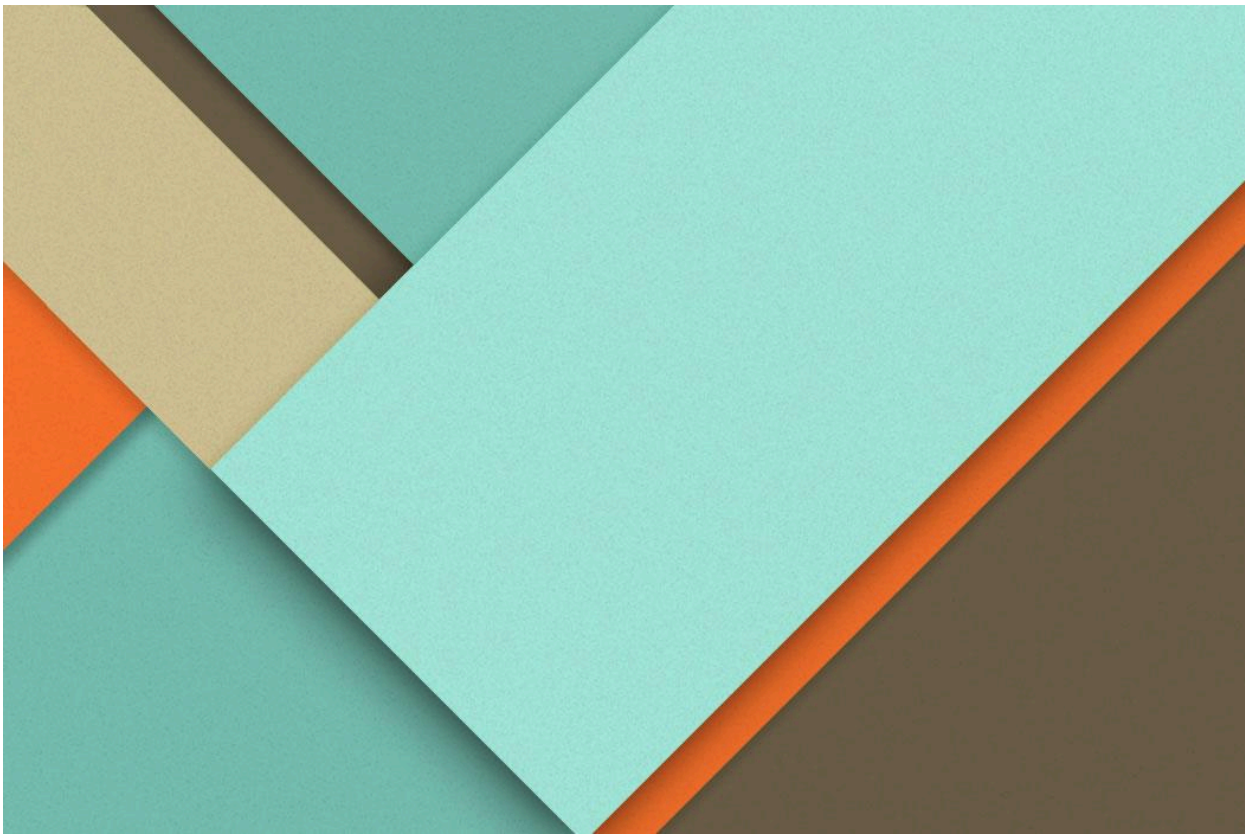


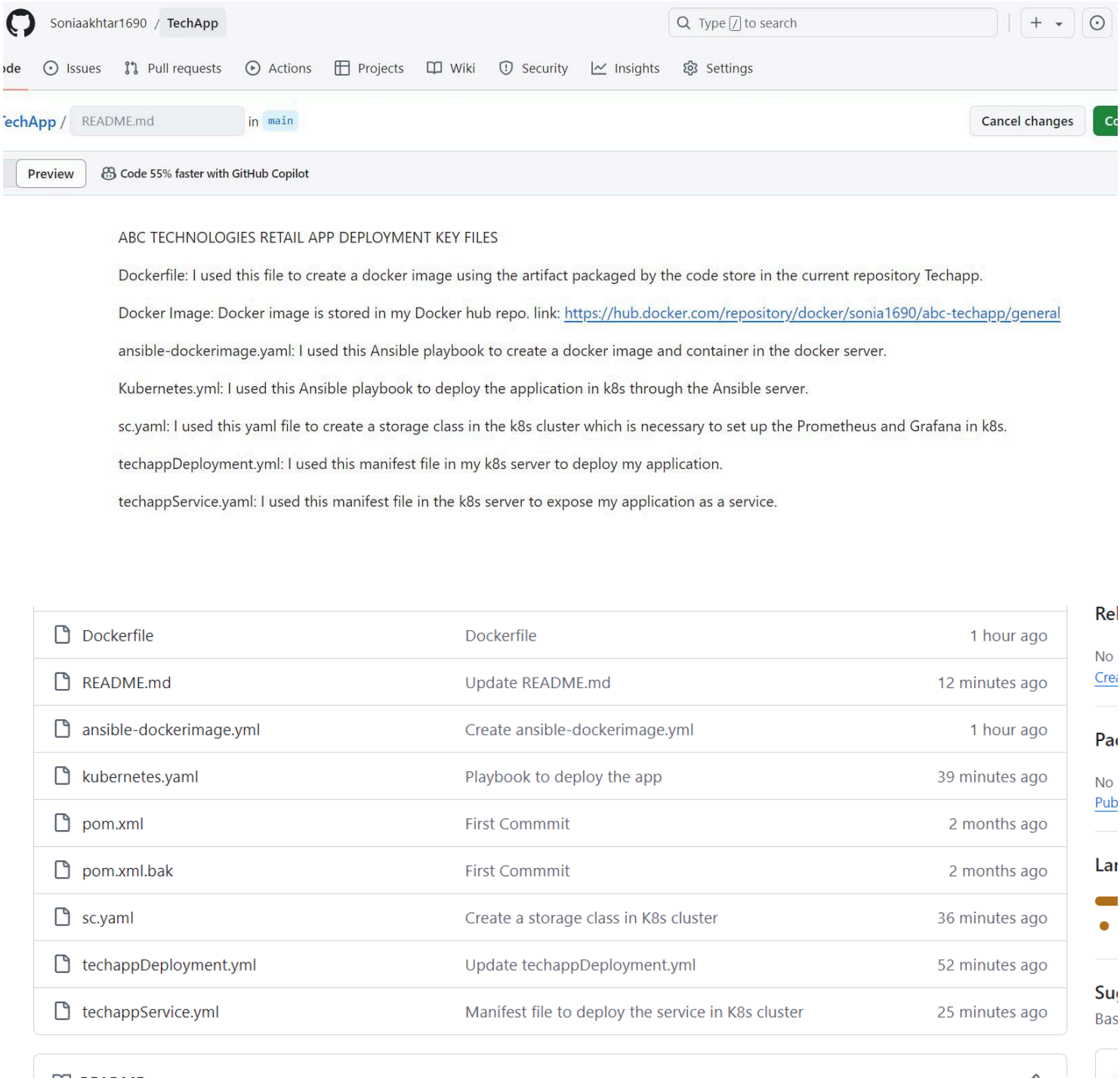
ABC TECHNOLOGIES APP

02-10-2024

Sonia Akhtar

The detail of key files used in this project are added to the git hub repository

https://github.com/Soniaakhtar1690/TechApp



Use Case

ABC Technologies is a leading online retail store, and it has recently acquired a large retail offline business store. The business store has many 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 build 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 integration with Ansible and Kubernetes for deployment. Use Docker Hub to pull and push images between Ansible and Kubernetes.

Goals

To implement CI/CD such that ABC Company can be

- highly available
- highly scalable
- highly performant
- easily built and maintained
- developed and deployed quickly

Problem Statements/Tasks

We need to develop a CI/CD pipeline to automate software development, testing, packaging, and deployment, reducing the time spent marketing the app and ensuring that end users experience good quality service. 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 Git repository.
- Write a 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.

Introduction

In this project, we develop a CI/CD pipeline to automate the software development, testing packaging, and deployment of an App for a retail company, reducing the time to market the app and ensuring good end users experience quality service.
Some of the technologies and tools used include:

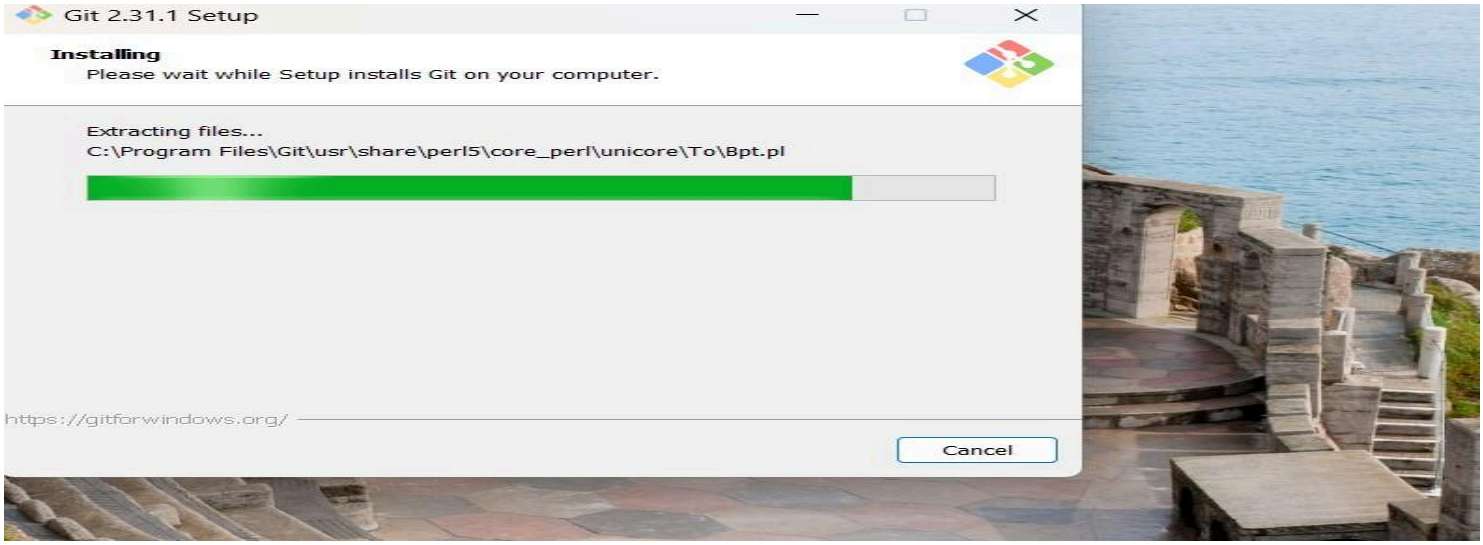
- Maven
- GitHub
- Jenkins
- Docker
- Ansible
- Kubernetes
- Grafana
- Prometheus

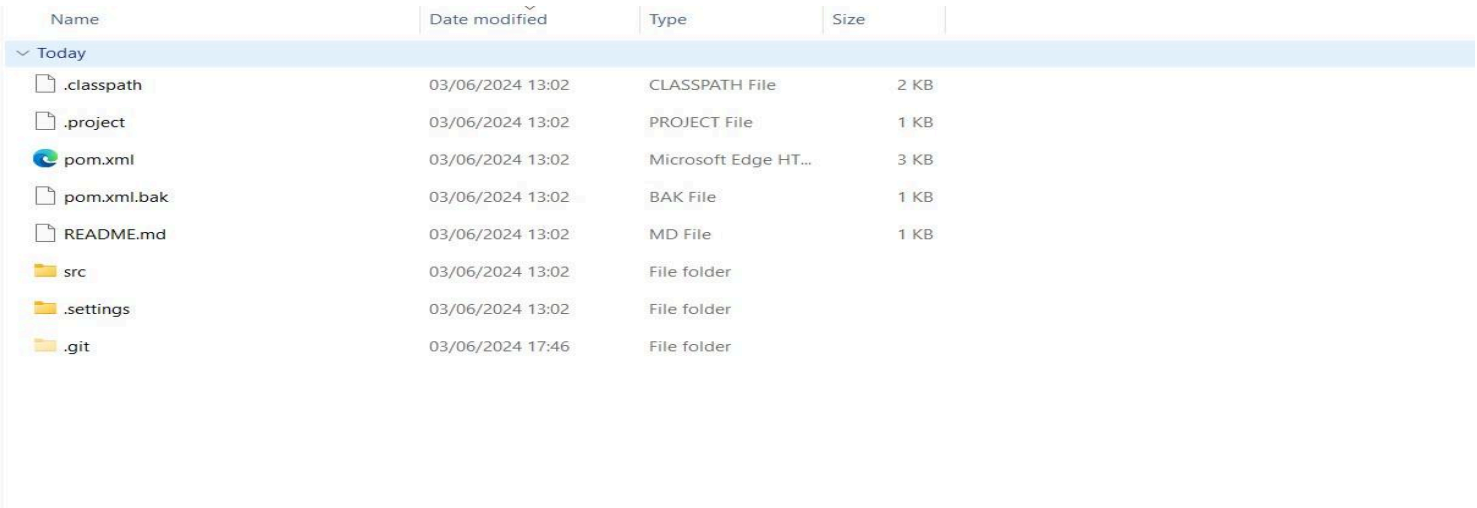
Prerequisites

- An AWS Account–
- A Source code
- Maven
- A GitHub account
- Jenkins
- Docker
- Ansible
- Kubernetes
- Grafana
- Prometheus

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

To achieve this task, first of all, I downloaded the Git Bash and provided Java Source Code on my Local PC and then created a GitHub Repository and uploaded all the project files to the repo.





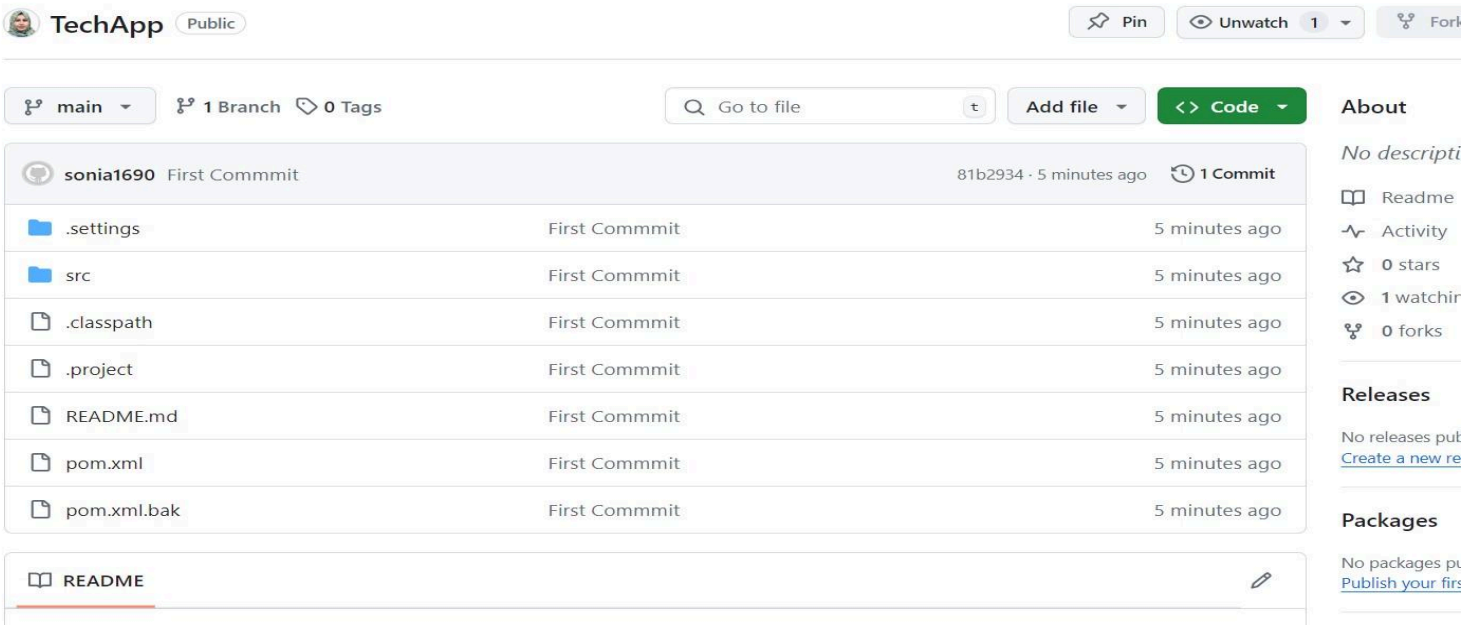
Name	Date modified	Type	Size
Today			
.classpath	03/06/2024 13:02	CLASSPATH File	2 KB
.project	03/06/2024 13:02	PROJECT File	1 KB
pom.xml	03/06/2024 13:02	Microsoft Edge HT...	3 KB
pom.xml.bak	03/06/2024 13:02	BAK File	1 KB
README.md	03/06/2024 13:02	MD File	1 KB
src	03/06/2024 13:02	File folder	
.settings	03/06/2024 13:02	File folder	
.git	03/06/2024 17:46	File folder	

Then I pushed the downloaded Java source code from from the local PC to the git repo by using the git bash command line.
https://github.com/Soniaakhtar1690/TechApp.git

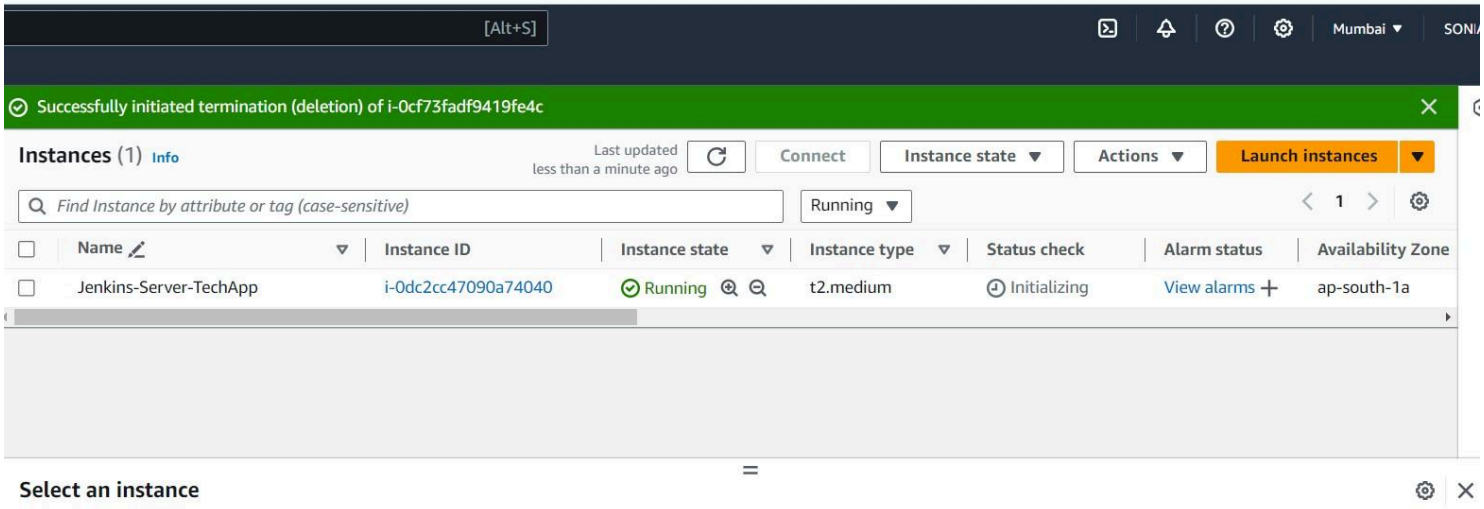
MINGW64:/c/Users/sonia/OneDrive/Documents/Industry Grade Project I - Java Project/ABC Technologies

```
sonia@Sonia MINGW64 ~/OneDrive/Documents/Industry Grade Project I - Java Project
/ABC Technologies (master)
$ git init
Initialized empty Git repository in C:/Users/sonia/OneDrive/Documents/Industry G
rade Project I - Java Project/ABC Technologies/.git/

sonia@Sonia MINGW64 ~/OneDrive/Documents/Industry Grade Project I - Java Project
/ABC Technologies (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .classpath
        .project
        .settings/
        README.md
        pom.xml
        pom.xml.bak
        src/

nothing added to commit but untracked files present (use "git add" to track)

sonia@Sonia MINGW64 ~/OneDrive/Documents/Industry Grade Project I - Java Project
/ABC Technologies (master)
$ git add .
warning: LF will be replaced by CRLF in src/main/webapp/WEB-INF/web.xml.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in src/main/webapp/index.jsp.
The file will have its original line endings in your working directory

sonia@Sonia MINGW64 ~/OneDrive/Documents/Industry Grade Project I - Java Project
/ABC Technologies (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   .classpath
        new file:   .project
        new file:   .settings/org.eclipse.jdt.core.prefs
        new file:   .settings/org.eclipse.m2e.core.prefs
        new file:   README.md
        new file:   pom.xml
        new file:   pom.xml.bak
        new file:   src/main/java/com/abc/RetailModule.java
        new file:   src/main/java/com/abc/dataAccessObject/RetailAccessObject.ja
va
        new file:   src/main/java/com/abc/dataAccessObject/RetailDataImp.java
        new file:   src/main/webapp/WEB-INF/web.xml
        new file:   src/main/webapp/index.jsp
        new file:   src/test/java/com/abc/dataAccessObject/ProductImpTest.java
```

```
sonia@Sonia MINGW64 ~/OneDrive/Documents/Industry Grade Project I - Java Project
/ABC Technologies (master)
$ git branch

sonia@Sonia MINGW64 ~/OneDrive/Documents/Industry Grade Project I - Java Project
/ABC Technologies (master)
$ git branch -M main

sonia@Sonia MINGW64 ~/OneDrive/Documents/Industry Grade Project I - Java Project
/ABC Technologies (main)
$ git commit -m "First Commmit"
[main (root-commit) 81b2934] First Commmit
 13 files changed, 313 insertions(+)
 create mode 100644 .classpath
 create mode 100644 .project
 create mode 100644 .settings/org.eclipse.jdt.core.prefs
 create mode 100644 .settings/org.eclipse.m2e.core.prefs
 create mode 100644 README.md
 create mode 100644 pom.xml
 create mode 100644 pom.xml.bak
 create mode 100644 src/main/java/com/abc/RetailModule.java
 create mode 100644 src/main/java/com/abc/dataAccessObject/RetailAccessObject.ja
va
 create mode 100644 src/main/java/com/abc/dataAccessObject/RetailDataImp.java
 create mode 100644 src/main/webapp/WEB-INF/web.xml
 create mode 100644 src/main/webapp/index.jsp
 create mode 100644 src/test/java/com/abc/dataAccessObject/ProductImpTest.java

sonia@Sonia MINGW64 ~/OneDrive/Documents/Industry Grade Project I - Java Project
/ABC Technologies (main)
$ git remote add origin https://github.com/Soniaakhtar1690/TechApp.git

sonia@Sonia MINGW64 ~/OneDrive/Documents/Industry Grade Project I - Java Project
/ABC Technologies (main)
$ git push -u origin main
Enumerating objects: 29, done.
Counting objects: 100% (29/29), done.
Delta compression using up to 8 threads
Compressing objects: 100% (20/20), done.
Writing objects: 100% (29/29), 4.43 KiB | 648.00 KiB/s, done.
Total 29 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Soniaakhtar1690/TechApp.git
 * [new branch]      main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.

sonia@Sonia MINGW64 ~/OneDrive/Documents/Industry Grade Project I - Java Project
/ABC Technologies (main)
$
```



Next, I launched an EC2 instance t2-medium (4GB RAM, 2 VCPU's).



Install Git

I connected to the server using MobaXterm and installed Git on the virtual machine with the command below:

```
apt-get install git
git clone https://github.com/Soniaakhtar1690/TechApp.git
```

```
root@jenkin-server:/# cd /opt
root@jenkin-server:/opt# ls
root@jenkin-server:/opt# git clone https://github.com/Soniaakhtar1690/TechApp.git
Cloning into 'TechApp'...
remote: Enumerating objects: 29, done.
remote: Counting objects: 100% (29/29), done.
remote: Compressing objects: 100% (20/20), done.
remote: Total 29 (delta 0), reused 29 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (29/29), 4.43 KiB | 4.43 MiB/s, done.
root@jenkin-server:/opt# ls
TechApp
root@jenkin-server:/opt#
```

Install Java

```
#To update the machine
sudo apt update
#To install java
sudo apt install openjdk-11-jdk -y
```

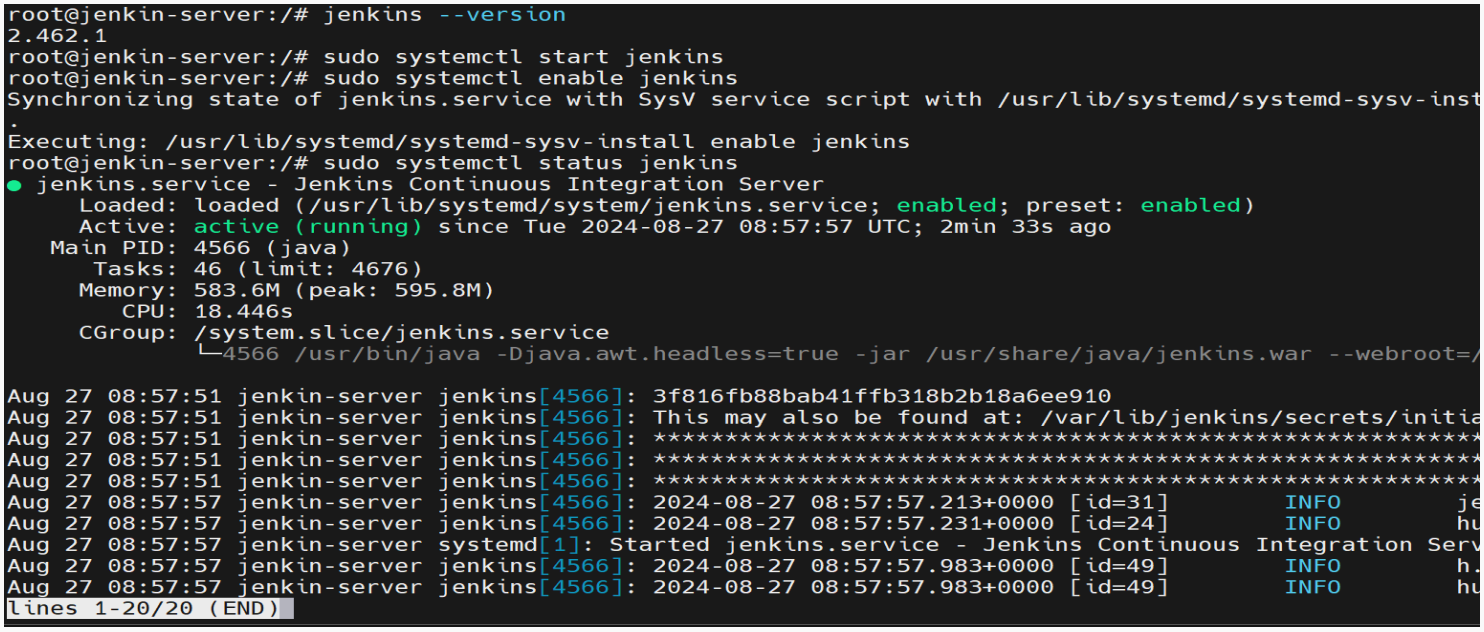
```
root@jenkin-server:/opt# sudo apt install openjdk-11-jdk -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  alsa-topology-conf alsa-ucm-conf at-spi2-common at-spi2-core ca-certificates-java
  dconf-gsettings-backend dconf-service fontconfig-config fonts-dejavu-core fonts-dejavu-extra
  fonts-dejavu-mono gsettings-desktop-schemas java-common libasound2-data libasound2t64
  libatk-bridge2.0-0t64 libatk-wrapper-java libatk-wrapper-java-jni libatk1.0-0t64 libatspi2.0-0t64
  libavahi-client3 libavahi-common-data libavahi-common3 libcups2t64 libdconf1 libdrm-amdgpu1
  libdrm-intel1 libdrm-nouveau2 libdrm-radeon1 libfontconfig1 libgif7 libgl1 libgl1-amber-dri
  libgl1-mesa-dri libglapi-mesa libglvnd0 libglx-mesa0 libglx0 libgraphite2-3 libharfbuzz0b libice-dev
  libice6 libjpeg-turbo8 libjpeg8 liblcms2-2 libllvm17t64 libpciaccess0 libpcsclite1
  libpthread-stubs0-dev libsm-dev libsm6 libvulkan1 libwayland-client0 libx11-dev libx11-xcb1 libxau-de
  libxaw7 libxcb-dri2-0 libxcb-dri3-0 libxcb-glx0 libxcb-present0 libxcb-randr0 libxcb-shape0
  libxcb-shm0 libxcb-sync1 libxcb-xfixes0 libxcb1-dev libxcomposite1 libxdmcp-dev libxfixes3 libxft2
  libxi6 libxinerama1 libxkbfile1 libxmu6 libxpm4 libxrandr2 libxrender1 libxshmfence1 libxt-dev
  libxt6t64 libxtst6 libxv1 libxxf86dga1 libxxf86vm1 mesa-vulkan-drivers openjdk-11-jdk-headless
  openjdk-11-jre openjdk-11-jre-headless session-migration x11-common x11-utils x11proto-dev
  xorg-sgml-doctools xtrans-dev
Suggested packages:
  default-jre alsa-utils libasound2-plugins cups-common libice-doc liblcms2-utils pcscd libsm-doc
```

Install Jenkins

Java is now installed successfully. To install Jenkins, visit the website pkg.jenkins.io and choose “debian-stable/” for the stable version. Then, follow the steps to install Jenkins.
Link: https://pkg.jenkins.io/debian-stable/

Start jenkins:

```
sudo systemctl start jenkins
sudo systemctl enable jenkins
sudo systemctl status jenkins
```



Install Maven:

Next, we must install Maven, which we will use to compile, test, and package the code. Run the following commands to install Maven.

Referenced: Here

```
sudo apt update
sudo apt install maven
mvn -version
```

Everything is set up now. It's time to start building.

Compile The Code

Change to the directory in which the source code and files are and then run the command below:

```
#To compile the source code
 mvn compile
```

```
Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/ple
/1.5.5/plexus-container-default-1.5.5.jar (217 kB at 1.4 MB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/junit/junit/3.8.2/junit
 745 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/log4j/log4j/1.2.12/log4
at 2.1 MB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/com/google/collections/
0/google-collections-1.0.jar (640 kB at 3.4 MB/s)
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 3 source files to /opt/TechApp/target/classes
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  6.444 s
[INFO] Finished at: 2024-08-27T09:07:32Z
[INFO] ------------------------------------------------------------------------
root@jenkin-server:/opt/TechApp#
```

Testing The Code

```
#To conduct test on the source code
mvn test
```

```
-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running com.abc.dataAccessObject.ProductImpTest
Tests run: 4, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.047 sec

Results :

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

[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  3.913 s
[INFO] Finished at: 2024-08-27T09:08:45Z
[INFO] ------------------------------------------------------------------------
root@jenkin-server:/opt/TechApp#
```

Packaging The Code

To package the code and build the war file, run the command below

```
#To build the war file
mvn package
```

```
[INFO] Packaging webapp
[INFO] Assembling webapp [ABCtechnologies] in [/opt/TechApp/target/ABCtechnologies-1.0]
[INFO] Processing war project
[INFO] Copying webapp resources [/opt/TechApp/src/main/webapp]
[INFO] Webapp assembled in [75 msecs]
[INFO] Building war: /opt/TechApp/target/ABCtechnologies-1.0.war
[INFO]
[INFO] --- jacoco-maven-plugin:0.8.6:report (jacoco-site) @ ABCtechnologies ---
[INFO] Loading execution data file /opt/TechApp/target/jacoco.exec
[INFO] Analyzed bundle 'RetailModule' with 2 classes
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  4.760 s
[INFO] Finished at: 2024-08-27T09:10:19Z
[INFO] ------------------------------------------------------------------------
root@jenkin-server:/opt/TechApp#
pport MobaXterm by subscribing to the professional edition here:  https://mobaxterm.mobatek.net
```

War File Created

The war file is created at this location
/opt/TechApp/target/ABCtechnologies-1.0

```
root@jenkin-server:/opt/TechApp# cd /opt/TechApp/target
root@jenkin-server:/opt/TechApp/target# ls
ABCtechnologies-1.0      generated-sources       maven-archiver  surefire-reports
ABCtechnologies-1.0.war  generated-test-sources  maven-status    test-classes
classes                  jacoco.exec             site
root@jenkin-server:/opt/TechApp/target#
```

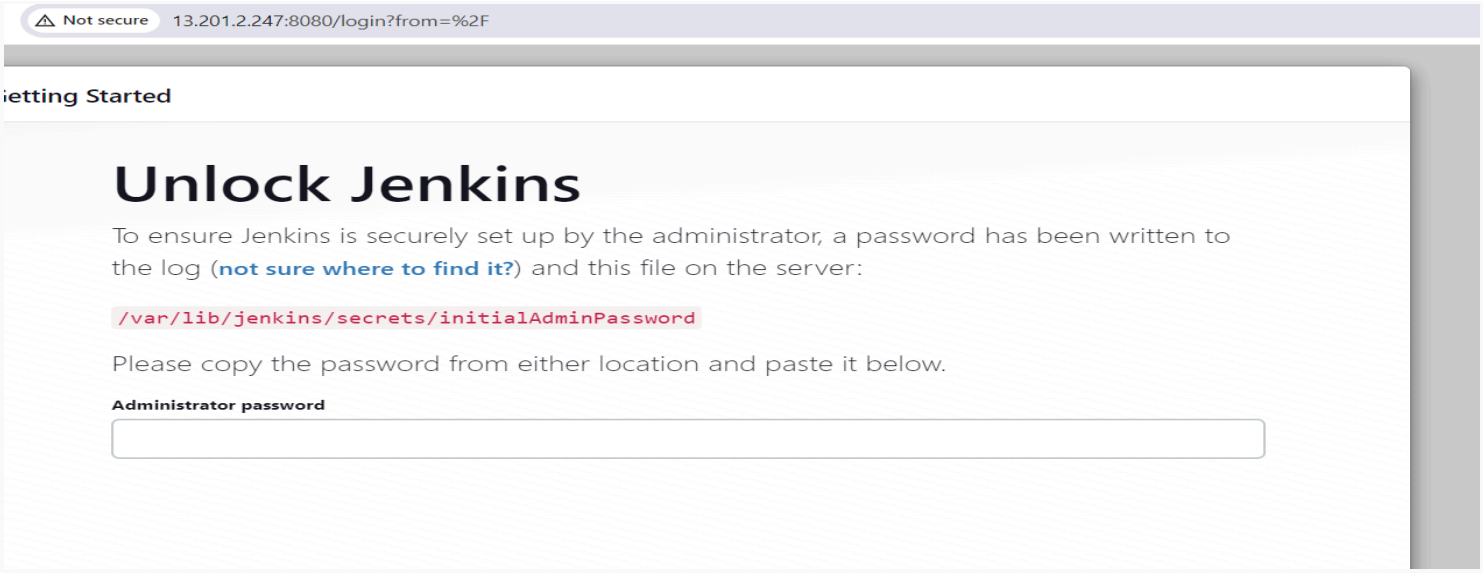
Task 2:Set up the git repository and push the source code. Login to Jenkins

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 CICD pipeline to execute the jobs created in step1**
3. **Set up a master-slave node to distribute the tasks in the pipeline.**

Setup Jenkins Dashboard

To access Jenkins in the browser use port 8080.

`public IP address:8080`
`13.201.2.247:8080`



To get the password, run the code below on your terminal

```
cat /var/lib/jenkins/secrets/initialAdminPassword
```

```
root@jenkin-server:/opt/TechApp/target# cat /var/lib/jenkins/secrets/initialAdminPassword
3f816fb88bab41ffb318b2b18a6ee910
root@jenkin-server:/opt/TechApp/target#
```

Copy the password above and paste it into the password section to access the Jenkins dashboard and browser. Follow the instructions until you have complete access to the dashboard.

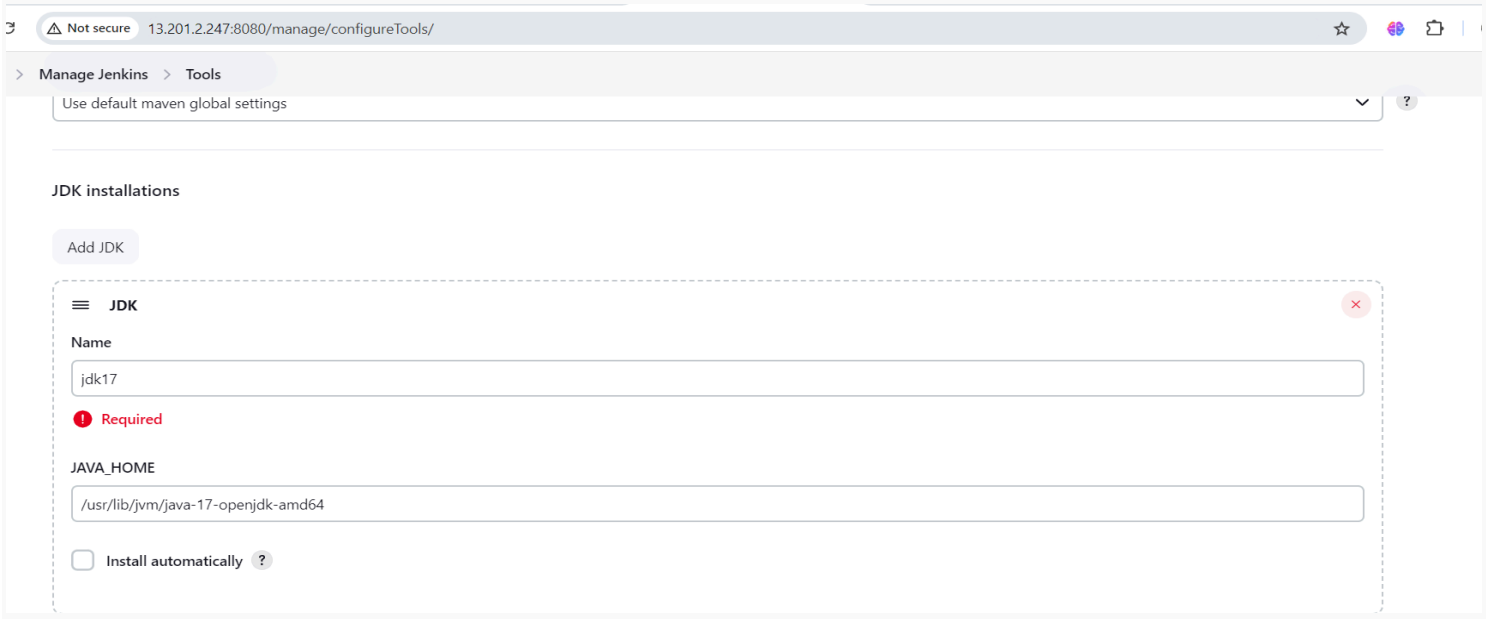
Setup the tools we need with Jenkins (Maven, Java, and Git)

This is to make sure Jenkins works with the tools we need. To do this, go to the Jenkins dashboard, click Manage Jenkins, and select tools.

Use the command below to find the JAVA_HOME

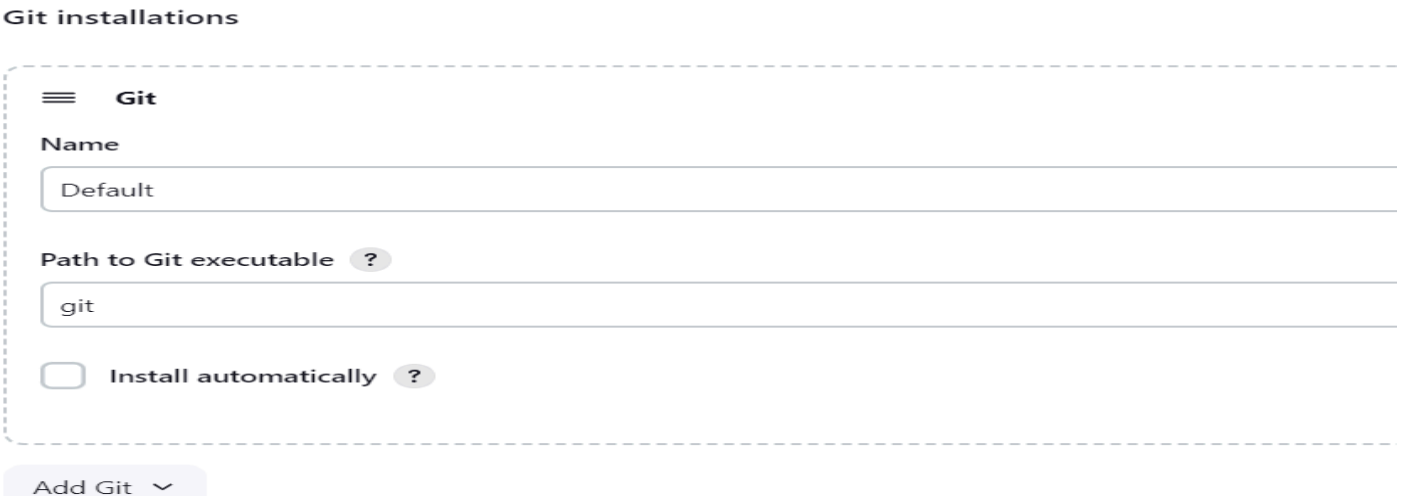
```
# For Ubuntu OS use:
update-alternatives --config java
```

```
root@ip-192-0-0-7:/home/ubuntu# update-alternatives --config java
There are 2 choices for the alternative java (providing /usr/bin/java).

  Selection    Path                                            Priority   Status
------------------------------------------------------------
* 0            /usr/lib/jvm/java-17-openjdk-amd64/bin/java   1711      auto mode
  1            /usr/lib/jvm/java-11-openjdk-amd64/bin/java   1111      manual mode
  2            /usr/lib/jvm/java-17-openjdk-amd64/bin/java   1711      manual mode

Press <enter> to keep the current choice[*], or type selection number:
```



Git Integration

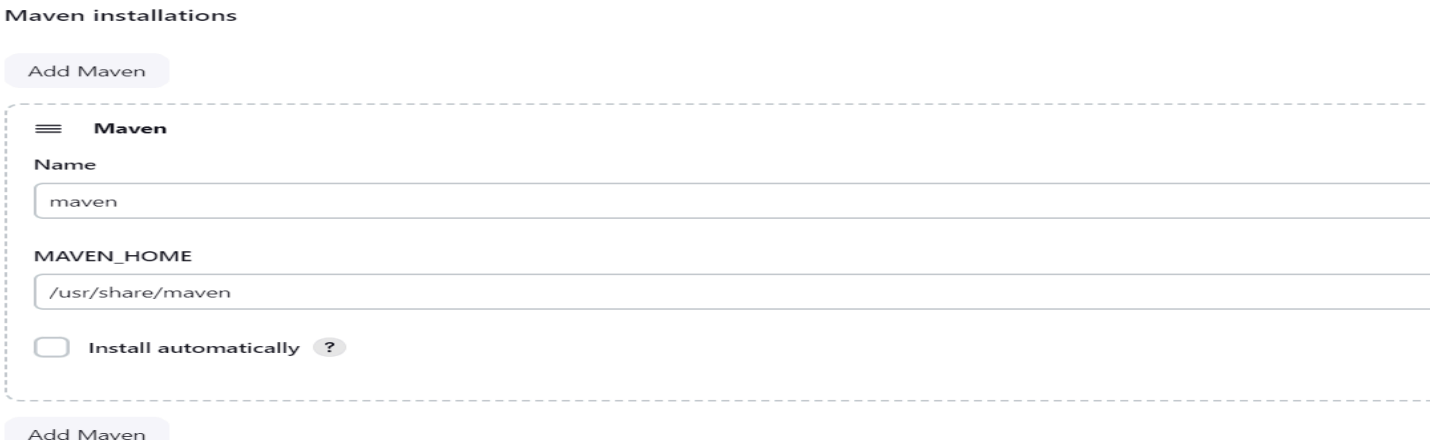
we do not need to make any changes, we work with the default settings.



For maven

we already installed it through the CLI, get the MAVEN_HOME as shown below and then click save

```
root@ip-192-0-0-7:/home/ubuntu# mvn --version
Apache Maven 3.8.7
Maven home: /usr/share/maven
Java version: 17.0.12, vendor: Ubuntu, runtime: /usr/lib/jvm/java-17-openjdk-amd64
Default locale: en, platform encoding: UTF-8
OS name: "linux", version: "6.8.0-1012-aws", arch: "amd64", family: "unix"
root@ip-192-0-0-7:/home/ubuntu#
```



After the tools are set up, we are ready to start the first job. The First Job is the Compile Job, so it becomes the Upstream Job for the Test Job and the Test becomes Upstream for the Package Job. Also, Compile Job is set to Build Trigger Every Hour.

1. create a build pipeline containing a job for each

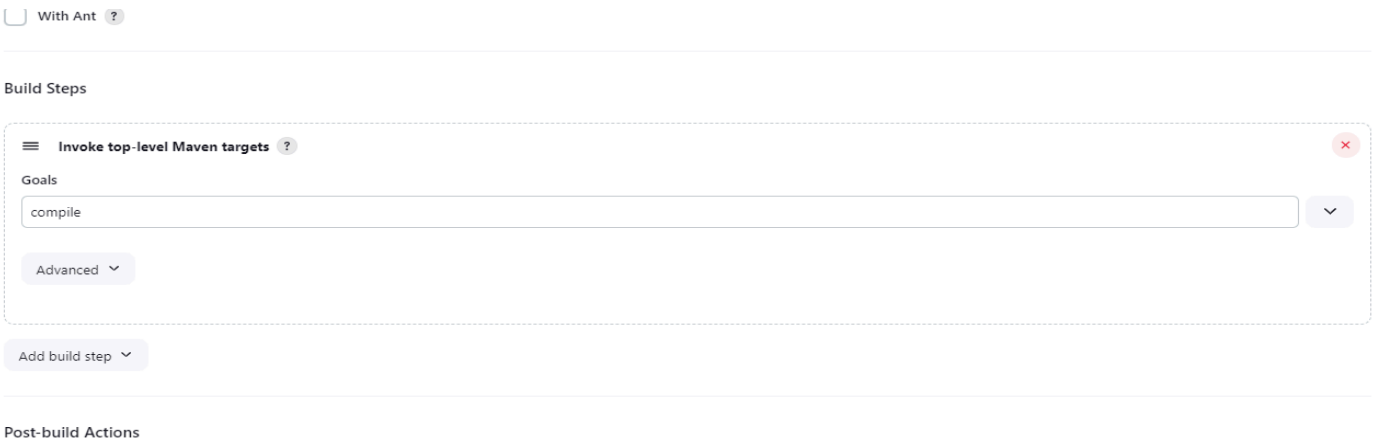
- **One for compiling source code**
- **Second for testing source code**
- **Third for packing the code**

Compile Job
The compile job will take the source code and turn it into class files that are in a language that machines can understand, using zeros and ones.

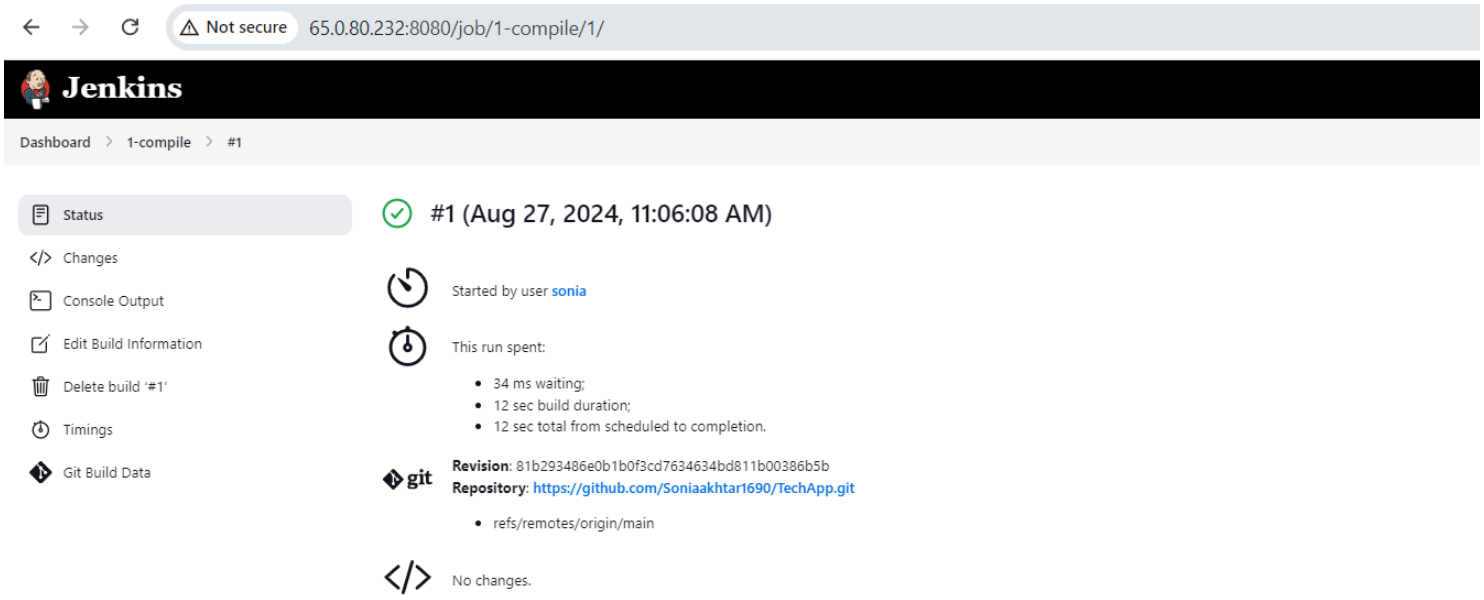
Now add the repository URL of TechApp under the git section

https://github.com/Soniaakhtar1690/TechApp.git

Then go to the build steps and select “Invoke top-level maven target”



Now build 1-compile job



The compiled code is stored at this location

/var/lib/jenkins/workspace/1-compile/target/classes

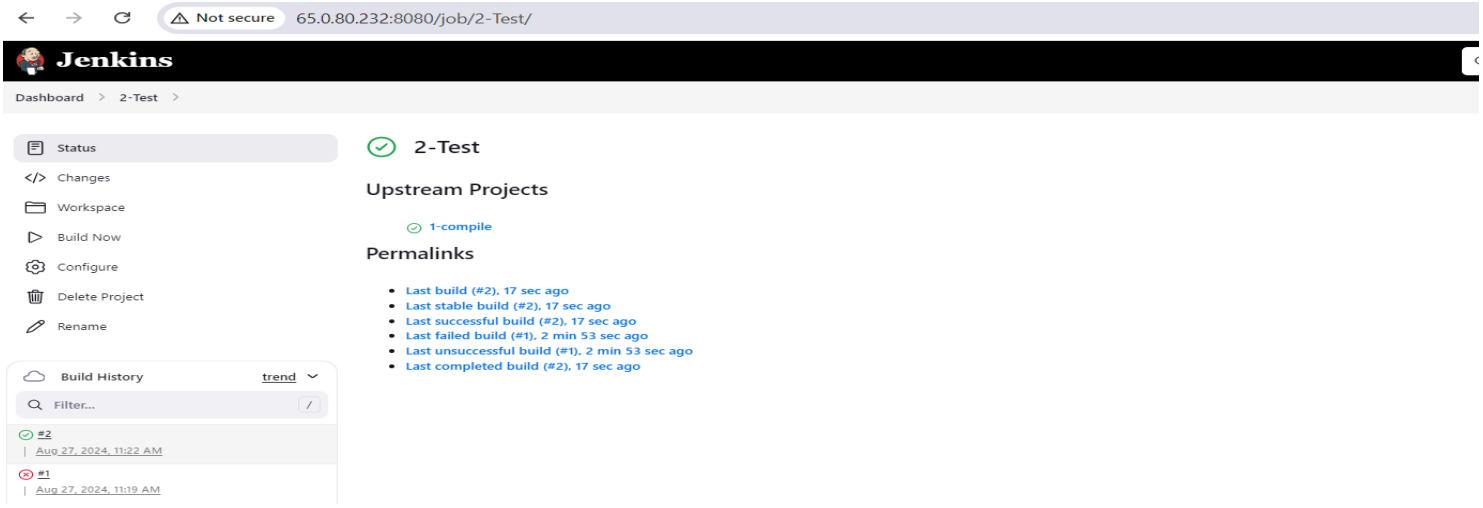
```
Progress (1): 602/640 kB
Progress (1): 606/640 kB
Progress (1): 610/640 kB
Progress (1): 614/640 kB
Progress (1): 618/640 kB
Progress (1): 623/640 kB
Progress (1): 627/640 kB
Progress (1): 631/640 kB
Progress (1): 635/640 kB
Progress (1): 639/640 kB
Progress (1): 640 kB

Downloaded from central: https://repo.maven.apache.org/maven2/com/google/collections/google-collections/1.0/google-collections-1.0.jar (640 kB at 4.7 MB/s)
[␛[1;34mINFO␛[m] Changes detected - recompiling the module!
[␛[1;34mINFO␛[m] Compiling 3 source files to /var/lib/jenkins/workspace/1-compile/target/classes
[␛[1;34mINFO␛[m] ␛[1m------------------------------------------------------------------------␛[m
[␛[1;34mINFO␛[m] ␛[1;32mBUILD SUCCESS␛[m
[␛[1;34mINFO␛[m] ␛[1m------------------------------------------------------------------------␛[m
[␛[1;34mINFO␛[m] Total time:  6.321 s
[␛[1;34mINFO␛[m] Finished at: 2024-08-27T11:06:20Z
[␛[1;34mINFO␛[m] ␛[1m------------------------------------------------------------------------␛[m
Finished: SUCCESS
```

Test Job

Now we will create a new job “2-Test” and add the “1-Compile” as an upstream job.To connect the test job to the Compile job, follow these steps:

1. Go to the build triggers section.
2. Select "Build after other projects are built."
3. In the box for the project to watch, enter "1-compile."
4. Save your changes and build the job.

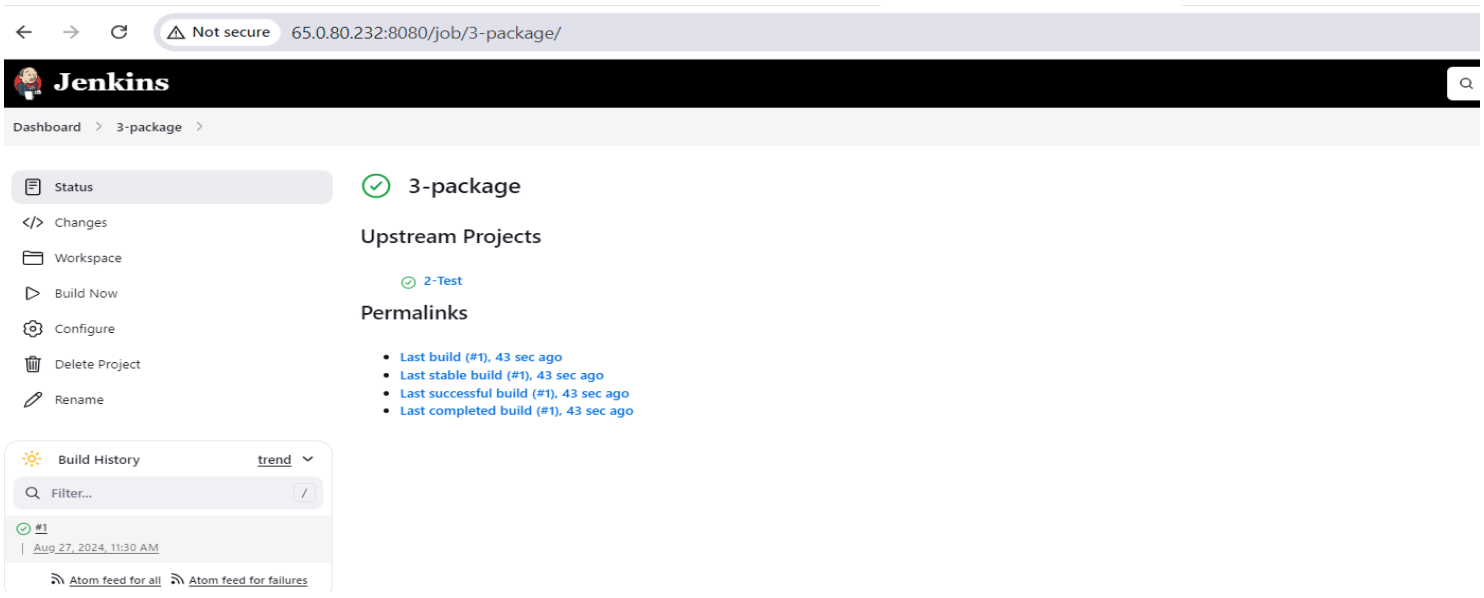
This will ensure that the unit testing runs right after the Compile job finishes.



```
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/surefire/surefire-junit4/2.12.4/surefire-junit4-2.12.4.jar (37 kB at 3.1 MB/s)

-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running com.abc.dataAccessObject.ProductImpTest
Tests run: 4, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.057 sec

Results :

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

[⍰[1;34mINFO⍰[m] ⍰[1m------------------------------------------------------------------------⍰[m
[⍰[1;34mINFO⍰[m] ⍰[1;32mBUILD SUCCESS⍰[m
[⍰[1;34mINFO⍰[m] ⍰[1m------------------------------------------------------------------------⍰[m
[⍰[1;34mINFO⍰[m] Total time:  4.255 s
[⍰[1;34mINFO⍰[m] Finished at: 2024-08-27T11:22:23Z
[⍰[1;34mINFO⍰[m] ⍰[1m------------------------------------------------------------------------⍰[m
Finished: SUCCESS
```

Packaging Job

The next step is to package the code into artifacts. Use the same steps as before, but now the goal is to package. Go to build triggers and select "Build after other projects are built." Enter "2-Test" as the upstream job to monitor, then save and build the job. The output will be as shown below.

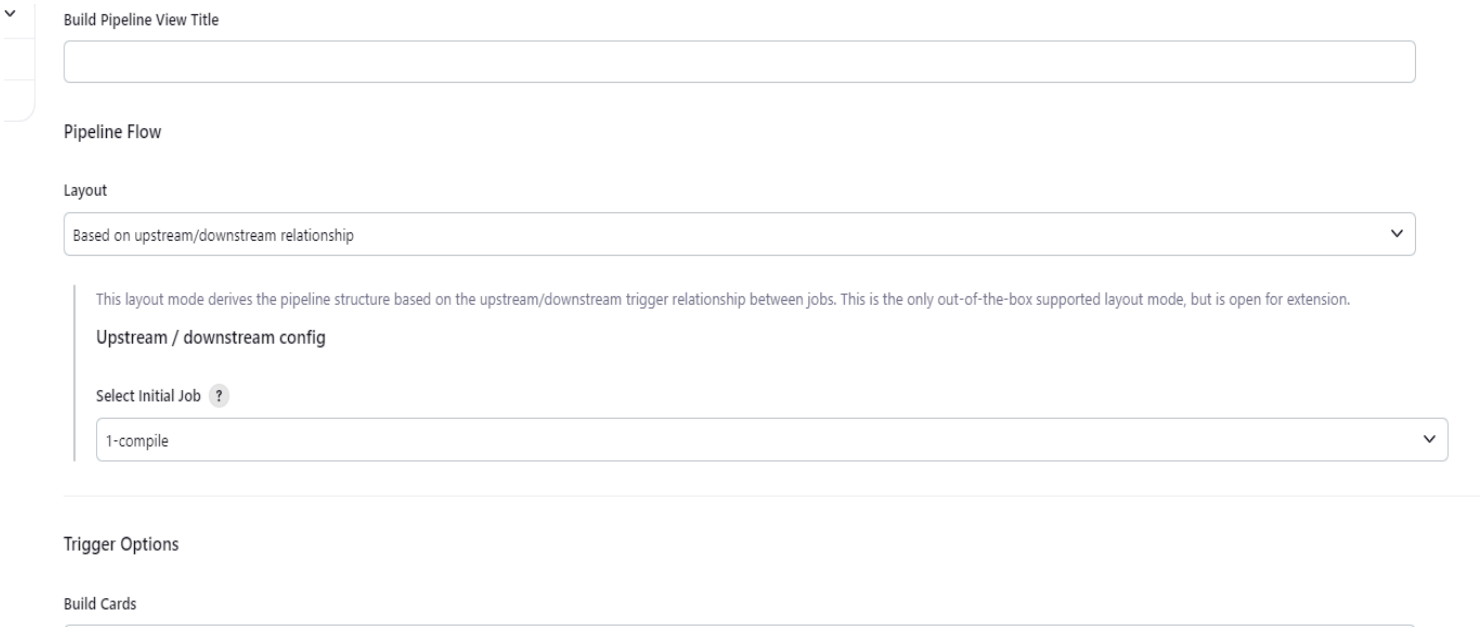


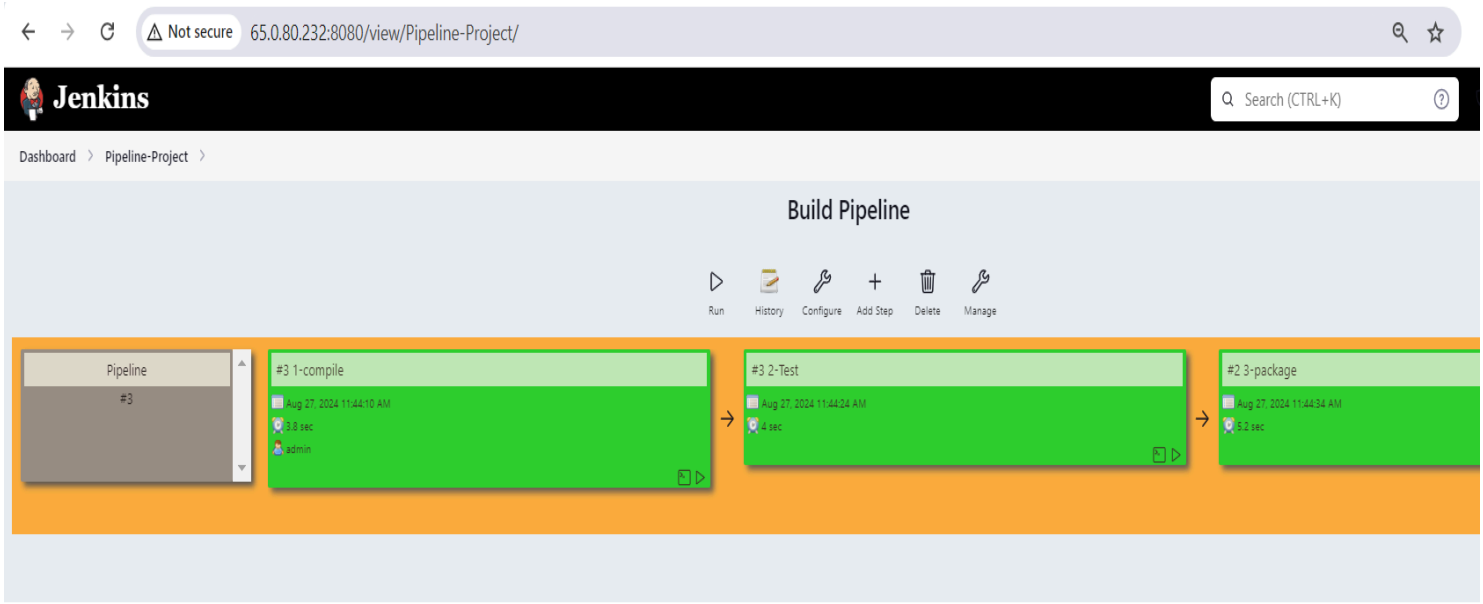
The war file is created at this location
/var/lib/jenkins/workspace/3-package/target/ABCtechnologies-1.0

```
Downloaded from central: https://repo.maven.apache.org/maven2/com/thoughtworks/xstream/xstream/1.4.10/xstream-1.4.10.jar (590 kB at 2.6 MB/s)
[␛[1;34mINFO␛[m] Packaging webapp
[␛[1;34mINFO␛[m] Assembling webapp [ABCtechnologies] in [/var/lib/jenkins/workspace/3-package/target/ABCtechnologies-1.0]
[␛[1;34mINFO␛[m] Processing war project
[␛[1;34mINFO␛[m] Copying webapp resources [/var/lib/jenkins/workspace/3-package/src/main/webapp]
[␛[1;34mINFO␛[m] Webapp assembled in [76 msecs]
[␛[1;34mINFO␛[m] Building war: /var/lib/jenkins/workspace/3-package/target/ABCtechnologies-1.0.war
[␛[1;34mINFO␛[m]
[␛[1;34mINFO␛[m] ␛[1m--- ␛[0;32mjacoco-maven-plugin:0.8.6:report␛[m ␛[1m(jacoco-site)␛[m @ ␛[36mABCtechnologies␛[0;1m ---␛[m
[␛[1;34mINFO␛[m] Loading execution data file /var/lib/jenkins/workspace/3-package/target/jacoco.exec
[␛[1;34mINFO␛[m] Analyzed bundle 'RetailModule' with 2 classes
[␛[1;34mINFO␛[m] ␛[1m------------------------------------------------------------------------␛[m
[␛[1;34mINFO␛[m] ␛[1;32mBUILD SUCCESS␛[m
[␛[1;34mINFO␛[m] ␛[1m------------------------------------------------------------------------␛[m
[␛[1;34mINFO␛[m] Total time:  5.692 s
[␛[1;34mINFO␛[m] Finished at: 2024-08-27T11:30:46Z
[␛[1;34mINFO␛[m] ␛[1m------------------------------------------------------------------------␛[m
Finished: SUCCESS
```

2. Execute the CICD pipeline to execute the jobs created in step1

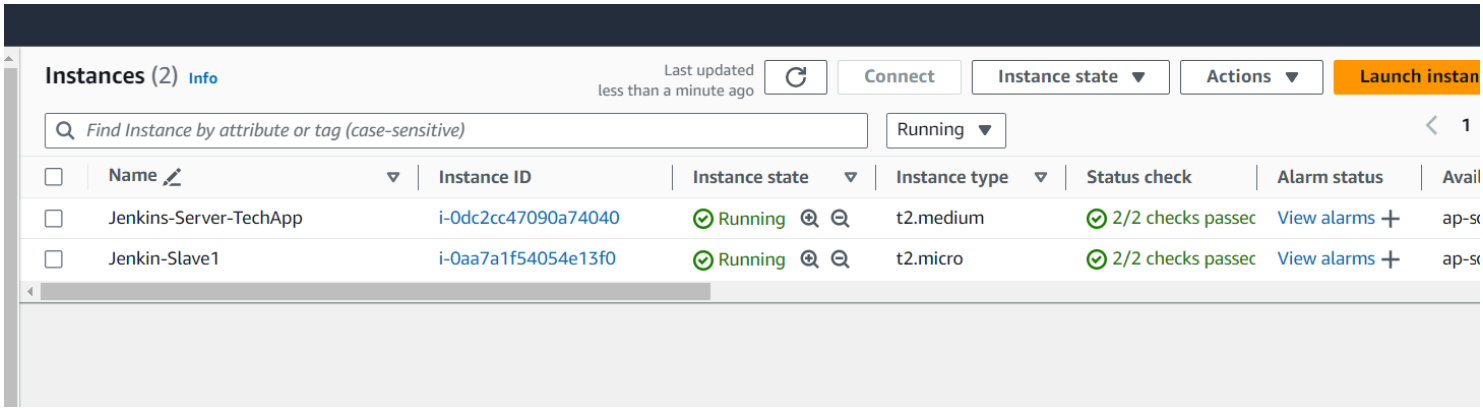
We have to download the **Build Pipeline** plugin to see all the jobs as a pipeline. Configure a pipeline “Pipeline-Project” and select “Build Pipeline View”. To configure the pipeline go to “Upstream/ downstream config” and then select > “1-compile” as required.





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

To create the slave. Go to Aws> Create new EC2 instance “Jenkin-Slave1” of Linux OS > using existing keypair > Security group rule of all traffic anywhere > instance type t2.micro(1 vcpu, 1 GiB memory) Create the VM> Connect to the machine CLI

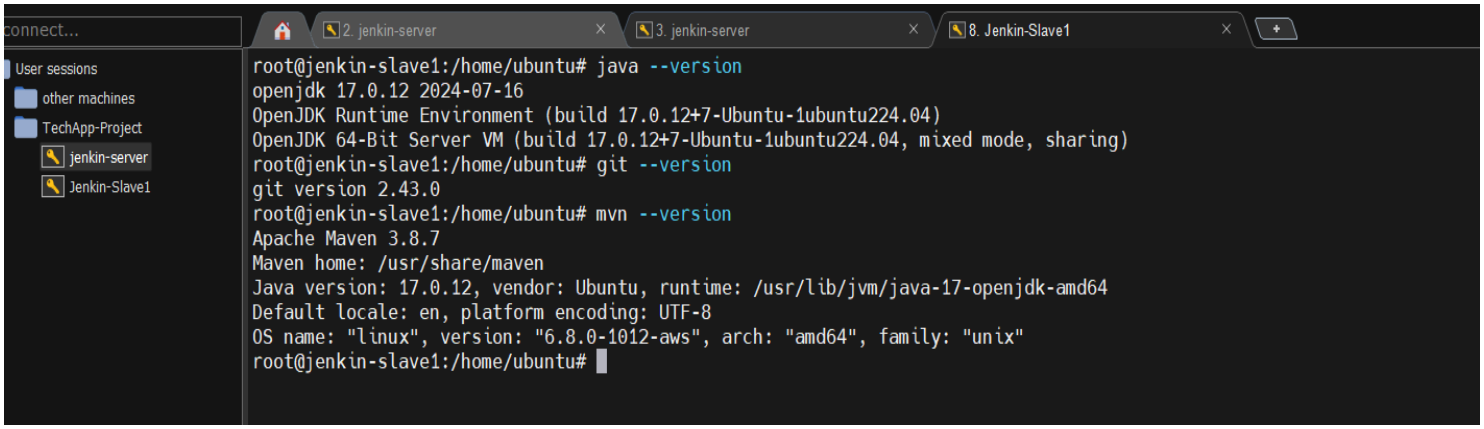


The **master machine** is the only server where Jenkins is installed. You create job configurations, build steps, source code management, and triggers on the master machine. You can specify which machine will do the job. The console output of jobs running on other machines can be viewed on the master machine. Java versions 11 and 17 should be present on the master and slave.

Slave machines are virtual machines (VMs) running any operating system, but they do not have Jenkins installed. Their main job is to receive tasks from the master machine and carry them out. Any tools required for the tasks must be installed on the slave machines. Each slave has an empty folder set up to serve as the workspace for the job.

Now install the following on slave:

```
#To Install Java
sudo apt install openjdk-17-jdk -y
#To Install Git
sudo apt install git -y
#To Install maven
sudo apt-get install maven
```



Root Directory for Jenkins to Store Temporary Data on Slave.

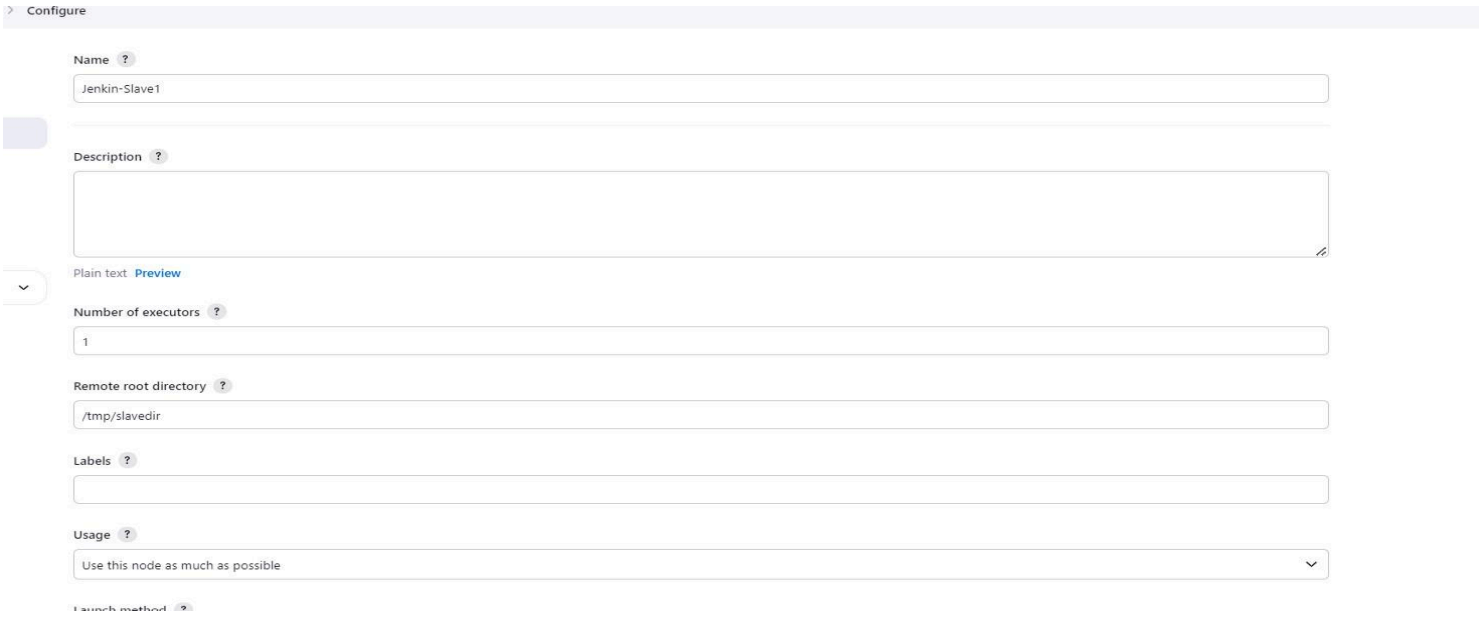
The next step is to create a root directory, this will be used exclusively by the Jenkins master to place remote files and create job workspace.

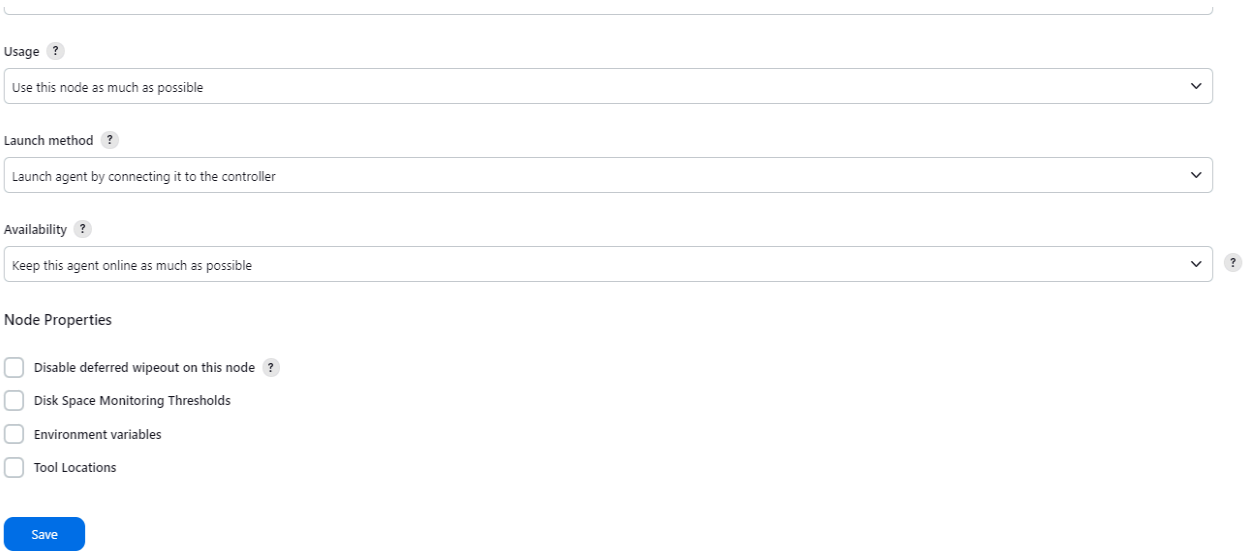
```
#To create a root directory
cd /tmp
mkdir slavedir
#To Give read/write/execute access to the root directory
chmod -R 777 /tmp/slavedir
```

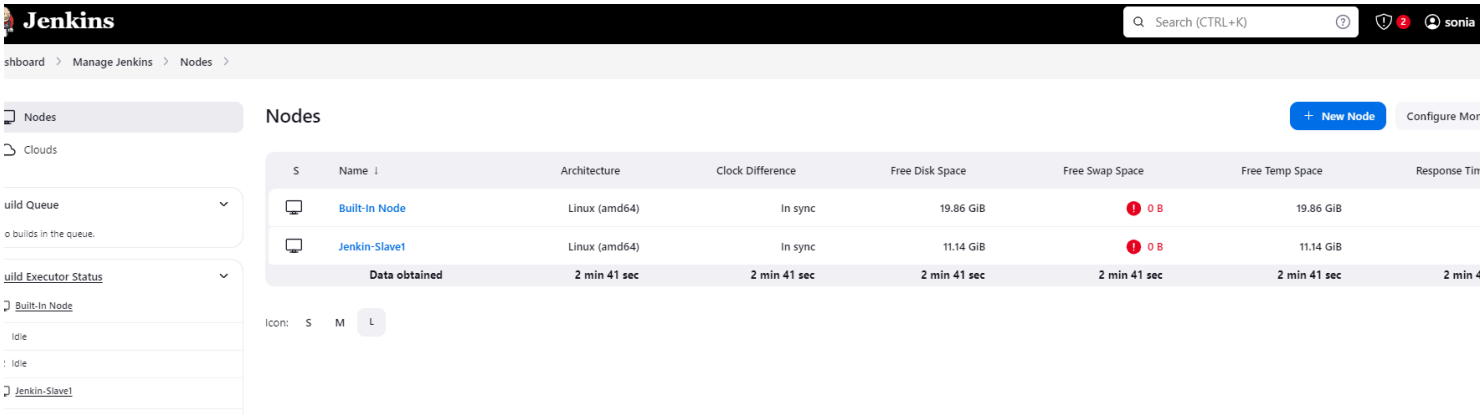
```
root@jenkin-slave1:/tmp# mkdir slavedir
root@jenkin-slave1:/tmp# ls
hsperfdata_root                                                 systemd-private-7f7b847719094060ab2fe5b08ecb13e3-chrony.service-SGZzrZ
slavedir                                                        systemd-private-7f7b847719094060ab2fe5b08ecb13e3-polkit.service-2iiZf9
snap-private-tmp                                                systemd-private-7f7b847719094060ab2fe5b08ecb13e3-systemd-logind.service-P83h8i
systemd-private-7f7b847719094060ab2fe5b08ecb13e3-ModemManager.service-7nupnE  systemd-private-7f7b847719094060ab2fe5b08ecb13e3-systemd-resolved.service-OQWgQa
root@jenkin-slave1:/tmp# chmod -R 777 /tmp/slavedir
root@jenkin-slave1:/tmp# ls -lh
total 32K
drwxr-xr-x 2 root root 4.0K Aug 27 12:19 hsperfdata_root
drwxrwxrwx 2 root root 4.0K Aug 27 12:21 slavedir
drwx------ 2 root root 4.0K Aug 27 12:11 snap-private-tmp
drwx------ 3 root root 4.0K Aug 27 12:11 systemd-private-7f7b847719094060ab2fe5b08ecb13e3-ModemManager.service-7nupnE
drwx------ 3 root root 4.0K Aug 27 12:11 systemd-private-7f7b847719094060ab2fe5b08ecb13e3-chrony.service-SGZzrZ
drwx------ 3 root root 4.0K Aug 27 12:11 systemd-private-7f7b847719094060ab2fe5b08ecb13e3-polkit.service-2iiZf9
drwx------ 3 root root 4.0K Aug 27 12:11 systemd-private-7f7b847719094060ab2fe5b08ecb13e3-systemd-logind.service-P83h8i
drwx------ 3 root root 4.0K Aug 27 12:11 systemd-private-7f7b847719094060ab2fe5b08ecb13e3-systemd-resolved.service-OQWgQa
root@jenkin-slave1:/tmp#
```

Setting Up Slave:

Go to Manage Jenkins > Nodes > Click on + New node
> Give node Name: Slave > Select Permanent agent > Click create



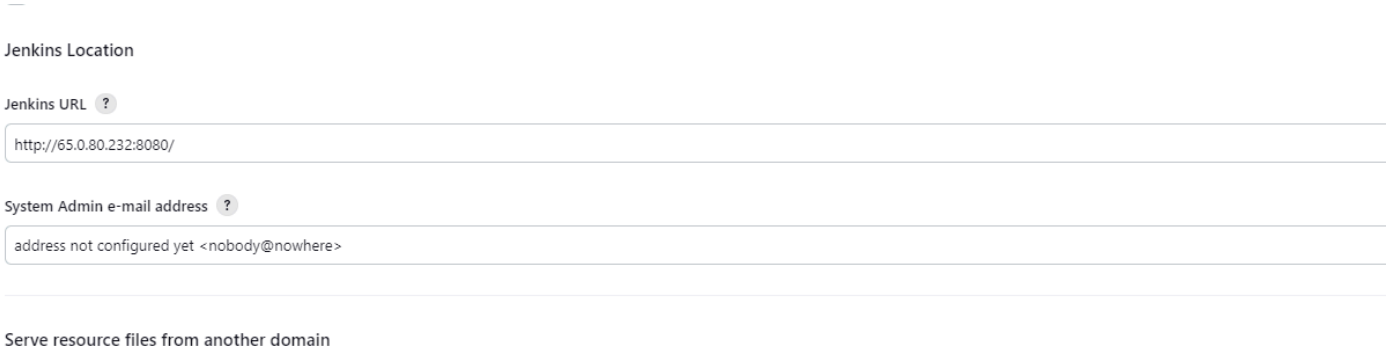




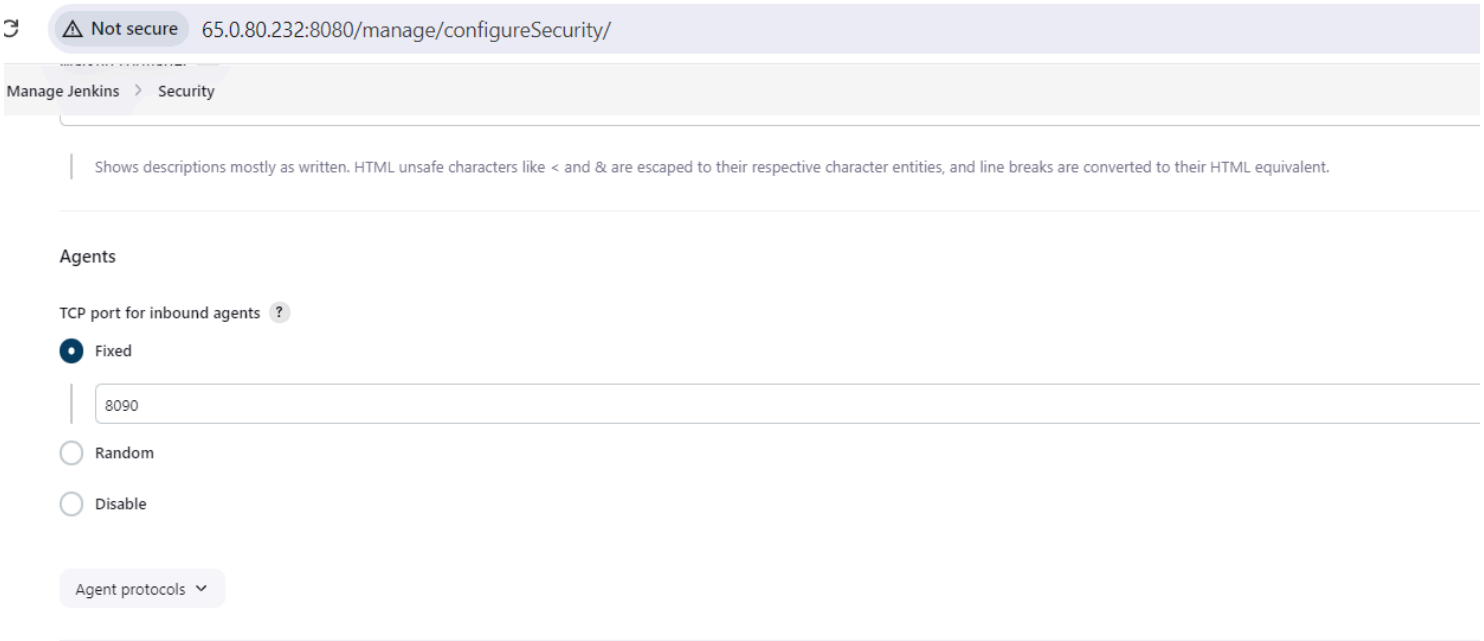
Setup on Jenkins Master:

Go to manage Jenkins> configure global security> scroll down to jenkins location and change the jenkins URL to the latest URL this is because public IP address changes all the time.

http://65.0.80.232:8080/ and then save the changes.



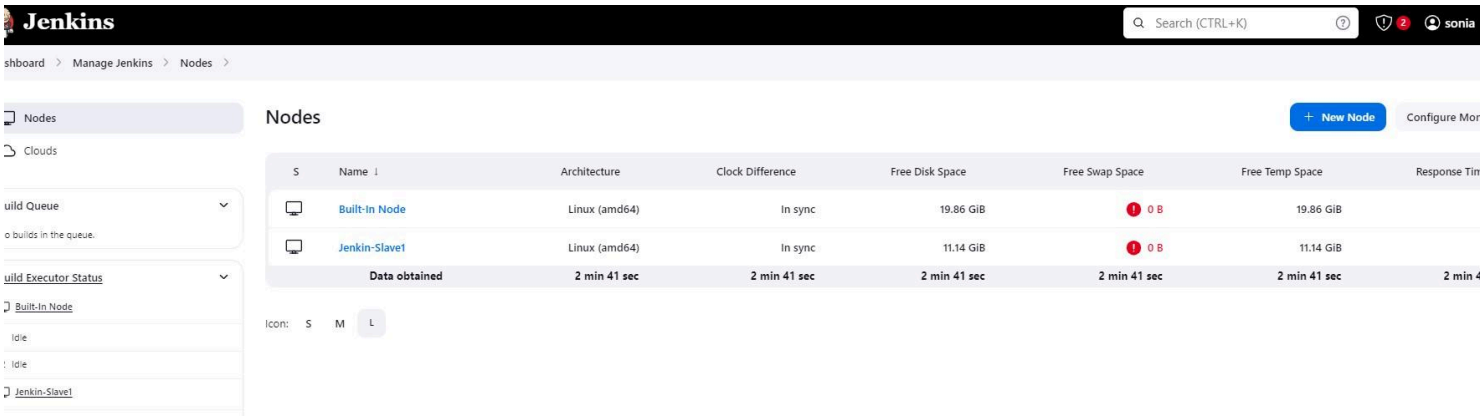
Now go to manage jenkins→ security→ Tcp port for inbound agent→ fixed →8090



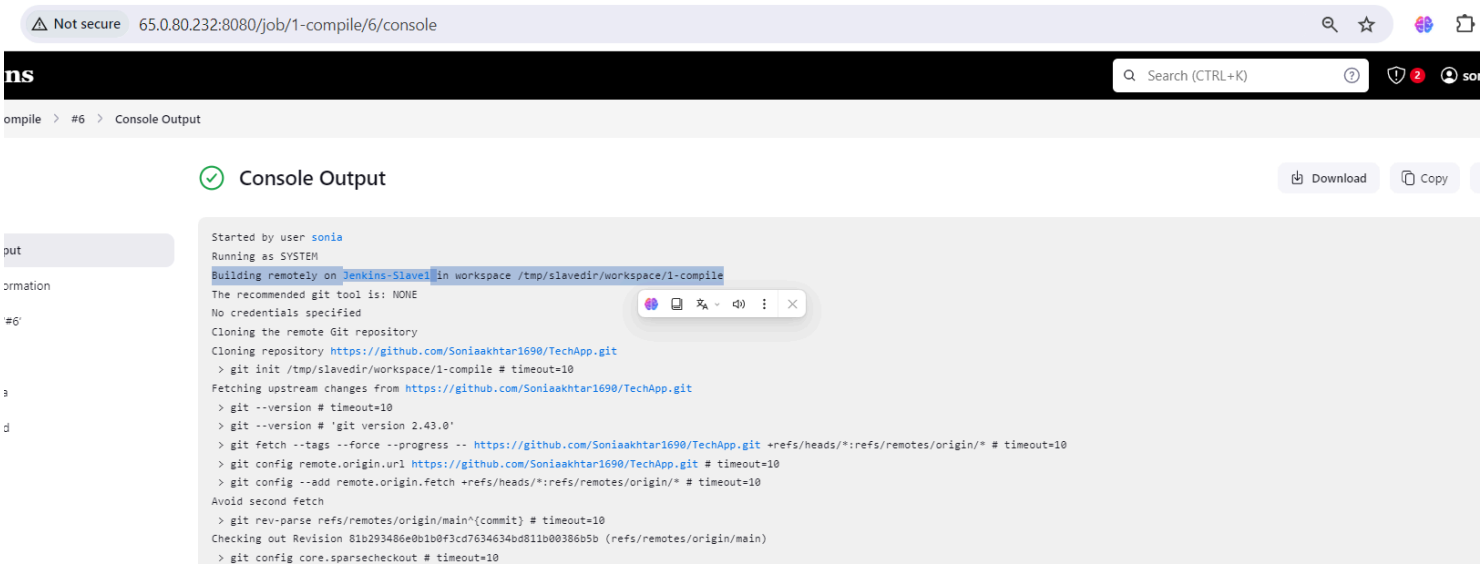
Now we are ready to connect the agent node to the master. Use the following command in agent cli

```
java -jar agent.jar -url http://65.0.80.232:8080/ -secret
1d3df6159f14b7eee6f76110f0ad724c329bcbb53811be9da5cf618a4f791379 -name
"Jenkin-Slave1" -workDir "/tmp/slavedir"
```

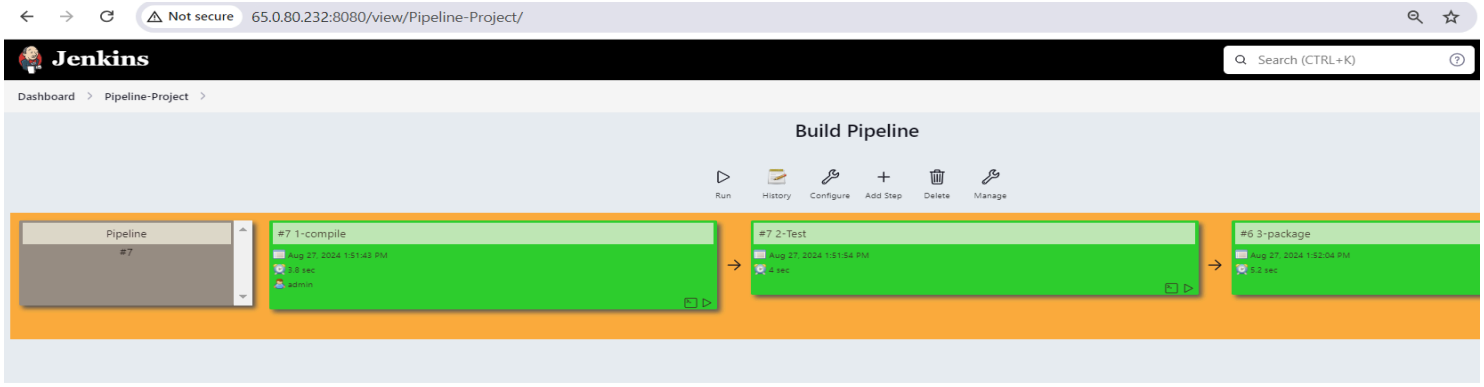
```
root@jenkin-slave1:/home/ubuntu# java -jar agent.jar -url http://65.0.80.232:8080/ -secret 1d3df6159f14b7eee6f76110f0ad724c329bcbb53811be9da5cf618a4f791379 -na
me "Jenkin-Slave1" -workDir "/tmp/slavedir"
Aug 27, 2024 1:14:16 PM org.jenkinsci.remoting.engine.WorkDirManager initializeWorkDir
INFO: Using /tmp/slavedir/remoting as a remoting work directory
Aug 27, 2024 1:14:16 PM org.jenkinsci.remoting.engine.WorkDirManager setupLogging
INFO: Both error and output logs will be printed to /tmp/slavedir/remoting
Aug 27, 2024 1:14:16 PM hudson.remoting.Launcher createEngine
INFO: Setting up agent: Jenkin-Slave1
Aug 27, 2024 1:14:16 PM hudson.remoting.Engine startEngine
INFO: Using Remoting version: 3248.3250.v3277a_8e88c9b_
Aug 27, 2024 1:14:16 PM org.jenkinsci.remoting.engine.WorkDirManager initializeWorkDir
INFO: Using /tmp/slavedir/remoting as a remoting work directory
Aug 27, 2024 1:14:16 PM hudson.remoting.Launcher$CuiListener status
INFO: Locating server among [http://65.0.80.232:8080/]
Aug 27, 2024 1:14:16 PM org.jenkinsci.remoting.engine.JnlpAgentEndpointResolver resolve
INFO: Remoting server accepts the following protocols: [JNLP4-connect, Ping]
Aug 27, 2024 1:14:16 PM hudson.remoting.Launcher$CuiListener status
INFO: Agent discovery successful
  Agent address: 65.0.80.232
  Agent port:    8090
  Identity:      79:57:24:fa:4c:92:4d:76:c5:a7:6b:74:ef:47:75:2e
Aug 27, 2024 1:14:16 PM hudson.remoting.Launcher$CuiListener status
INFO: Handshaking
Aug 27, 2024 1:14:16 PM hudson.remoting.Launcher$CuiListener status
INFO: Connecting to 65.0.80.232:8090
Aug 27, 2024 1:14:16 PM hudson.remoting.Launcher$CuiListener status
INFO: Server reports protocol JNLP4-connect-proxy not supported, skipping
Aug 27, 2024 1:14:16 PM hudson.remoting.Launcher$CuiListener status
INFO: Trying protocol: JNLP4-connect
Aug 27, 2024 1:14:16 PM org.jenkinsci.remoting.protocol.impl.BIONetworkLayer$Reader run
INFO: Waiting for ProtocolStack to start.
Aug 27, 2024 1:14:17 PM hudson.remoting.Launcher$CuiListener status
INFO: Remote identity confirmed: 79:57:24:fa:4c:92:4d:76:c5:a7:6b:74:ef:47:75:2e
Aug 27, 2024 1:14:17 PM hudson.remoting.Launcher$CuiListener status
INFO: Connected
```



See the Compile job console output on the slave machine below



See the Pipeline view of all 3 jobs below



Task 3: Write a Docker file. Create an image and container for the Docker host. Integrate docker host with Jenkins. Create a 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.**

First, install the Docker on the slave machine, Ref

```
root@jenkin-slave1:/# sudo systemctl status docker
● docker.service - Docker Application Container Engine
     Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; preset: enabled)
     Active: active (running) since Tue 2024-08-27 14:47:39 UTC; 24s ago
TriggeredBy: ● docker.socket
       Docs: https://docs.docker.com
   Main PID: 8775 (dockerd)
      Tasks: 9
     Memory: 20.8M (peak: 21.6M)
        CPU: 221ms
     CGroup: /system.slice/docker.service
             └─8775 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

Aug 27 14:47:39 jenkin-slave1 systemd[1]: Starting docker.service - Docker Application Container Engine...
Aug 27 14:47:39 jenkin-slave1 dockerd[8775]: time="2024-08-27T14:47:39.300715955Z" level=info msg="Starting up"
Aug 27 14:47:39 jenkin-slave1 dockerd[8775]: time="2024-08-27T14:47:39.301756638Z" level=info msg="detected 127.0.0.53 nameserver, assumin
Aug 27 14:47:39 jenkin-slave1 dockerd[8775]: time="2024-08-27T14:47:39.395803785Z" level=info msg="Loading containers: start."
Aug 27 14:47:39 jenkin-slave1 dockerd[8775]: time="2024-08-27T14:47:39.640637010Z" level=info msg="Loading containers: done."
Aug 27 14:47:39 jenkin-slave1 dockerd[8775]: time="2024-08-27T14:47:39.654446601Z" level=info msg="Docker daemon" commit=f9522e5 container
Aug 27 14:47:39 jenkin-slave1 dockerd[8775]: time="2024-08-27T14:47:39.654527944Z" level=info msg="Daemon has completed initialization"
Aug 27 14:47:39 jenkin-slave1 dockerd[8775]: time="2024-08-27T14:47:39.689868538Z" level=info msg="API listen on /run/docker.sock"
Aug 27 14:47:39 jenkin-slave1 systemd[1]: Started docker.service - Docker Application Container Engine.
lines 1-21/21 (END)
```

I used Tomcat Image from the Docker Hub Repository to Create the Docker file.
I am creating a docker image in Tomcat's latest image in the FROM command. Then add the war file to the tomcat $CATALINA_HOME folder which /usr/local/tomcat/webapps

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

To Enhance the Package Job to Create Docker File and Build Docker Image:

Execute shell ?

Command

See the list of available environment variables

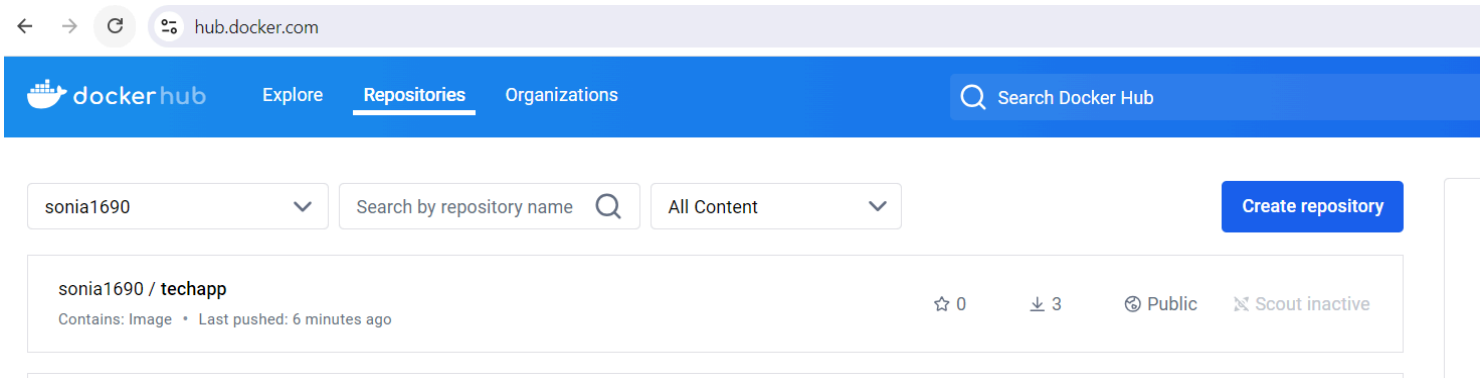
```
cd /home/ubuntu
rm -rf warfile
mkdir warfile
cd warfile
cp /tmp/slavedir/workspace/3-package/target/ABCtechnologies-1.0.war .
touch Dockerfile
cat > Dockerfile <<EOT
FROM tomcat:9.0.73-jre8
ADD . /usr/local/tomcat/webapps
EOT
docker build -t techapp .
docker tag techapp sonia1690/techapp:lts
docker login --username sonia1690 --password Sonia1690$
docker push sonia1690/techapp:lts
```

The steps below are carried out after the maven task of Packaging, hence, to enhance the Job.

- Execute Shell Command Step in Jenkins
- Create a new Directory at Home (that is the warfile)
- Move ABCtechnologies-1.0.war file to the warfile directory
- Create a Docker file as shown in the screenshot above.
- Run the Docker build command to build the image
- I tagged the image with my private Docker

```
Started by user sonia
Running as SYSTEM
Building remotely on Jenkins-Slave1 in workspace /tmp/slavedir/workspace/3-package
The recommended git tool is: NONE
No credentials specified
 > git rev-parse --resolve-git-dir /tmp/slavedir/workspace/3-package/.git # timeout=10
Fetching changes from the remote Git repository
 > git config remote.origin.url https://github.com/Soniaakhtar1690/TechApp.git # timeout=10
Fetching upstream changes from https://github.com/Soniaakhtar1690/TechApp.git
 > git --version # timeout=10
 > git --version # 'git version 2.43.0'
 > git fetch --tags --force --progress -- https://github.com/Soniaakhtar1690/TechApp.git +refs/heads/*:refs/remotes/origin/* # timeout=10
 > git rev-parse refs/remotes/origin/main^{commit} # timeout=10
Checking out Revision 81b293486e0b1b0f3cd7634634bd811b00386b5b (refs/remotes/origin/main)
 > git config core.sparsecheckout # timeout=10
 > git checkout -f 81b293486e0b1b0f3cd7634634bd811b00386b5b # timeout=10
Commit message: "First Commmit"
 > git rev-list --no-walk 81b293486e0b1b0f3cd7634634bd811b00386b5b # timeout=10
[3-package] $ mvn package
[␛[1;34mINFO␛[m] Scanning for projects...
```

```
d7849882b980: Waiting
37159e8293d8: Waiting
04d1dcab20cb: Waiting
b93c1bd012ab: Waiting
30c504649c00: Mounted from library/tomcat
61a4e99a7859: Mounted from library/tomcat
e11606ab2e41: Mounted from library/tomcat
b3a672dc471e: Mounted from library/tomcat
d7849882b980: Mounted from library/tomcat
37159e8293d8: Mounted from library/tomcat
04d1dcab20cb: Mounted from library/tomcat
2966cf2468d5: Pushed
b93c1bd012ab: Mounted from library/tomcat
lts: digest: sha256:52bace31d1534590df1af6ccaaab9c83b3a3201b3442eade09eec60bbc087782 size: 2207
Finished: SUCCESS
```



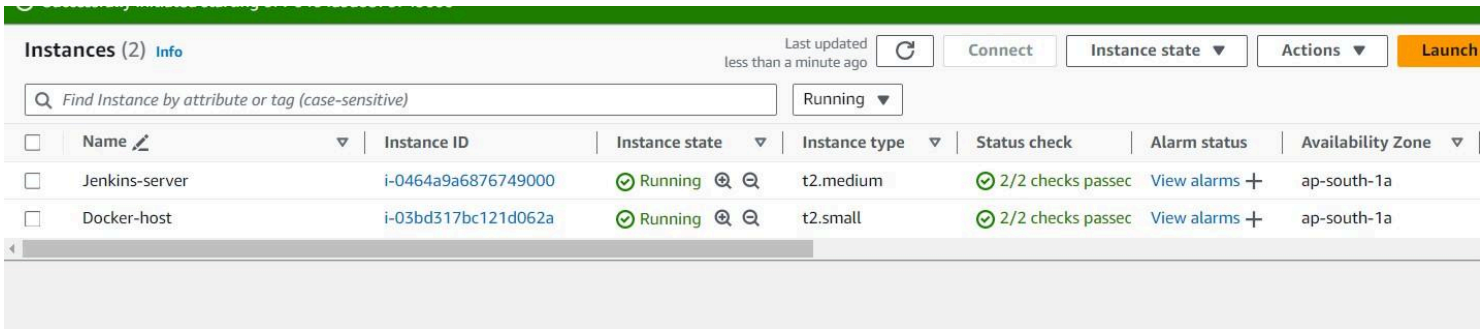
After the push, I downloaded my image to test on the Slave machine if I could access the APP or not

```
Status: Image is up to date for sonia1690/techapp:lts
docker.io/sonia1690/techapp:lts
root@jenkin-slave1:/home/ubuntu/warfile# docker images
REPOSITORY          TAG       IMAGE ID       CREATED          SIZE
techapp             latest    1f0d39460bf1   7 minutes ago    248MB
sonia1690/techapp   lts       1f0d39460bf1   7 minutes ago    248MB
<none>              <none>    d7004e6825bd   14 minutes ago   248MB
root@jenkin-slave1:/home/ubuntu/warfile# docker run -d --name techapp-container -p 8585=8080 sonia1690/techapp
docker: no port specified: :/tcp<empty>.
See 'docker run --help'.
root@jenkin-slave1:/home/ubuntu/warfile# docker run -d --name techapp-container -p 8585:8080 sonia1690/techapp
Unable to find image 'sonia1690/techapp:latest' locally
latest: Pulling from sonia1690/techapp
128d54f2c9b1: Pull complete
ed02fb405f69: Pull complete
ef1a03968e31: Pull complete
fb79e073623a: Pull complete
d232b5c03d49: Pull complete
126ec62633ec: Pull complete
Digest: sha256:32a240d6abb03d7080c1bb328b0ed674da3072d7ca743d4b92005334d3c75cac
Status: Downloaded newer image for sonia1690/techapp:latest
427aa71a85a1b950ef29935bd6bae1bb5813cbd6ef2b1f6db095a59cb1578193
```

```
Note: Waiting for the process to end and use of the -force option require that $CATALINA_PID is defined
root@f4e897a2dacb:/usr/local/tomcat/bin# exit
exit
root@jenkin-slave1:/home/ubuntu/warfile# docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
root@jenkin-slave1:/home/ubuntu/warfile# docker run -d -p 8583:8080 techapp
203c1226debfae7ac16b2e8831f3c6398fd98ded0693573199c2a1a37bc926a0
root@jenkin-slave1:/home/ubuntu/warfile# docker ps
CONTAINER ID   IMAGE     COMMAND             CREATED         STATUS         PORTS                                       NAMES
203c1226debf   techapp   "catalina.sh run"   3 seconds ago   Up 3 seconds   0.0.0.0:8583->8080/tcp, :::8583->8080/tcp   sweet_dirac
root@jenkin-slave1:/home/ubuntu/warfile#
```

Integrate docker host with Jenkins over SSH

Create a new ec2 instance, Docker-host with instance type t2.small



install the Docker on the Docker host, [Ref]

```
root@docker-host:/home/ubuntu# docker --version
Docker version 27.3.1, build ce12230
root@docker-host:/home/ubuntu#
```

Now we will establish an SSH connection between both machines as the jenkins server wants to go inside the Docker server so jenkins is a client and Docker is a server in this case. We will create SSH keys in the client-server which is jenkins and then we will add the public key to the docker server.

I will create a dockeradmin user and dockeradmin directory on both machines.

Jenkins-server

```
useradd dockeradmin
passwd dockeradmin
visudo    #add dockeradmin to the sudo file
mkdir /home/dockeradmin
cd /home/dockeradmin
mkdir .ssh
chmod 700 .ssh
chown dockeradmin .ssh
su dockeradmin
```

Now run the following commands as a dockeradmin user

```
ssh docker-host-ip
```

```
root@jenkins-server:/home/dockeradmin# pwd
/home/dockeradmin
root@jenkins-server:/home/dockeradmin# mkdir .ssh
root@jenkins-server:/home/dockeradmin# chmod 700 .ssh
root@jenkins-server:/home/dockeradmin# chown dockeradmin .ssh
root@jenkins-server:/home/dockeradmin# ls -la
total 12
```

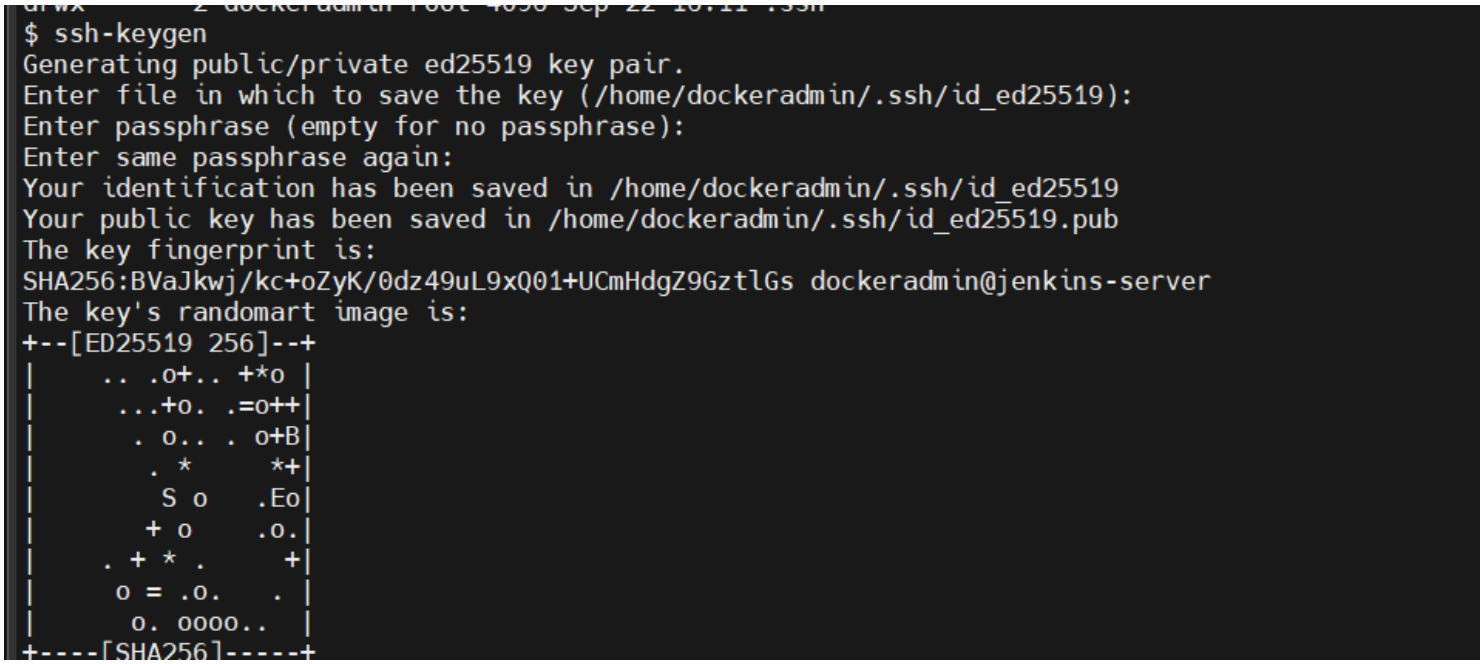
```
su dockeradmin
ssh docker-host-ip
```

```
$ ssh 172.0.5.39
The authenticity of host '172.0.5.39 (172.0.5.39)' can't be established.
ED25519 key fingerprint is SHA256:7ozilfwmQ4dSNXUcX97tT0ZEhMRKsmVSNEyzoeiFNXo.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '172.0.5.39' (ED25519) to the list of known hosts.
dockeradmin@172.0.5.39: Permission denied (publickey).
$ ls -la
```

Note: ssh docker-host-ip will create a file known_host in .ssh directory and add the docker-host.

The following command will create rsa keys in .ssh directory

```
ssh-keygen -t rsa
ls
```



Docker-server

Create a new user dockeradmin and add this to docker group, also create a new directory dockeradmin in /home directory

Note: As a root user

```
useradd dockeradmin
usermod -aG docker dockeradmin
visudo          #add dockeradmin user to sudo file
mkdir /home/dockeradmin
```

```
root@docker-host:/home# ls
dockeradmin  ubuntu
root@docker-host:/home# id dockeradmin/
id: ‘dockeradmin/’: no such user
root@docker-host:/home# id dockeradmin
uid=1002(dockeradmin) gid=1002(dockeradmin) groups=1002(dockeradmin),999(docker)
root@docker-host:/home# ls -la
total 16
drwxr-xr-x  4 root   root   4096 Sep 22 12:19 .
drwxr-xr-x 20 root   root   4096 Sep 22 15:46 ..
drwxr-xr-x  3 root   root   4096 Sep 22 12:19 dockeradmin
drwxr-x---  4 ubuntu ubuntu 4096 Sep 22 15:48 ubuntu
```

cd dockeradmin
mkdir .ssh
chmod 700 .ssh
chown dockeradmin .ssh

```
root@docker-host:/home/dockeradmin# mkdir .ssh
root@docker-host:/home/dockeradmin# chmod 700 .ssh
root@docker-host:/home/dockeradmin# chown dockeradmin .ssh
root@docker-host:/home/dockeradmin# ls
root@docker-host:/home/dockeradmin# ls -la
total 12
```

cd .ssh
touch authorized_keys
vi authorized_keys

```
root@docker-host:/home/dockeradmin/.ssh# touch authorized_keys
root@docker-host:/home/dockeradmin/.ssh# ls
authorized_keys
root@docker-host:/home/dockeradmin/.ssh# chmod 700 authorized_keys
root@docker-host:/home/dockeradmin/.ssh# chown dockeradmin authorized_keys
root@docker-host:/home/dockeradmin/.ssh# ls -la
total 8
drwx------ 2 dockeradmin root 4096 Sep 22 16:14 .
drwxr-xr-x 3 root        root 4096 Sep 22 16:12 ..
-rwx------ 1 dockeradmin root    0 Sep 22 16:14 authorized_keys
root@docker-host:/home/dockeradmin/.ssh# vi authorized_keys
```

Now copy the content of public key in jenkins server (client) and add the key to the authorized_key file in docker server(server)

```
root@jenkins-server:/home/dockeradmin/.ssh# ls
id_ed25519  id_ed25519.pub  known_hosts
root@jenkins-server:/home/dockeradmin/.ssh# cat id_ed25519.pub
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAI03iRuHTU92/mzqytWqt2QH1t2M+06g7Ed+FxuYrIqcM dockeradmin@jenkins-server
root@jenkins-server:/home/dockeradmin/.ssh#
```

```
root@docker-host:/home/dockeradmin/.ssh# ls
authorized_keys
root@docker-host:/home/dockeradmin/.ssh# cat authorized_keys
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAI03iRuHTU92/mzqytWqt2QH1t2M+06g7Ed+FxuYrIqcM dockeradmin@jenkins-server
root@docker-host:/home/dockeradmin/.ssh#
```

Jenkins server:

Now we will try to check the SSH connection

Note: as dockeradmin user

`ssh dockeradmin@docker-host-ip`

```
$ ssh dockeradmin@15.206.94.16
The authenticity of host '15.206.94.16 (15.206.94.16)' can't be established.
ED25519 key fingerprint is SHA256:7ozilfwmQ4dSNXUcX97tT0ZEhMRKsmVSNEyzoeiFNXo.
This host key is known by the following other names/addresses:
    ~/.ssh/known_hosts:1: [hashed name]
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '15.206.94.16' (ED25519) to the list of known hosts.
Welcome to Ubuntu 22.04.4 LTS (GNU/Linux 6.5.0-1022-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Sun Sep 22 16:16:27 UTC 2024

  System load:  0.0               Processes:             112
  Usage of /:   10.4% of 24.05GB  Users logged in:       1
  Memory usage: 12%               IPv4 address for eth0: 172.0.5.39
```

SSH connection is established successfully.

Docker host

Go to the /etc/ssh/sshd_config file and uncomment the following

```
vi /etc/ssh/sshd_config
```

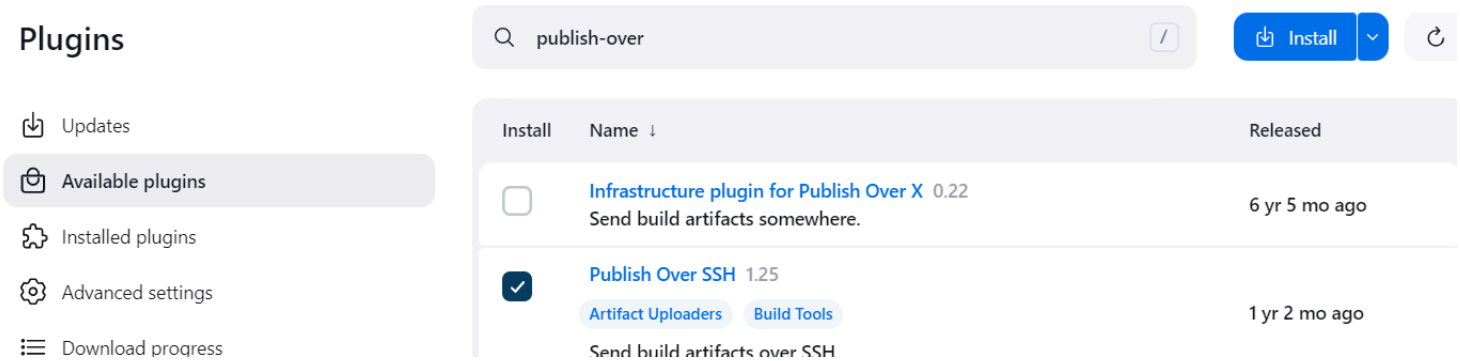
```
#IgnoreRhosts yes

# To disable tunneled clear text passwords, change to no here!
PasswordAuthentication yes
#PermitEmptyPasswords no

# Change to yes to enable challenge-response passwords (beware issues with
```

```
systemctl restart ssh
```

Configure Jenkins:

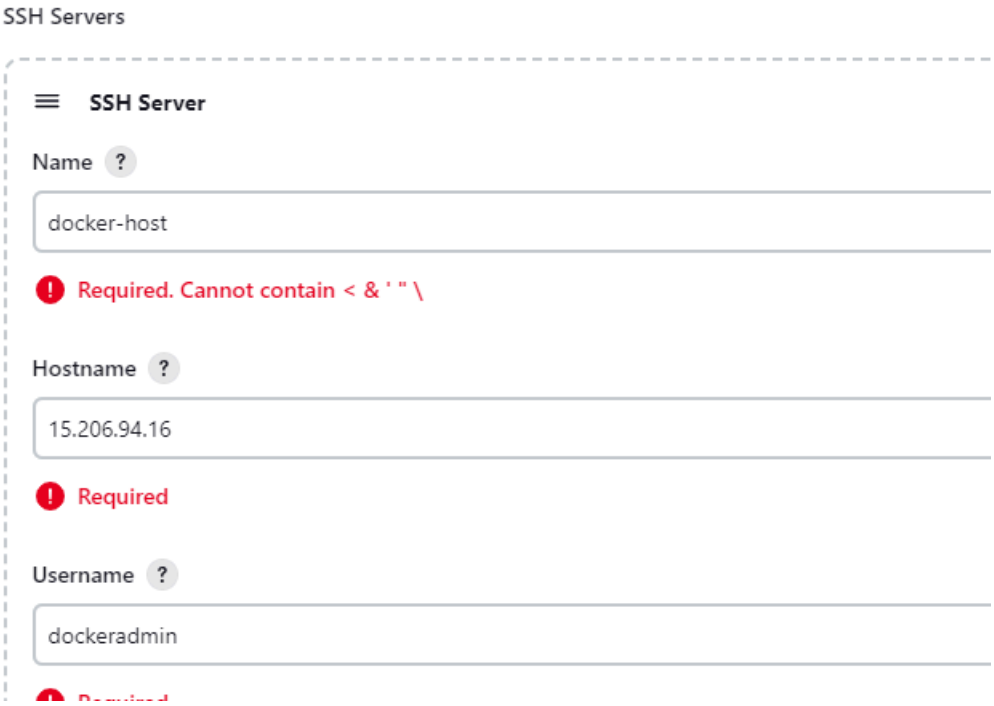
Install necessary Jenkins plugins

Navigate to Manage Jenkins > Manage Plugins and install the **Docker plugin, Docker-build-steps, Cloudbease Docker Build and publish** and **Docker Pipeline plugin and Publish-over-ssh**



Go to manage jenkins>systems> ssh server

Add the following detail to the server
Name: docker-host (hostname of server which we want to connect over SSH)
Hostname: Public ip of docker host
username: dockeradmin (the user which we want to connect over the docker host)

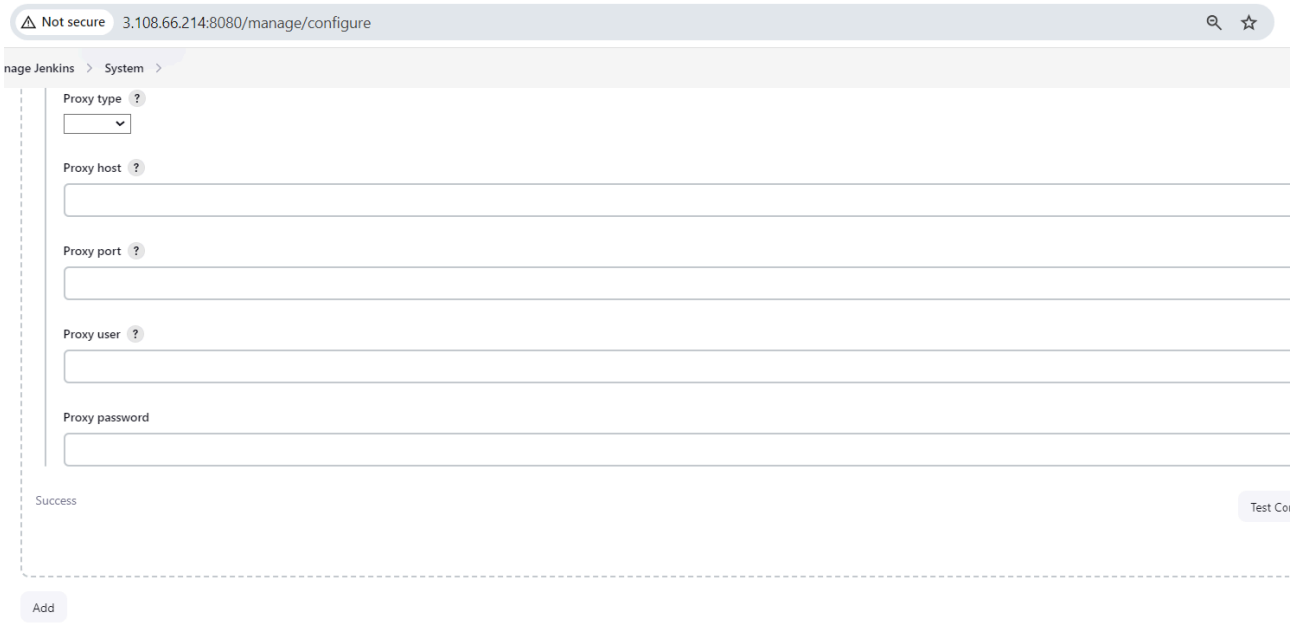


Tick the following box:
Use password authentication, or use a different key

Key: give the private key of the jenkins server (client)

```
root@jenkins-server:/home/dockeradmin/.ssh# cat id_ed25519
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAAAMwAAAAtzc2gtZW
QyNTUxOQAAACDt4kbh01Pdv5s6srVqrdkB9bdjPtOoOxHfhcbmKyKnDAAAAKAl3iMfJd4j
HwAAAAtzc2gtZWQyNTUxOQAAACDt4kbh01Pdv5s6srVqrdkB9bdjPtOoOxHfhcbmKyKnDA
AAAEAPW1QjPkPDcZzqUTiEKb1FMp4778ySo0r2F1bTCtSP9+3iRuHTU92/mzqytWqt2QH1
t2M+06g7Ed+FxuYrIqcMAAAAGmRvY2tlcmFkbWluQGplbmtpbnMtc2VydmVyAQID
-----END OPENSSH PRIVATE KEY-----
root@jenkins-server:/home/dockeradmin/.ssh# ls -la
```

Test the configuration
It should be successful

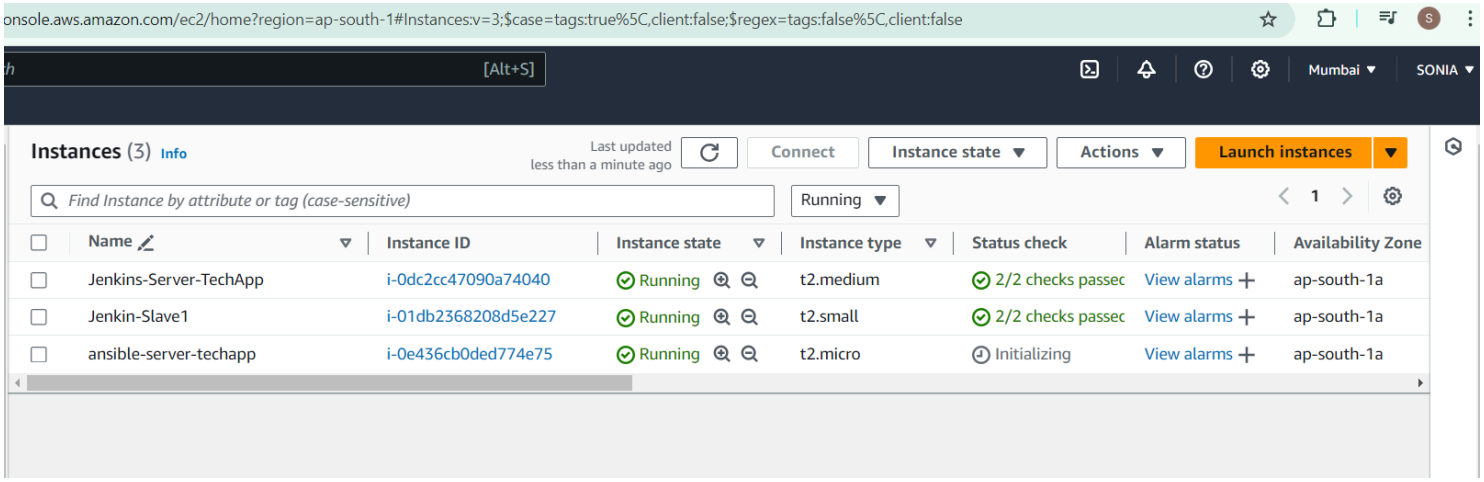


Task 4: Integrate Docker host with Ansible. Write an Ansible playbook to create an Image and create a continuer. Integrate Ansible with Jenkins. Deploy ansible-playbook. CI/CD job to build code on Ansible and deploy it on the docker container

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**

Docker host Integration with Ansible

First, I created a new Ec2 instance t2.micro




```
sudo apt update
sudo apt install software-properties-common
sudo add-apt-repository --yes --update ppa:ansible/ansible
sudo apt install ansible
ansible ––version
```

```
root@ansible-server:/home/ubuntu# ansible --version
ansible [core 2.16.10]
  config file = /etc/ansible/ansible.cfg
  configured module search path = ['/root/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python3/dist-packages/ansible
  ansible collection location = /root/.ansible/collections:/usr/share/ansible/collections
  executable location = /usr/bin/ansible
  python version = 3.12.3 (main, Apr 10 2024, 05:33:47) [GCC 13.2.0] (/usr/bin/python3)
  jinja version = 3.1.2
  libyaml = True
root@ansible-server:/home/ubuntu#
```

Next, we create a new user “ansadmin” and set a password for this user.

```
root@ansible-server:/etc/ansible# cd /etc/ansible
root@ansible-server:/etc/ansible# ls
ansible.cfg  hosts  roles
root@ansible-server:/etc/ansible# useradd ansadmin
root@ansible-server:/etc/ansible# passwd ansadmin
New password:
Retype new password:
passwd: password updated successfully
root@ansible-server:/etc/ansible#
```

We have to add this user to the sudo file so that it can run root commands

visudo

asadmin ALL=(ALL:ALL) ALL

```
# User alias specification

# Cmnd alias specification

# User privilege specification
root    ALL=(ALL:ALL) ALL

# Members of the admin group may gain root privileges
%admin ALL=(ALL) ALL

# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL
 ansadmin ALL=(ALL:ALL) ALL
# See sudoers(5) for more information on "@include" directives:

@includedir /etc/sudoers.d
```

Now we have to install docker on ansible-server to use ansible-playbook to create images and containers.
Reference: Here

```
root@ansible-server:/etc/ansible# docker --version
Docker version 27.2.0, build 3ab4256
root@ansible-server:/etc/ansible# systemctl status docker
● docker.service - Docker Application Container Engine
     Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; preset: enabled)
     Active: active (running) since Thu 2024-08-29 06:03:13 UTC; 1min 26s ago
TriggeredBy: ● docker.socket
       Docs: https://docs.docker.com
   Main PID: 3907 (dockerd)
      Tasks: 8
     Memory: 31.0M (peak: 31.1M)
        CPU: 336ms
     CGroup: /system.slice/docker.service
             └─3907 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

Aug 29 06:03:13 ansible-server systemd[1]: Starting docker.service - Docker Application Container Engine...
Aug 29 06:03:13 ansible-server dockerd[3907]: time="2024-08-29T06:03:13.170018851Z" level=info msg="Starting up"
Aug 29 06:03:13 ansible-server dockerd[3907]: time="2024-08-29T06:03:13.171559089Z" level=info msg="detected 127.0.0.53 names
Aug 29 06:03:13 ansible-server dockerd[3907]: time="2024-08-29T06:03:13.441958745Z" level=info msg="Loading containers: start
Aug 29 06:03:13 ansible-server dockerd[3907]: time="2024-08-29T06:03:13.824557451Z" level=info msg="Loading containers: done.
Aug 29 06:03:13 ansible-server dockerd[3907]: time="2024-08-29T06:03:13.851140186Z" level=info msg="Docker daemon" commit=3ab
Aug 29 06:03:13 ansible-server dockerd[3907]: time="2024-08-29T06:03:13.851270345Z" level=info msg="Daemon has completed init
Aug 29 06:03:13 ansible-server dockerd[3907]: time="2024-08-29T06:03:13.909723246Z" level=info msg="API listen on /run/docker
Aug 29 06:03:13 ansible-server systemd[1]: Started docker.service - Docker Application Container Engine.
lines 1-21/21 (END)
```

We have to add ansadmin user to the docker group so that ansadmin user can perform docker commands.

```
usermod -aG docker ansadmin
```

PasswordAuthentication Configuration:

We have to enable password authentication settings so that only that machine can communicate with the ansible-server with a public key.

```
vi /etc/ssh/sshd_config
```

```
# Don't read the user's ~/.rhosts and ~/.shosts files
#IgnoreRhosts yes

# To disable tunneled clear text passwords, change to no here!
PasswordAuthentication yes
#PermitEmptyPasswords no

# Change to yes to enable challenge-response passwords (beware issues with
# some PAM modules and threads)
```

Install and start the OPENSSH-Server on ansible-server:

```
sudo apt update && sudo apt upgrade
sudo apt install openssh-server -y
sudo systemctl enable ssh
sudo systemctl start ssh
```

```
Synchronizing state of ssh.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable ssh
Created symlink /etc/systemd/system/sshd.service → /usr/lib/systemd/system/ssh.service.
Created symlink /etc/systemd/system/multi-user.target.wants/ssh.service → /usr/lib/systemd/system/ssh.service.
root@ansible-server:/etc/ssh# sudo systemctl start ssh
root@ansible-server:/etc/ssh# sudo systemctl status ssh
● ssh.service - OpenBSD Secure Shell server
     Loaded: loaded (/usr/lib/systemd/system/ssh.service; enabled; preset: enabled)
    Drop-In: /usr/lib/systemd/system/ssh.service.d
             └─ec2-instance-connect.conf
     Active: active (running) since Thu 2024-08-29 06:20:54 UTC; 19s ago
TriggeredBy: ● ssh.socket
       Docs: man:sshd(8)
             man:sshd_config(5)
    Process: 29652 ExecStartPre=/usr/sbin/sshd -t (code=exited, status=0/SUCCESS)
   Main PID: 29654 (sshd)
      Tasks: 1 (limit: 1130)
     Memory: 2.0M (peak: 2.3M)
        CPU: 23ms
     CGroup: /system.slice/ssh.service
             └─29654 "sshd: /usr/sbin/sshd -D -o AuthorizedKeysCommand /usr/share/ec2-instance-connect/eic_run_authorized_keys

Aug 29 06:20:54 ip-192-0-0-5 systemd[1]: Starting ssh.service - OpenBSD Secure Shell server...
Aug 29 06:20:54 ip-192-0-0-5 sshd[29654]: Server listening on :: port 22.
Aug 29 06:20:54 ip-192-0-0-5 systemd[1]: Started ssh.service - OpenBSD Secure Shell server.
```

Docker-server Integration with Ansible-server:

Go to the Docker server, install Docker create a user with the same name, and add this user to the Docker group.

```
useradd ansadmin
passwd ansadmin
usermod -aG docker ansadmin
```

```
root@ip-192-0-0-12:/home/ubuntu# useradd ansadmin
root@ip-192-0-0-12:/home/ubuntu# passwd ansadmin
New password:
Retype new password:
passwd: password updated successfully
root@ip-192-0-0-12:/home/ubuntu# usermod -aG docker ansadmin
root@ip-192-0-0-12:/home/ubuntu#
```

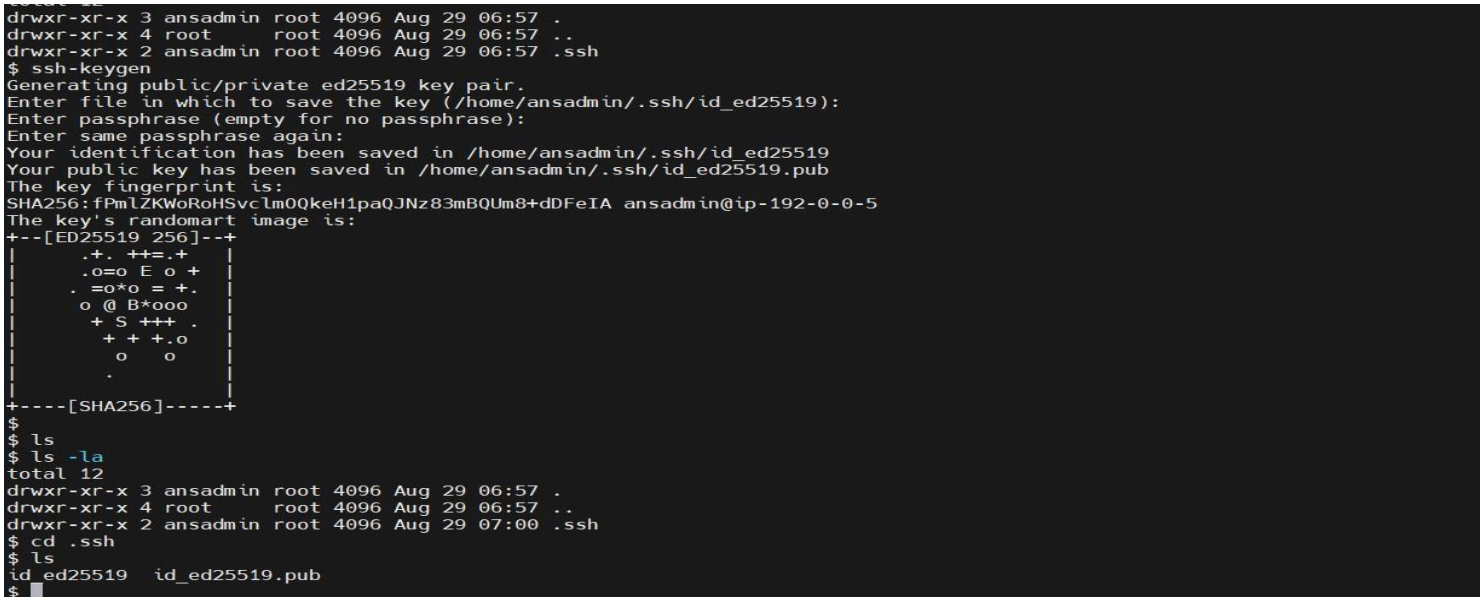
SSH Key Setup ansible-server:

In an ansible server, we have to generate a key for the ansadmin user to connect it with the docker, Kubernetes, and jenkins servers. For this switch to ansadmin user and create the following directories to save the ssh key.

```
su ansadmin
mkdir -p ~/.ssh
touch ~/.ssh/authorized_keys
chmod 700 ~/.ssh
chmod 600 ~/.ssh/authorized_keys
```

Now reload the sshd service and generate the ssh keys.

```
systemctl reload sshd
ssh-keygen
```



Copy the keys to Docker-server:

The same public key should be present in the docker server to enable a passwordless SSH connection between the ansible server and the docker server. We are using ssh-copy-id, a script that installs your public SSH key on a remote server's authorized_keys file. This allows you to log in without entering a

password. This method enhances security and improves convenience, especially for tasks requiring frequent server access.

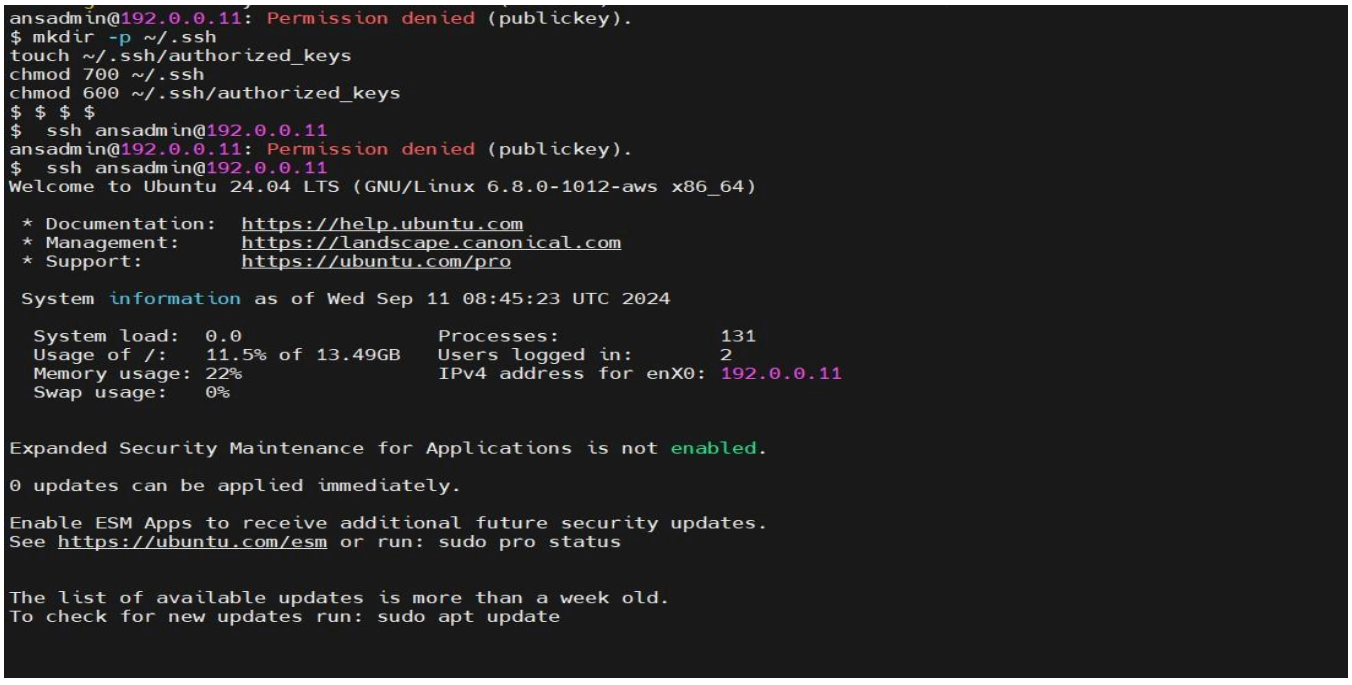
As root user

```
mkdir -p ~/.ssh
touch ~/.ssh/authorized_keys
chmod 700 ~/.ssh
chmod 600 ~/.ssh/authorized_keys
```

nano authorized_keys (Copy the public key from ansible server and paste in this file)

```
systemctl restart ssh
```

192.0.0.11 is the private IP of the docker-server, as both the ansible-server and docker-server are in the same VPC. Otherwise, we have to use public IP.



Now we are going to add the IP of docker-server to the hosts file.

Exit from the session and ansadmin user. Run the following commands as root user on ansible server.

```
$cd /etc/ansible
$vi hosts
#add the following to the file
 192.0.0.11
localhost
 $su ansadmin
$ansible all -m ping
```

```
root@ip-192-0-0-9:/etc/ansible# ls
ansible.cfg  hosts  roles
root@ip-192-0-0-9:/etc/ansible# su ansadmin
$ ansible all -m ping
The authenticity of host 'localhost (127.0.0.1)' can't be established.
ED25519 key fingerprint is SHA256:ox19/+xkOcUoATdMZaNqJZOsn8aD+05PyvoKZARDynM.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? 192.0.0.11 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
```

The Docker server is successfully integrated with Ansible server.

To create a directory as a workspace in the Ansible server- so that jenkins can save its artifacts.

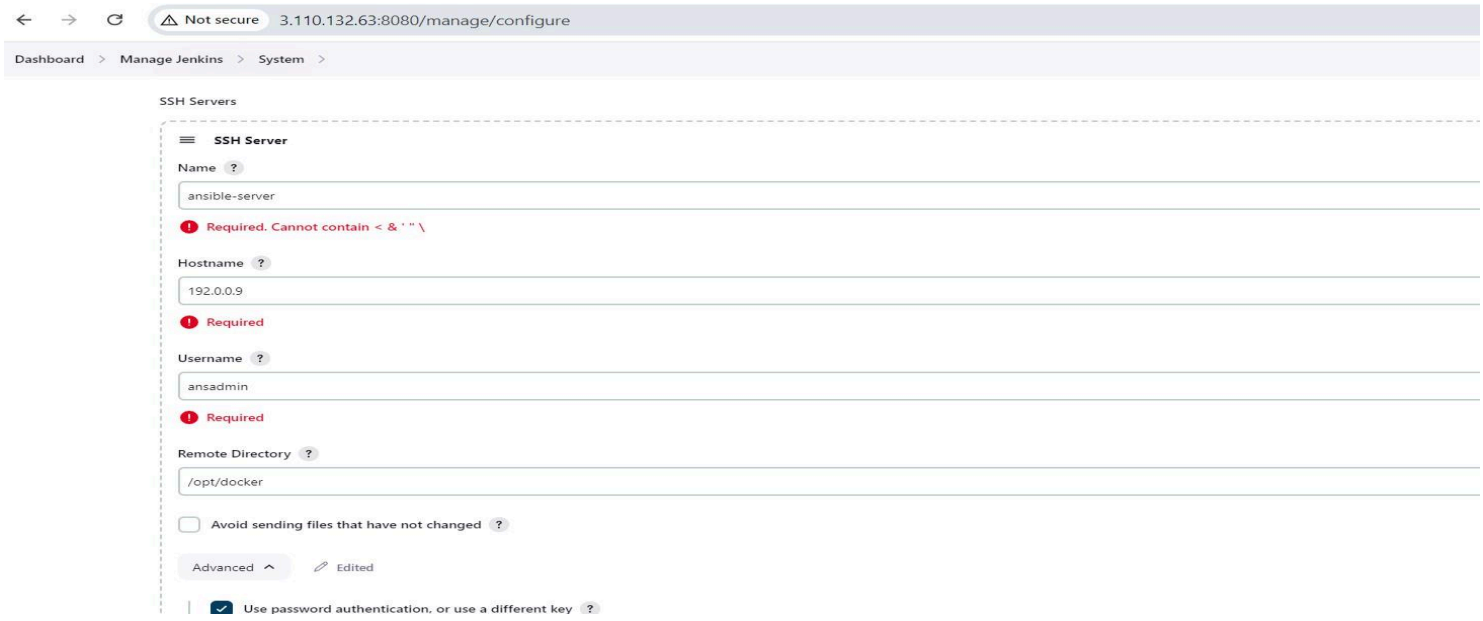
```
$cd /opt
$sudo mkdir docker

#now give full access to an admin over this directory
$sudo chown -R ansadmin:ansadmin /opt/docker
$ls -la
```

```
root@ip-192-0-0-9:/opt# ls
containerd
root@ip-192-0-0-9:/opt# sudo mkdir docker
root@ip-192-0-0-9:/opt# sudo chown -R ansadmin:ansadmin /opt/docker
root@ip-192-0-0-9:/opt# ls -la
total 16
drwxr-xr-x  4 root     root     4096 Sep 11 10:39 .
drwxr-xr-x 22 root     root     4096 Sep 11 06:35 ..
drwx--x--x  4 root     root     4096 Aug 30 07:10 containerd
drwxr-xr-x  2 ansadmin ansadmin 4096 Sep 11 10:39 docker
root@ip-192-0-0-9:/opt#
```

Ansible integration with jenkins

Integrate jenkins server with Ansible server over SSH.

install plugin
publish over ssh
Menage jenkins--> configuration--> Publish over ssh (at end)
Name: ansible-server
Hostname: private ip (in case both machines are in same vpc)
username: ansadmin
advance
 use password authentication
 add password
 test connection apply-->save



The Ansible server is successfully integrated with jenkins.

Ansible playbook to create an image and create a continuer:

Docker-server

First, I created a Dockerfile in Docker-server which will run the techapp.war file on Tomcat server.

Note: I already copied the war file from the target directory to the current directory /opt/docke where i am creating the Dockerfile.

```
vi Dockerfile

# Use the official Tomcat image from Docker Hub
FROM tomcat:latest

# Copy the WAR file into the Tomcat webapps directory
COPY ./techapp.war /usr/local/tomcat/webapps/

# Expose the default port for Tomcat
EXPOSE 8080

# Set the CMD to run Tomcat
CMD ["catalina.sh", "run"]
```

Ansible-Server

Write a host file containing the IPs of the target host where we want to implement the Ansible playbook. For this, i created a new file in /opt/docker/hosts and private IP of Docker-server 192.0.0.11.
Now we have to change the default location of the host file. We have to edit the file /etc/ansible/ansible.cfg.

```
# Since Ansible 2.12 (core):
# To generate an example config file (a "disabled" one with all default settings, commented out):
#               $ ansible-config init --disabled > ansible.cfg
#
# Also you can now have a more complete file by including existing plugins:
# ansible-config init --disabled -t all > ansible.cfg

# For previous versions of Ansible you can check for examples in the 'stable' branches of each version
# Note that this file was always incomplete  and lagging changes to configuration settings

# for example, for 2.9: https://github.com/ansible/ansible/blob/stable-2.9/examples/ansible.cfg
#
[defaults]
inventory = /opt/docker/hosts
host_key_checking = False
~
~
~
```

Now create the ansible-playbook in ansible-server to create and run the docker container on docker-server.

$vi ansible-dockerimage.yml

```
root@ip-192-0-0-9:/opt/docker# ls
ansible-dockerimage.yml  hosts
root@ip-192-0-0-9:/opt/docker# cat hosts
192.0.0.11

root@ip-192-0-0-9:/opt/docker# cat ansible-dockerimage.yml
---
- name: Create Docker image and container
  hosts: all
  become: true

  tasks:
    - name: Remove unwanted Docker images
      docker_image:
        name: abc-techapp-image
        state: absent
      ignore_errors: yes

    - name: Build Docker image
      docker_image:
        name: abc-techapp-image
        tag: lts
        source: build
        build:
          path: /opt/docker/  # Directory containing your Dockerfile
        state: present

    - name: Create and Run Docker container
      docker_container:
        name: techapp_container
        image: abc-techapp-image:lts  # Specify the tag for the image
        state: started
        published_ports:
          - "80:80"
root@ip-192-0-0-9:/opt/docker#
```

```
---
- name: Create Docker image and container
  hosts: all
  become: true

  tasks:
    - name: Remove unwanted Docker images
      docker_image:
        name: abc-techapp-image
        state: absent
      ignore_errors: yes

    - name: Build Docker image
      docker_image:
        name: abc-techapp-image
        tag: lts
        source: build
        build:
          path: /opt/docker/  # Directory containing your Dockerfile
        state: present

    - name: Create and Run Docker container
      docker_container:
        name: techapp_container
        image: abc-techapp-image:lts  # Specify the tag for the image
        state: started
        published_ports:
          - "80:80"


  —----------------------------
```

To run the ansible playbook, switch to ansadmin in ansible-server and run the following command

```
$su ansadmin
 $ansible-playbook -i hosts ansible-dockerimage.yml -K
```

```
root@ip-192-0-0-9:/opt/docker# vi ansible-dockerimage.yml
root@ip-192-0-0-9:/opt/docker# su ansadmin
$ ansible-playbook -i hosts ansible-dockerimage.yml -K
aBECOME password:

PLAY [Create Docker image and container] ***************************************************************************

TASK [Gathering Facts] *********************************************************************************************
ok: [192.0.0.11]

TASK [Remove unwanted Docker images] *******************************************************************************
ok: [192.0.0.11]

TASK [Build Docker image] ******************************************************************************************
ok: [192.0.0.11]

TASK [Create and Run Docker container] *****************************************************************************
changed: [192.0.0.11]

PLAY RECAP *********************************************************************************************************
192.0.0.11                 : ok=4    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

$
```

Docker-Server

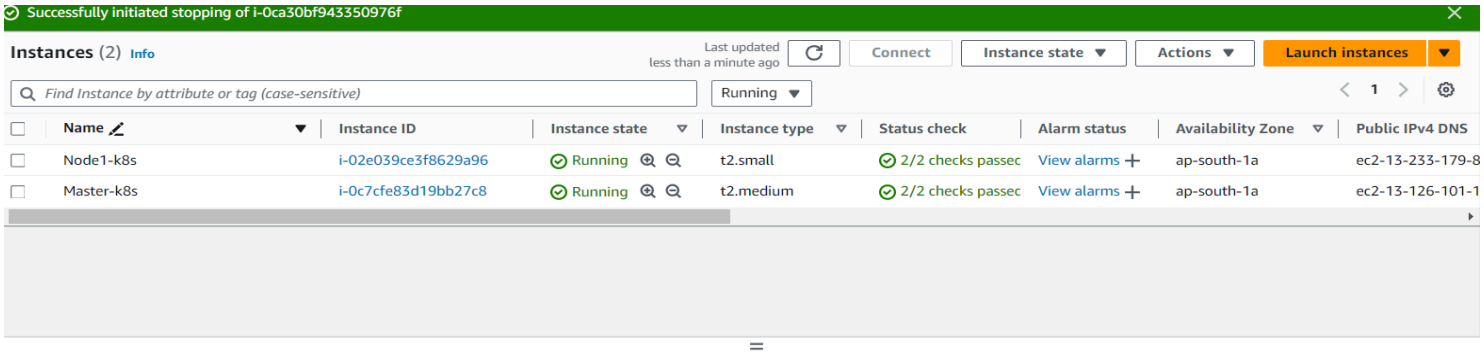
$docker ps

$docker images

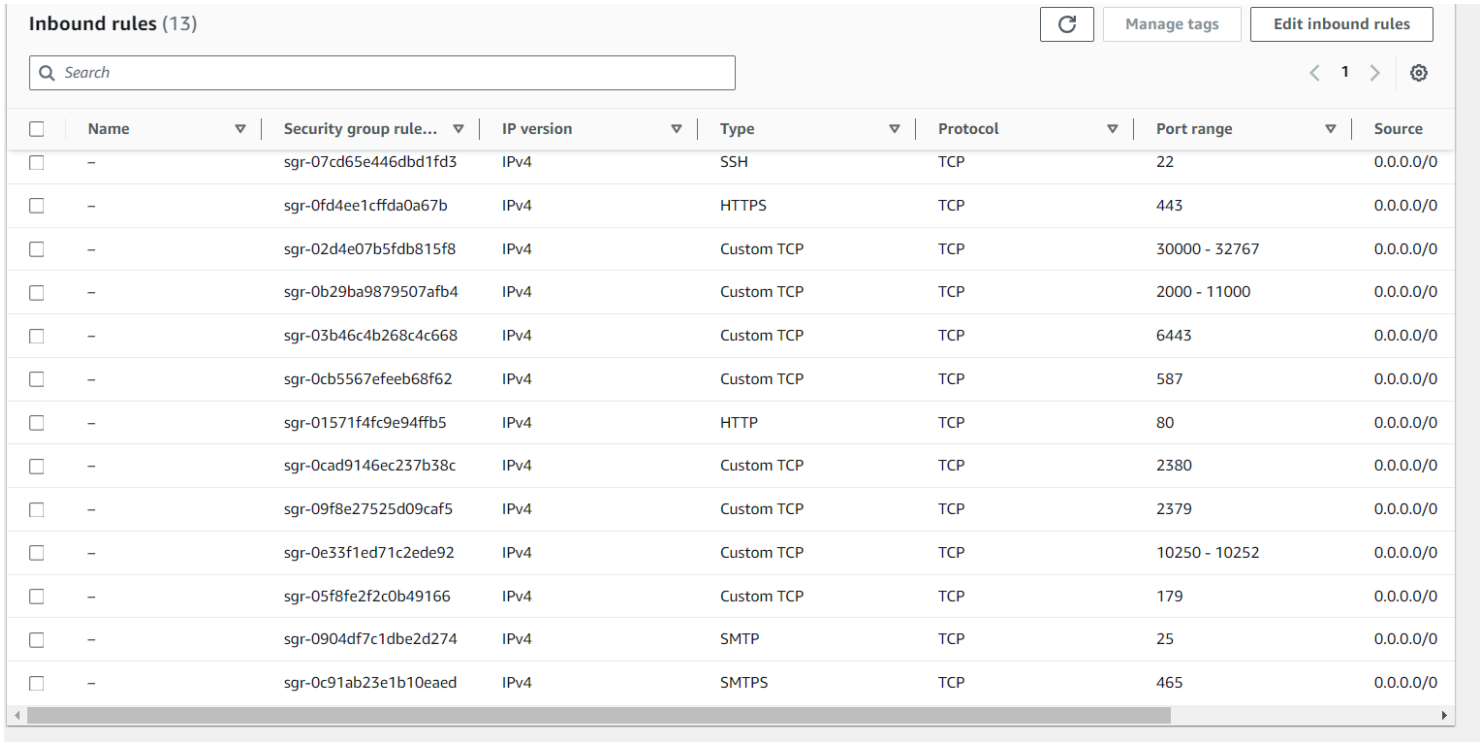
```
root@ip-192-0-0-11:/opt/docker# docker ps
CONTAINER ID   IMAGE                    COMMAND             CREATED          STATUS          PORTS                           NAMES
f956e01b55a9   abc-techapp-image:lts    "catalina.sh run"   12 seconds ago   Up 11 seconds   0.0.0.0:80->80/tcp, 8080/tcp    techapp_container
root@ip-192-0-0-11:/opt/docker# docker images
REPOSITORY          TAG       IMAGE ID       CREATED          SIZE
abc-techapp-image   lts       0bc79d6c27b6   41 minutes ago   519MB
tomcat              latest    a8515e7a8d2e   2 days ago       512MB
root@ip-192-0-0-11:/opt/docker# cat Dockerfile
FROM tomcat:latest
COPY ./techapp.war /usr/local/tomcat/webapps
root@ip-192-0-0-11:/opt/docker# ls
Dockerfile  techapp.war
root@ip-192-0-0-11:/opt/docker#
```

Kubernetes Setup

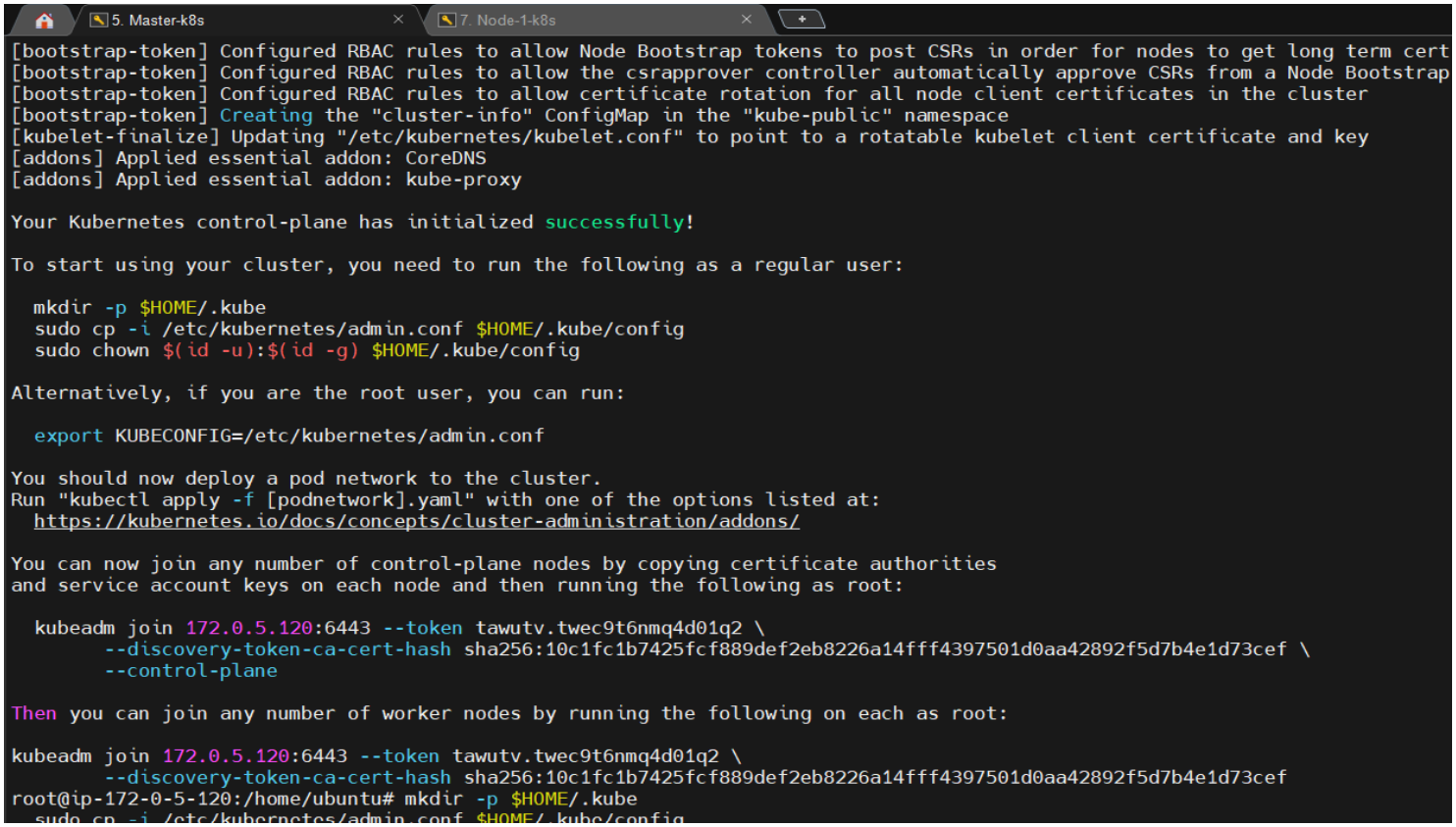
Create 2 new EC2 instances for Kubernetes master and slave node.

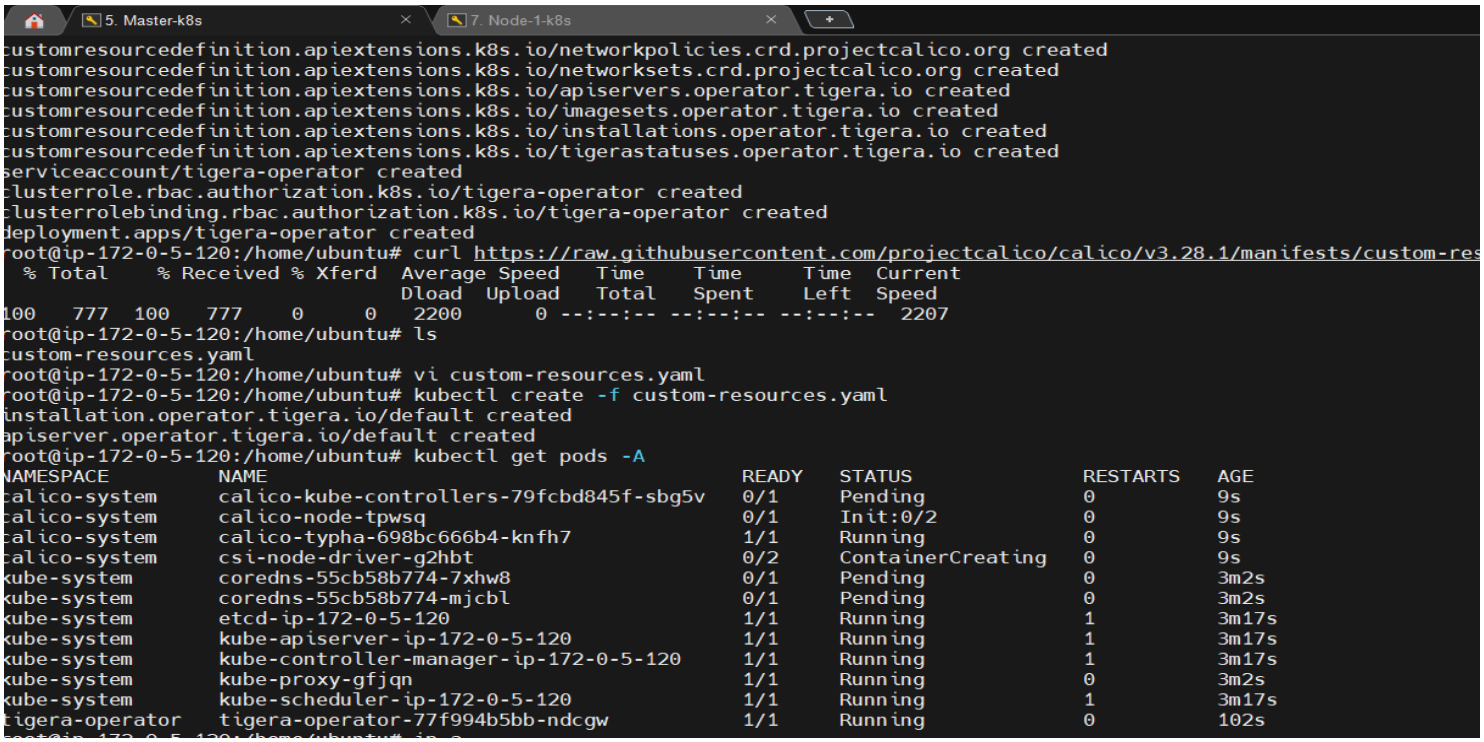


AMI - Ubuntu Server 22.04
Instance type- t2.medium (master), t2.small (Node1)
Security group Inbound rules are following

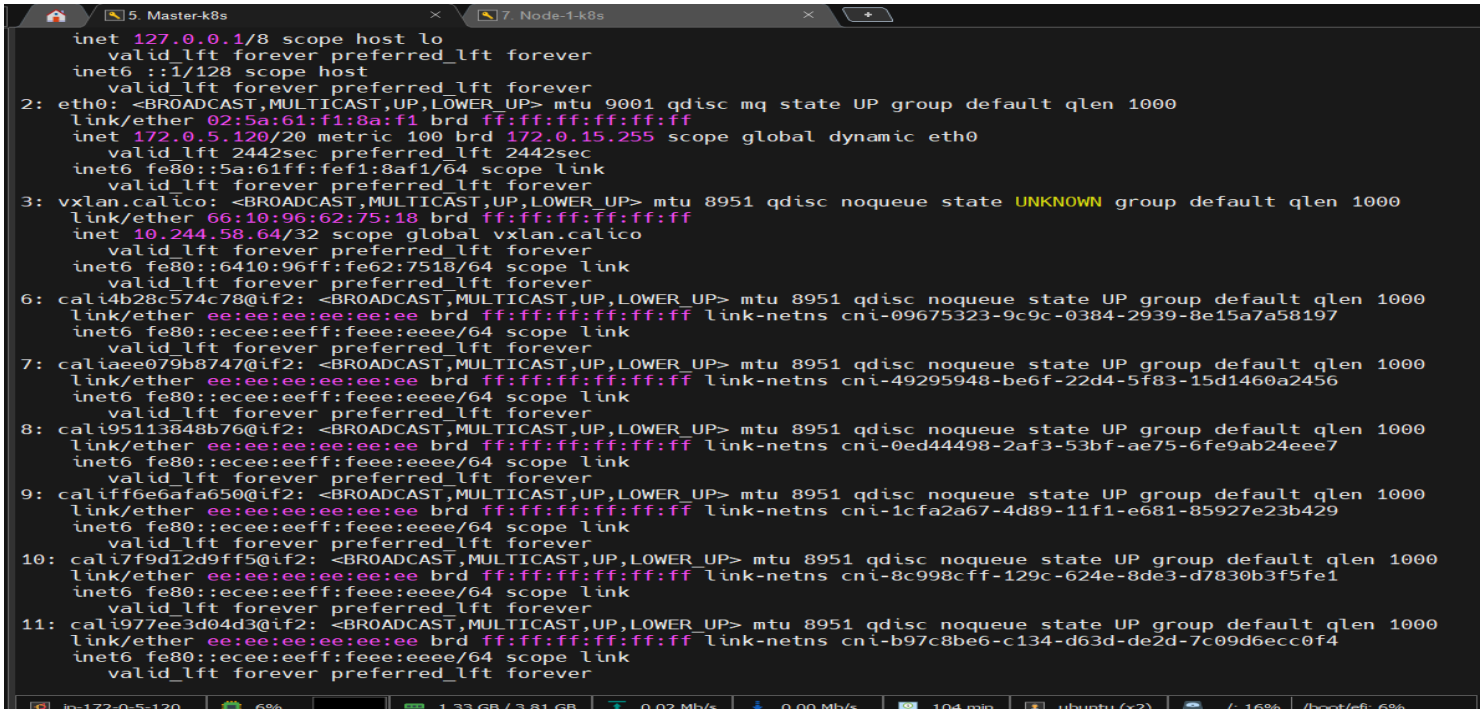


Now install the Kubernetes cluster by following this instruction
https://kanzal.com/kubernetes-with-kubeadm-and-calico/

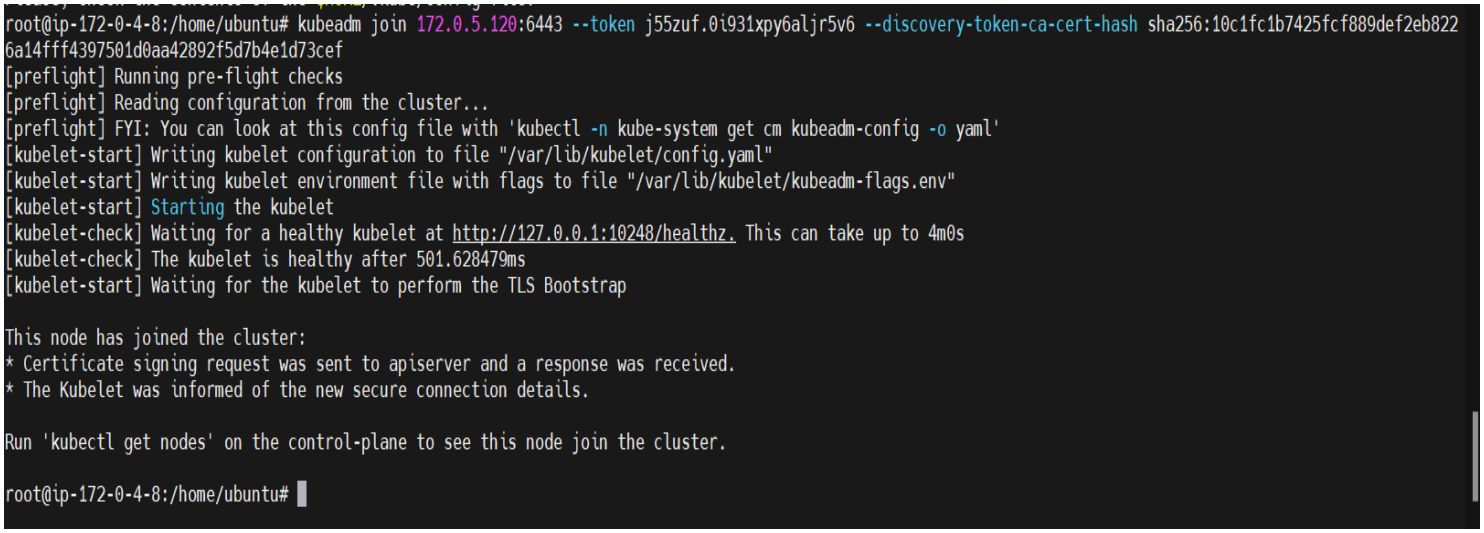




An overlay network is created for the Kubernetes cluster



The node1 has joined the cluster



See the image below showing the Connected Master and Slave

```
root@ip-172-0-5-120:~# kubectl get pods -A
NAMESPACE          NAME                                       READY   STATUS    RESTARTS   AGE
calico-apiserver   calico-apiserver-69d54bb465-fjfmh          1/1     Running   0          96m
calico-apiserver   calico-apiserver-69d54bb465-sthkh          1/1     Running   0          96m
calico-system      calico-kube-controllers-79fcbd845f-sbg5v   1/1     Running   0          97m
calico-system      calico-node-tmv2m                          0/1     Running   0          70m
calico-system      calico-node-tpwsq                          0/1     Running   0          97m
calico-system      calico-typha-698bc666b4-knfh7              1/1     Running   0          97m
calico-system      csi-node-driver-g2hbt                      2/2     Running   0          97m
calico-system      csi-node-driver-wkphj                      2/2     Running   0          70m
kube-system        coredns-55cb58b774-7xhw8                   1/1     Running   0          100m
kube-system        coredns-55cb58b774-mjcbl                   1/1     Running   0          100m
kube-system        etcd-ip-172-0-5-120                        1/1     Running   1          100m
kube-system        kube-apiserver-ip-172-0-5-120              1/1     Running   1          100m
kube-system        kube-controller-manager-ip-172-0-5-120     1/1     Running   1          100m
kube-system        kube-proxy-7wthc                           1/1     Running   0          70m
kube-system        kube-proxy-gfjqn                           1/1     Running   0          100m
kube-system        kube-scheduler-ip-172-0-5-120              1/1     Running   1          100m
tigera-operator    tigera-operator-77f994b5bb-ndcgw           1/1     Running   0          99m
root@ip-172-0-5-120:~# kubectl get nodes -A
NAME            STATUS   ROLES           AGE    VERSION
ip-172-0-4-8    Ready    <none>          70m    v1.30.5
ip-172-0-5-120  Ready    control-plane   101m   v1.30.5
root@ip-172-0-5-120:~#
```

Once Kubernetes was installed and the Master and Node Instances were connected, I Created a Manifests file for Service and Deployment to test whether everything worked properly.

2. Write pod, service, and deployment manifest file

1. Deploy Artifacts on Kubernetes

I create deployment and service manifest files to deploy the artifact on Kubernetes.
Service Manifest File is as shown below:

```
root@ip-172-0-5-120:/home/ubuntu# cat techappService.yml
apiVersion: v1
kind: Service
metadata:
  name: techappservice
spec:
  type: NodePort
  selector:
    tyep: webserver
  ports:
    - port: 80
      targetPort: 8080
      nodePort: 30018
root@ip-172-0-5-120:/home/ubuntu#
```

Run the following command to apply the service.yml file

$kubectl apply -f techappService.yml

```
      nodePort: 30018
root@ip-172-0-5-120:/home/ubuntu# kubectl create -f techappService.yml
service/techappservice created
root@ip-172-0-5-120:/home/ubuntu# kubectl get all
NAME                    TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
service/kubernetes      ClusterIP   10.96.0.1       <none>        443/TCP        5h4m
service/techappservice  NodePort    10.110.51.105   <none>        80:30018/TCP   18s
root@ip-172-0-5-120:/home/ubuntu#
```

I created a new docker image and pushed it to my docker hub account.
https://hub.docker.com/repository/docker/sonia1690/abc-techapp/general

See the deployment manifest **techappDeployment.yml** below:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: techappdeployment
spec:
  replicas: 3  # Desired number of pod replicas
  selector:
    matchLabels:
      type: webserver
  template:
    metadata:
      labels:
        type: webserver
    spec:
      containers:
      - name:  container1
        image: sonia1690/abc-techapp:lts  # Define the Docker image to use
```

—----------

I have added 3 replicas to the deployment for high availability.

```
root@ip-172-0-5-120:/home/ubuntu# cat techappDeployment.yml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: techappdeployment
spec:
  replicas: 3  # Desired number of pod replicas
  selector:
    matchLabels:
      type: webserver
  template:
    metadata:
      labels:
        type: webserver
    spec:
      containers:
      - name:  container1
        image: sonia1690/abc-techapp:lts  # Define the Docker image to use
root@ip-172-0-5-120:/home/ubuntu#
```

Run the following command to apply this file:

$kubectl apply -f techappDeployment.yml

$kubectl expose deployment techappdeployment --port 80 --target-port 8080 --type NodePort (Instead of using techappService file, I applied the service manually by the above command)

Use the following commands to check the newly added resources.

$kubectl get nodes
$kubectl get svc
$kubectl get ep

```
root@k8s-server:/home/ubuntu# kubectl apply -f techappDeployment.yml
deployment.apps/techappdeployment created
root@k8s-server:/home/ubuntu# kubectl expose deployment techappdeployment --port 80 --target-port 8080 --type NodePort
service/techappdeployment exposed
root@k8s-server:/home/ubuntu# kubectl get nodes
NAME           STATUS   ROLES           AGE   VERSION
ip-172-0-4-8   Ready    <none>          16h   v1.30.5
k8s-server     Ready    control-plane   16h   v1.30.5
root@k8s-server:/home/ubuntu# kubectl get pods
NAME                                  READY   STATUS    RESTARTS   AGE
techappdeployment-549f54cf78-9w6dz    1/1     Running   0          46s
techappdeployment-549f54cf78-httv6    1/1     Running   0          46s
techappdeployment-549f54cf78-pzwh5    1/1     Running   0          46s
root@k8s-server:/home/ubuntu# kubectl get ep
NAME               ENDPOINTS                                                  AGE
kubernetes         172.0.5.120:6443                                           16h
techappdeployment  10.10.185.153:8080,10.10.213.78:8080,10.10.213.79:8080   31s
root@k8s-server:/home/ubuntu# kubectl get svc
NAME               TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
kubernetes         ClusterIP   10.96.0.1        <none>        443/TCP        16h
techappdeployment  NodePort    10.109.180.246   <none>        80:32564/TCP   57s
root@k8s-server:/home/ubuntu#
```

After deploying the pods I entered into the pod and copied the files of **webapps.dist** directory to **webapps.** This is because the latest version of Tomcat checks the webapps directory to execute the application but by default, all the Tomcat files are available in webapps.dist. That is why we have to copy them to webapps directory.

$kubectl exec -it techappdeployment-549f54cf78-md58l -- /bin/bash

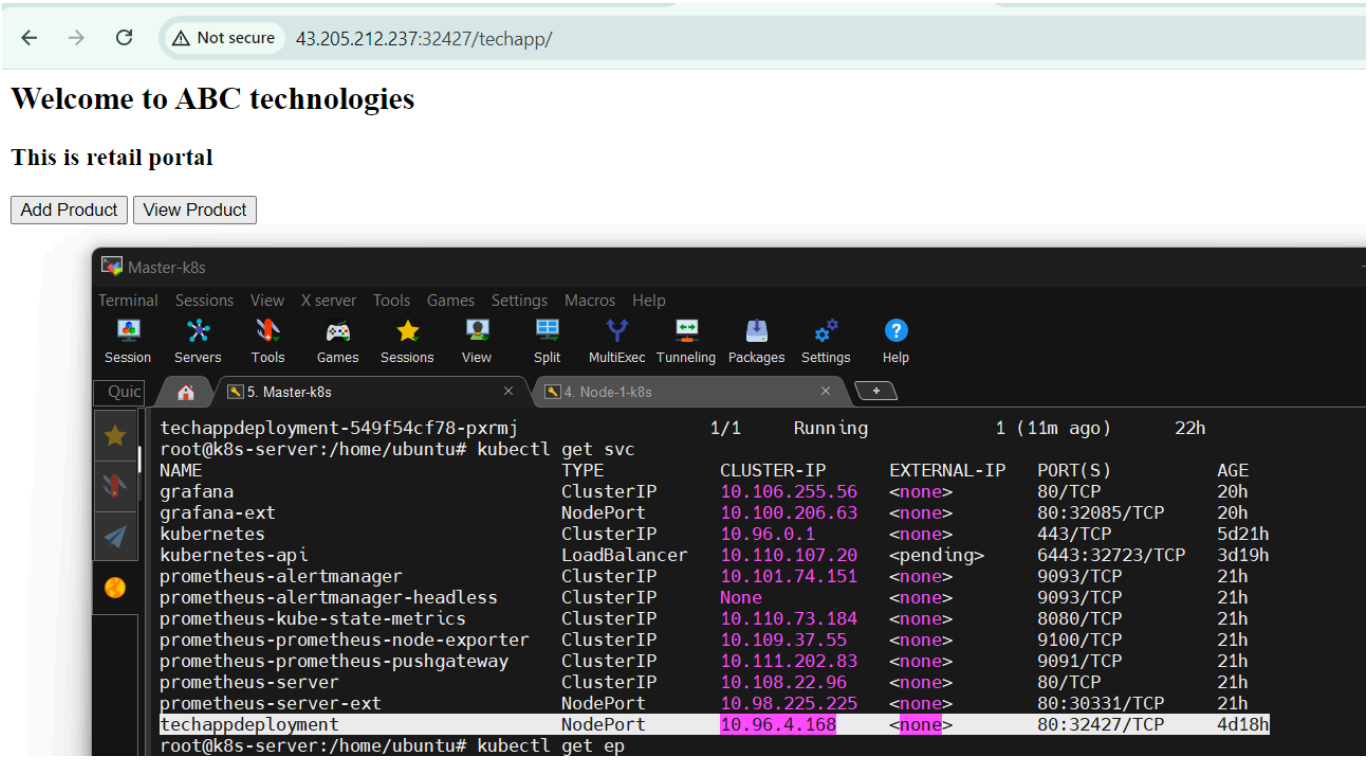
In webapps.dist directory I used the following command to copy all files

$cp -R * ../webapps

```
root@techappdeployment-549f54cf78-md58l:/usr/local/tomcat/webapps.dist# cp -R * ../webapps
root@techappdeployment-549f54cf78-md58l:/usr/local/tomcat/webapps.dist# cd ..
root@techappdeployment-549f54cf78-md58l:/usr/local/tomcat# ls
bin  BUILDING.txt  conf  CONTRIBUTING.md  lib  LICENSE  logs  native-jni-lib  NOTICE  README.md  RELEASE-NOTES  RUNNING.txt  temp  webapps  webapps.dist  work
root@techappdeployment-549f54cf78-md58l:/usr/local/tomcat# cd webapps
root@techappdeployment-549f54cf78-md58l:/usr/local/tomcat/webapps# ls
docs  examples  host-manager  manager  ROOT  techapp  techapp.war
root@techappdeployment-549f54cf78-md58l:/usr/local/tomcat/webapps# exit
exit
root@ip-172-0-5-120:/home/ubuntu# ls
```

```
root@k8s-server:/home/ubuntu# kubectl get pods -A
NAMESPACE          NAME                                       READY   STATUS    RESTARTS        AGE
calico-apiserver   calico-apiserver-55cf75748f-tjvrr          1/1     Running   2 (23m ago)     16h
calico-apiserver   calico-apiserver-55cf75748f-zngbc          1/1     Running   2 (23m ago)     16h
calico-system      calico-kube-controllers-84696d5dd-jwk6k    1/1     Running   2 (23m ago)     16h
calico-system      calico-node-gwfv6                          1/1     Running   2 (23m ago)     16h
calico-system      calico-node-wvqf4                          1/1     Running   2 (23m ago)     16h
calico-system      calico-typha-6675cf86d7-rkqqx              1/1     Running   2 (23m ago)     16h
calico-system      csi-node-driver-5wpgs                      2/2     Running   4 (23m ago)     16h
calico-system      csi-node-driver-wrx6d                      2/2     Running   4 (23m ago)     16h
default            techappdeployment-549f54cf78-9w6dz         1/1     Running   0               7m36s
default            techappdeployment-549f54cf78-httv6         1/1     Running   0               7m36s
default            techappdeployment-549f54cf78-pzwh5         1/1     Running   0               7m36s
kube-system        coredns-55cb58b774-jhrhk                   1/1     Running   2 (23m ago)     16h
kube-system        coredns-55cb58b774-vk745                   1/1     Running   2 (23m ago)     16h
kube-system        ebs-csi-controller-57fbb9d8cf-c7h8p        5/5     Running   9 (21m ago)     43m
kube-system        ebs-csi-controller-57fbb9d8cf-zlb66        5/5     Running   10 (21m ago)    43m
kube-system        ebs-csi-node-4sj47                         3/3     Running   5 (21m ago)     43m
kube-system        ebs-csi-node-vxgfv                         3/3     Running   6 (21m ago)     43m
kube-system        etcd-k8s-server                            1/1     Running   2 (23m ago)     16h
kube-system        kube-apiserver-k8s-server                  1/1     Running   1 (23m ago)     54m
kube-system        kube-controller-manager-k8s-server         1/1     Running   3 (23m ago)     16h
kube-system        kube-proxy-48mmv                           1/1     Running   2 (23m ago)     16h
kube-system        kube-proxy-k772d                           1/1     Running   2 (23m ago)     16h
kube-system        kube-scheduler-k8s-server                  1/1     Running   7 (23m ago)     16h
tigera-operator    tigera-operator-77b6b446bb-fj9jk           1/1     Running   5 (22m ago)     16h
root@k8s-server:/home/ubuntu#
```

At this point, I have successfully created Pods, Deployment, and Service via Kubernetes.



The techappdeployment service is exposed on Nodeport 32427. To access this on the browser http://worker-node-ip:32427/techapp —----------

3. Integrate Kubernetes with Ansible

We will use the same procedure while integrating the Docker host with Ansible or jenkins in previous tasks. The only difference is that now we want the ansible server to access the root user of the Kubernetes server because only root user can run Kubernetes commands fully. For this, we will create a pair in the root directory of the ansible server and add the public key in the authorized file present in /root/.ssh/authorized_keys of the Kubernetes server.

In the root directory of the ansible server, go to the .ssh directory and create a keypair using ssh-keygen

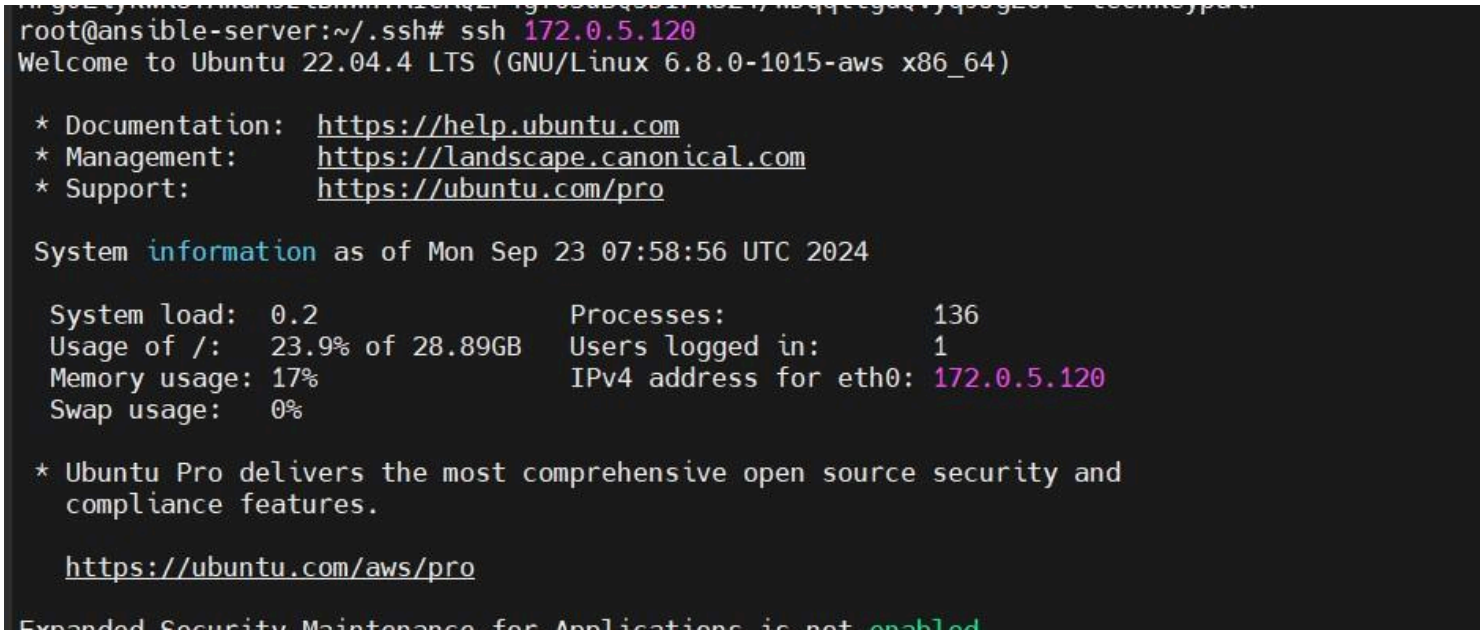
$cd /root

$cd .ssh

$ssh-keygen

```
root@ansible-server:/# cd
root@ansible-server:~# ls
ansible-kubernetes  snap
root@ansible-server:~# ls -la
total 56
drwx------  6 root root  4096 Sep 21 14:54 .
drwxr-xr-x 22 root root  4096 Sep 23 07:49 ..
drwxr-xr-x  4 root root  4096 Sep 21 11:56 .ansible
-rw-------  1 root root  4432 Sep 21 15:46 .bash_history
-rw-r--r--  1 root root  3106 Apr 22 13:04 .bashrc
-rw-r--r--  1 root root   161 Apr 22 13:04 .profile
drwx------  2 root root  4096 Sep 21 10:59 .ssh
-rw-------  1 root root 16116 Sep 21 14:54 .viminfo
drwxr-xr-x  4 root root  4096 Sep 21 11:56 ansible-kubernetes
drwx------  3 root root  4096 Sep 21 10:55 snap
root@ansible-server:~# cd .ssh
root@ansible-server:~/.ssh# ls
authorized_keys  id_ed25519  id_ed25519.pub  known_hosts  known_hosts.old
root@ansible-server:~/.ssh# cat id_ed25519.pub
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIOTVV1eGHC1UHFCO7SPHFyPABZs6dUQjnCbRhwirOeP2 root@ip-172-0-9-208
root@ansible-server:~/.ssh# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
```

```
root@k8s-server:~/.ssh# cd
root@k8s-server:~# ls -la
total 104
drwx------ 11 root root  4096 Sep 21 11:42 .
drwxr-xr-x 19 root root  4096 Sep 23 07:49 ..
drwxr-xr-x  4 root root  4096 Sep 21 11:23 .ansible
-rw-------  1 root root 23185 Sep 21 15:46 .bash_history
-rw-r--r--  1 root root  3106 Oct 15  2021 .bashrc
drwx------  3 root root  4096 Sep 21 11:57 .cache
drwx------  3 root root  4096 Sep 19 10:02 .config
drwxr-xr-x  3 root root  4096 Sep 20 11:54 .kube
drwx------  3 root root  4096 Sep 18 12:15 .launchpadlib
-rw-------  1 root root    20 Sep 19 10:13 .lesshst
drwxr-xr-x  3 root root  4096 Sep 18 11:53 .m2
-rw-r--r--  1 root root   161 Jul  9  2019 .profile
drwx------  2 root root  4096 Sep 21 11:35 .ssh
-rw-r--r--  1 root root     0 Sep 18 09:52 .sudo_as_admin_successful
-rw-------  1 root root 15402 Sep 21 11:35 .viminfo
-rw-r--r--  1 root root   164 Sep 18 11:49 .wget-hsts
drwxr-xr-x  2 root root  4096 Sep 21 11:42 ansible-kubernetes
drwx------  4 root root  4096 Sep 18 09:51 snap
root@k8s-server:~# cd .ssh
root@k8s-server:~/.ssh# ls
authorized_keys  id_rsa  id_rsa.pub  known_hosts  known_hosts.old
root@k8s-server:~/.ssh# cat authorized_keys
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIOTVV1eGHC1UHFCO7SPHFyPABZs6dUQjnCbRhwirOeP2 root@ip-172-0-9-208

ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQDe2BvGl3ZOKXFbqcw/6vkY45mRSDwKnO9hrkIgwtKxTsD4quHbRLr0vASDX0xDr7E
lHZYd6jjiLZqYGuzqTkfOGEkmWUiTI8vplkU5BFxyywKn4p4Eco3LCq+9SlRQCAoe50WxIaW/8V3o1Hpl0G1D+pMXTvxIxGfhWn5k/j
zjKRckTLzkGSQ+2gG07w5S4jIYg47XtOVxHOqmaPiqPvIG/WotJxWlq8vw543ISsIWT8oEbbGFNvqMhX90FNRNPKwYo9r/JLRGl/XcA
swkobQ0L1JQMjeYmdfWhlYLUCb8iVccdxPIwLhDlR3Lkx4Zn0tIQdGoWAK1zDnV1mOlQf9aLxc= root@ip-172-0-5-120
root@k8s-server:~/.ssh#
```



Setting Up the Inventory File

Ansible takes the default configuration from **/etc/ansible/ansible.cfg** . However we do not want to mess up with the default configurations if things go wrong. So we create a new configuration with the following configs. Also, ansible takes the host inventory from **/etc/ansible/hosts** where the hosts are defined which we will configure to override the values as follows:

We are creating new ansible.cfg and host file in /root/ansible-kubernetes directory

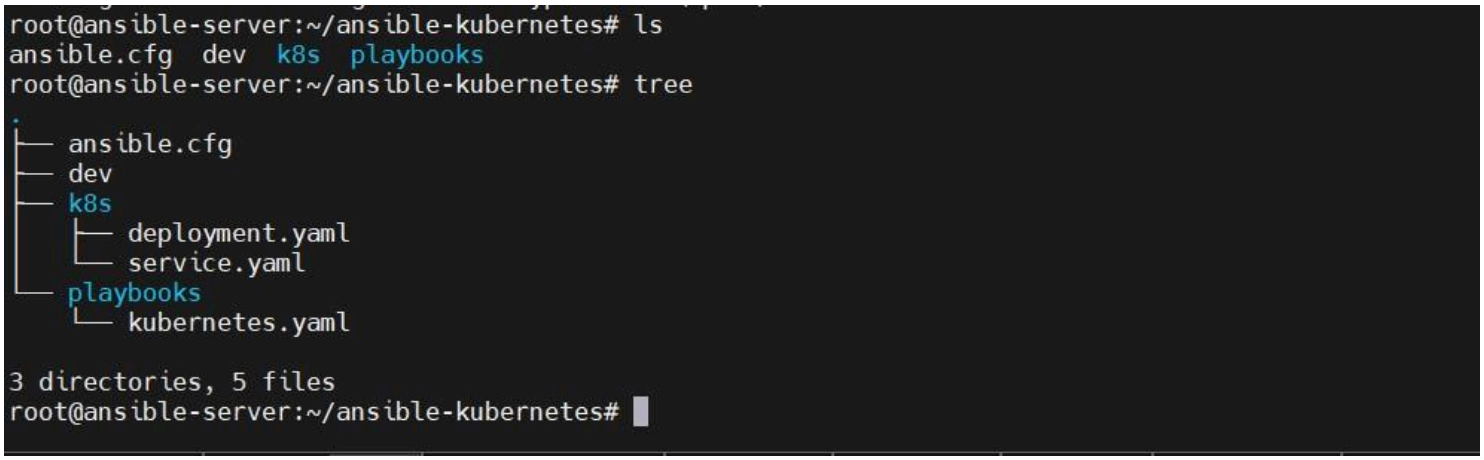
```
cd /root
mkdir ansible-kubernetes
touch ansible.cfg
touch dev      # dev file will be used as the default host file
```

```
root@k8s-server:~/ansible-kubernetes# ls
ansible.cfg  dev
root@k8s-server:~/ansible-kubernetes# cat ansible.cfg
[defaults]
inventory = ./dev
root@k8s-server:~/ansible-kubernetes# cat dev
[masters]
k8s-master ansible_host=172.0.5.120 ansible_user=root

root@k8s-server:~/ansible-kubernetes#
```

In the dev file, we added the k8s-master private IP, user name, and name of the host.

4. Ansible playbook to create deployment and service

I created a new directory ansible-Kubernetes and a few other directories inside ansible-Kubernetes.




```
$mkdir ansible-kubernetes
$cd ansible-kubernetes
```

Now create another directory k8s for deployment.yaml and service.yaml files

```
$mkdir k8s
$cd k8s
$vi deployment.yaml
```

Add the following content to this file

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: retailappdeployment
spec:
 replicas: 1  # Desired number of pod replicas
 selector:
  matchLabels:
   type: webserver
 template:
  metadata:
   labels:
    type: webserver
```

spec:
containers:
- name: retailapp-cont
image: sonia1690/abc-techapp:lts # Define the Docker image to use

```
root@ansible-server:~/ansible-kubernetes# ls
ansible.cfg  dev  k8s  playbooks
root@ansible-server:~/ansible-kubernetes# cd k8s
root@ansible-server:~/ansible-kubernetes/k8s# ls
deployment.yaml  service.yaml
root@ansible-server:~/ansible-kubernetes/k8s# cat deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: retailappdeployment
spec:
  replicas: 1  # Desired number of pod replicas
  selector:
    matchLabels:
      type: webserver
  template:
    metadata:
      labels:
        type: webserver
    spec:
      containers:
      - name:  retailapp-cont
        image: sonia1690/abc-techapp:lts  # Define the Docker image to use
root@ansible-server:~/ansible-kubernetes/k8s#
```

For service.yaml file
$vi service.yaml

```
root@ansible-server:~/ansible-kubernetes/k8s# cat service.yaml
apiVersion: v1
kind: Service
metadata:
  name: retailappdeployment
spec:
  type: NodePort
  ports:
    - port: 80
      targetPort: 8080
      nodePort: 30001  # This port is valid
  selector:
    app: retailapp
root@ansible-server:~/ansible-kubernetes/k8s#
```

We will create a playbook to deploy our application on the Kubernetes cluster through our ansible server.
Now we will create another directory inside the ansible-kubernetes directory to store the playbooks.

```
$cd ..
$mkdir playbooks
$cd playbooks
$vi kubernetes.yaml

---
- name: Deploy Retail App on Kubernetes
  hosts: '{{ host }}'  # Specify the host where kubectl is configured
  tasks:
    # Create a test namespace on the cluster without any manifest files.
    - name: "Create a k8s namespace"
      k8s:
        name: retail
        api_version: v1
        kind: Namespace
        state: present

    # Copying the Service.yaml and deployment.yaml in the remote node.
    - name: "Copying deployment.yaml file"
      copy:
        src: ../k8s/deployment.yaml
        dest: /tmp/deployment.yaml

    - name: "Copying service.yaml file"
      copy:
        src: ../k8s/service.yaml
        dest: /tmp/service.yaml

    # Creating a Kubernetes deployment in test using file stored locally
    - name: "Create a deployment"
      k8s:
        state: present
        namespace: retail
        src: /tmp/deployment.yaml

    # Creating a Kubernetes service in retail using file stored on local.
```

```
- name: "Create a Service"
  k8s:
    state: present
    namespace: retail
    src: /tmp/service.yaml

# Checking if the Kubernetes service are running on the cluster.
- name: "Status of the service"
  k8s:
    api_version: v1
    kind: Service
    name: retailappdeployment
    namespace: retail
  register: web_service

# CleanUP all the applied configurations
- name: "Ansible file module to delete multiple files"
  file:
    path: "{{ item }}"
    state: absent   # to delete the files
  with_items:
    - /tmp/deployment.yaml
    - /tmp/service.yaml
```

```
root@ansible-server:~/ansible-kubernetes# ls
ansible.cfg  dev  k8s  playbooks
root@ansible-server:~/ansible-kubernetes# cd playbooks/
root@ansible-server:~/ansible-kubernetes/playbooks# ls
kubernetes.yaml
root@ansible-server:~/ansible-kubernetes/playbooks# cat kubernetes.yaml
---
- name: Deploy Retail App on Kubernetes
  hosts: '{{ host }}'  # Specify the host where kubectl is configured
  tasks:
    # Create a test namespace on the cluster without any manifest files.
    - name: "Create a k8s namespace"
      k8s:
        name: retail
        api_version: v1
        kind: Namespace
        state: present

    # Copying the Service.yaml and deployment.yaml in the remote node.
    - name: "Copying deployment.yaml file"
      copy:
        src: ../k8s/deployment.yaml
        dest: /tmp/deployment.yaml

    - name: "Copying service.yaml file"
      copy:
        src: ../k8s/service.yaml
        dest: /tmp/service.yaml

    # Creating a Kubernetes deployment in test using file stored locally
    - name: "Create a deployment"
      k8s:
        state: present
        namespace: retail
        src: /tmp/deployment.yaml

    # Creating a Kubernetes service in retail using file stored on local.
    - name: "Create a Service"
      k8s:
```

```
    # Creating a Kubernetes service in retail using file stored on local.
    - name: "Create a Service"
      k8s:
        state: present
        namespace: retail
        src: /tmp/service.yaml

    # Checking if the Kubernetes service are running on the cluster.
    - name: "Status of the service"
      k8s:
        api_version: v1
        kind: Service
        name: retailappdeployment
        namespace: retail
      register: web_service

    # CleanUP all the applied configurations
    - name: "Ansible file module to delete multiple files"
      file:
        path: "{{ item }}"
        state: absent   # to delete the files
      with_items:
        - /tmp/deployment.yaml
        - /tmp/service.yaml
```

Now go back to the ansible-kubernetes directory and run the following command to run the ansible-playbook

```
ansible-playbook playbooks/kubernetes.yaml -e host=k8s-master
```

Note: host=k8s-master
K8s-master is the same name that i give to Kubernetes master in the dev file

```
root@ansible-server:~/ansible-kubernetes# ls
ansible.cfg  dev  k8s  playbooks
root@ansible-server:~/ansible-kubernetes# cat dev
[master]
k8s-master ansible_ssh_host=172.0.5.120 ansible_ssh_user=root
root@ansible-server:~/ansible-kubernetes#
```

We successfully executed the Ansible playbook to create deployment and service.

```
root@ansible-server:~/ansible-kubernetes# ls
ansible.cfg  dev  k8s  playbooks
root@ansible-server:~/ansible-kubernetes# ansible-playbook playbooks/kubernetes.yaml -e host=k8s-master

PLAY [Deploy Retail App on Kubernetes] ***************************************************************

TASK [Gathering Facts] *******************************************************************************
ok: [k8s-master]

TASK [Create a k8s namespace] ************************************************************************
ok: [k8s-master]

TASK [Copying deployment.yaml file] ******************************************************************
changed: [k8s-master]

TASK [Copying service.yaml file] *********************************************************************
changed: [k8s-master]

TASK [Create a deployment] ***************************************************************************
ok: [k8s-master]

TASK [Create a Service] ******************************************************************************
ok: [k8s-master]

TASK [Status of the service] *************************************************************************
ok: [k8s-master]

PLAY RECAP *******************************************************************************************
k8s-master                 : ok=7    changed=2    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

root@ansible-server:~/ansible-kubernetes#
```

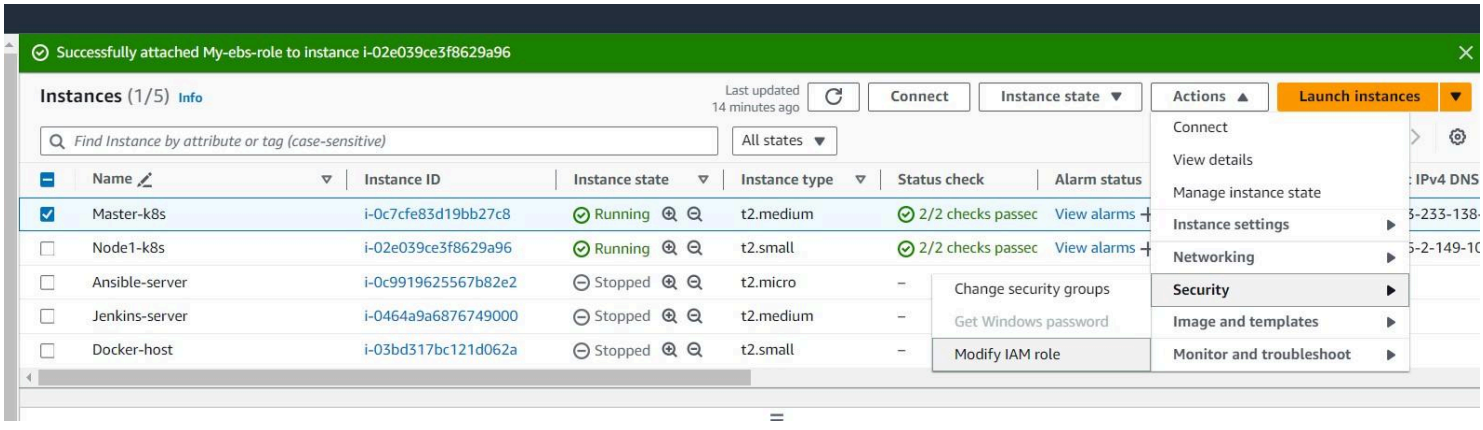
Task 5: Using Prometheus, monitor the resources like CPU utilization:

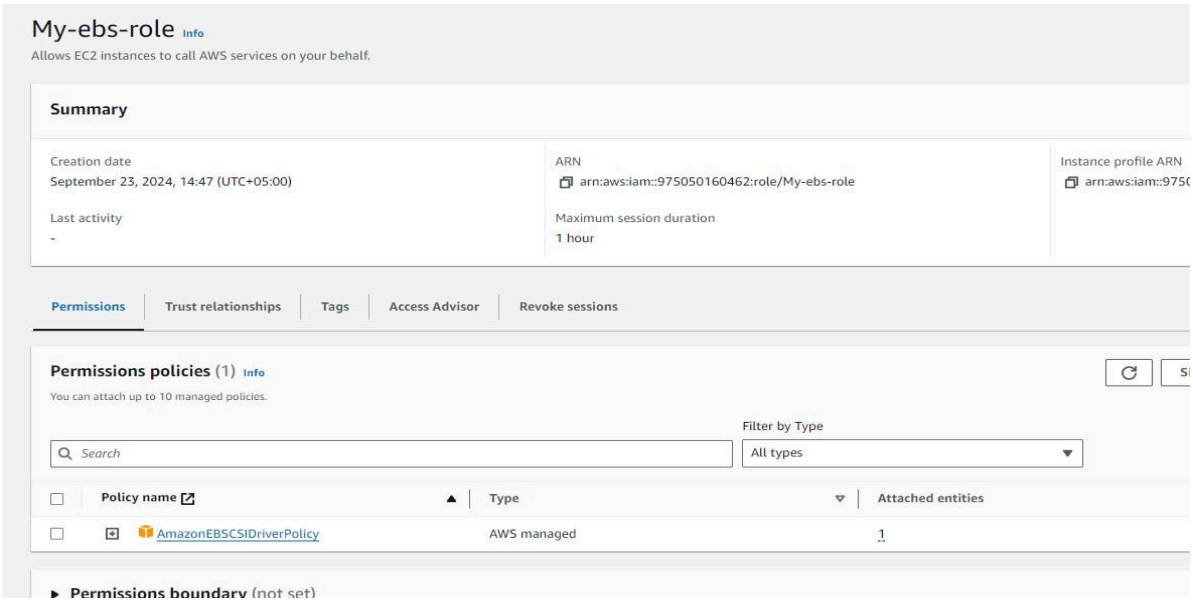
1. **Total Usage, Usage per core, usage breakdown, memory, and network on the instance by providing the endpoints on the local host.**
2. **Install the node exporter and add the URL to the target in Prometheus.**
3. **Using this data, log in to Grafana and create a dashboard to show the metrics.**

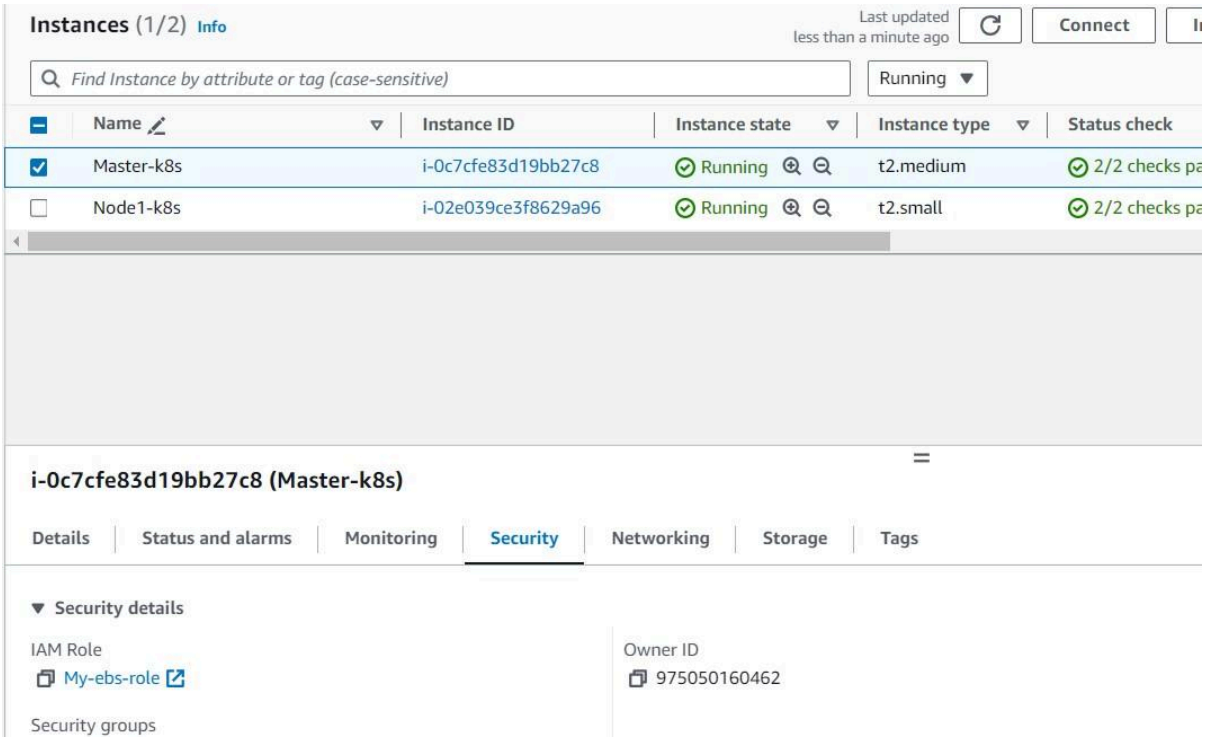
Installing Helm:
Installed the Helm repo by following this link
https://helm.sh/docs/intro/install/

Adding New Role:

Before installing Prometheus and Grafana we should have a storage class. We will follow the steps to create a role and storage file to assign a storage class to our machines.
First, we have to add a new role with the policy **`AmazonEBSCSIDriverPolicy`**, and add this role to the Kubernetes master and node machine.







Now open **/etc/kubernetes/manifests/kube-apiserver.yaml** file and add fowwoling flag

--allow-privileged=true

```
root@k8s-server:/home/ubuntu# cd /etc/kubernetes/manifests/
root@k8s-server:/etc/kubernetes/manifests# ls
etcd.yaml  kube-apiserver.yaml  kube-controller-manager.yaml  kube-scheduler.yaml
root@k8s-server:/etc/kubernetes/manifests# cat kube-apiserver.yaml
apiVersion: v1
kind: Pod
metadata:
  annotations:
    kubeadm.kubernetes.io/kube-apiserver.advertise-address.endpoint: 172.0.5.120:6443
  creationTimestamp: null
  labels:
    component: kube-apiserver
    tier: control-plane
  name: kube-apiserver
  namespace: kube-system
spec:
  containers:
  - command:
    - kube-apiserver

    - --allow-privileged=true
    - --advertise-address=172.0.5.120
    - --allow-privileged=true
    - --authorization-mode=Node,RBAC
    - --client-ca-file=/etc/kubernetes/pki/ca.crt
    - --enable-admission-plugins=NodeRestriction
```

Creat Storage Class:

Now create a new YAML file for the storage class and add the following into this file.

```
vi sc.yaml
```

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ebs-sc
provisioner: ebs.csi.aws.com
volumeBindingMode: Immediate
parameters:
  csi.storage.k8s.io/fstype: ext4
  type: gp3
  iopsPerGB: "50"
  encrypted: "false"
allowVolumeExpansion: true
reclaimPolicy: Delete
```

```
root@k8s-server:/home/ubuntu# cd
root@k8s-server:~# ls
ansible-kubernetes  aws-ebs-csi-driver  aws-ebs-csi-driver-2.35.1.tgz  get_helm.sh  pod.yaml  pvc.yaml  sc.yaml  snap
root@k8s-server:~# cat sc.yaml

---

apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ebs-sc
provisioner: ebs.csi.aws.com
volumeBindingMode: Immediate
parameters:
  csi.storage.k8s.io/fstype: ext4
  type: gp3
  iopsPerGB: "50"
  encrypted: "false"
allowVolumeExpansion: true
reclaimPolicy: Delete
root@k8s-server:~#
```

#Apply the sc.yaml file then a new storage class will be created. but it is not a default storage class.

kubectl apply -f sc.yaml

#To make it the default storage class we have to run an additional command

kubectl patch storageclass my-sc -p '{"metadata": {"annotations":{"storageclass.kubernetes.io/is-default-class":"true"}}}'

```
root@k8s-server:~# kubectl apply -f sc.yaml
storageclass.storage.k8s.io/my-sc created
root@k8s-server:~# kubectl get pod
NAME                                    READY   STATUS    RESTARTS   AGE
techappdeployment-549f54cf78-5kcnm   1/1     Running   0          68m
techappdeployment-549f54cf78-9gbs5   1/1     Running   0          69m
techappdeployment-549f54cf78-pxrmj   1/1     Running   0          68m
root@k8s-server:~# kubectl get sc
NAME    PROVISIONER       RECLAIMPOLICY   VOLUMEBINDINGMODE   ALLOWVOLUMEEXPANSION   AGE
my-sc   ebs.csi.aws.com   Delete          Immediate           true                   20s
root@k8s-server:~# kubectl patch storageclass my-sc -p '{"metadata": {"annotations":{"storageclass.kubernetes.io/is-default-class":"true"}}}'
storageclass.storage.k8s.io/my-sc patched
root@k8s-server:~# kubectl get sc
NAME             PROVISIONER       RECLAIMPOLICY   VOLUMEBINDINGMODE   ALLOWVOLUMEEXPANSION   AGE
my-sc (default)   ebs.csi.aws.com   Delete          Immediate           true                   2m14s
```

Install Prometheus:

#Update helm repo

```
helm repo update
```

#Install Prometheus

```
helm install prometheus prometheus-community/prometheus
```

```
root@k8s-server:~# helm install prometheus prometheus-community/prometheus
NAME: prometheus
LAST DEPLOYED: Mon Sep 23 10:00:51 2024
NAMESPACE: default
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
The Prometheus server can be accessed via port 80 on the following DNS name from within your cluster:
prometheus-server.default.svc.cluster.local


Get the Prometheus server URL by running these commands in the same shell:
  export POD_NAME=$(kubectl get pods --namespace default -l "app.kubernetes.io/name=prometheus,app.kubernetes.io/instance=p
etadata.name}")
  kubectl --namespace default port-forward $POD_NAME 9090


The Prometheus alertmanager can be accessed via port 9093 on the following DNS name from within your cluster:
prometheus-alertmanager.default.svc.cluster.local


Get the Alertmanager URL by running these commands in the same shell:
```

```
Get the Prometheus server URL by running these commands in the same shell:
  export POD_NAME=$(kubectl get pods --namespace default -l "app.kubernetes.io/name=prometheus,app.kubernetes.io/instance=prome
etadata.name}")
  kubectl --namespace default port-forward $POD_NAME 9090


The Prometheus alertmanager can be accessed via port 9093 on the following DNS name from within your cluster:
prometheus-alertmanager.default.svc.cluster.local


Get the Alertmanager URL by running these commands in the same shell:
  export POD_NAME=$(kubectl get pods --namespace default -l "app.kubernetes.io/name=alertmanager,app.kubernetes.io/instance=pro
.metadata.name}")
  kubectl --namespace default port-forward $POD_NAME 9093
#################################################################################
######   WARNING: Pod Security Policy has been disabled by default since    #####
######            it deprecated after k8s 1.25+. use                        #####
######            (index .Values "prometheus-node-exporter" "rbac"          #####
###### .          "pspEnabled") with (index .Values                         #####
######            "prometheus-node-exporter" "rbac" "pspAnnotations")       #####
######            in case you still need it.                                #####
#################################################################################


The Prometheus PushGateway can be accessed via port 9091 on the following DNS name from within your cluster:
prometheus-prometheus-pushgateway.default.svc.cluster.local


Get the PushGateway URL by running these commands in the same shell:
  export POD_NAME=$(kubectl get pods --namespace default -l "app=prometheus-pushgateway,component=pushgateway" -o jsonpath="{.i
  kubectl --namespace default port-forward $POD_NAME 9091
```

```
root@k8s-server:~# kubectl get pods
NAME                                            READY   STATUS              RESTARTS   AGE
prometheus-alertmanager-0                       0/1     ContainerCreating   0          22s
prometheus-kube-state-metrics-74cdb59bff-4c982  1/1     Running             0          23s
prometheus-prometheus-node-exporter-grbbs       1/1     Running             0          23s
prometheus-prometheus-node-exporter-sb629       1/1     Running             0          23s
prometheus-prometheus-pushgateway-66fc55f8d-rb2t5   1/1     Running             0          23s
prometheus-server-dd484f8d9-trzck               0/2     ContainerCreating   0          23s
techappdeployment-549f54cf78-9w6dz              1/1     Running             0          17m
techappdeployment-549f54cf78-httv6              1/1     Running             0          17m
techappdeployment-549f54cf78-pzwh5              1/1     Running             0          17m
root@k8s-server:~# kubectl get ep
NAME                                ENDPOINTS                                                  AGE
kubernetes                          172.0.5.120:6443                                           16h
prometheus-alertmanager             <none>                                                     28s
prometheus-alertmanager-headless    <none>                                                     28s
prometheus-kube-state-metrics       10.10.213.80:8080                                          28s
prometheus-prometheus-node-exporter 172.0.4.8:9100,172.0.5.120:9100                            28s
prometheus-prometheus-pushgateway   10.10.213.81:9091                                          28s
prometheus-server                   <none>                                                     28s
techappdeployment                   10.10.185.153:8080,10.10.213.78:8080,10.10.213.79:8080     17m
root@k8s-server:~#
```

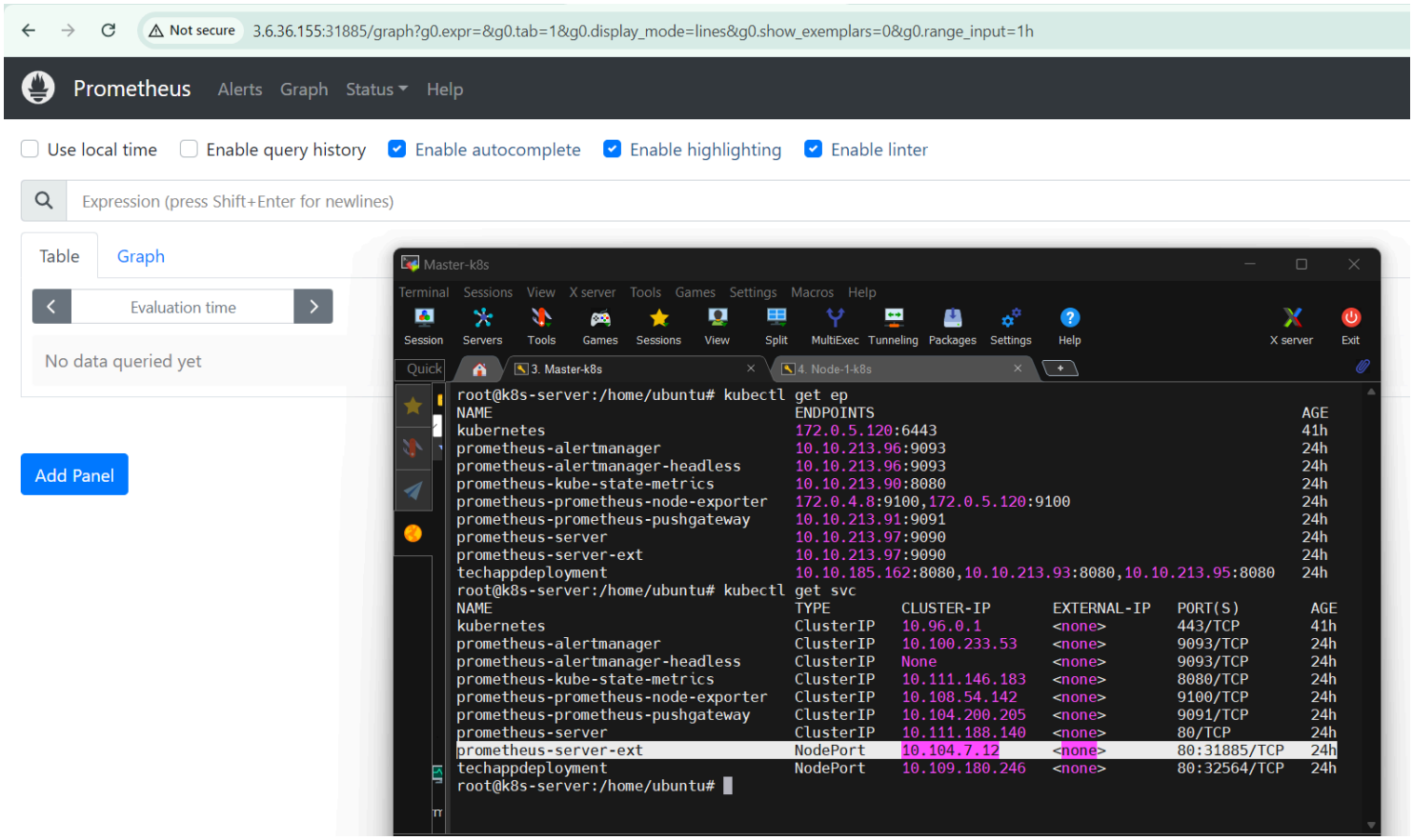
The Prometheus server is installed successfully but it is not exposed to the NodePort yet. So to expose it the the Node port we will run the following command

Expose Prometheus Service:

```
kubectl expose service prometheus-server --type=NodePort --target-port=9090 --name=prometheus-server-ext
```

```
techappdeployment                   NodePort    10.109.180.248   <none>        80:32564/TCP   24h
root@k8s-server:/home/ubuntu# kubectl get pods
NAME                                            READY   STATUS    RESTARTS      AGE
prometheus-alertmanager-0                       1/1     Running   0             24h
prometheus-kube-state-metrics-74cdb59bff-4c982  1/1     Running   3 (13m ago)   24h
prometheus-prometheus-node-exporter-grbbs       1/1     Running   1 (29m ago)   24h
prometheus-prometheus-node-exporter-sb629       1/1     Running   2 (13m ago)   24h
prometheus-prometheus-pushgateway-66fc55f8d-rb2t5   1/1     Running   2 (13m ago)   24h
prometheus-server-dd484f8d9-trzck               2/2     Running   0             24h
techappdeployment-549f54cf78-9w6dz              1/1     Running   1 (29m ago)   25h
techappdeployment-549f54cf78-httv6              1/1     Running   2 (13m ago)   25h
techappdeployment-549f54cf78-pzwh5              1/1     Running   2 (13m ago)   25h
root@k8s-server:/home/ubuntu# kubectl get ep
NAME                                ENDPOINTS                                                  AGE
kubernetes                          172.0.5.120:6443                                           41h
prometheus-alertmanager             10.10.213.96:9093                                          24h
prometheus-alertmanager-headless    10.10.213.96:9093                                          24h
prometheus-kube-state-metrics       10.10.213.90:8080                                          24h
prometheus-prometheus-node-exporter 172.0.4.8:9100,172.0.5.120:9100                            24h
prometheus-prometheus-pushgateway   10.10.213.91:9091                                          24h
prometheus-server                   10.10.213.97:9090                                          24h
prometheus-server-ext               10.10.213.97:9090                                          24h
techappdeployment                   10.10.185.162:8080,10.10.213.93:8080,10.10.213.95:8080     25h
root@k8s-server:/home/ubuntu#
```

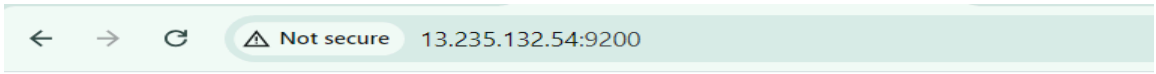
The Prometheus server is exposed on port **31885**



Install Node-exporter

Node Exporter collects data on various resources, including CPU, memory, disk usage, and network statistics. Node Exporter is specifically built to expose data in a format that Prometheus can understand. Follow the steps in this link to install and set Node Exporter on Node and master
https://developer.couchbase.com/tutorial-node-exporter-setup

```
root@k8s-server:/usr/lib/systemd/system# cat node_exporter.service
[Unit]
Description=Node Exporter
Documentation=https://prometheus.io/docs/guides/node-exporter/
Wants=network-online.target
After=network-online.target

[Service]
User=node_exporter
Group=node_exporter
Type=simple
Restart=on-failure
ExecStart=/usr/bin/node_exporter \
  --web.listen-address=:9200

[Install]
WantedBy=multi-user.target
root@k8s-server:/usr/lib/systemd/system# pwd
/usr/lib/systemd/system
root@k8s-server:/usr/lib/systemd/system#
```



Node Exporter

Metrics


```
# HELP go_gc_duration_seconds A summary of the pause duration of garbage collection cycles.
# TYPE go_gc_duration_seconds summary
go_gc_duration_seconds{quantile="0"} 0
go_gc_duration_seconds{quantile="0.25"} 0
go_gc_duration_seconds{quantile="0.5"} 0
go_gc_duration_seconds{quantile="0.75"} 0
go_gc_duration_seconds{quantile="1"} 0
go_gc_duration_seconds_sum 0
go_gc_duration_seconds_count 0
# HELP go_goroutines Number of goroutines that currently exist.
# TYPE go_goroutines gauge
go_goroutines 9
# HELP go_info Information about the Go environment.
# TYPE go_info gauge
go_info{version="go1.14.4"} 1
# HELP go_memstats_alloc_bytes Number of bytes allocated and still in use.
# TYPE go_memstats_alloc_bytes gauge
go_memstats_alloc_bytes 1.218016e+06
# HELP go_memstats_alloc_bytes_total Total number of bytes allocated, even if freed.
# TYPE go_memstats_alloc_bytes_total counter
go_memstats_alloc_bytes_total 1.218016e+06
# HELP go_memstats_buck_hash_sys_bytes Number of bytes used by the profiling bucket hash table.
# TYPE go_memstats_buck_hash_sys_bytes gauge
go_memstats_buck_hash_sys_bytes 4247
# HELP go_memstats_frees_total Total number of frees.
# TYPE go_memstats_frees_total counter
go_memstats_frees_total 626
# HELP go_memstats_gc_cpu_fraction The fraction of this program's available CPU time used by the GC since the program starte
# TYPE go_memstats_gc_cpu_fraction gauge
go_memstats_gc_cpu_fraction 0
# HELP go_memstats_gc_sys_bytes Number of bytes used for garbage collection system metadata.
# TYPE go_memstats_gc_sys_bytes gauge
go_memstats_gc_sys_bytes 3.436808e+06
# HELP go_memstats_heap_alloc_bytes Number of heap bytes allocated and still in use.
# TYPE go_memstats_heap_alloc_bytes gauge
go_memstats_heap_alloc_bytes 1.218016e+06
# HELP go_memstats_heap_idle_bytes Number of heap bytes waiting to be used
```

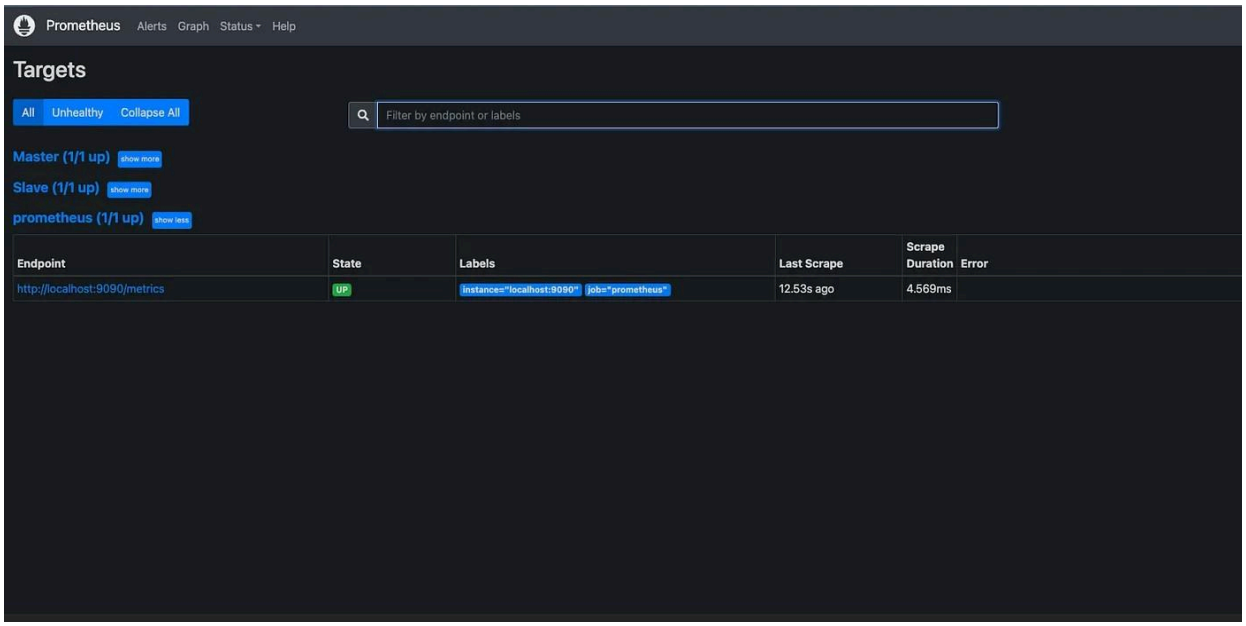
Configure Prometheus to Scrape Node Exporter Metrics

Create values.yaml file to configure the Prometheus and add the nod exporter as a scraping target by adding the IP of the master and node as targets.

```
serverFiles:
  prometheus.yml:
    global:
      scrape_interval: 15s
    scrape_configs:
      - job_name: 'node_exporter'
        static_configs:
          - targets: ['k8s-server:9200','172.0.4.8:9200']
```

To apply this file use the following command and restart the Prometheus.

```
helm upgrade prometheus prometheus-community/prometheus -f values.yaml
```



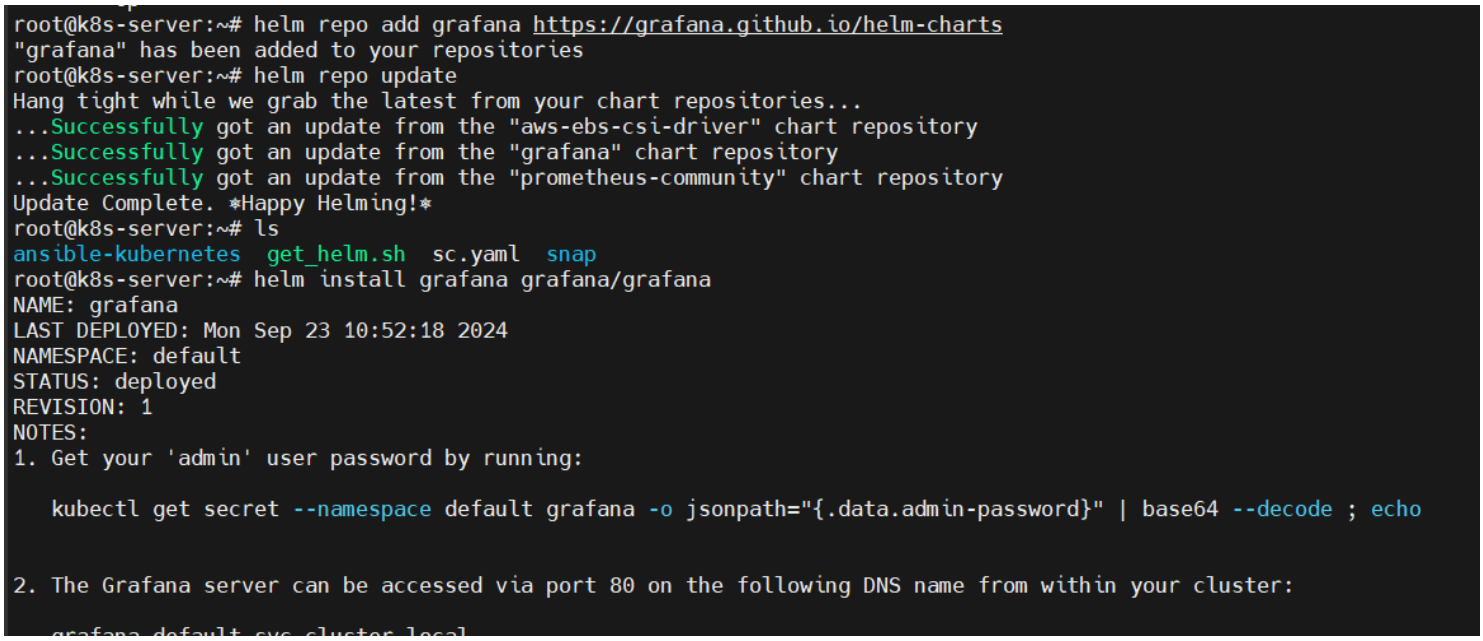
Install Grafana:

```
helm repo add grafana https://grafana.github.io/helm-charts
```

#Update helm repo
```
helm repo update
```

#Install Grafana
```
helm install grafana grafana/grafana
```

#Expose Grafana Service
```
kubectl expose service grafana --type=NodePort --target-port=3000
--name=grafana-ext
```

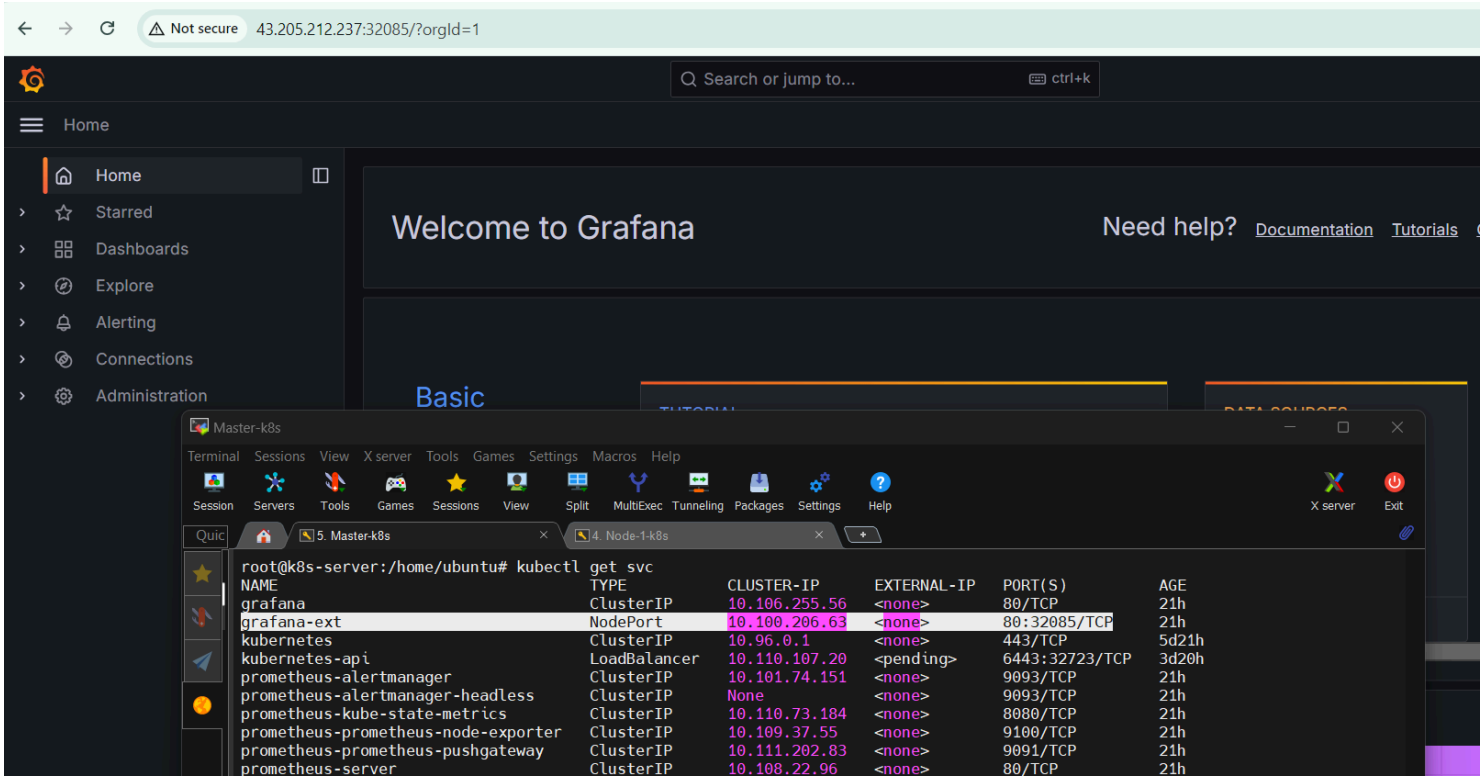


```
2. The Grafana server can be accessed via port 80 on the following DNS name from within your cluster:

   grafana.default.svc.cluster.local

   Get the Grafana URL to visit by running these commands in the same shell:
     export POD_NAME=$(kubectl get pods --namespace default -l "app.kubernetes.io/name=grafana,app.kubernetes
     kubectl --namespace default port-forward $POD_NAME 3000

3. Login with the password from step 1 and the username: admin
#################################################################################
######   WARNING: Persistence is disabled!!! You will lose your data when   #####
######            the Grafana pod is terminated.                            #####
#################################################################################
root@k8s-server:/home/ubuntu# kubectl get pods
NAME                                            READY   STATUS    RESTARTS      AGE
grafana-54658758c9-rblnb                        1/1     Running   0             11s
prometheus-alertmanager-0                       1/1     Running   0             24h
prometheus-kube-state-metrics-74cdb59bff-4c982  1/1     Running   3 (16m ago)   24h
prometheus-prometheus-node-exporter-grbbs       1/1     Running   1 (31m ago)   24h
prometheus-prometheus-node-exporter-sb629       1/1     Running   2 (16m ago)   24h
prometheus-prometheus-pushgateway-66fc55f8d-rb2t5   1/1     Running   2 (16m ago)   24h
prometheus-server-dd484f8d9-trzck               2/2     Running   0             24h
techappdeployment-549f54cf78-9w6dz              1/1     Running   1 (31m ago)   25h
techappdeployment-549f54cf78-httv6              1/1     Running   2 (16m ago)   25h
techappdeployment-549f54cf78-pzwh5              1/1     Running   2 (16m ago)   25h
root@k8s-server:/home/ubuntu# kubectl expose service grafana --type=NodePort --target-port=3000 --name=grafan
service/grafana-ext exposed
root@k8s-server:/home/ubuntu# kubectl get svc
NAME                                TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
grafana                             ClusterIP   10.96.37.198     <none>        80/TCP         20s
grafana-ext                         NodePort    10.101.98.213    <none>        80:30571/TCP   5s
kubernetes                          ClusterIP   10.96.0.1        <none>        443/TCP        41h
prometheus-alertmanager             ClusterIP   10.100.233.53    <none>        9093/TCP       24h
prometheus-alertmanager-headless    ClusterIP   None             <none>        9093/TCP       24h
prometheus-kube-state-metrics       ClusterIP   10.111.146.183   <none>        8080/TCP       24h
prometheus-prometheus-node-exporter ClusterIP   10.108.54.142    <none>        9100/TCP       24h
prometheus-prometheus-pushgateway   ClusterIP   10.104.200.205   <none>        9091/TCP       24h
prometheus-server                   ClusterIP   10.111.188.140   <none>        80/TCP         24h
prometheus-server-ext               NodePort    10.104.7.12      <none>        80:31885/TCP   24h
techappdeployment                   NodePort    10.109.180.246   <none>        80:32564/TCP   25h
root@k8s-server:/home/ubuntu#
```

```
kubectl get secret --namespace default grafana -o
jsonpath="{.data.admin-password}" | base64 --decode ; echo
```

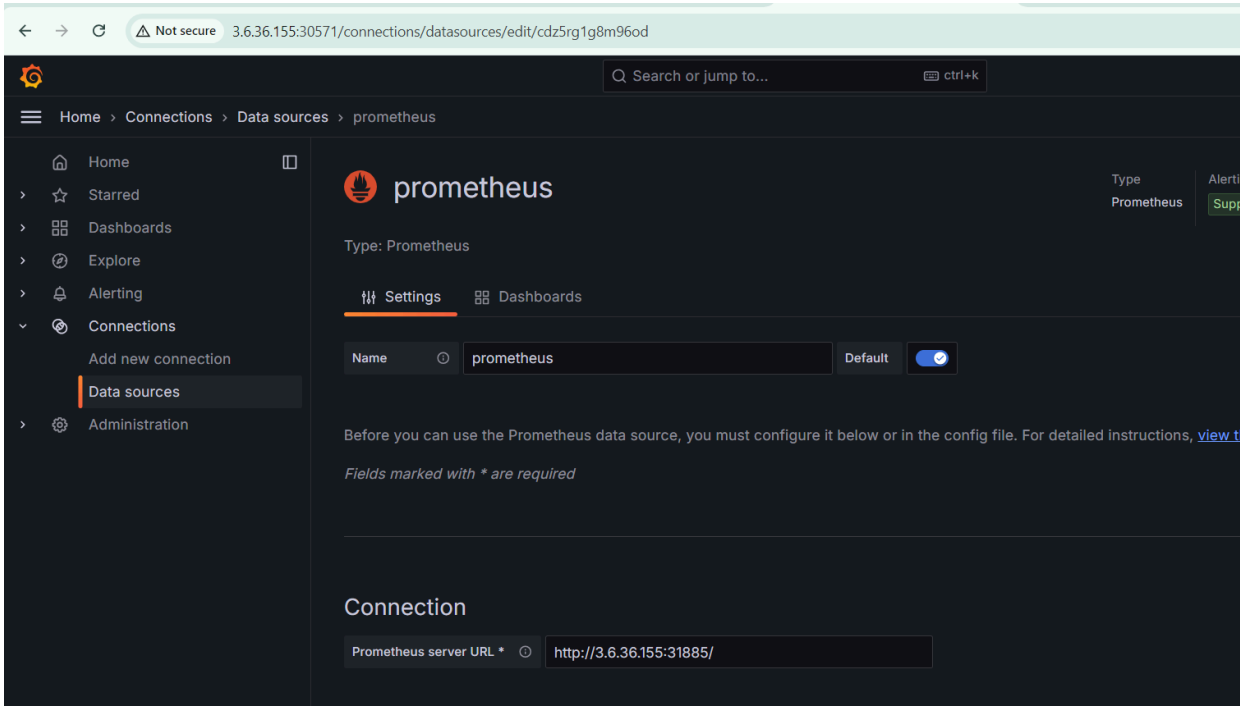
The above command will generate a password for the Grafane dashboard login.
3LCJzOMKDfGRTo9V69Wrm1wQnulArk6Dsk9mhLXh

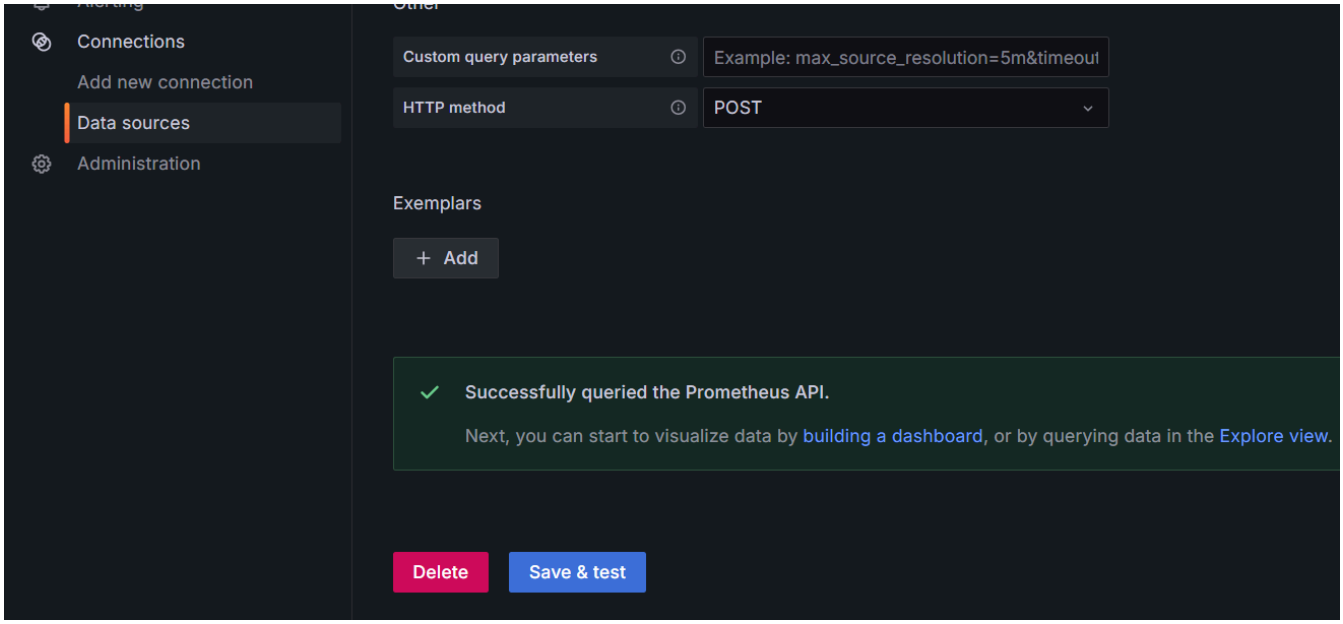
The Grafana service is exposed over port 32085, we can access the dashboard through the browser http://node-ip:32085



Now go to Grafana dashboard, click on Data source, add the prometheus as data source.
Prometheus server URL: http://3.6.36.155:31885/

Test the connection



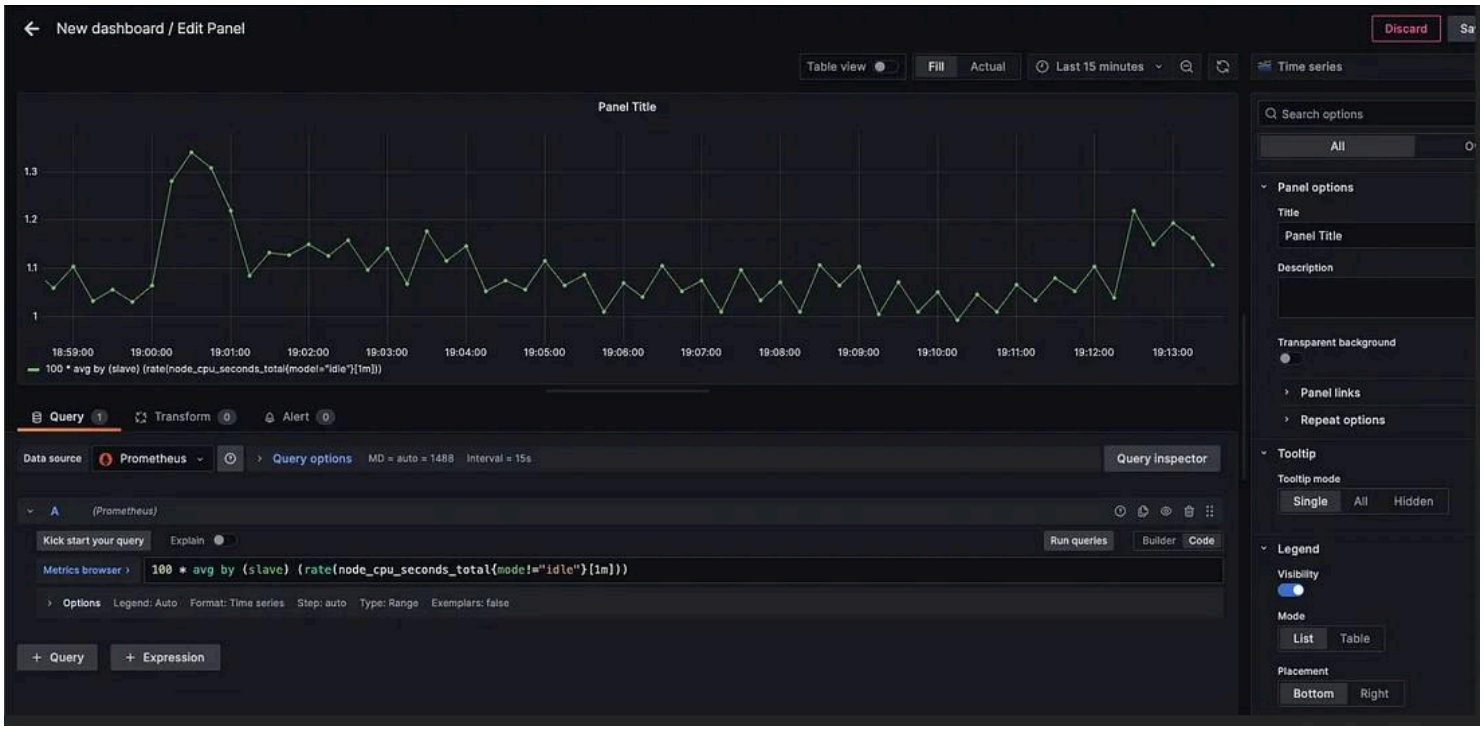


Now add the dashboard in prometheus. click on dashboard, Import dashboard,

Monitor the metrics at the Grafana dashboard:

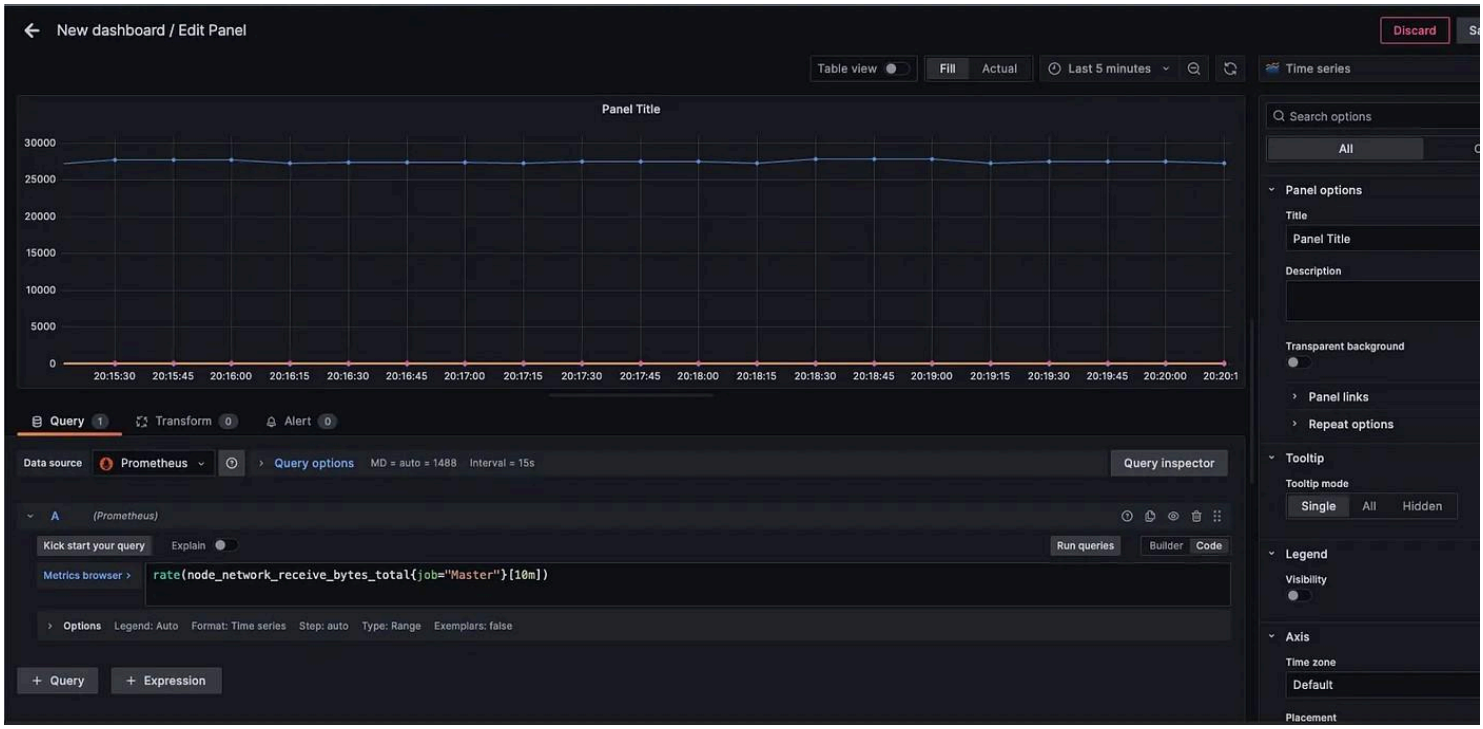
Below are the metrics captured in grafana dashboard

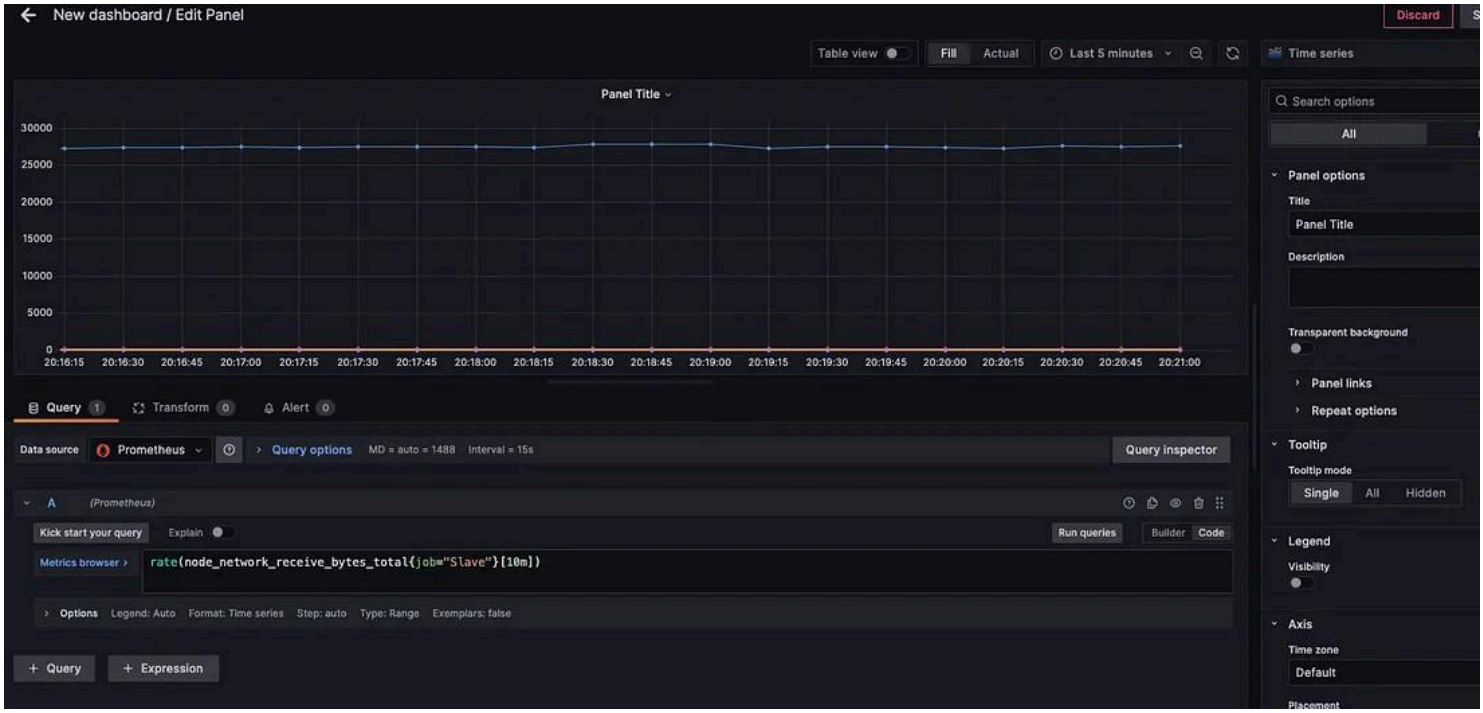
For CPU



For Memory Usage

For Network


New dashboard / Edit Panel
Discard
Table view
Fill
Actual
Last 5 minutes
Time series
Panel Title
30000
25000
20000
15000
10000
5000
0
20:15:30 20:15:45 20:16:00 20:16:15 20:16:30 20:16:45 20:17:00 20:17:15 20:17:30 20:17:45 20:18:00 20:18:15 20:18:30 20:18:45 20:19:00 20:19:15 20:19:30 20:19:45 20:20:00
Query 1
Transform 0
Alert 0
Data source
Prometheus
Query options
MD = auto = 1488
Interval = 15s
Query inspector
A
(Prometheus)
Kick start your query
Explain
Run queries
Builder
Code
Metrics browser
rate(node_network_receive_bytes_total{job="Master"}[10m])
Options
Legend: Auto
Format: Time series
Step: auto
Type: Range
Exemplars: false
+ Query
+ Expression
Search options
All
Panel options
Title
Panel Title
Description
Transparent background
Panel links
Repeat options
Tooltip
Tooltip mode
Single
All
Hidden
Legend
Visibility
Axis
Time zone
Default
Placement


New dashboard / Edit Panel
Discard
Table view
Fill
Actual
Last 5 minutes
Time series
Panel Title
30000
25000
20000
15000
10000
5000
0
20:16:15 20:16:30 20:16:45 20:17:00 20:17:15 20:17:30 20:17:45 20:18:00 20:18:15 20:18:30 20:18:45 20:19:00 20:19:15 20:19:30 20:19:45 20:20:00 20:20:15 20:20:30 20:20:45 20:21:00
Query 1
Transform 0
Alert 0
Data source
Prometheus
Query options
MD = auto = 1488
Interval = 15s
Query inspector
A
(Prometheus)
Kick start your query
Explain
Run queries
Builder
Code
Metrics browser
rate(node_network_receive_bytes_total{job="Slave"}[10m])
Options
Legend: Auto
Format: Time series
Step: auto
Type: Range
Exemplars: false
+ Query
+ Expression
Search options
All
Panel options
Title
Panel Title
Description
Transparent background
Panel links
Repeat options
Tooltip
Tooltip mode
Single
All
Hidden
Legend
Visibility
Axis
Time zone
Default
Placement

Conclusion

ABC Technologies has successfully finished the initial phase of the project aimed at acquiring data from the offline business store for analytics and predictive purposes. The team employed the DevOps model to automate the development, deployment, and testing of servlets and HTML pages. This approach has led to the creation of a more reliable, scalable, and efficient system, making maintenance simpler. Some of the advantages that ABC Technologies can expect include:

1. **Highly available**
2. **Highly scalable**
3. **Highly Performant**
4. **Easily built and maintained**
5. **Developed and deployed quickly**
6. **Lower production bugs**
7. **Frequent releases**
8. **Better customer experiences**
9. **Lesser time to market**

THE END