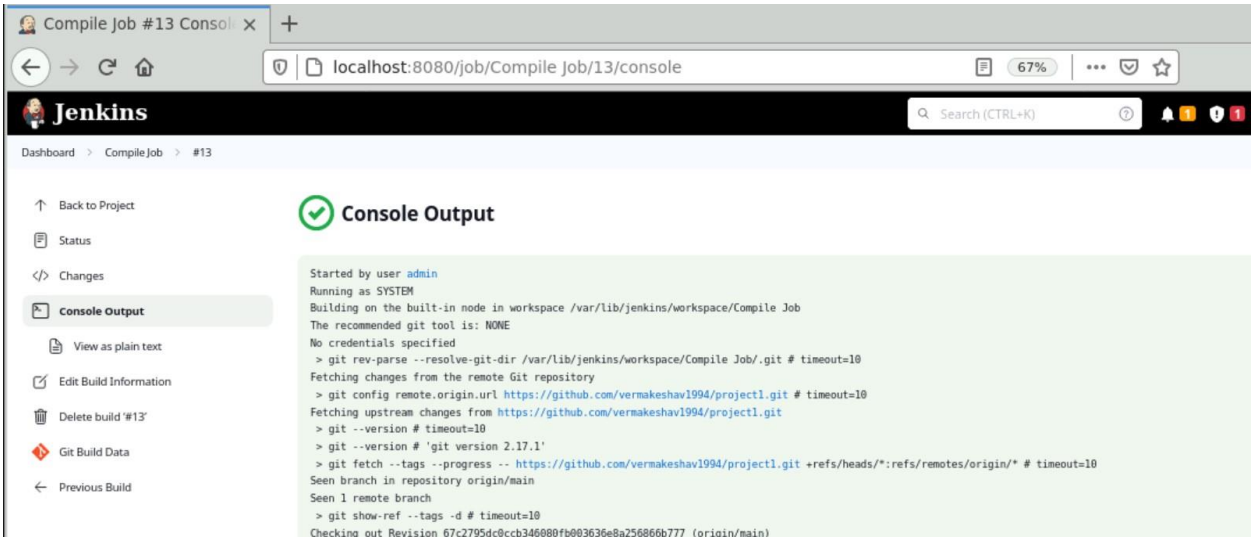
Step 3: Set up a master-slave node to distribute the tasks in the pipeline

Here, My Compile job will be running on Master Node. Master Node is the default Node so, I didn't change the name and it is showing as Built-In node.

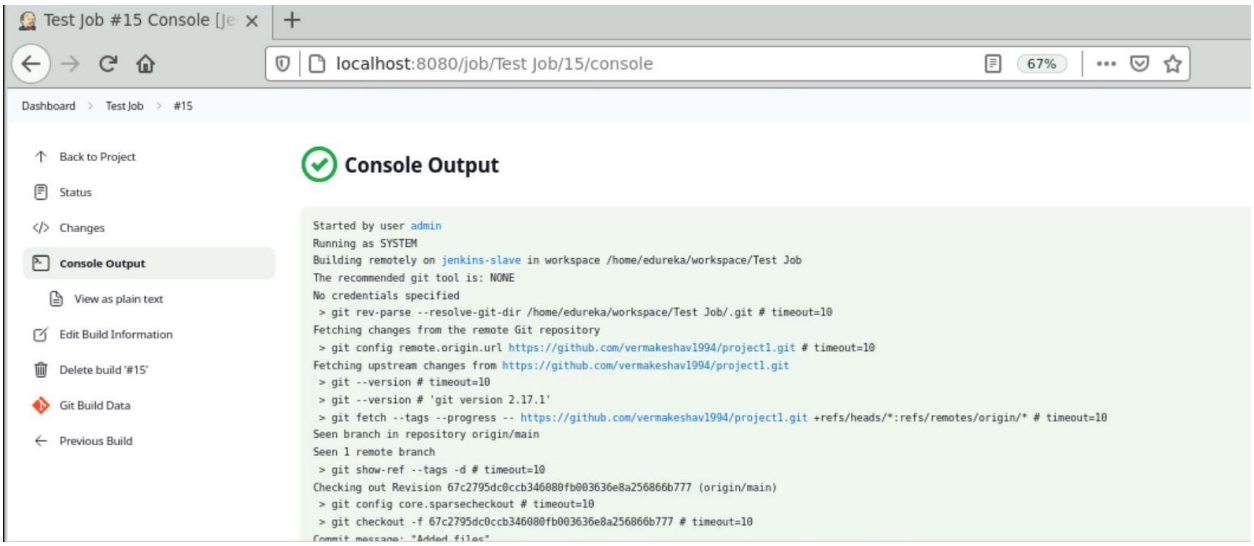
Test Job will be running on Slave Node and then again Package Job will be running on Master Node.



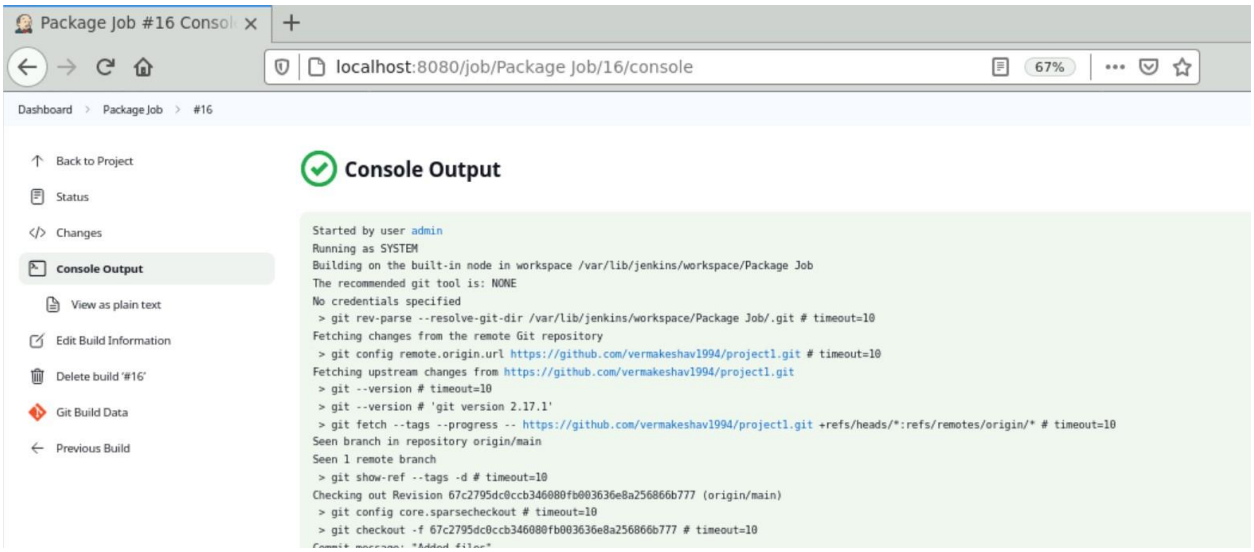
Screenshot of Compile Job running on Master :



Screenshot of Test Job running on Slave :



Screenshot of Package job running on Master :



Task 3: Write a Docket 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.

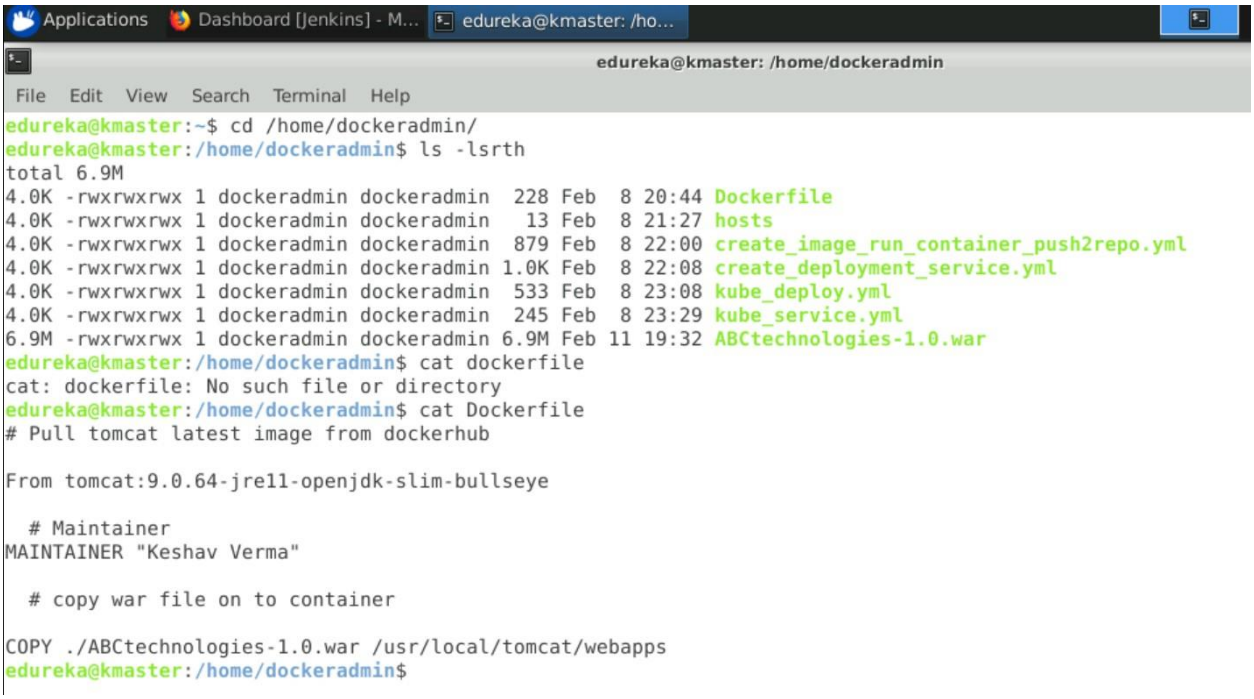
2. In the Docker image, add code to move the war file to the Tomcat server and build the image.

Approach i have followed: -

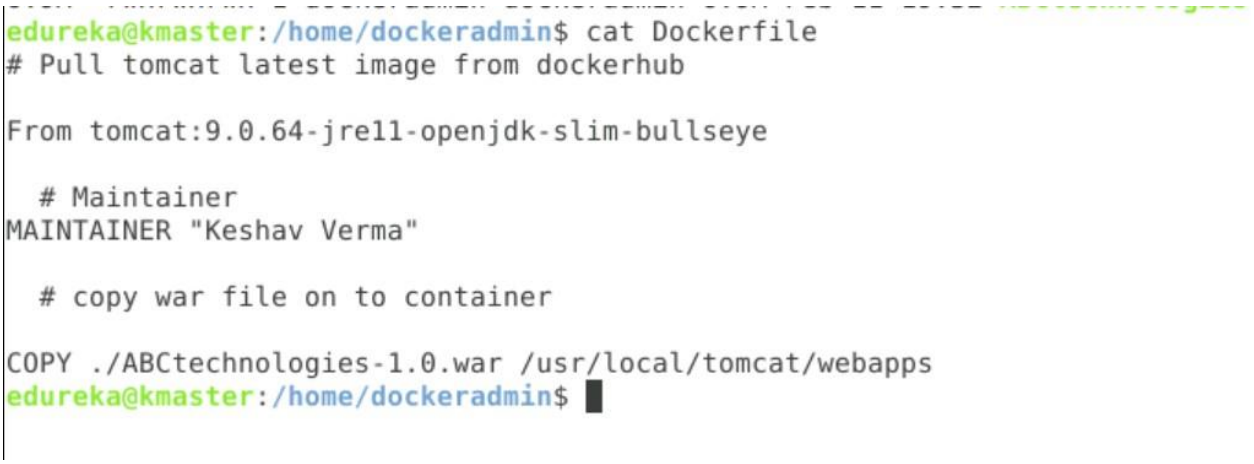
Jenkins and Docker are already installed in master machine and slave machine. - Now using earlier Jenkins package job with more enhancement with docker integration and doing two things here

- **i)** Deploying .war file generated from package job into the tomcat server
- **ii)** And creating the docker build and docker container with above .war file generated from package command and uploading the docker image to the docker hub and running this docker image as container.

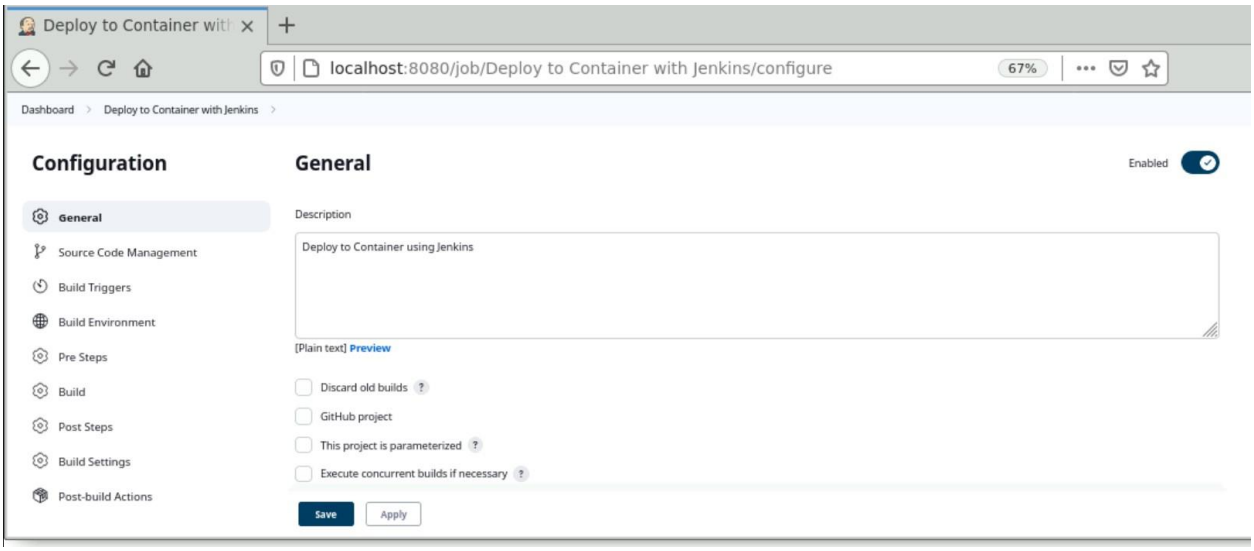
Screenshot of Docker File

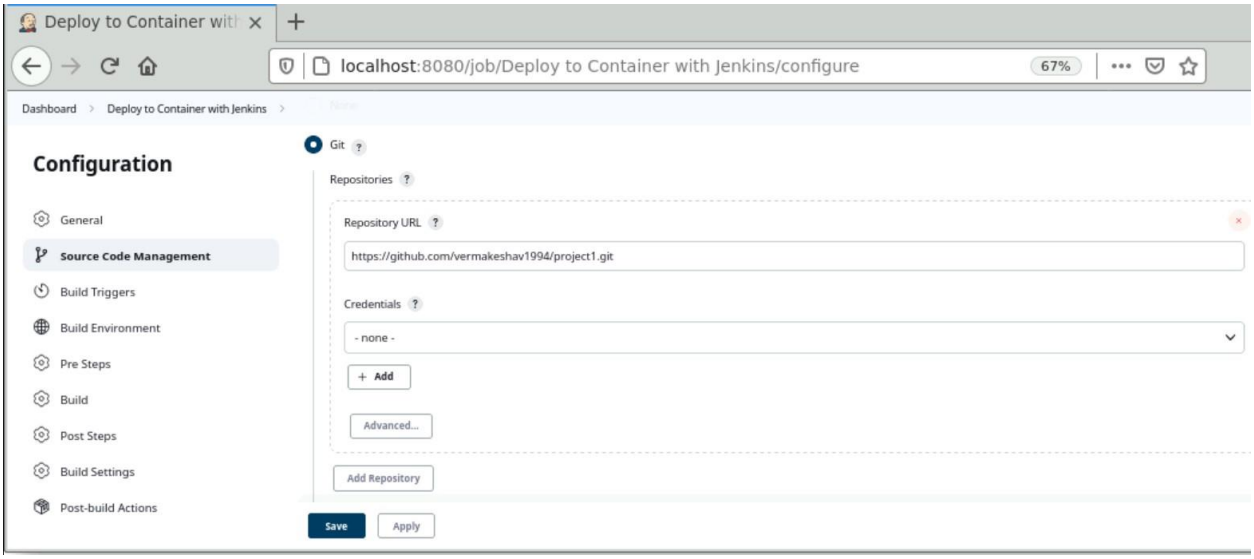


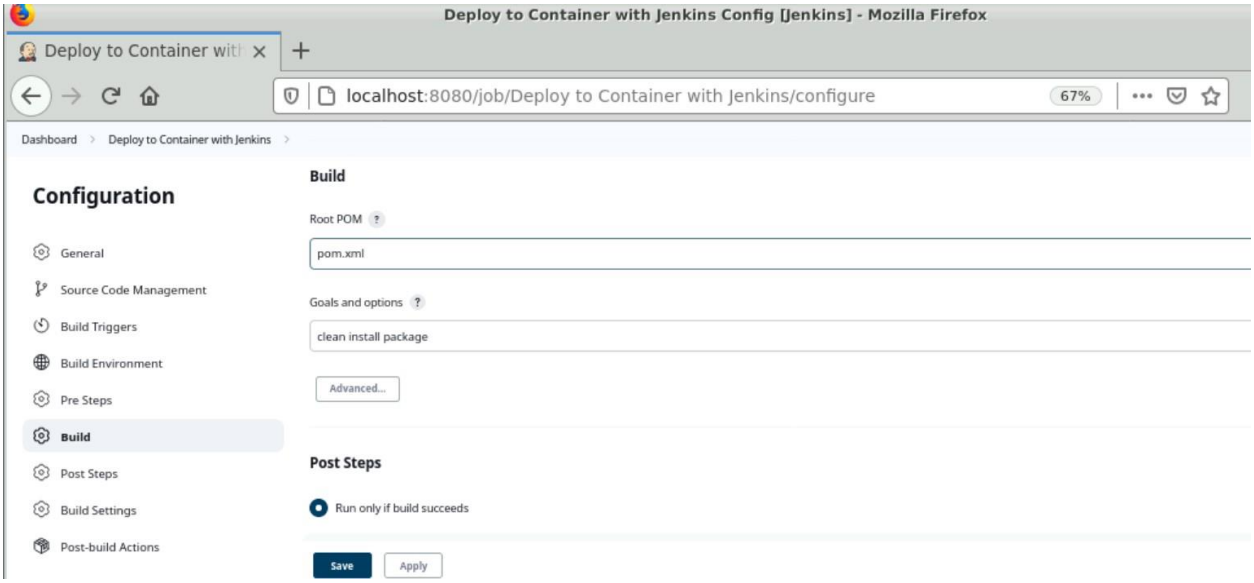
Screenshot of Code to move War file to Tomcat Server

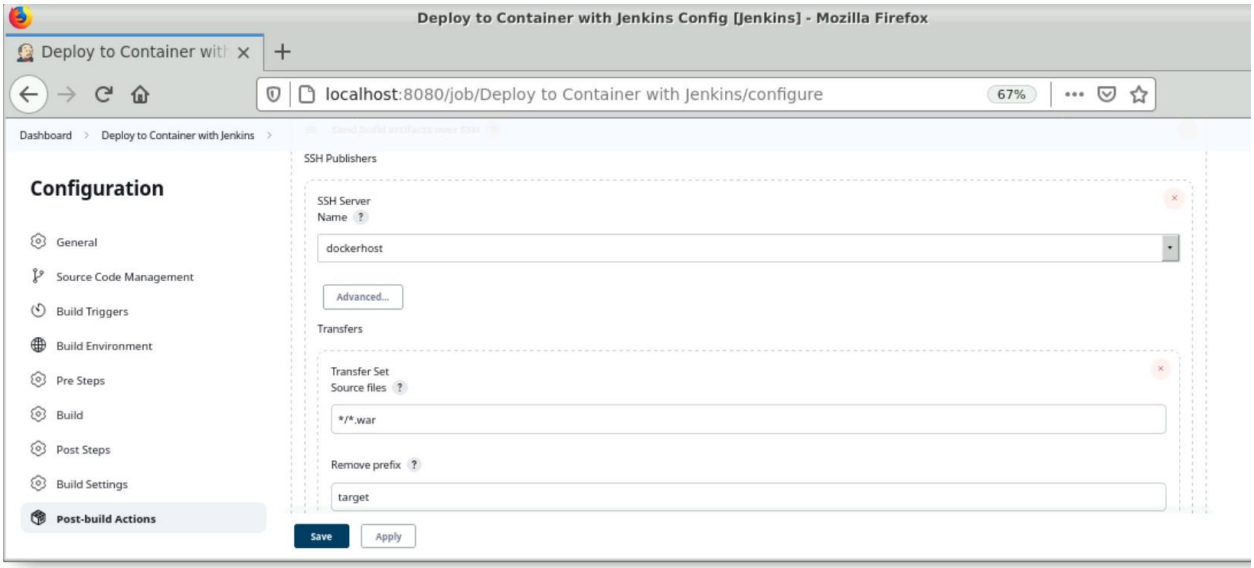


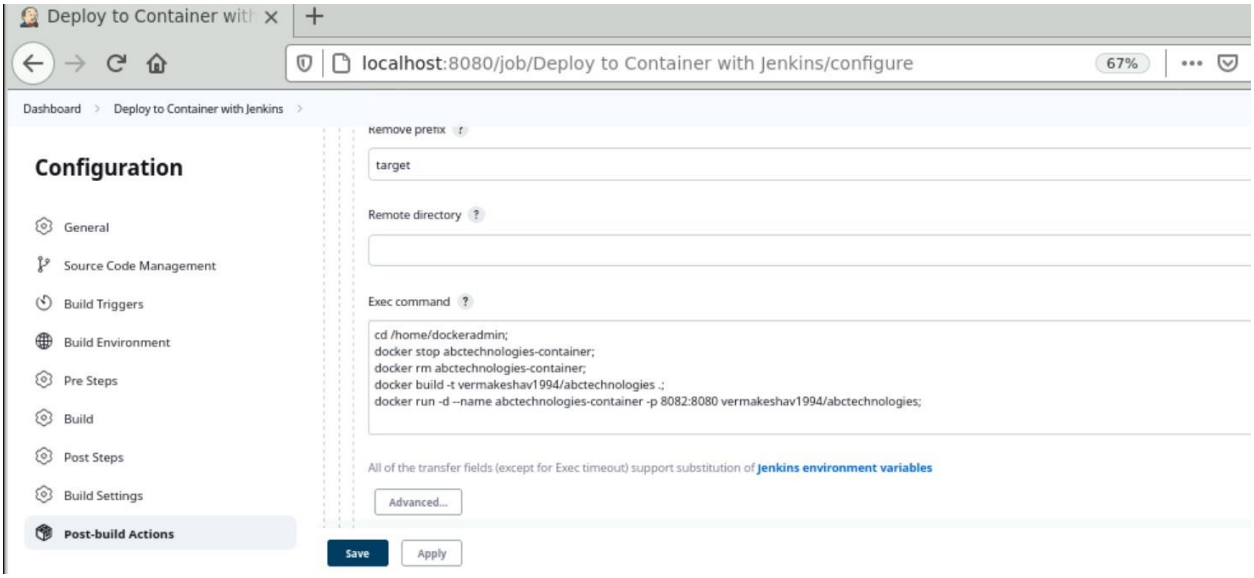
Screenshot of Deploying to Container with Jenkins Job Configuration :-


Deploy to Container with
localhost:8080/job/Deploy to Container with Jenkins/configure
67%
Dashboard
Deploy to Container with Jenkins
Configuration
General
Enabled
General
Source Code Management
Build Triggers
Build Environment
Pre Steps
Build
Post Steps
Build Settings
Post-build Actions
Description
Deploy to Container using Jenkins
[Plain text] Preview
Discard old builds
GitHub project
This project is parameterized
Execute concurrent builds if necessary
Save
Apply


Deploy to Container with
localhost:8080/job/Deploy to Container with Jenkins/configure
67%
Dashboard
Deploy to Container with Jenkins
Configuration
General
Source Code Management
Build Triggers
Build Environment
Pre Steps
Build
Post Steps
Build Settings
Post-build Actions
Git
Repositories
Repository URL
https://github.com/vermakeshav1994/project1.git
Credentials
- none -
Add
Advanced...
Add Repository
Save
Apply

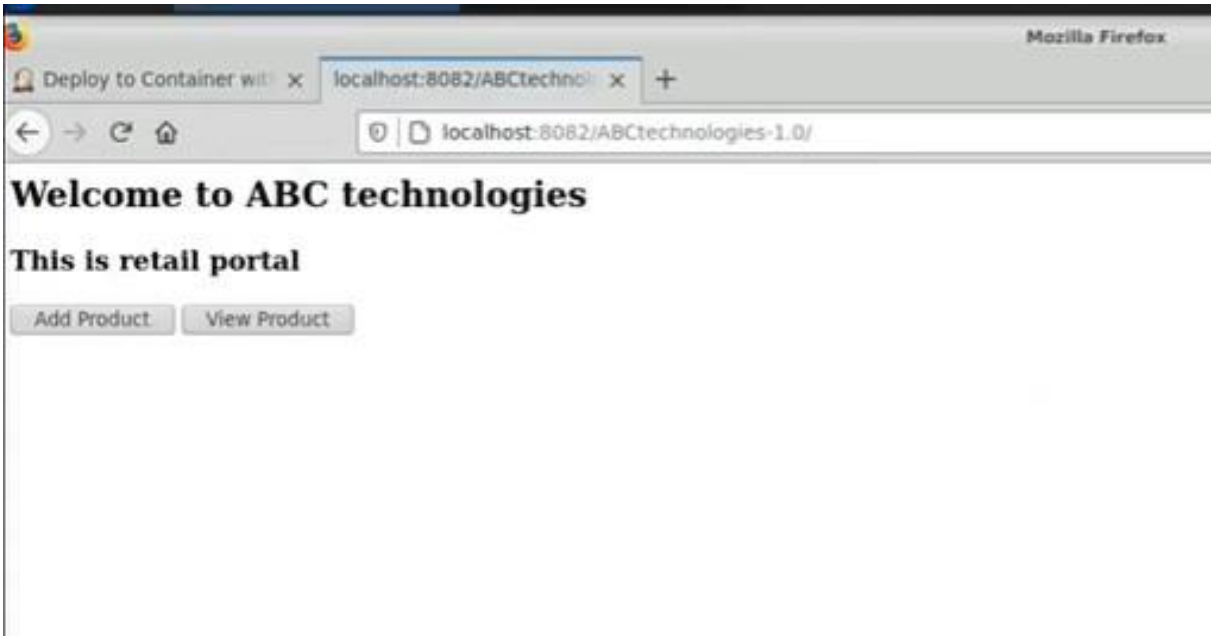

Deploy to Container with Jenkins Config [Jenkins] - Mozilla Firefox
Deploy to Container with
localhost:8080/job/Deploy to Container with Jenkins/configure
67%
Dashboard
Deploy to Container with Jenkins
Configuration
General
Source Code Management
Build Triggers
Build Environment
Pre Steps
Build
Post Steps
Build Settings
Post-build Actions
Build
Root POM
pom.xml
Goals and options
clean install package
Advanced...
Post Steps
Run only if build succeeds
Save
Apply


Deploy to Container with Jenkins Config [Jenkins] - Mozilla Firefox
Deploy to Container with
localhost:8080/job/Deploy to Container with Jenkins/configure
67%
Dashboard
Deploy to Container with Jenkins
SSH Publishers
Configuration
General
Source Code Management
Build Triggers
Build Environment
Pre Steps
Build
Post Steps
Build Settings
Post-build Actions
SSH Server
Name
dockerhost
Advanced...
Transfers
Transfer Set
Source files
/.war
Remove prefix
target
Save
Apply



Deploy to Container with Jenkins Job console Output :

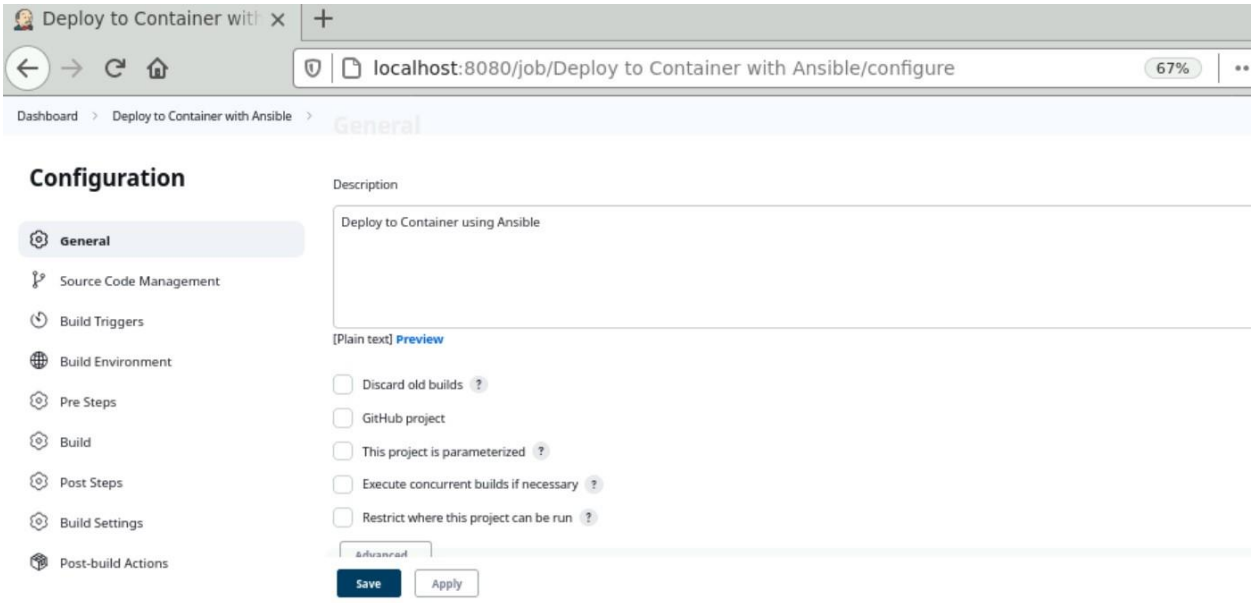
Deploy to container with Jenkins result

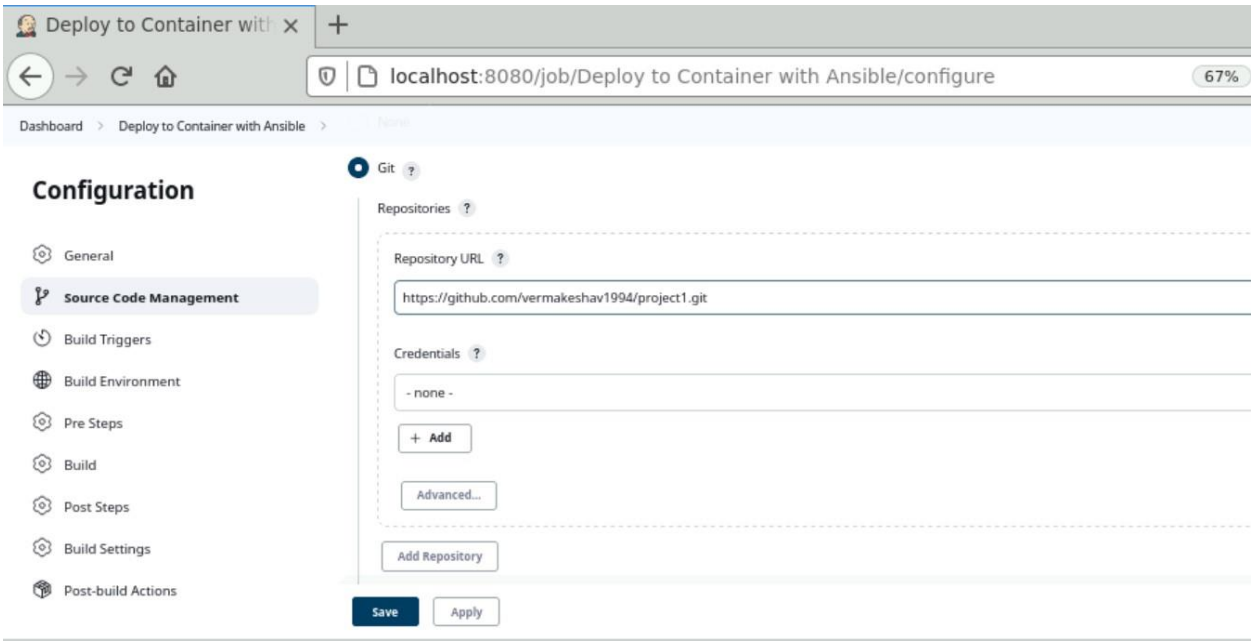


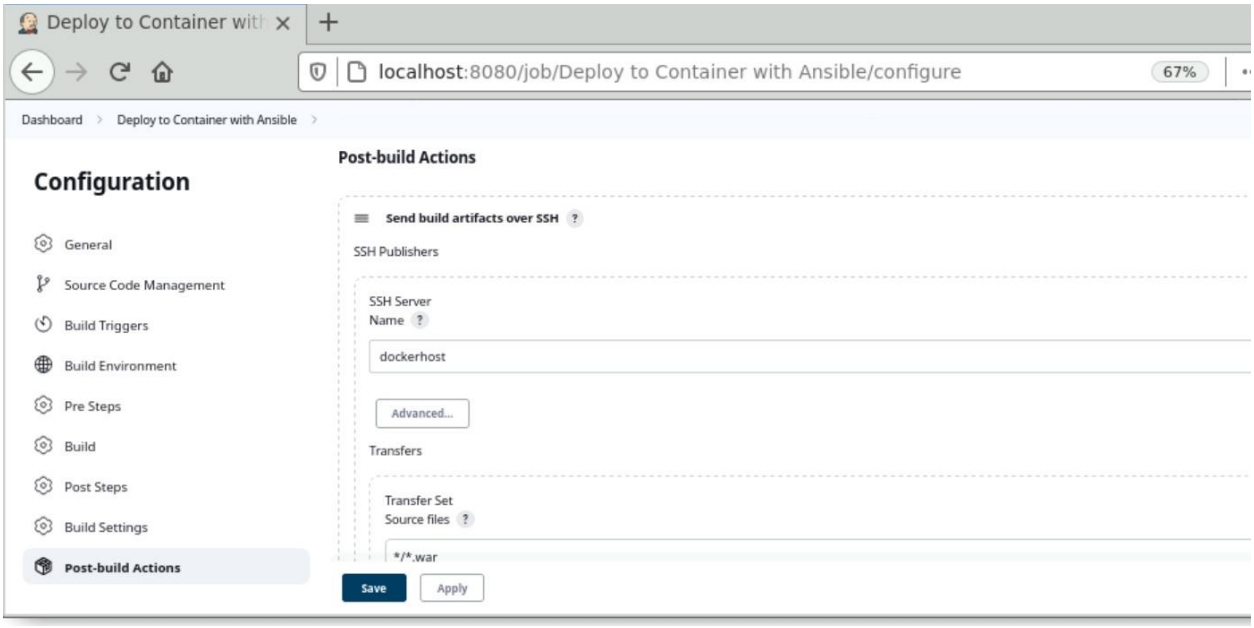
Task 4: Integrate the Docker host with Ansible. Write an Ansible playbook to create an image and create a continuer. Integrate Ansible with Jenkins. Deploy Ansible-playbook. CI/CD job to build code on ansible and deploy it on docker container

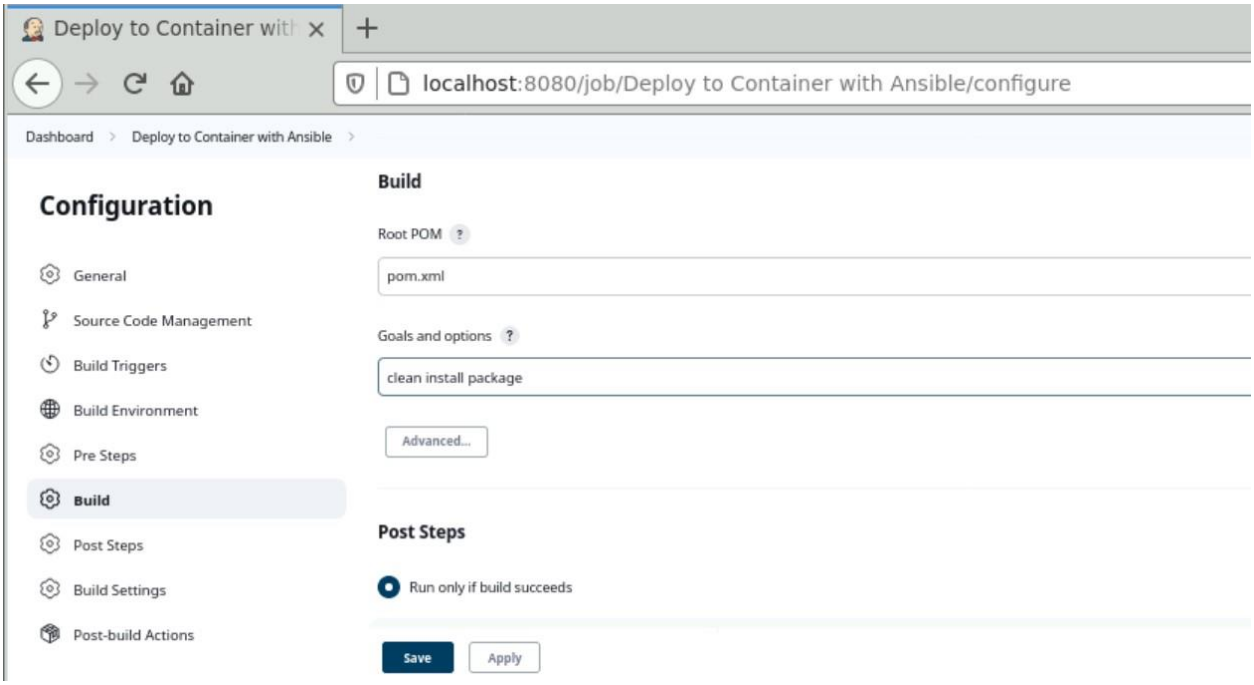
1. Deploy Artifacts on Kubernetes
2. Write pod, service, and deployment manifest file
3. Integrate Kubernetes with Ansible
4. Ansible playbook to create deployment and service

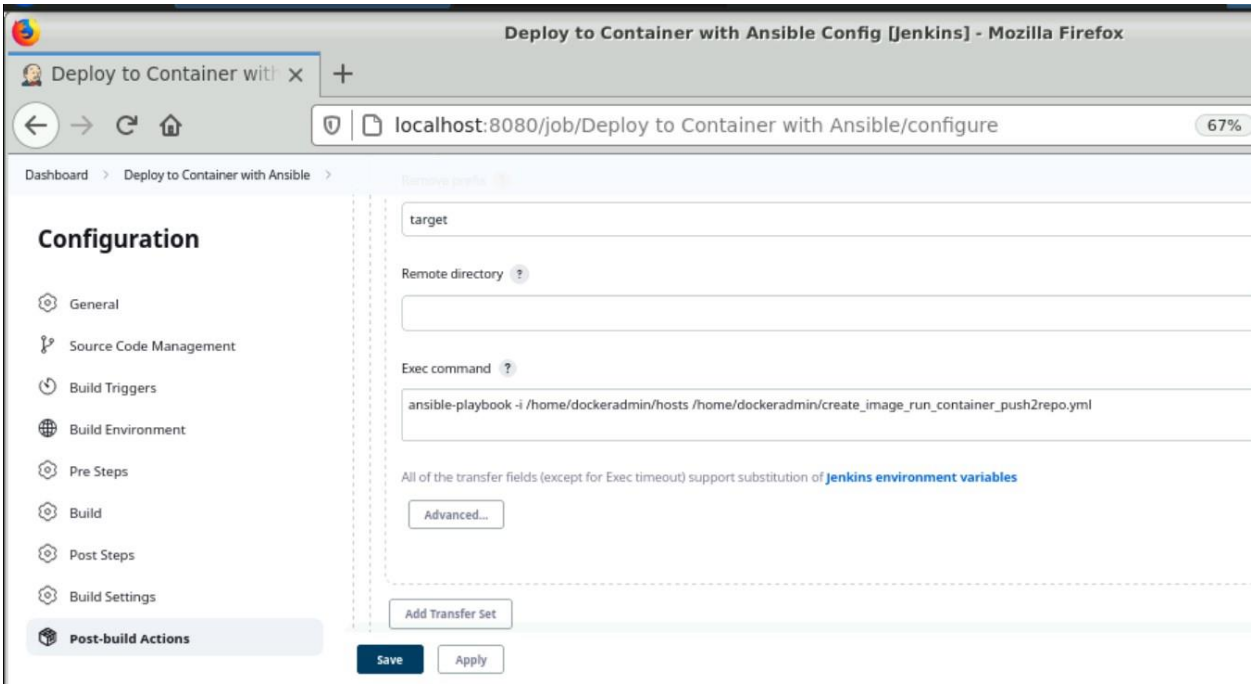
Screenshot of Deploying to Container with Ansible Job Configuration :-


Deploy to Container with
localhost:8080/job/Deploy to Container with Ansible/configure
67%
Dashboard
Deploy to Container with Ansible
Configuration
General
Source Code Management
Build Triggers
Build Environment
Pre Steps
Build
Post Steps
Build Settings
Post-build Actions
Description
Deploy to Container using Ansible
[Plain text] Preview
Discard old builds
GitHub project
This project is parameterized
Execute concurrent builds if necessary
Restrict where this project can be run
Save
Apply

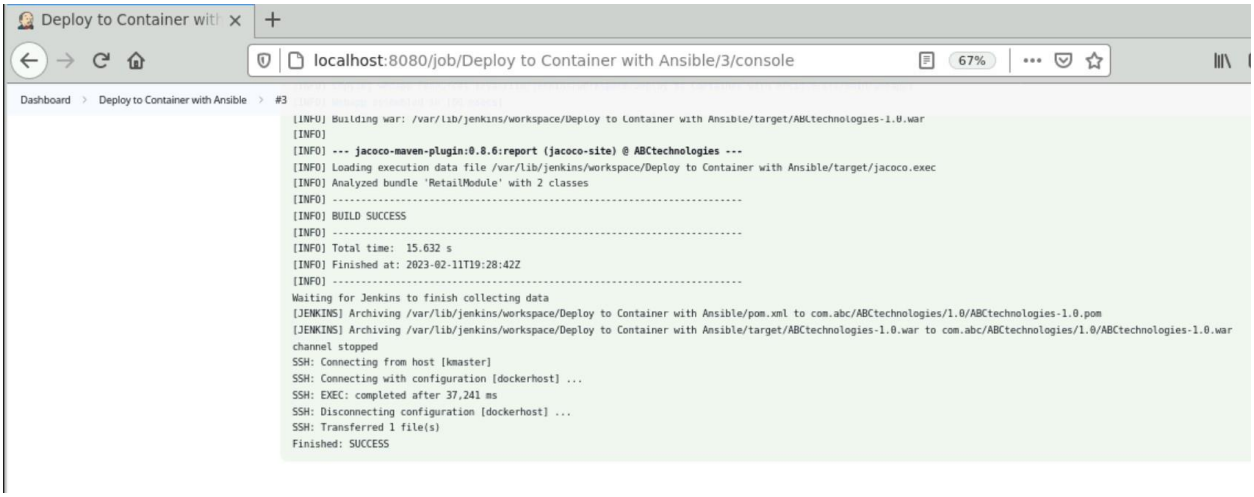

Deploy to Container with
localhost:8080/job/Deploy to Container with Ansible/configure
67%
Dashboard
Deploy to Container with Ansible
Configuration
General
Source Code Management
Build Triggers
Build Environment
Pre Steps
Build
Post Steps
Build Settings
Post-build Actions
Git
Repositories
Repository URL
https://github.com/vermakeshav1994/project1.git
Credentials
- none -
Add
Advanced...
Add Repository
Save
Apply


Deploy to Container with
localhost:8080/job/Deploy to Container with Ansible/configure
67%
Dashboard
Deploy to Container with Ansible
Configuration
General
Source Code Management
Build Triggers
Build Environment
Pre Steps
Build
Post Steps
Build Settings
Post-build Actions
Post-build Actions
Send build artifacts over SSH
SSH Publishers
SSH Server
Name
dockerhost
Advanced...
Transfers
Transfer Set
Source files
/.war
Save
Apply


Deploy to Container with
localhost:8080/job/Deploy to Container with Ansible/configure
Dashboard
Deploy to Container with Ansible
Configuration
General
Source Code Management
Build Triggers
Build Environment
Pre Steps
Build
Post Steps
Build Settings
Post-build Actions
Build
Root POM
pom.xml
Goals and options
clean install package
Advanced...
Post Steps
Run only if build succeeds
Save
Apply

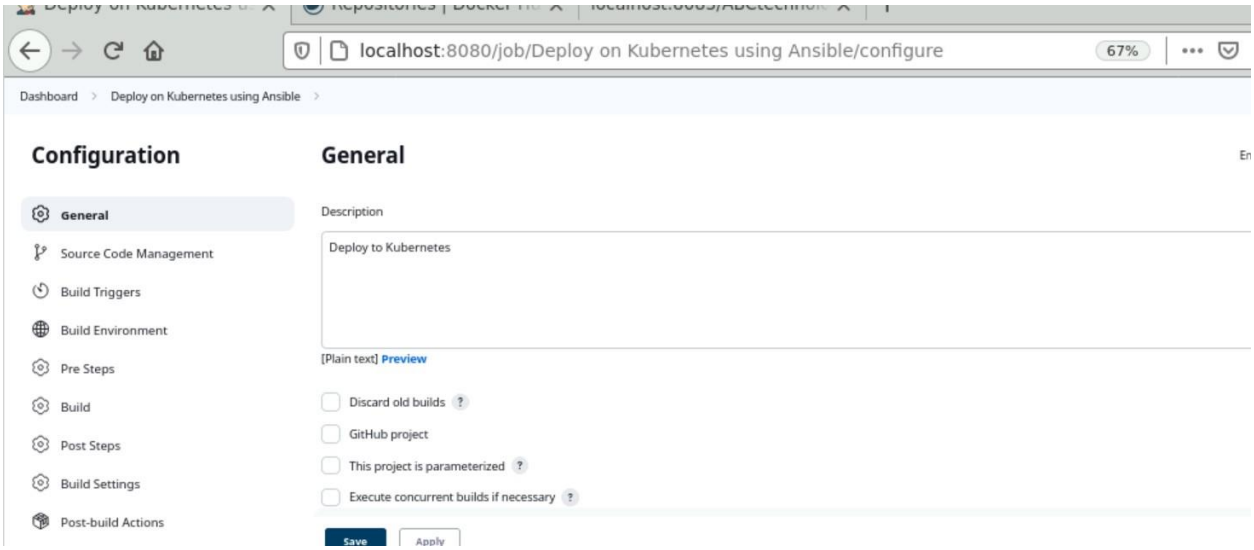


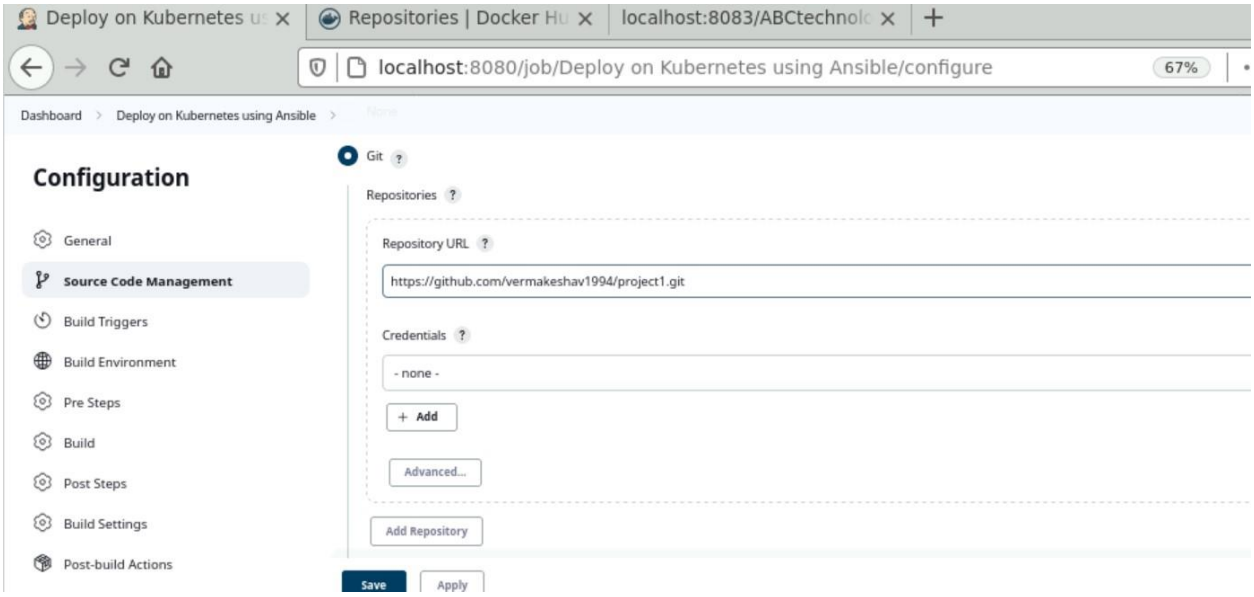
Deploy to Container with Ansible Job console Output :

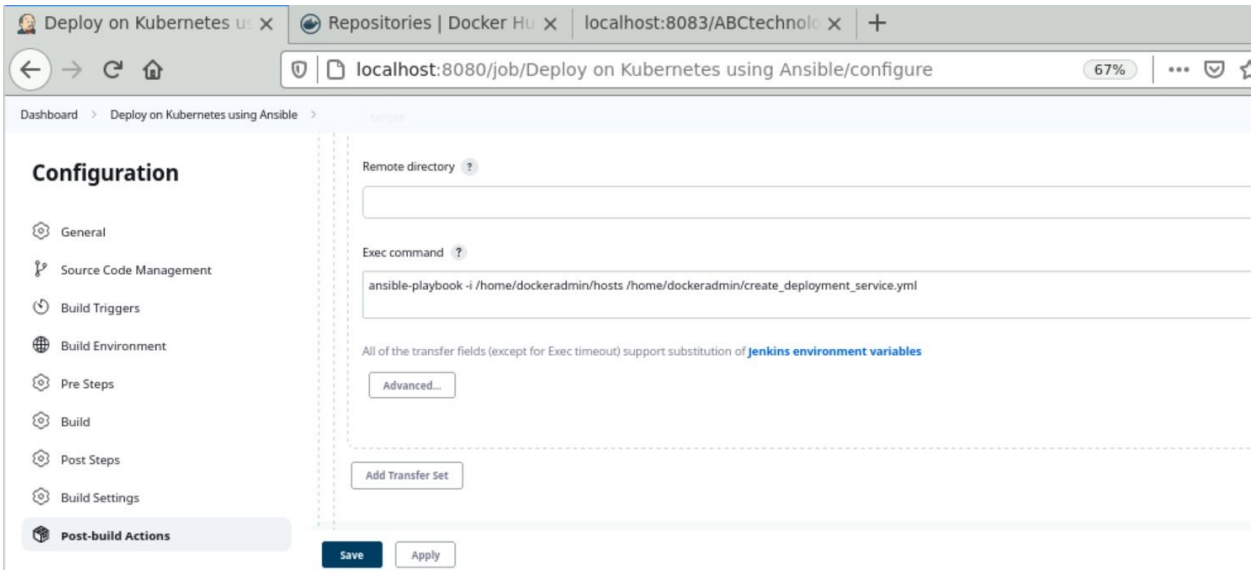


Deploy to Container with ansible final output :-

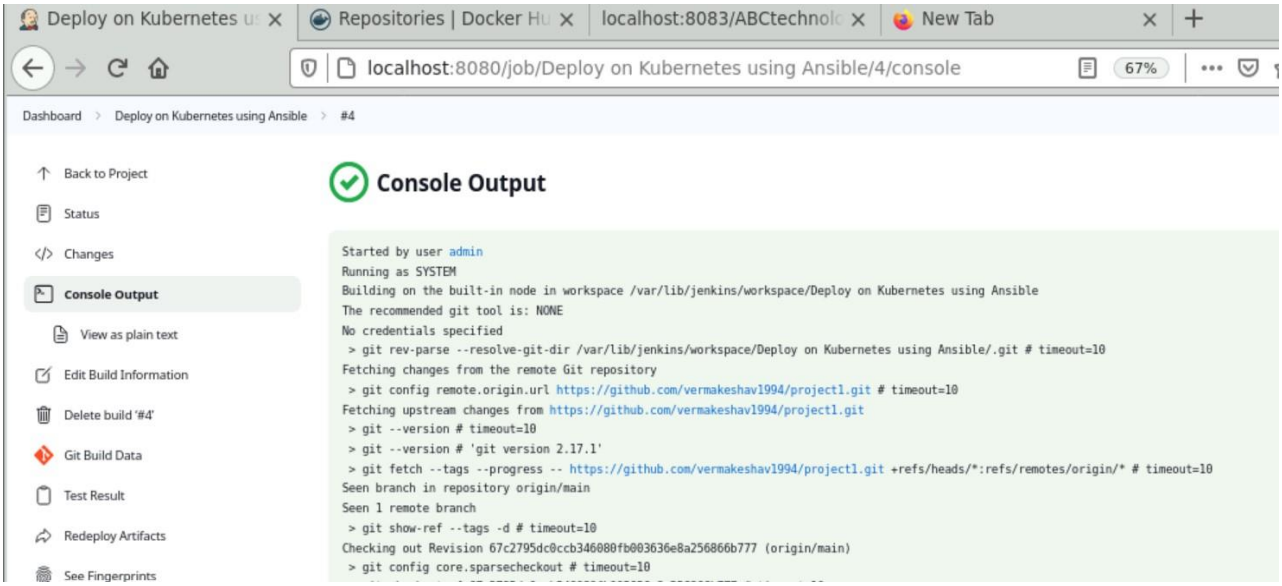
Screenshot of Deploying to Kubernetes with Ansible Job Configuration

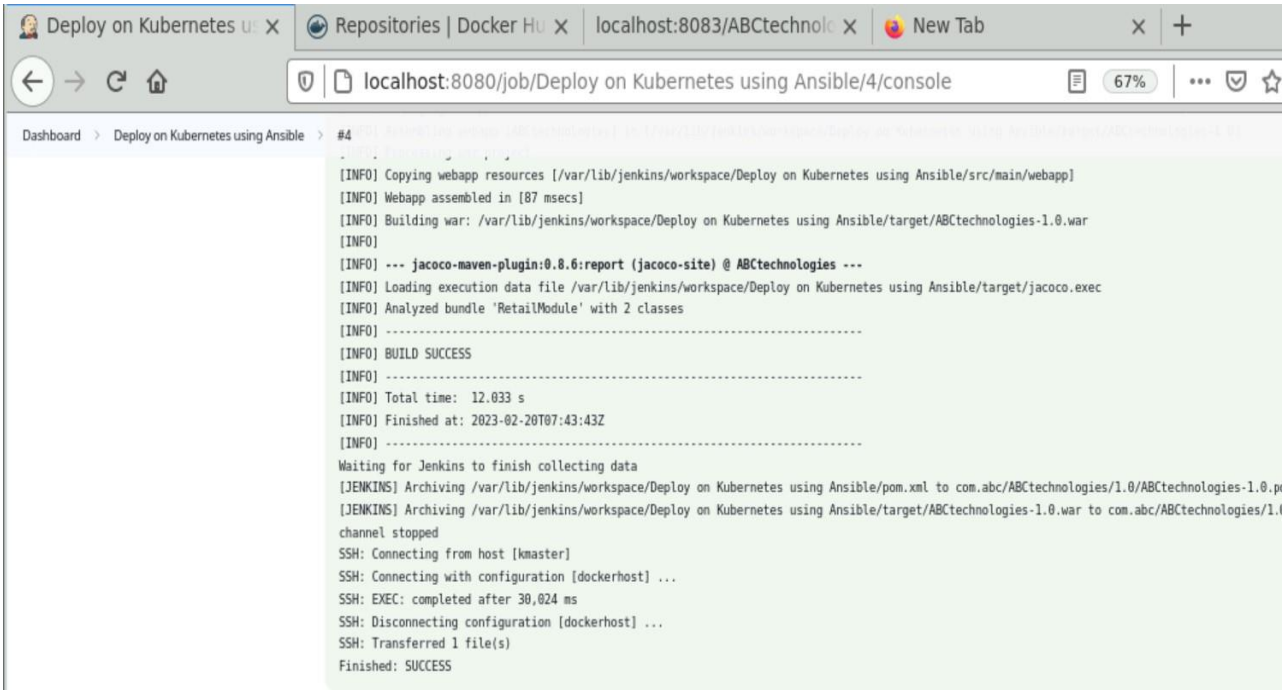



Deploy on Kubernetes us
Repositories | Docker Hu
localhost:8083/ABCtechnolo
localhost:8080/job/Deploy on Kubernetes using Ansible/configure
67%
Dashboard
Deploy on Kubernetes using Ansible
Configuration
General
Source Code Management
Build Triggers
Build Environment
Pre Steps
Build
Post Steps
Build Settings
Post-build Actions
Git
Repositories
Repository URL
https://github.com/vermakeshav1994/project1.git
Credentials
- none -
Add
Advanced...
Add Repository
Save
Apply

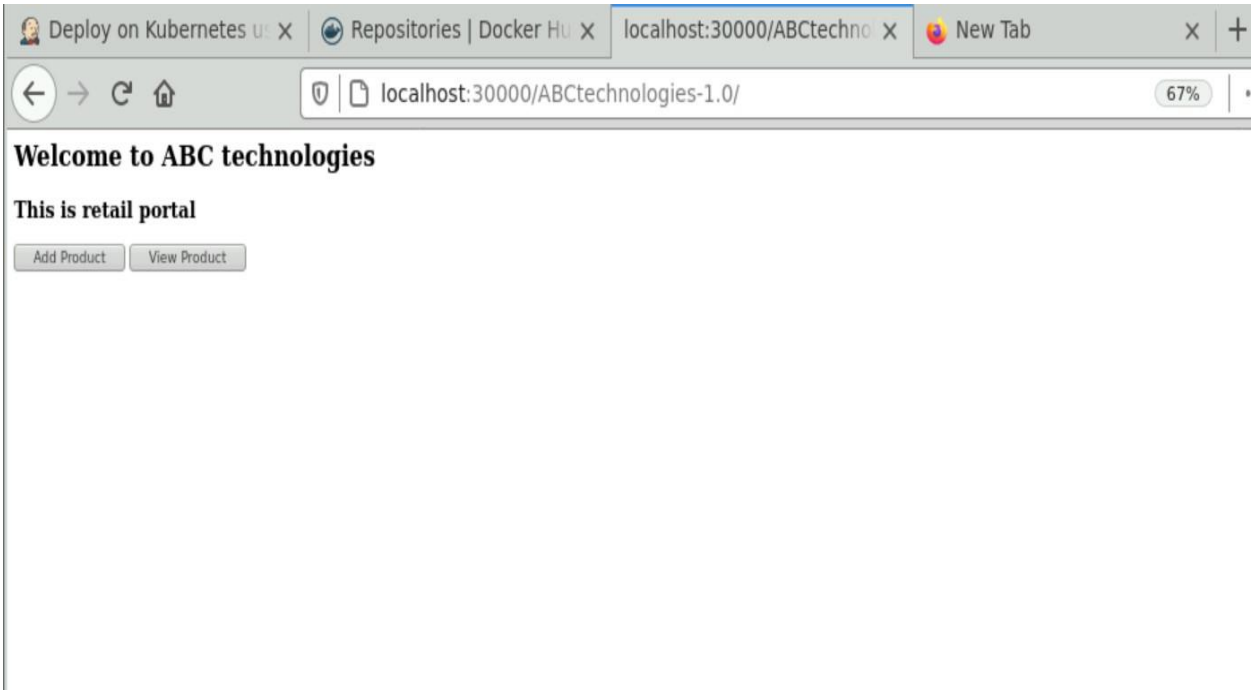

Deploy on Kubernetes us
Repositories | Docker Hu
localhost:8083/ABCtechnolo
localhost:8080/job/Deploy on Kubernetes using Ansible/configure
67%
Dashboard
Deploy on Kubernetes using Ansible
Configuration
General
Source Code Management
Build Triggers
Build Environment
Pre Steps
Build
Post Steps
Build Settings
Post-build Actions
Remote directory
Exec command
ansible-playbook -i /home/dockeradmin/hosts /home/dockeradmin/create_deployment_service.yml
All of the transfer fields (except for Exec timeout) support substitution of Jenkins environment variables
Advanced...
Add Transfer Set
Save
Apply

Deploy to Kubernetes using Ansible Console Output :-

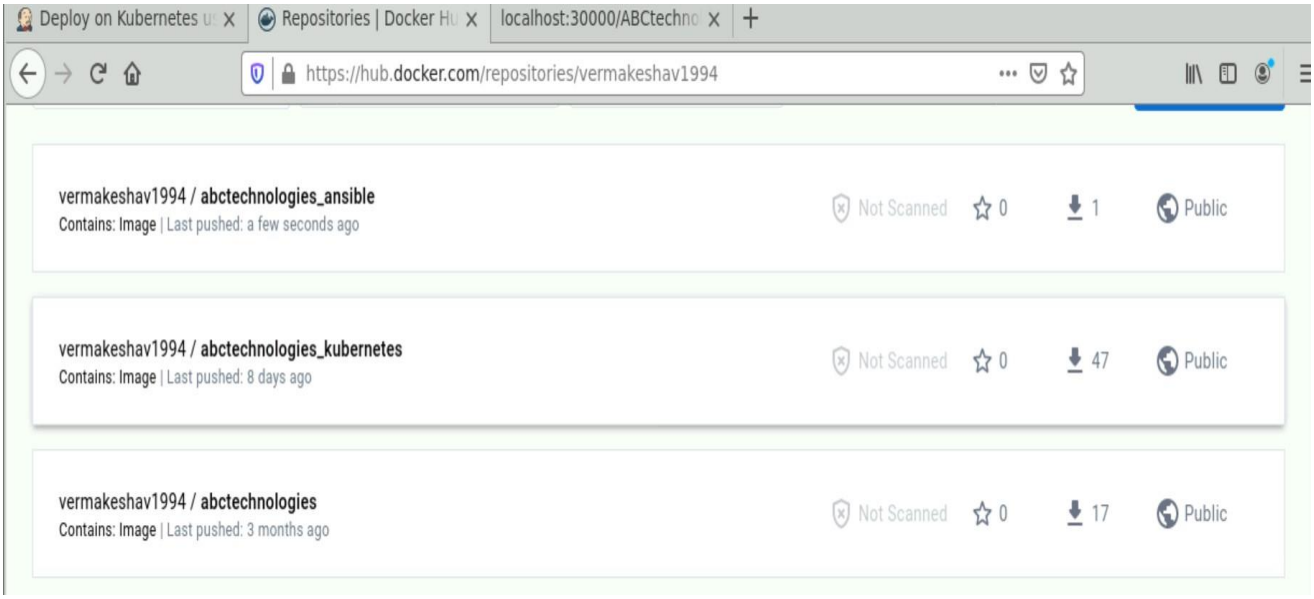




Deploy to Kubernetes using Ansible final Output :-



Docker hub repositories ---



Run Container to push to repositories yml file

edureka@kmaster:/home/dockeradmin$ cat create_image_run_container_push2repo.yml

---

- hosts: all

  become: true

  tasks:

  - name: Stopping running container

```
  command: docker stop abctechnologies_ansible-container

- name: Removing Container

  command: docker rm abctechnologies_ansible-container

- name: Building Docker Image

  command: docker build -t abctechnologies_ansible .

- name: Creating Docker container

  command: docker run -d --name abctechnologies_ansible-container -p
8083:8080 abctechnologies_ansible

- name: Login to Docker Hub
  docker_login:
    username: vermakeshav1994
    password: Keshav@12345
    email: vermakeshav1994@gmail.com

- name: Create tag to push image on docker hub
```

```
    command: docker tag abctechnologies_ansible:latest
vermakeshav1994/abctechnologies_ansible:latest


  - name: Push Docker Image to docker hub


    command: docker push vermakeshav19
```

Kube Service Yml file

```
edureka@kmaster:/home/dockeradmin$ cat kube_service.yml
apiVersion: v1
kind: Service
metadata:
  name: edureka-service
  labels:
    app: edureka-devops-project
spec:
  selector:
    app: edureka-devops-project
  type: NodePort
  ports:
    - port: 8080
      targetPort: 8080
      nodePort: 30000
```

edureka@kmaster:/home/dockeradmin$

Kube_Deploy yml File

edureka@kmaster:~$ cd /home/dockeradmin

edureka@kmaster:/home/dockeradmin$ ls -lrth

total 6.9M

-rwxrwxrwx 1 dockeradmin dockeradmin 228 Feb 8 20:44 Dockerfile

-rwxrwxrwx 1 dockeradmin dockeradmin   13 Feb 8 21:27 hosts

-rwxrwxrwx 1 dockeradmin dockeradmin 879 Feb 8 22:00 create_image_run_container_push2repo.yml

-rwxrwxrwx 1 dockeradmin dockeradmin 1.0K Feb 8 22:08 create_deployment_service.yml

-rwxrwxrwx 1 dockeradmin dockeradmin 533 Feb 8 23:08 kube_deploy.yml

-rwxrwxrwx 1 dockeradmin dockeradmin 245 Feb 8 23:29 kube_service.yml

-rwxrwxrwx 1 dockeradmin dockeradmin 6.9M Feb 20 07:43 ABCtechnologies-1.0.war

edureka@kmaster:/home/dockeradmin$ cat kube_deploy.yml

apiVersion: apps/v1

kind: Deployment

metadata:

  name: edureka-deployment

spec:

  replicas: 2

```
  selector:
    matchLabels:
      app: edureka-devops-project

  template:
    metadata:
      labels:
        app: edureka-devops-project
    spec:
      containers:
      - name: edureka-devops-project
        image: vermakeshav1994/abctechnologies_kubernetes
        imagePullPolicy: Always
        ports:
        - containerPort: 8080
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxSurge: 1
      maxUnavailable: 1
edureka@kmaster:/home/dockeradmin$
```

Task 5: Using Prometheus, monitor the resources like CPU utilisation: 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.

First of all we need to have prometheus,Grafana but edureka lab already installed these tools. So I had installed the node-exporter in master node to monitor the metrics of a node and added the target url in the prometheus.yml file so that prometheus will start to monitor

And I have monitored and captured the CPU,memory and network of the target node. And I have created a dashboard by selecting the prometheus app from Grafana and then created a Panel for visualizing the metrics in Grafana

After installing node_exporter I have added the below job in the prometheus.yml file to get the metrics from the target.

—---------

- job_name: 'node_exporter_Metrics'

scrape_interval: 5s

static_configs:

- targets: ['localhost:9100']

—------

/opt/prometheus-2.27/prometheus.yml

 # my global config

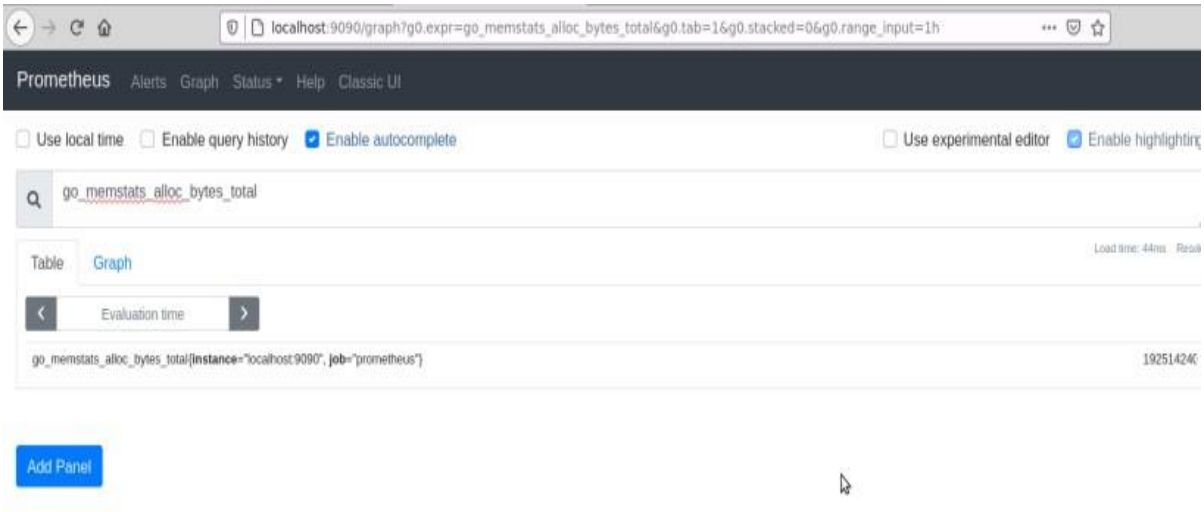
global:

scrape_interval: 15s # Set the scrape interval to every 15 seconds. Default is every 1 minute.

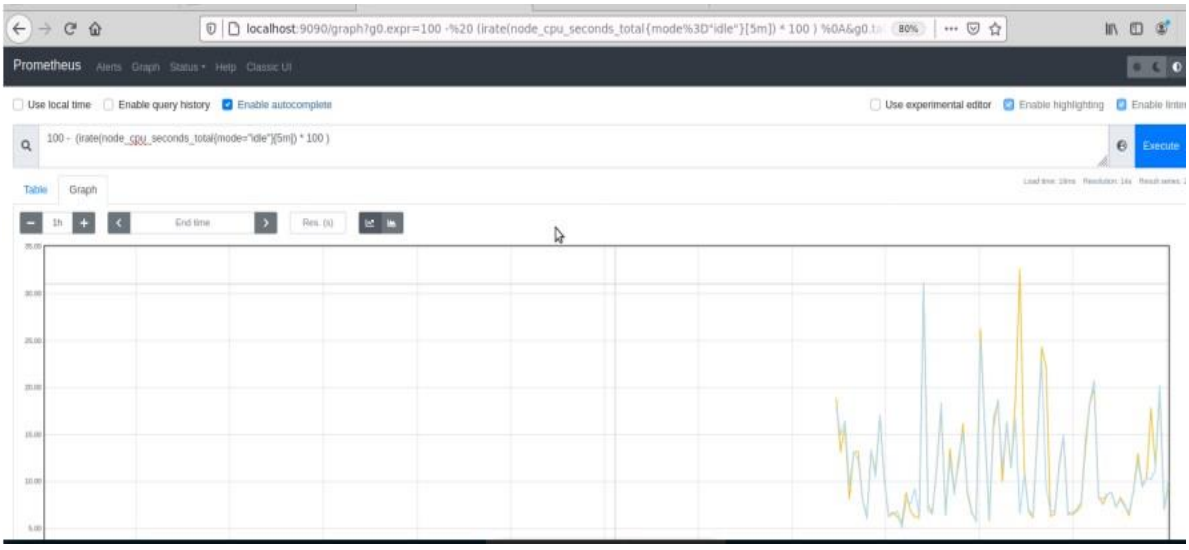
evaluation_interval: 15s # Evaluate rules every 15 seconds. The default is every 1 minute.

#scrape_timeout is set to the global default (10s).

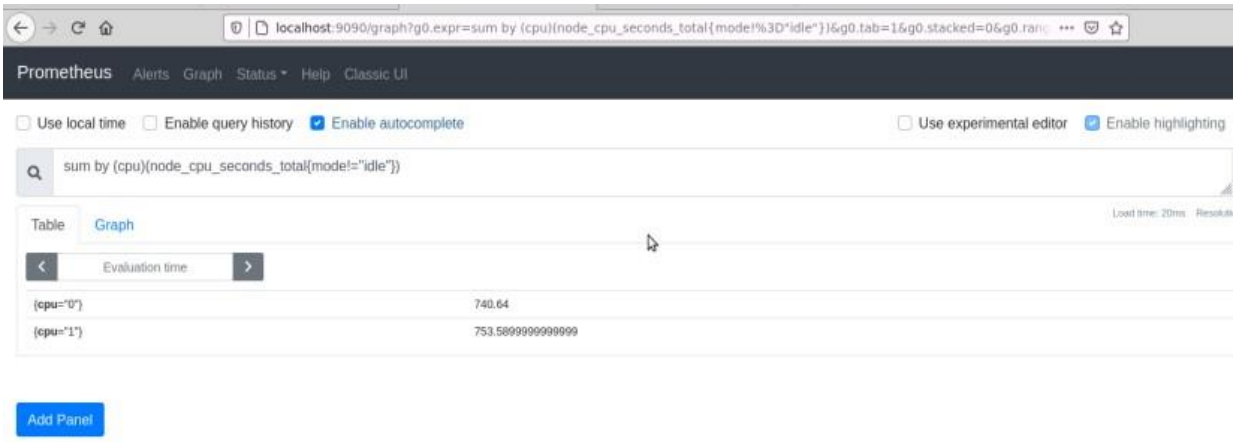
Prometheus self Monitoring Screenshot :-



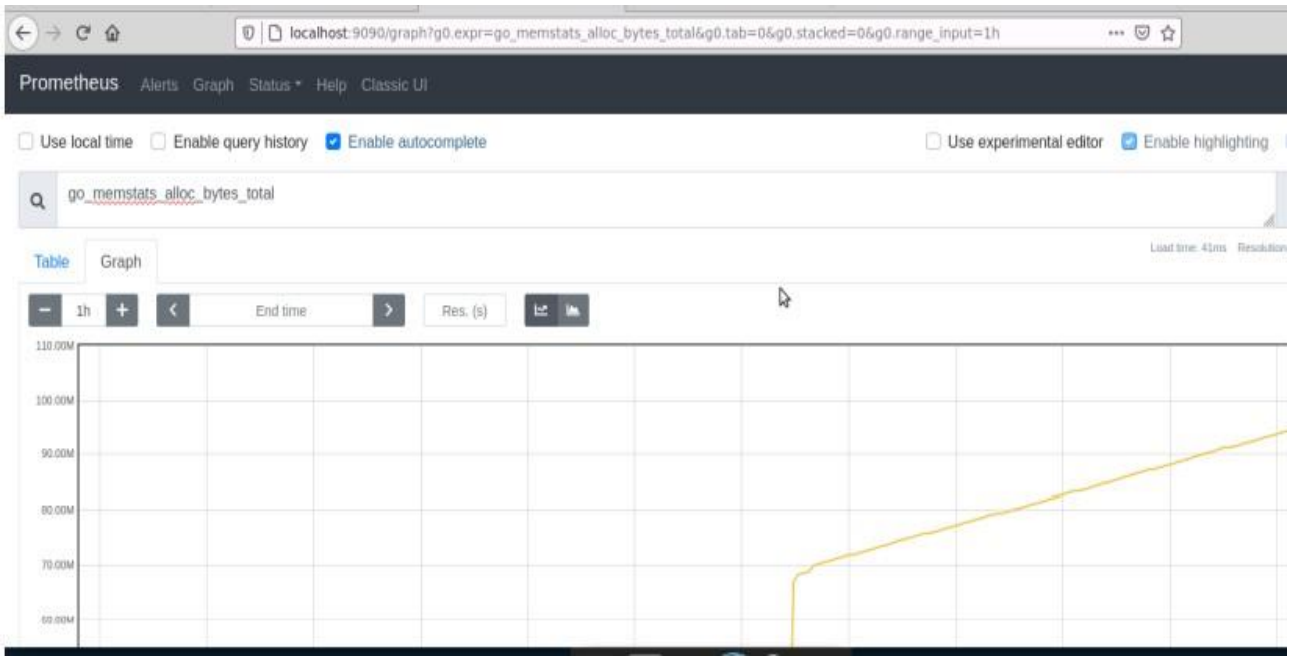
CPU utilization graph :-



CPU Usage screnshots –



Graph Visualization metrics in Prometheus



Rate of Growth of CPU and Total count of CPU from starting