

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>

A screenshot of a GitHub repository page. The top navigation bar shows the repository name 'TechApp' under 'Soniaakhtar1690'. The main content area displays the 'README.md' file. The file content is as follows:

```
ABC TECHNOLOGIES RETAIL APP DEPLOYMENT KEY FILES

Dockerfile: I used this file to create a docker image using the artifact packaged by the code store in the current repository Techapp.

Docker Image: Docker image is stored in my Docker hub repo. link: https://hub.docker.com/repository/docker/sonia1690/abc-techapp/general

ansible-dockerimage.yaml: I used this Ansible playbook to create a docker image and container in the docker server.

Kubernetes.yaml: I used this Ansible playbook to deploy the application in k8s through the Ansible server.

sc.yaml: I used this yaml file to create a storage class in the k8s cluster which is necessary to set up the Prometheus and Grafana in k8s.

techappDeployment.yaml: I used this manifest file in my k8s server to deploy my application.

techappService.yaml: I used this manifest file in the k8s server to expose my application as a service.
```

File	Description	Time Ago
Dockerfile	Dockerfile	1 hour ago
README.md	Update README.md	12 minutes ago
ansible-dockerimage.yaml	Create ansible-dockerimage.yaml	1 hour ago
kubernetes.yaml	Playbook to deploy the app	39 minutes ago
pom.xml	First Commit	2 months ago
pom.xml.bak	First Commit	2 months ago
sc.yaml	Create a storage class in K8s cluster	36 minutes ago
techappDeployment.yaml	Update techappDeployment.yaml	52 minutes ago
techappService.yaml	Manifest file to deploy the service in K8s cluster	25 minutes ago

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:

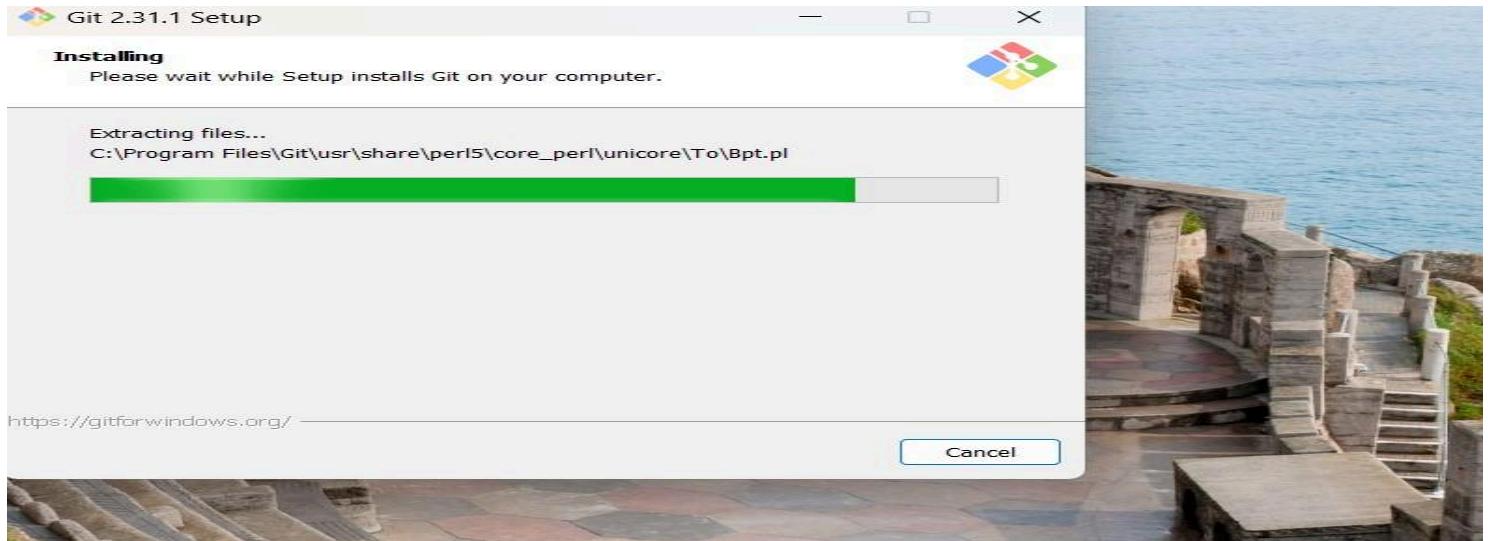
- Maven
- GitHub
- Jenkins
- Docker
- Ansible
- Kubernetes
- Grafana
- Prometheus

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 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
$ 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
$ 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
$ 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
$ 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
    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
va

sonia@Sonia MINGW64 ~/OneDrive/Documents/Industry Grade Project I - Java Project
$ git branch
sonia@Sonia MINGW64 ~/OneDrive/Documents/Industry Grade Project I - Java Project
$ git branch -M main
sonia@Sonia MINGW64 ~/OneDrive/Documents/Industry Grade Project I - Java Project
$ git commit -m "First Commit"
[main (root-commit) 81b2934] First Commit
 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
    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
va

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

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 alsound-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-jni libatk1.0-0t64 libatspi2.0-0t64
libavahi-client3 libavahi-common-data libavahi-common3 libcups2t64 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 libpciauth0 libpcslite1
libpthread-stubs0-dev libsm-dev libsm6 libvulkan1 libwayland-client0 libx11-dev libx11-xcb1 libxau-dev
libxaw7 libxcb-dri2-0 libxcb-dri3-0 libxcb-glx0 libxcb-present0 libxcb-randr0 libxcb-shape0
libxcb-shm0 libxcb-sync1 libxcb-xfixes0 libxcb1-dev libcomposite1 libxdmcp-dev libxfixes3 libxft
libxi6 libxinerama1 libxkbfile1 libxmu6 libxpm4 libxrandr2 libxrender1 libxshmfence1 libxt-dev
libxt6t64 libxtst6 libxv1 libxf86dga1 libxf86vm1 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 alsound-conf 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

root@jenkin-server:/# jenkins --version
2.462.1
root@jenkin-server:/# sudo systemctl start jenkins
root@jenkin-server:/# sudo systemctl enable jenkins
Synchronizing state of jenkins.service with SysV service script with /usr/lib/systemd/systemd-sysv-inst...
.
Executing: /usr/lib/systemd/systemd-sysv-install enable jenkins
root@jenkin-server:/# sudo systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
   Loaded: loaded (/usr/lib/systemd/system/jenkins.service; enabled; preset: enabled)
     Active: active (running) since Tue 2024-08-27 08:57:57 UTC; 2min 33s ago
       Main PID: 4566 (java)
          Tasks: 46 (limit: 4676)
         Memory: 583.6M (peak: 595.8M)
            CPU: 18.446s
        CGroup: /system.slice/jenkins.service
                  └─4566 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=...

Aug 27 08:57:51 jenkin-server jenkins[4566]: 3f816fb88bab41ffb318b2b18a6ee910
Aug 27 08:57:51 jenkin-server jenkins[4566]: This may also be found at: /var/lib/jenkins/secrets/initia...
Aug 27 08:57:51 jenkin-server jenkins[4566]: ****
Aug 27 08:57:51 jenkin-server jenkins[4566]: ****
Aug 27 08:57:51 jenkin-server jenkins[4566]: ****
Aug 27 08:57:57 jenkin-server jenkins[4566]: 2024-08-27 08:57:57.213+0000 [id=31]      INFO    jenkins
Aug 27 08:57:57 jenkin-server jenkins[4566]: 2024-08-27 08:57:57.231+0000 [id=24]      INFO    jenkins
Aug 27 08:57:57 jenkin-server systemd[1]: Started jenkins.service - Jenkins Continuous Integration Serv...
Aug 27 08:57:57 jenkin-server jenkins[4566]: 2024-08-27 08:57:57.983+0000 [id=49]      INFO    jenkins
Aug 27 08:57:57 jenkin-server jenkins[4566]: 2024-08-27 08:57:57.983+0000 [id=49]      INFO    jenkins
[Lines 1-20/20 (END)]
```

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

root@jenkin-server:/# 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@jenkin-server:/#
```

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/plexus/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/log4j-1.2.12.jar (44 kB at 2.1 MB/s)  
Downloaded from central: https://repo.maven.apache.org/maven2/com/google/collections/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#
```

Support 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

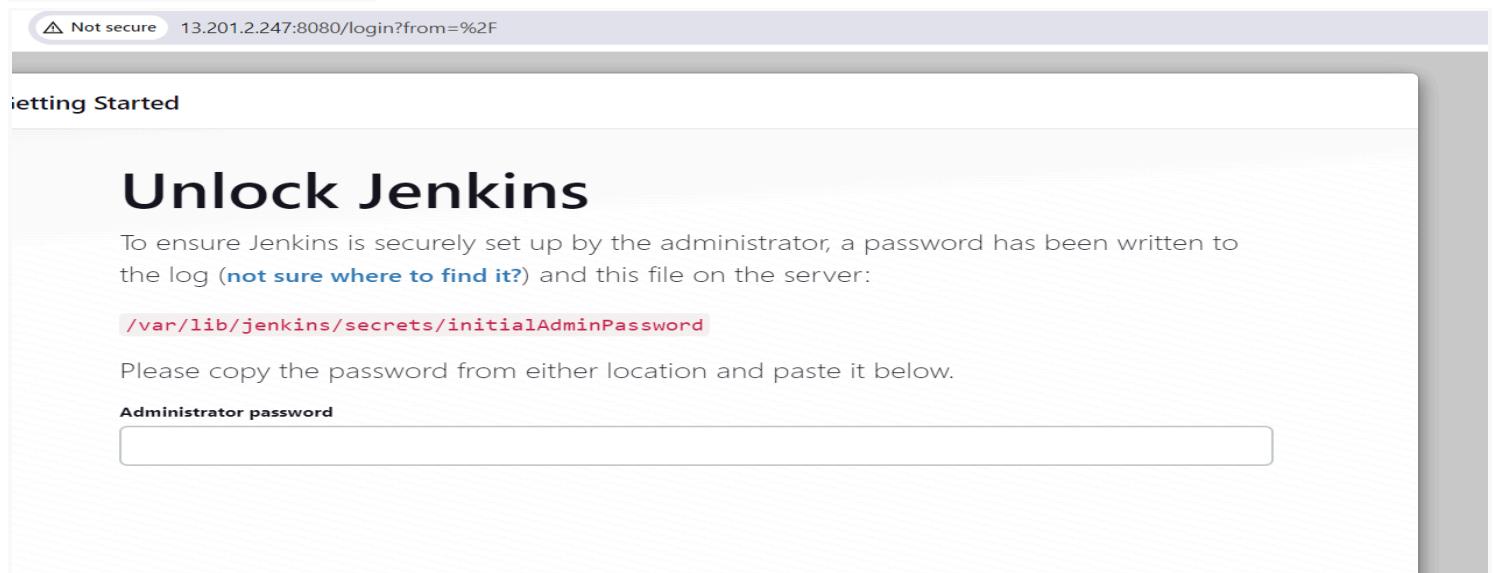
1. create a build pipeline containing a job for each
 - One for compiling source code
 - Second for testing source code
 - Third for packing the code
2. Execute the CICD pipeline to execute the jobs created in step1
3. Set up a master-slave node to distribute the tasks in the pipeline.

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@rp-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
*	/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: █
```

The screenshot shows the Jenkins 'Configure Tools' page at the URL `13.201.2.247:8080/manage/configureTools/`. The page title is 'Manage Jenkins > Tools'. A sub-menu item 'Use default maven global settings' is visible. The main section is titled 'JDK installations' and contains a form for adding a new JDK. The form fields are: 'Name' (value: 'jdk17'), which is marked as 'Required' with a red exclamation icon. Below it is the 'JAVA_HOME' field (value: '/usr/lib/jvm/java-17-openjdk-amd64'). At the bottom of the form is a checkbox labeled 'Install automatically' with a question mark icon next to it.

Git Integration

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

Git installations

The screenshot shows a configuration panel for 'Git' with the following fields:

- Name:** Default
- Path to Git executable:** git
- Install automatically:**

At the bottom left is a button labeled "Add Git".

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#
```

Maven installations

The screenshot shows a configuration panel for 'Maven' with the following fields:

- Name:** maven
- MAVEN_HOME:** /usr/share/maven
- Install automatically:**

At the bottom left is a button labeled "Add Maven".

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

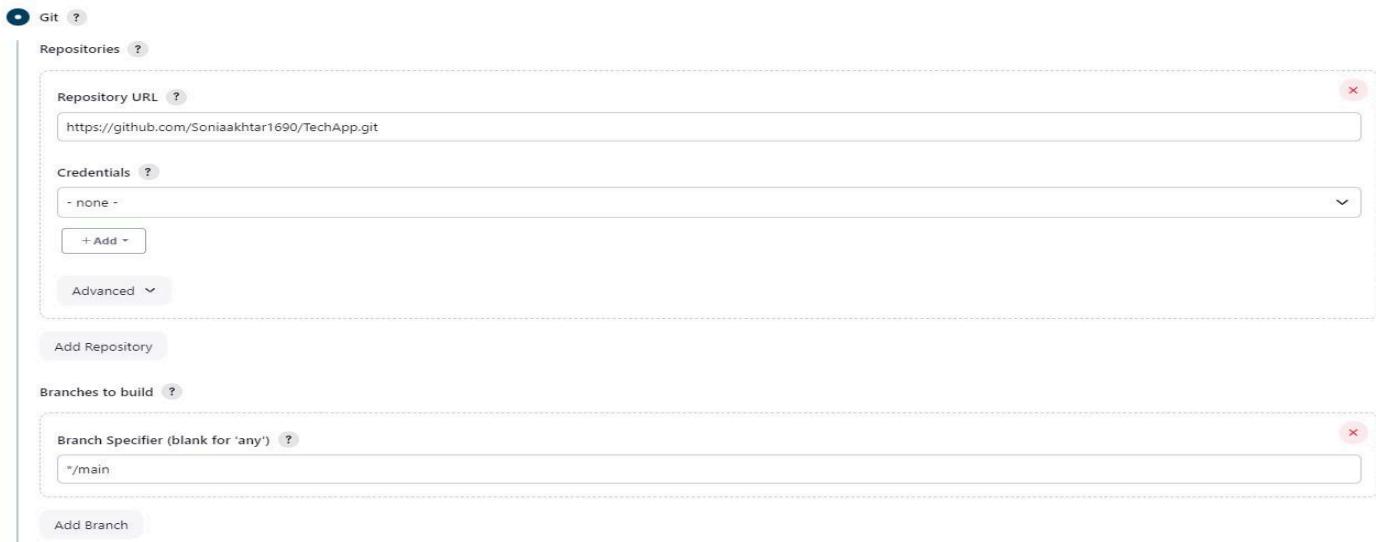
- One for compiling source code
- Second for testing source code
- Third for packing the code

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”

With Ant [?](#)

Build Steps

☰ Invoke top-level Maven targets [?](#) ✖

Goals

compile

Advanced ▾

Add build step ▾

Post-build Actions

Now build 1-compile job

← → ⚡ Not secure 65.0.80.232:8080/job/1-compile/1/



Dashboard > 1-compile > #1

Status ✓ #1 (Aug 27, 2024, 11:06:08 AM)

</> Changes

☒ Console Output

📝 Edit Build Information

🗑 Delete build '#1'

⌚ Timings

⚡ git Revision: 81b293486e0b1b0f3cd7634634bd811b00386b5b
Repository: <https://github.com/Soniaakhtar1690/TechApp.git>
• refs/remotes/origin/main

</> No changes.

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.

The screenshot shows the Jenkins interface for the '2-Test' job. At the top, there's a navigation bar with back, forward, and search icons, followed by a 'Not secure' warning and the URL '65.0.80.232:8080/job/2-Test/'. Below the header is the Jenkins logo and the project name '2-Test'. A sidebar on the left contains links for Status, Changes, Workspace, Build Now, Configure, Delete Project, and Rename. The main content area has a 'Status' bar with a green checkmark and the text '2-Test'. Underneath is a 'Upstream Projects' section showing '1-compile' with a green circular icon. The 'Permalinks' section lists several build links. At the bottom, there's a 'Build History' table with two entries: '#2 Aug 27, 2024, 11:22 AM' and '#1 Aug 27, 2024, 11:19 AM'.

Build	Date
#2	Aug 27, 2024, 11:22 AM
#1	Aug 27, 2024, 11:19 AM

```

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

[0[1;34mINFO[m] 0[1m-----0[m
[0[1;34mINFO[m] 0[1;32mBUILD SUCCESS0[m
[0[1;34mINFO[m] 0[1m-----0[m
[0[1;34mINFO[m] Total time: 4.255 s
[0[1;34mINFO[m] Finished at: 2024-08-27T11:22:23Z
[0[1;34mINFO[m] 0[1m-----0[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 screenshot shows the Jenkins dashboard for the '3-package' job. The job status is 'Status' (green checkmark) and 'Upstream Projects' is '2-Test'. The 'Build History' table shows a single build (#1) from Aug 27, 2024, at 11:30 AM. Permalinks for the last four builds are listed as follows:

- Last build (#1), 43 sec ago
- Last stable build (#1), 43 sec ago
- Last successful build (#1), 43 sec ago
- Last completed build (#1), 43 sec ago

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] 0[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-----
[1;34mINFO[m] 1;32mBUILD SUCCESS[m
[1;34mINFO[m] 1m-----
[1;34mINFO[m] Total time: 5.692 s
[1;34mINFO[m] Finished at: 2024-08-27T11:30:46Z
[1;34mINFO[m] 1m-
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.

The screenshot shows the Jenkins Pipeline configuration interface. At the top, there's a section for 'Build Pipeline View Title' with a dropdown menu and a text input field containing 'Build Pipeline View Title'. Below this is the 'Pipeline Flow' section, which includes a 'Layout' dropdown set to 'Based on upstream/downstream relationship'. A note below explains that this layout mode derives the pipeline structure based on trigger relationships between jobs. Under 'Upstream / downstream config', there's a 'Select Initial Job' dropdown with '1-compile' selected. The bottom sections, 'Trigger Options' and 'Build Cards', are partially visible.

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

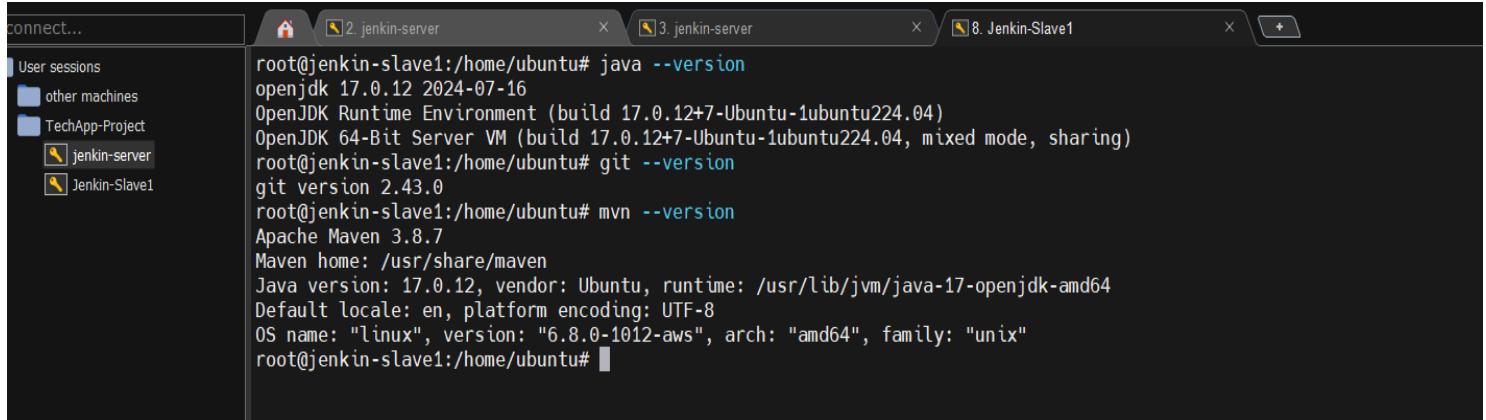
Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Available
Jenkins-Server-TechApp	i-0dc2cc47090a74040	Running	t2.medium	2/2 checks passed	View alarms	ap-south-1
Jenkin-Slave1	i-0aa7a1f54054e13f0	Running	t2.micro	2/2 checks passed	View alarms	ap-south-1

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
```



The screenshot shows a terminal window with three tabs: 'jenkin-server' (tab 2), 'jenkin-server' (tab 3), and 'Jenkins-Slave1' (tab 8). The 'Jenkins-Slave1' tab contains the following command-line output:

```
root@jenkin-slave1:/home/ubuntu# java --version  
openjdk 17.0.12 2024-07-16  
OpenJDK Runtime Environment (build 17.0.12+7-Ubuntu-1ubuntu224.04)  
OpenJDK 64-Bit Server VM (build 17.0.12+7-Ubuntu-1ubuntu224.04, mixed mode, sharing)  
root@jenkin-slave1:/home/ubuntu# git --version  
git version 2.43.0  
root@jenkin-slave1:/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@jenkin-slave1:/home/ubuntu#
```

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
slavedir
snap-private-tmp
systemd-private-7f7b847719094060ab2fe5b08ecb13e3-ModemManager.service-7nupnE
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-SGZrZ
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-0QWgQa
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

The screenshot shows the Jenkins 'Configure' screen for a new node. The node is named 'Jenkin-Slave1'. It has a description field, which is empty. The 'Number of executors' is set to 1. The 'Remote root directory' is set to '/tmp/slavedir'. There are empty 'Labels' and 'Usage' fields. The 'Launch method' dropdown is visible at the bottom.

Name: Jenkin-Slave1

Description:

Plain text Preview

Number of executors: 1

Remote root directory: /tmp/slavedir

Labels:

Usage: Use this node as much as possible

Launch method:

Usage ?

Use this node as much as possible

Launch method ?

Launch agent by connecting it to the controller

Availability ?

Keep this agent online as much as possible

Node Properties

- Disable deferred wipeout on this node ?
- Disk Space Monitoring Thresholds
- Environment variables
- Tool Locations

Save

Jenkins

Search (CTRL+K) ⚡ 2 sonia

Dashboard > Manage Jenkins > Nodes >

Nodes

+ New Node Configure More

S	Name	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
	Built-in Node	Linux (amd64)	In sync	19.86 GiB	1 0 B	19.86 GiB	
	Jenkin-Slave1	Linux (amd64)	In sync	11.14 GiB	1 0 B	11.14 GiB	
	Data obtained	2 min 41 sec	2 min 41 sec	2 min 41 sec	2 min 41 sec	2 min 41 sec	2 min 41 sec

Icon: S M L

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.

Jenkins Location

Jenkins URL ?

http://65.0.80.232:8080/

System Admin e-mail address ?

address not configured yet <nobody@nowhere>

Serve resource files from another domain

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

⚠ Not secure 65.0.80.232:8080/manage/configureSecurity/

Manage Jenkins > Security

Shows descriptions mostly as written. HTML unsafe characters like < and & are escaped to their respective character entities, and line breaks are converted to their HTML equivalent.

Agents

TCP port for inbound agents ?

Fixed

8090

Random

Disable

Agent protocols ▾

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_8e80c9b
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

```

Jenkins

Search (CTRL+K) ? 2 sonia

Dashboard > Manage Jenkins > Nodes >

Nodes							
S	Name	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
idle	Built-In Node	Linux (amd64)	In sync	19.86 GiB	0 B	19.86 GiB	
idle	Jenkin-Slave1	Linux (amd64)	In sync	11.14 GiB	0 B	11.14 GiB	
	Data obtained	2 min 41 sec	2 min 41 sec	2 min 41 sec	2 min 41 sec	2 min 41 sec	2 min 41 sec

Icon: S M L

Build Queue: 0 builds in the queue.

Build Executor Status: Built-In Node (idle), Jenkin-Slave1 (idle).

+ New Node Configure More

See the Compile job console output on the slave machine below

⚠ Not secure 65.0.80.232:8080/job/1-compile/6/console

ns

Search (CTRL+K) ⚡ 2 soi

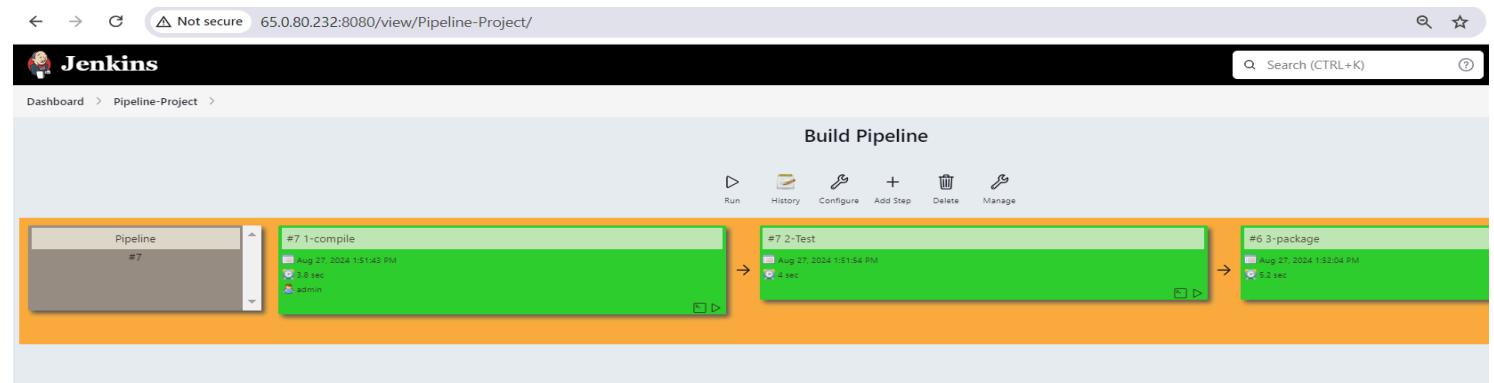
compile > #6 > Console Output

Console Output

Download Copy

```
Started by user sonia
Running as SYSTEM
Building remotely on Jenkins-Slave[1] in workspace /tmp/slavedir/workspace/1-compile
The recommended git tool is: NONE
No credentials specified
Cloning the remote Git repository
Cloning repository https://github.com/Soniaakhtar1690/TechApp.git
> git init /tmp/slavedir/workspace/1-compile # 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/*
> git config remote.origin.url https://github.com/Soniaakhtar1690/TechApp.git # timeout=10
> git config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=10
Avoid second fetch
> git rev-parse refs/remotes/origin/main^(commit) # timeout=10
Checking out Revision 81b293406e0b1b0f3cd7634634bd811b00386b5b (refs/remotes/origin/main)
> git config core.sparsecheckout # timeout=10
```

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, assuming it is a local network"
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 containerdVersion=1.8.18
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 Commit"
> 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

```

The screenshot shows the Docker Hub interface. At the top, there are navigation links for 'hub.docker.com' (with a refresh icon), 'Explore', 'Repositories' (which is underlined in blue), and 'Organizations'. A search bar on the right contains the placeholder 'Search Docker Hub'. Below the header, there's a dropdown menu set to 'sonia1690' and a search input field with 'sonia1690 / techapp' typed into it. To the right of the search input is a blue button labeled 'Create repository'. Underneath, a card displays the repository details: 'sonia1690 / techapp', 'Contains: Image', 'Last pushed: 6 minutes ago', '0 stars', '3 forks', 'Public', and 'Scout inactive'.

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@f4e897a2dab:/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
203c1226debfae7ac16b2e8831f3c6398fd98ded0693573199c2a1a37bc926a0
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

Instances (2) Info		Last updated	C	Connect	Instance state ▾	Actions ▾	Launch
		less than a minute ago					
<input type="checkbox"/> Find Instance by attribute or tag (case-sensitive)		Running ▾					
<input type="checkbox"/>	Name ↗	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone ▾
<input type="checkbox"/>	Jenkins-server	i-0464a9a6876749000	Running C Q	t2.medium	2/2 checks passec View alarms +		ap-south-1a
<input type="checkbox"/>	Docker-host	i-03bd317bc121d062a	Running C Q	t2.small	2/2 checks passec View alarms +		ap-south-1a

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
```

```
root@jenkins-server:~# ssh 172.0.5.39
$ 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
```

```
drwxr-x--- 2 dockeradmin root 4096 Sep 22 10:11 .ssh  
$ ssh-keygen  
Generating public/private ed25519 key pair.  
Enter file in which to save the key (/home/dockeradmin/.ssh/id_ed25519):  
Enter passphrase (empty for no passphrase):  
Enter same passphrase again:  
Your identification has been saved in /home/dockeradmin/.ssh/id_ed25519  
Your public key has been saved in /home/dockeradmin/.ssh/id_ed25519.pub  
The key fingerprint is:  
SHA256:BVaJkwj/kc+oZyK/0dz49uL9xQ01+UCmHdgZ9GztlGs dockeradmin@jenkins-server  
The key's randomart image is:  
++-[ED25519 256]--  
| ...o+.. +*o |  
| ...+o.. .=o++|  
| . o... . o+B|  
| . *      **|  
| S o      .Eo|  
| + o      .o.|  
| . + * .    +|  
| o = .o.   . |  
| o. 0000.. . |  
+---[SHA256]---
```

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
root@docker-host:/home# cd dockeradmin/
```

```
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
root@docker-host:/home/dockeradmin/.ssh# id dockeradmin
```

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 AAAAC3NzaC1lZDI1NTE5AAAI03iRuHTU92/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 AAAAC3NzaC1lZDI1NTE5AAAI03iRuHTU92/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
```

```
lost host key verification failed.
$ 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**

The screenshot shows the Jenkins Manage Plugins interface. On the left, there's a sidebar with options: Updates, Available plugins (which is selected and highlighted in grey), Installed plugins, Advanced settings, and Download progress. The main area has a search bar at the top with the text 'publish-over'. Below the search bar, there are two tabs: 'Install' and 'Name' (sorted in descending order). A list of plugins is displayed, each with a checkbox, the plugin name, its version, its last release date, and a brief description. The 'Publish Over SSH' plugin is selected for installation, indicated by a checked checkbox.

Install	Name	Released
<input type="checkbox"/>	Infrastructure plugin for Publish Over X 0.22 Send build artifacts somewhere.	6 yr 5 mo ago
<input checked="" type="checkbox"/>	Publish Over SSH 1.25 Artifact Uploaders Build Tools Send build artifacts over SSH	1 yr 2 mo ago

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)

SSH Servers

≡ **SSH Server**

Name ?
docker-host

! Required. Cannot contain < & ' " \

Hostname ?
15.206.94.16

! Required

Username ?
dockeradmin

[Use password authentication, or use a different key](#)

Tick the following box:

Use password authentication, or use a different key

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

Use password authentication, or use a different key ?

Passphrase / Password ?

Path to key ?

Key ?


```
-----BEGIN OPENSSH PRIVATE KEY-----  
b3BlnNzaC1rZXktdjEAAAAABG5vbmUAAAAEb9uZQAAAAAAAAABAAAAMwAAAAtzc2gtZW  
QyNTUxOQAAACDt4kbh01Pdv5s6srVqrkB9bdjPtOoOxHfhcbmKyKnDAAAAl3iMfd4j  
HwAAAAtzc2gtZWQyNTUxOQAAACDt4kbh01Pdv5s6srVqrkB9bdjPtOoOxHfhcbmKyKnDA  
AAAEAPW1QjPkPdcZzqUTiEKb1FMp4778ySo0r2F1bTCtSP9+3iRuHTU92/mzqtWqt2QH1  
-----END OPENSSH PRIVATE KEY-----
```

The private key.

```
root@jenkins-server:/home/dockeradmin/.ssh# cat id_ed25519
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEb9uZQAAAAAAAAABAAAAMwAAAAtzc2gtZW
QyNTUx0QAAACDt4kbh01Pdv5s6srVqrdkB9bdjPt0o0xHfhcbmKyKnDAAAAL3iMfJd4j
HwAAAAtzc2gtZWQyNTUx0QAAACDt4kbh01Pdv5s6srVqrdkB9bdjPt0o0xHfhcbmKyKnDA
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

The screenshot shows the Jenkins system configuration page at <http://3.108.66.214:8080/manage/configure>. The page title is "Manage Jenkins > System >". It contains fields for proxy configuration:

- Proxy type: A dropdown menu currently set to "None".
- Proxy host: An empty input field.
- Proxy port: An empty input field.
- Proxy user: An empty input field.
- Proxy password: An empty input field.

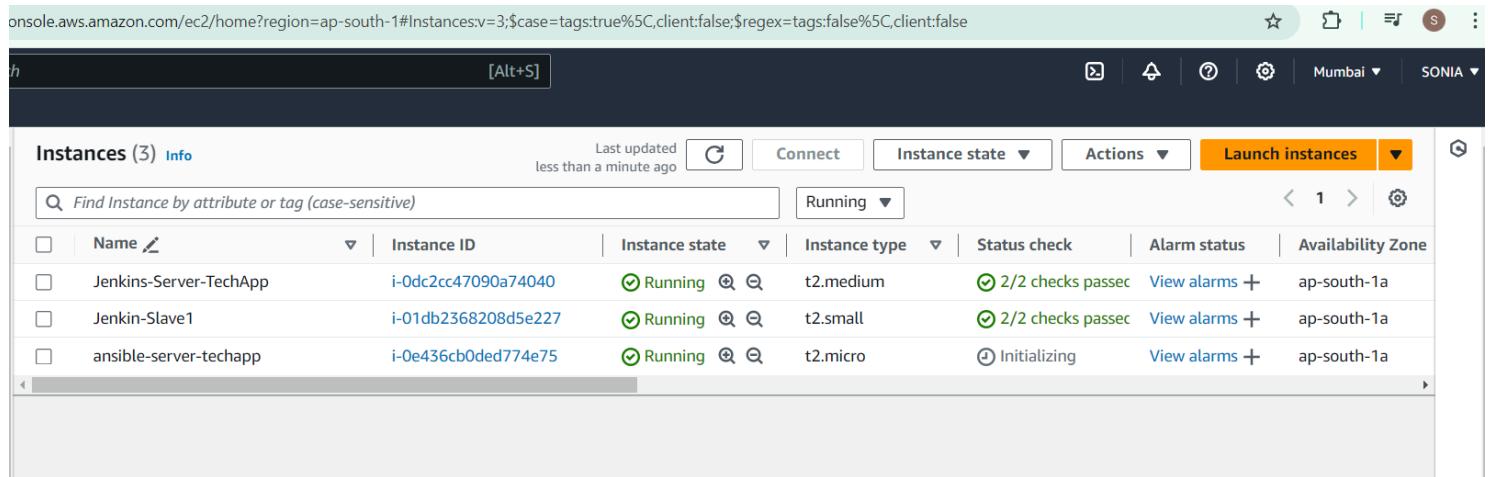
Below the form, there is a "Success" message and a "Test Configuration" button. At the bottom left, there is an "Add" button.

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



The screenshot shows the AWS Management Console EC2 Instances page. The URL in the address bar is `console.aws.amazon.com/ec2/home?region=ap-south-1#Instances:v=3;$case=true%5C,client:false;$regex=tags:false%5C,client:false`. The page displays three EC2 instances:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
Jenkins-Server-TechApp	i-0dc2cc47090a74040	Running	t2.medium	2/2 checks passed	View alarms	ap-south-1a
Jenkin-Slave1	i-01db2368208d5e227	Running	t2.small	2/2 checks passed	View alarms	ap-south-1a
ansible-server-techapp	i-0e436cb0ded774e75	Running	t2.micro	Initializing	View alarms	ap-south-1a

```
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
ansadmin  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) 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: dockerd.service
   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="2024-08-29T06:03:13.442000000Z" level=info msg="Loading containers: done"
Aug 29 06:03:13 ansible-server dockerd[3907]: time="2024-08-29T06:03:13.851440186Z" 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 inti
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
```

```

# Ignore 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
```

```
drwxr-xr-x 3 ansadmin root 4096 Aug 29 06:57 .
drwxr-xr-x 4 root      root 4096 Aug 29 06:57 ..
drwxr-xr-x 2 ansadmin root 4096 Aug 29 06:57 .ssh
$ ssh-keygen
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/ansadmin/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ansadmin/.ssh/id_ed25519
Your public key has been saved in /home/ansadmin/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:fPm1ZKWoRoHSvc1m0QkeH1paQJNz83mBQUm8+dDFeIA ansadmin@ip-192-0-0-5
The key's randomart image is:
+--[ED25519 256]--+
 .+. ++.+
 .o=o E o +
 . =o*o = +
 o @ B*ooo
 + S +++
 + + +.o
 o o
+
+---[SHA256]---+
$ ls
$ ls -la
total 12
drwxr-xr-x 3 ansadmin root 4096 Aug 29 06:57 .
drwxr-xr-x 4 root      root 4096 Aug 29 06:57 ..
drwxr-xr-x 2 ansadmin root 4096 Aug 29 07:00 .ssh
$ cd .ssh
$ ls
id_ed25519  id_ed25519.pub
$
```

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.

```
ansadmin@ip-192-0-0-11:~$ mkdir -p ~/.ssh  
touch ~/.ssh/authorized_keys  
chmod 700 ~/.ssh  
chmod 600 ~/.ssh/authorized_keys
```

```
ansadmin@192.0.0.11: Permission denied (publickey).  
$ mkdir -p ~/.ssh  
touch ~/.ssh/authorized_keys  
chmod 700 ~/.ssh  
chmod 600 ~/.ssh/authorized_keys  
$ $ $  
$ ssh ansadmin@192.0.0.11  
ansadmin@192.0.0.11: Permission denied (publickey).  
$ ssh ansadmin@192.0.0.11  
Welcome to Ubuntu 24.04 LTS (GNU/Linux 6.8.0-1012-aws x86_64)  
  
 * Documentation:  https://help.ubuntu.com  
 * Management:    https://landscape.canonical.com  
 * Support:       https://ubuntu.com/pro  
  
System information as of Wed Sep 11 08:45:23 UTC 2024  
  
System load: 0.0          Processes: 131  
Usage of /: 11.5% of 13.49GB  Users logged in: 2  
Memory usage: 22%          IPv4 address for enX0: 192.0.0.11  
Swap usage: 0%  
  
Expanded Security Maintenance for Applications is not enabled.  
0 updates can be applied immediately.  
Enable ESM Apps to receive additional future security updates.  
See https://ubuntu.com/esm or run: sudo pro status  
  
The list of available updates is more than a week old.  
To check for new updates run: sudo apt update
```

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/+xk0cUoATdMZAjqJZ0sn8aD+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 screenshot shows the Jenkins configuration interface for managing SSH servers. The URL is 3.110.132.63:8080/manage/configure. The page title is "SSH Servers". A sub-section titled "SSH Server" is active. The fields filled are:

- Name:** ansible-server (with a validation error message: "Required. Cannot contain < & ' \" \")
- Hostname:** 192.0.0.9 (with a validation error message: "Required")
- Username:** ansadmin (with a validation error message: "Required")
- Remote Directory:** /opt/docker
- Avoid sending files that have not changed:** An unchecked checkbox.
- Advanced:** A dropdown menu showing "Edited".
- Use password authentication, or use a different key:** A checked checkbox.

Proxy host	<input type="text"/>	?
Proxy port	<input type="text"/>	?
Proxy user	<input type="text"/>	?
Proxy password	<input type="text"/>	?
Success	<input type="button" value="Test Configuration"/>	<input type="button" value="Delete"/>

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.

Instances (2) Info										
		Last updated less than a minute ago		Connect	Instance state ▾	Actions ▾	Launch instances ▾			
<input type="checkbox"/> Name ▾		Instance ID	Instance state ▾	Instance type ▾	Status check	Alarm status	Availability Zone ▾	Public IPv4 DNS		
<input type="checkbox"/>	Node1-k8s	i-02e039ce3f8629a96	Running View details Logs	t2.small	2/2 checks passed View alarms +		ap-south-1a	ec2-13-233-179-8		
<input type="checkbox"/>	Master-k8s	i-0c7cf83d19bb27c8	Running View details Logs	t2.medium	2/2 checks passed View alarms +		ap-south-1a	ec2-13-126-101-1		

AMI - Ubuntu Server 22.04

Instance type- t2.medium (master), t2.small (Node1)

Security group Inbound rules are following

Inbound rules (13)							
<input type="checkbox"/> Search							
	Name	Security group rule...	IP version	Type	Protocol	Port range	Source
<input type="checkbox"/>	-	sgr-07cd65e446dbd1fd3	IPv4	SSH	TCP	22	0.0.0.0/0
<input type="checkbox"/>	-	sgr-0fd4ee1cffda0a67b	IPv4	HTTPS	TCP	443	0.0.0.0/0
<input type="checkbox"/>	-	sgr-02d4e07b5fdb815f8	IPv4	Custom TCP	TCP	30000 - 32767	0.0.0.0/0
<input type="checkbox"/>	-	sgr-0b29ba9879507afb4	IPv4	Custom TCP	TCP	2000 - 11000	0.0.0.0/0
<input type="checkbox"/>	-	sgr-03b46c4b268c4c668	IPv4	Custom TCP	TCP	6443	0.0.0.0/0
<input type="checkbox"/>	-	sgr-0cb5567efeeb68f62	IPv4	Custom TCP	TCP	587	0.0.0.0/0
<input type="checkbox"/>	-	sgr-01571f4fc9e94ffb5	IPv4	HTTP	TCP	80	0.0.0.0/0
<input type="checkbox"/>	-	sgr-0cad9146ec237b38c	IPv4	Custom TCP	TCP	2380	0.0.0.0/0
<input type="checkbox"/>	-	sgr-09f8e2752d09caf5	IPv4	Custom TCP	TCP	2379	0.0.0.0/0
<input type="checkbox"/>	-	sgr-0e33f1ed71c2ede92	IPv4	Custom TCP	TCP	10250 - 10252	0.0.0.0/0
<input type="checkbox"/>	-	sgr-05f8fe2f2c0b49166	IPv4	Custom TCP	TCP	179	0.0.0.0/0
<input type="checkbox"/>	-	sgr-0904df7c1dbe2d274	IPv4	SMTP	TCP	25	0.0.0.0/0
<input type="checkbox"/>	-	sgr-0c91ab23e1b10eaed	IPv4	SMTPS	TCP	465	0.0.0.0/0

Now install the Kubernetes cluster by following this instruction

<https://kanzal.com/kubernetes-with-kubeadm-and-calico/>

```
[bootstrapping-token] Configured RBAC rules to allow Node Bootstrap tokens to post CSRs in order for nodes to get long term cert
[bootstrapping-token] Configured RBAC rules to allow the csrapprover controller automatically approve CSRs from a Node Bootstrap
[bootstrapping-token] Configured RBAC rules to allow certificate rotation for all node client certificates in the cluster
[bootstrapping-token] Creating the "cluster-info" ConfigMap in the "kube-public" namespace
[kubelet-finalize] Updating "/etc/kubernetes/kubelet.conf" to point to a rotatable kubelet client certificate and key
[addons] Applied essential addon: CoreDNS
[addons] Applied essential addon: kube-proxy

Your Kubernetes control-plane has initialized successfully!

To start using your cluster, you need to run the following as a regular user:

mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config

Alternatively, if you are the root user, you can run:

export KUBECONFIG=/etc/kubernetes/admin.conf

You should now deploy a pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
https://kubernetes.io/docs/concepts/cluster-administration/addons/

You can now join any number of control-plane nodes by copying certificate authorities
and service account keys on each node and then running the following as root:

kubeadm join 172.0.5.120:6443 --token tawutv.twec9t6nmq4d01q2 \
    --discovery-token-ca-cert-hash sha256:10c1fc1b7425fcf889def2eb8226a14fff4397501d0aa42892f5d7b4e1d73cef \
    --control-plane

Then you can join any number of worker nodes by running the following on each as root:

kubeadm join 172.0.5.120:6443 --token tawutv.twec9t6nmq4d01q2 \
    --discovery-token-ca-cert-hash sha256:10c1fc1b7425fcf889def2eb8226a14fff4397501d0aa42892f5d7b4e1d73cef
root@ip-172-0-5-120:/home/ubuntu# mkdir -p $HOME/.kube
root@ip-172-0-5-120:/home/ubuntu# sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
```

```
customresourcedefinition.apiextensions.k8s.io/networkpolicies.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/networksets.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/apiservers.operator.tigera.io created
customresourcedefinition.apiextensions.k8s.io/imagesets.operator.tigera.io created
customresourcedefinition.apiextensions.k8s.io/installations.operator.tigera.io created
customresourcedefinition.apiextensions.k8s.io/tigerastatuses.operator.tigera.io created
serviceaccount/tigera-operator created
clusterrole.rbac.authorization.k8s.io/tigera-operator created
clusterrolebinding.rbac.authorization.k8s.io/tigera-operator created
deployment.apps/tigera-operator created
root@ip-172-0-5-120:/home/ubuntu# curl https://raw.githubusercontent.com/projectcalico/calico/v3.28.1/manifests/custom-resources.yaml
% Total    % Received % Xferd  Average Speed   Time   Time     Current
          Dload  Upload Total Spent   Left Speed
100  777 100  777  0     0  2200      0 ---:--- ---:--- 2207
root@ip-172-0-5-120:/home/ubuntu# ls
custom-resources.yaml
root@ip-172-0-5-120:/home/ubuntu# vi custom-resources.yaml
root@ip-172-0-5-120:/home/ubuntu# kubectl create -f custom-resources.yaml
installation.operator.tigera.io/default created
apiserver.operator.tigera.io/default created
root@ip-172-0-5-120:/home/ubuntu# kubectl get pods -A
NAMESPACE     NAME                               READY   STATUS    RESTARTS   AGE
calico-system  calico-kube-controllers-79fcbd845f-sbg5v  0/1     Pending   0          9s
calico-system  calico-node-tpwsq                0/1     Init:0/2  0          9s
calico-system  calico-typha-698bc666b4-knfh7    1/1     Running   0          9s
calico-system  csi-node-driver-g2hb7              0/2     ContainerCreating  0          9s
kube-system    coredns-55ccb58b774-7xhw8        0/1     Pending   0          3m2s
kube-system    coredns-55ccb58b774-mjcb1        0/1     Pending   0          3m2s
kube-system    etcd-ip-172-0-5-120                 1/1     Running   1          3m17s
kube-system    kube-apiserver-ip-172-0-5-120       1/1     Running   1          3m17s
kube-system    kube-controller-manager-ip-172-0-5-120 1/1     Running   1          3m17s
kube-system    kube-proxy-grfjqn                  1/1     Running   0          3m2s
kube-system    kube-scheduler-ip-172-0-5-120       1/1     Running   1          3m17s
tigera-operator tigera-operator-77f994b5bb-ndcgw   1/1     Running   0          102s
```

An overlay network is created for the Kubernetes cluster

```

inet 127.0.0.1/8 scope host lo
    valid_lft forever preferred_lft forever
inet6 ::1/128 scope host
    valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9001 qdisc mq state UP group default qlen 1000
    link/ether 02:5a:61:f1:8a:f1 brd ff:ff:ff:ff:ff:ff
        inet 172.0.5.120/20 metric 100 brd 172.0.15.255 scope global dynamic eth0
            valid_lft 2442sec preferred_lft 2442sec
        inet6 fe80::5a:61ff:fe1:8af1/64 scope link
            valid_lft forever preferred_lft forever
3: vxlan.calico: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 8951 qdisc noqueue state UNKNOWN group default qlen 1000
    link/ether 66:10:96:62:75:18 brd ff:ff:ff:ff:ff:ff
        inet 10.244.58.64/32 scope global vxlan.calico
            valid_lft forever preferred_lft forever
        inet6 fe80::6410:96ff:fe62:7518/64 scope link
            valid_lft forever preferred_lft forever
6: cali4b28c574c78@if2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 8951 qdisc noqueue state UP group default qlen 1000
    link/ether ee:ee:ee:ee:ee:ee brd ff:ff:ff:ff:ff:ff link-netns cni-09675323-9c9c-0384-2939-8e15a7a58197
        inet6 fe80::eece:efff:feee:eeee/64 scope link
            valid_lft forever preferred_lft forever
7: caliaeef79b8747@if2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 8951 qdisc noqueue state UP group default qlen 1000
    link/ether ee:ee:ee:ee:ee:ee brd ff:ff:ff:ff:ff:ff link-netns cni-49295948-be6f-22d4-5f83-15d1460a2456
        inet6 fe80::ecee:efff:feee:eeee/64 scope link
            valid_lft forever preferred_lft forever
8: cali95113848b76@if2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 8951 qdisc noqueue state UP group default qlen 1000
    link/ether ee:ee:ee:ee:ee:ee brd ff:ff:ff:ff:ff:ff link-netns cni-0ed44498-2af3-53bf-ae75-6fe9ab24eee7
        inet6 fe80::ecee:efff:feee:eeee/64 scope link
            valid_lft forever preferred_lft forever
9: califf6e6afa650@if2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 8951 qdisc noqueue state UP group default qlen 1000
    link/ether ee:ee:ee:ee:ee:ee brd ff:ff:ff:ff:ff:ff link-netns cni-1cfa2a67-4d89-11f1-e681-85927e23b429
        inet6 fe80::ecee:efff:feee:eeee/64 scope link
            valid_lft forever preferred_lft forever
10: cali7ff9d12d9ff5@if2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 8951 qdisc noqueue state UP group default qlen 1000
    link/ether ee:ee:ee:ee:ee:ee brd ff:ff:ff:ff:ff:ff link-netns cni-8c998cff-129c-624e-8de3-d7830b3f5fe1
        inet6 fe80::ecee:efff:feee:eeee/64 scope link
            valid_lft forever preferred_lft forever
11: cali977e3d04d3@if2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 8951 qdisc noqueue state UP group default qlen 1000
    link/ether ee:ee:ee:ee:ee:ee brd ff:ff:ff:ff:ff:ff link-netns cni-b97c8be6-c134-d63d-de2d-7c09d6ecc0f4
        inet6 fe80::ecee:efff:feee:eeee/64 scope link
            valid_lft forever preferred_lft forever

```

The node1 has joined the cluster

```

root@ip-172-0-4-8:/home/ubuntu# kubeadm join 172.0.5.120:6443 --token j55zuf.0i931xpy6aljr5v6 --discovery-token-ca-cert-hash sha256:10c1fc1b7425fcf889def2eb822
6a14fff4397501d0aa42892f5d7b4e1d73cef
[preflight] Running pre-flight checks
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o yaml'
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-check] Waiting for a healthy kubelet at http://127.0.0.1:10248/healthz. This can take up to 4m0s
[kubelet-check] The kubelet is healthy after 501.628479ms
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap

This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was received.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the control-plane to see this node join the cluster.

root@ip-172-0-4-8:/home/ubuntu#

```

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-stkhk   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:~# 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:~# kubectl get pods
NAME                               READY   STATUS    RESTARTS   AGE
techappdeployment-549f54cf78-9w6dz 1/1     Running   0          46s
techappdeployment-549f54cf78-htt6v  1/1     Running   0          46s
techappdeployment-549f54cf78-pzwh5 1/1     Running   0          46s
root@k8s-server:~# 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:~# 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:~#

```

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-md581 -- /bin/bash
```

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

```
$cp -R * ../webapps
```

```

root@techappdeployment-549f54cf78-md581:/usr/local/tomcat/webapps.dist# cp -R * ../webapps
root@techappdeployment-549f54cf78-md581:/usr/local/tomcat/webapps.dist# cd ..
root@techappdeployment-549f54cf78-md581:/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-md581:/usr/local/tomcat# cd webapps
root@techappdeployment-549f54cf78-md581:/usr/local/tomcat/webapps# ls
docs examples host-manager manager ROOT techapp techapp.war
root@techappdeployment-549f54cf78-md581:/usr/local/tomcat/webapps# exit
exit
root@ip-172-0-5-120:~/home/ubuntu# ls

```

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-typa-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-htt6	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

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

```

Master-k8s
Terminal Sessions View Xserver Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help
Quid 5. Master-k8s 4. Node-1-k8s
techappdeployment-549f54cf78-pxrmj 1/1 Running 1 (11m ago) 22h
root@k8s-server:/home/ubuntu# kubectl get svc
NAME           TYPE      CLUSTER-IP   EXTERNAL-IP  PORT(S)   AGE
grafana        ClusterIP 10.106.255.56 <none>       80/TCP     20h
grafana-ext    NodePort   10.100.206.63 <none>       80:32085/TCP 20h
kubernetes     ClusterIP 10.96.0.1    <none>       443/TCP    5d21h
kubernetes-api LoadBalancer 10.110.107.20 <pending>   6443:32723/TCP 3d19h
prometheus-alertmanager ClusterIP 10.101.74.151 <none>       9093/TCP   21h
prometheus-alertmanager-headless ClusterIP None <none>       9093/TCP   21h
prometheus-kube-state-metrics ClusterIP 10.110.73.184 <none>       8080/TCP   21h
prometheus-prometheus-node-exporter ClusterIP 10.109.37.55 <none>       9100/TCP   21h
prometheus-prometheus-pushgateway ClusterIP 10.111.202.83 <none>       9091/TCP   21h
prometheus-server ClusterIP 10.108.22.96 <none>       80/TCP     21h
prometheus-server-ext NodePort 10.98.225.225 <none>       80:30331/TCP 21h
techappdeployment NodePort 10.96.4.168 <none>       80:32427/TCP 4d18h
root@k8s-server:/home/ubuntu# kubectl get ep

```

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:/home/ubuntu# cd /
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 AAAAC3NzaC1lZDI1NTE5AAAI0TVV1eGHC1UHFC07SPHFyPABZs6dUQjnCbRhvir0eP2 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-r--r-- 1 root root 20 Sep 19 10:13 .lesshist
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 pub known_hosts known_hosts.old
root@k8s-server:~/ssh# cat authorized_keys
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAI0TVV1eGHC1UHFC07SPHFyPABZs6dUQjnCbRhvir0eP2 root@ip-172-0-9-208

ssh-rsa AAAAB3NzaC1yc2EAAAQABAAQDe2BvG13Z0KXFbqcw/6vkY45mRSDwKn09hrkIgwtKxTsD4quHbRLr0vASDX0xDr7E
1HZYd6jjjLzqYGuzqTkf0GEkmWUiTI8vplkU5BFxywKn4p4Eco3LCq+9SLRQCAoe50WxIaW/8V3o1Hpl0G1D+pMXTvxIxGfhWn5k/j
zjKRckTLzkGS0+2gG07w5S4jIYg47Xt0VxH0qmaPiqPvIG/WotJxwIq8vw543ISsIW78oEbbGFNvqMhX90FNRPKwYo9r/JLRGl/XcA
svkobQ0L1J0MjeYmdfwHlyLUCb8iVccdxPiwLhDIR3Lkx4Zn0tIQdGoWAK1zDnV1m0Lqf9aLxc= root@ip-172-0-5-120
root@k8s-server:~/ssh#
```

```

root@ansible-server:~/ssh# ssh 172.0.5.120
Welcome to Ubuntu 22.04.4 LTS (GNU/Linux 6.8.0-1015-aws x86_64)

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

System information as of Mon Sep 23 07:58:56 UTC 2024

System load: 0.2              Processes:          136
Usage of /: 23.9% of 28.89GB  Users logged in:    1
Memory usage: 17%             IPv4 address for eth0: 172.0.5.120
Swap usage: 0%               

* Ubuntu Pro delivers the most comprehensive open source security and
  compliance features.

https://ubuntu.com/aws/pro

Expanded Security Maintenance for Applications is not enabled

```

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.

```
root@ansible-server:~/ansible-kubernetes# ls
ansible.cfg  dev  k8s  playbooks
root@ansible-server:~/ansible-kubernetes# tree
.
├── ansible.cfg
├── dev
└── k8s
    ├── deployment.yaml
    └── service.yaml
└── playbooks
    └── kubernetes.yaml

3 directories, 5 files
root@ansible-server:~/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:

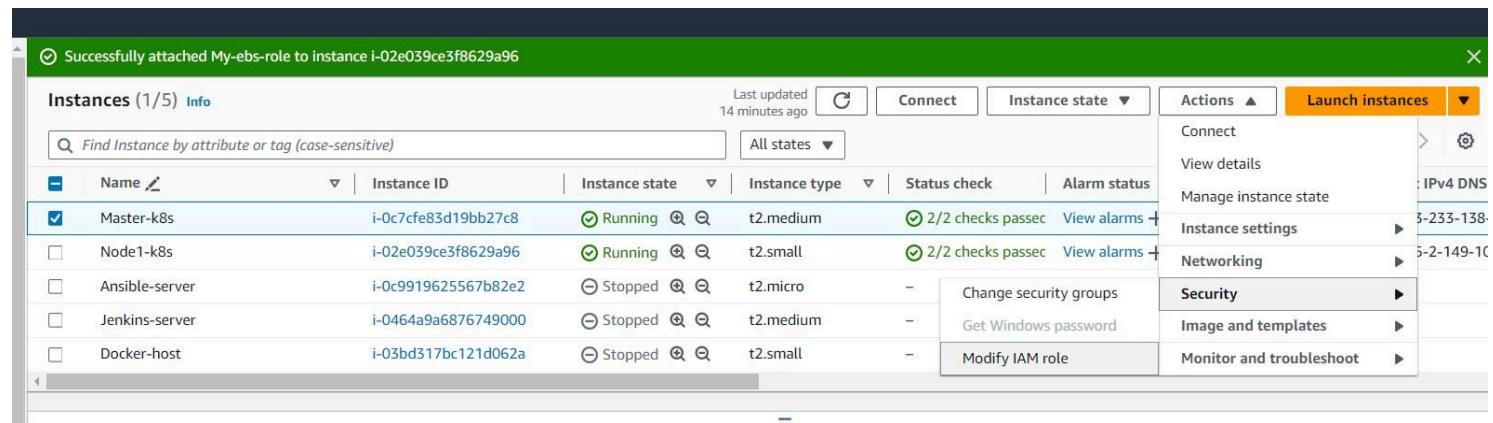
Installed the Helm repo by following this link

<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.



My-ebs-role [Info](#)

Allows EC2 instances to call AWS services on your behalf.

Summary

Creation date September 23, 2024, 14:47 (UTC+05:00)	ARN arn:aws:iam::975050160462:role/My-ebs-role	Instance profile ARN arn:aws:iam::975050160462:instance-profile/My-ebs-role
Last activity -	Maximum session duration 1 hour	

Permissions [Trust relationships](#) [Tags](#) [Access Advisor](#) [Revoke sessions](#)

Permissions policies (1) [Info](#)

You can attach up to 10 managed policies.

Policy name	Type	Attached entities
AmazonEBSCSIDDriverPolicy	AWS managed	1

Permissions boundary (not set)

Instances (1/2) [Info](#)

Last updated less than a minute ago

Name	Instance ID	Instance state	Instance type	Status check
<input checked="" type="checkbox"/> Master-k8s	i-0c7cf83d19bb27c8	Running Details Logs	t2.medium	2/2 checks passed
<input type="checkbox"/> Node1-k8s	i-02e039ce3f8629a96	Running Details Logs	t2.small	2/2 checks passed

i-0c7cf83d19bb27c8 (Master-k8s)

[Details](#) [Status and alarms](#) [Monitoring](#) [Security](#) [Networking](#) [Storage](#) [Tags](#)

▼ Security details

IAM Role My-ebs-role	Owner ID 975050160462
Security groups	

Now open `/etc/kubernetes/manifests/kube-apiserver.yaml` file and add following 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=prometheus,app.kubernetes.io/name=prometheus,app.kubernetes.io/instance=prometheus")  
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=alertmanager,app.kubernetes.io/name=alertmanager,app.kubernetes.io/instance=alertmanager")  
kubectl --namespace default port-forward $POD_NAME 9093
```

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=alertmanager,app.kubernetes.io/name=alertmanager,app.kubernetes.io/instance=alertmanager")  
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=".spec.containers[0].image")  
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:

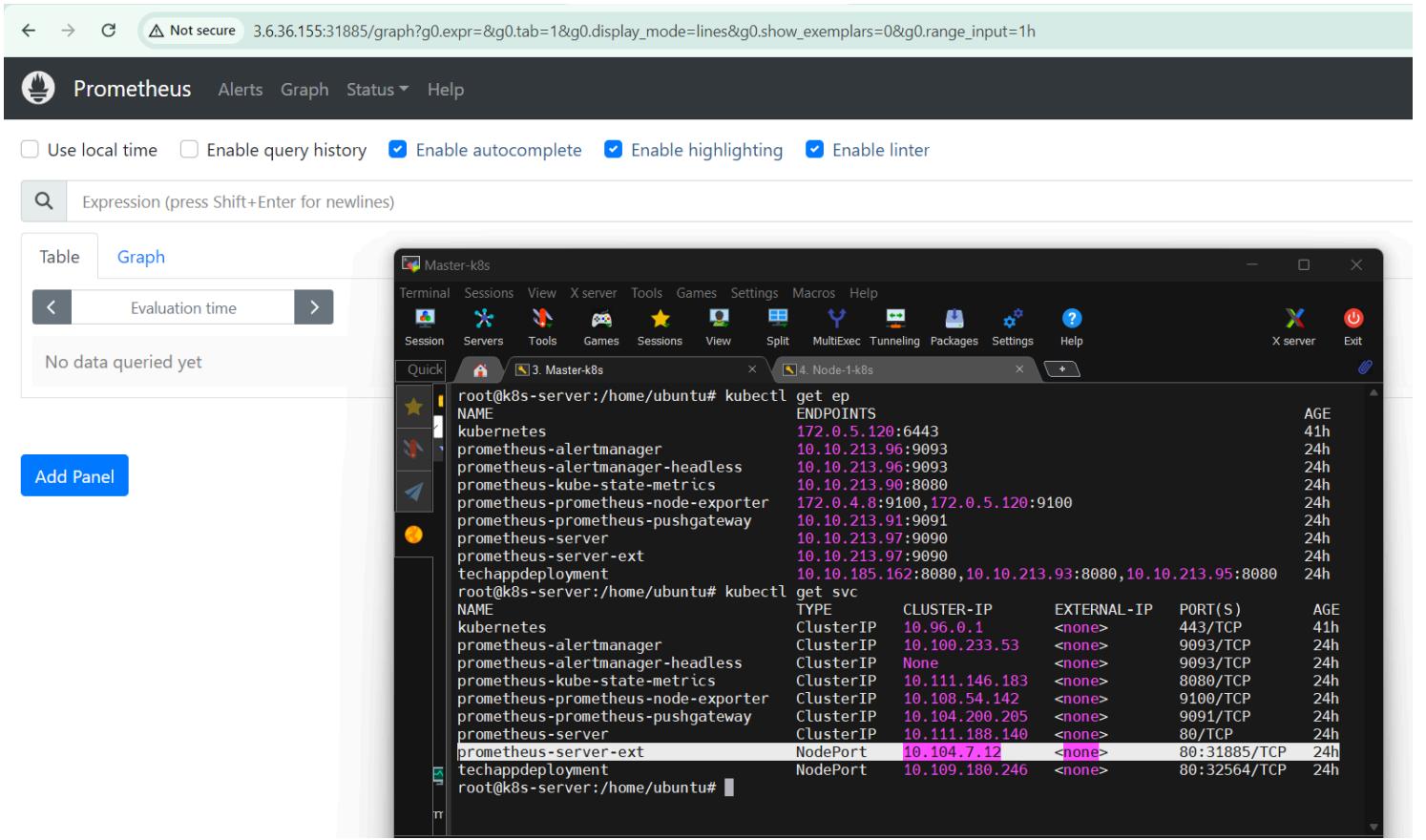
```
kubectl expose service prometheus-server --type=NodePort --target-port=9090
--name=prometheus-server-ext
```

```

root@k8s-server:/home/ubuntu# kubectl get pods
NAME                                         NODEPORT   10.109.180.240   <none>           80:32564/TCP   24h
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#
```

← → ⌂ ⚠ Not secure 13.235.132.54:9200

Node Exporter

[Metrics](#)

← → ⌂ ⚠ Not secure 13.235.132.54:9200/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 started.
# 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']
```

```
root@k8s-server:~# vi values.yaml  
root@k8s-server:~# cat values.yaml  
serverFiles:  
  prometheus.yml:  
    global:  
      scrape_interval: 15s  
    scrape_configs:  
      - job_name: 'node_exporter'  
        static_configs:  
          - targets: ['k8s-server:9200','172.0.4.8:9200']  
root@k8s-server:~#
```

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

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

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://localhost:9090/metrics	UP	Instance="localhost:9090", job="prometheus"	12.53s ago	4.569ms	none

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
```

```
root@k8s-server:~# helm repo add grafana https://grafana.github.io/helm-charts
"grafana" has been added to your repositories
root@k8s-server:~# helm repo update
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "aws-ebs-csi-driver" chart repository
...Successfully got an update from the "grafana" chart repository
...Successfully got an update from the "prometheus-community" chart repository
Update Complete. *Happy Helming!*
root@k8s-server:~# ls
ansible-kubernetes get_helm.sh sc.yaml snap
root@k8s-server:~# helm install grafana grafana/grafana
NAME: grafana
LAST DEPLOYED: Mon Sep 23 10:52:18 2024
NAMESPACE: default
STATUS: deployed
REVISION: 1
NOTES:
1. Get your 'admin' user password by running:

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

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

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.io/name=grafana" --field-selector status.phase=Running | grep grafana | awk '{print $1}')  
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-htt6       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=grafana  
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 Grafana dashboard login.

3LCJzOMKDfGRT09V69Wrm1wQnulArk6Dsk9mhLXh

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

Welcome to Grafana

Need help? Documentation Tutorials

Basic TUTORIAL DATA SOURCES

Master-k8s

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
grafana	ClusterIP	10.106.255.56	<none>	80/TCP	21h
grafana-ext	NodePort	10.100.206.63	<none>	80:32085/TCP	21h
kubernetes	ClusterIP	10.96.0.1	<none>	443/TCP	5d21h
kubernetes-apiservice	LoadBalancer	10.110.107.20	<pending>	6443:32723/TCP	3d20h
prometheus-alertmanager	ClusterIP	10.101.74.151	<none>	9093/TCP	21h
prometheus-alertmanager-headless	ClusterIP	None	<none>	9093/TCP	21h
prometheus-kube-state-metrics	ClusterIP	10.110.73.184	<none>	8080/TCP	21h
prometheus-prometheus-node-exporter	ClusterIP	10.109.37.55	<none>	9100/TCP	21h
prometheus-prometheus-pushgateway	ClusterIP	10.111.202.83	<none>	9091/TCP	21h
prometheus-server	ClusterIP	10.108.22.96	<none>	80/TCP	21h

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

Home > Connections > Data sources > prometheus

Type: Prometheus

Settings Dashboards

Name: prometheus Default:

Before you can use the Prometheus data source, you must configure it below or in the config file. For detailed instructions, [view t](#)

Fields marked with * are required

Connection

Prometheus server URL *

The screenshot shows the Grafana Alerting interface. On the left, there's a sidebar with options: Connections, Add new connection, Data sources (which is selected and highlighted in orange), and Administration. The main area has a "Custom query parameters" input field with the placeholder "Example: max_source_resolution=5m&timeout=10s" and an "HTTP method" dropdown set to POST. Below this is a section titled "Exemplars" with a "+ Add" button. A green success message box contains the text "Successfully queried the Prometheus API." and "Next, you can start to visualize data by [building a dashboard](#), or by querying data in the [Explore view](#)." At the bottom are two buttons: "Delete" and "Save & test".

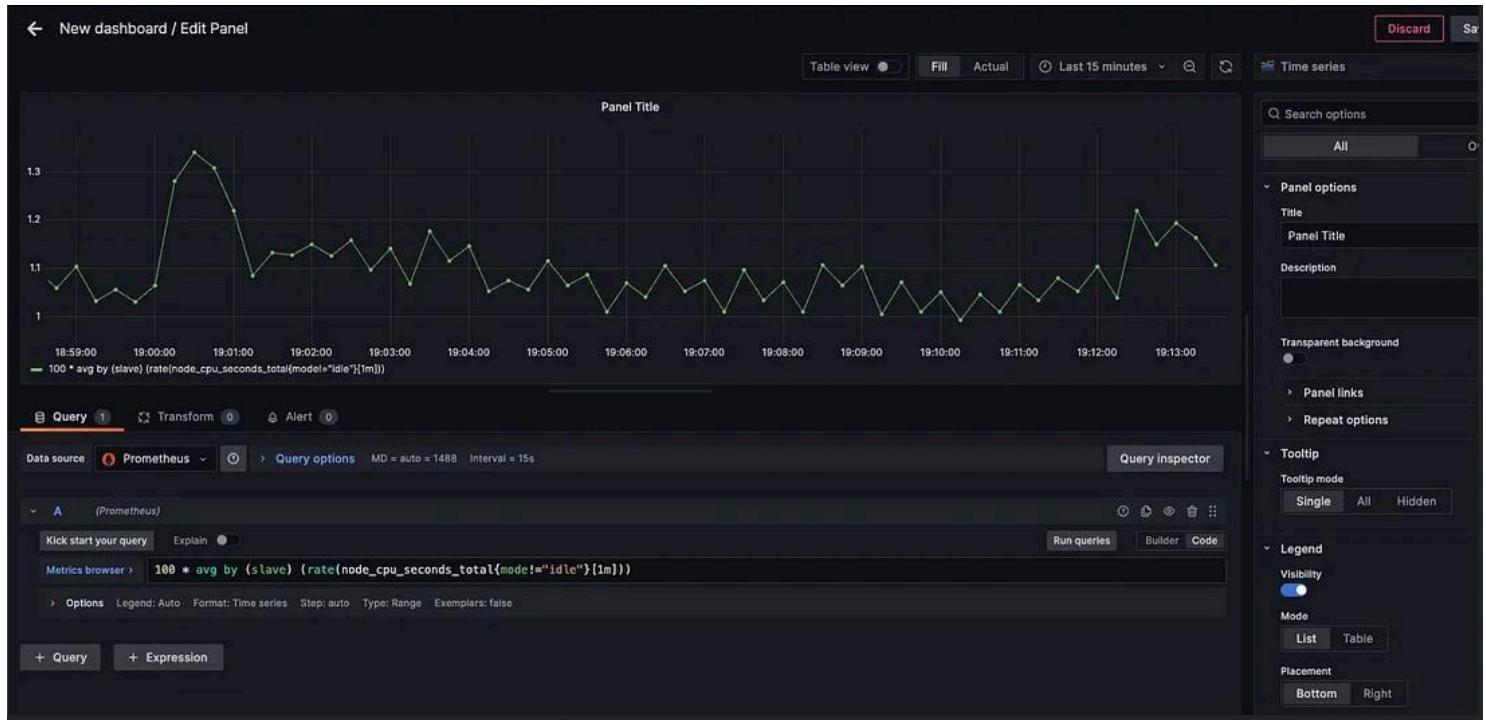
Now add the dashboard in prometheus. click on dashboard, Import dashboard,

Monitor the metrics at the Grafana dashboard:

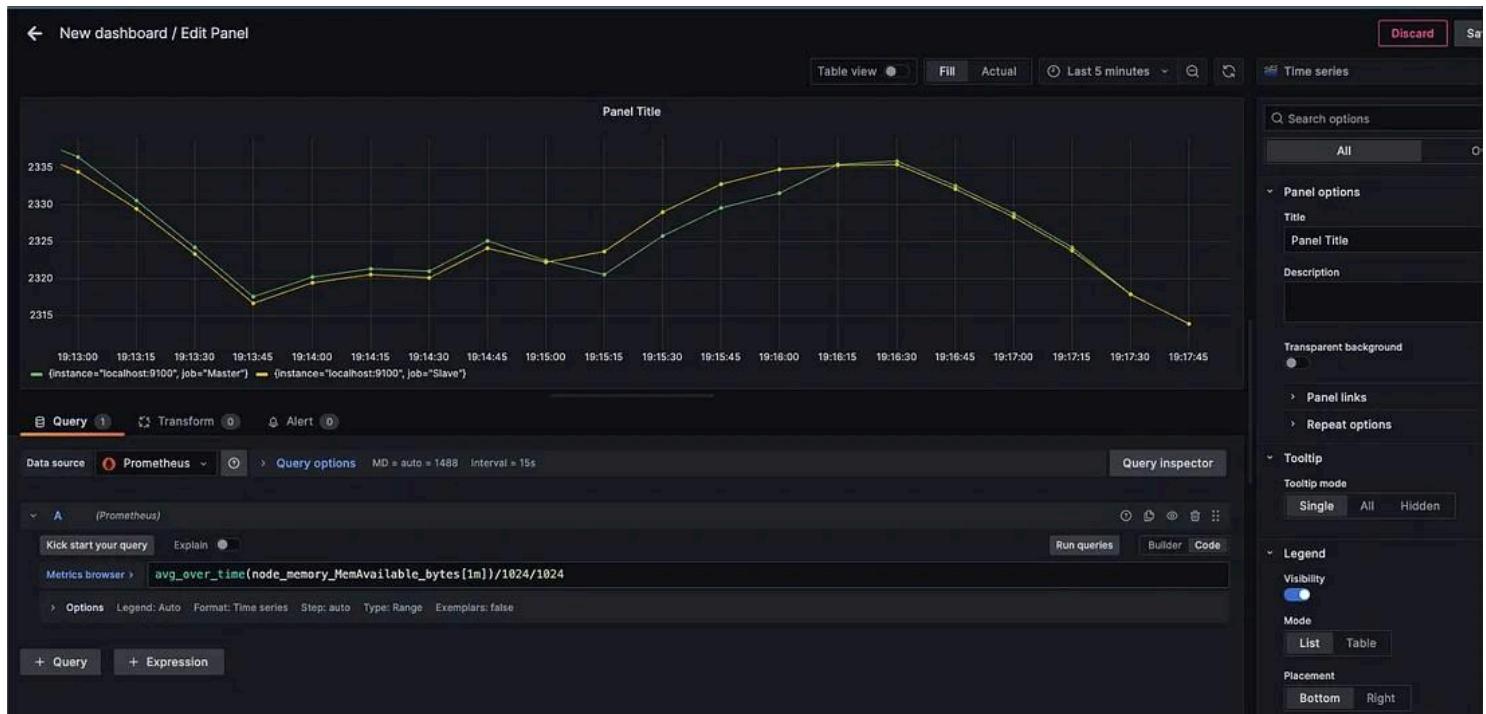
Below are the metrics captured in grafana dashboard

For CPU

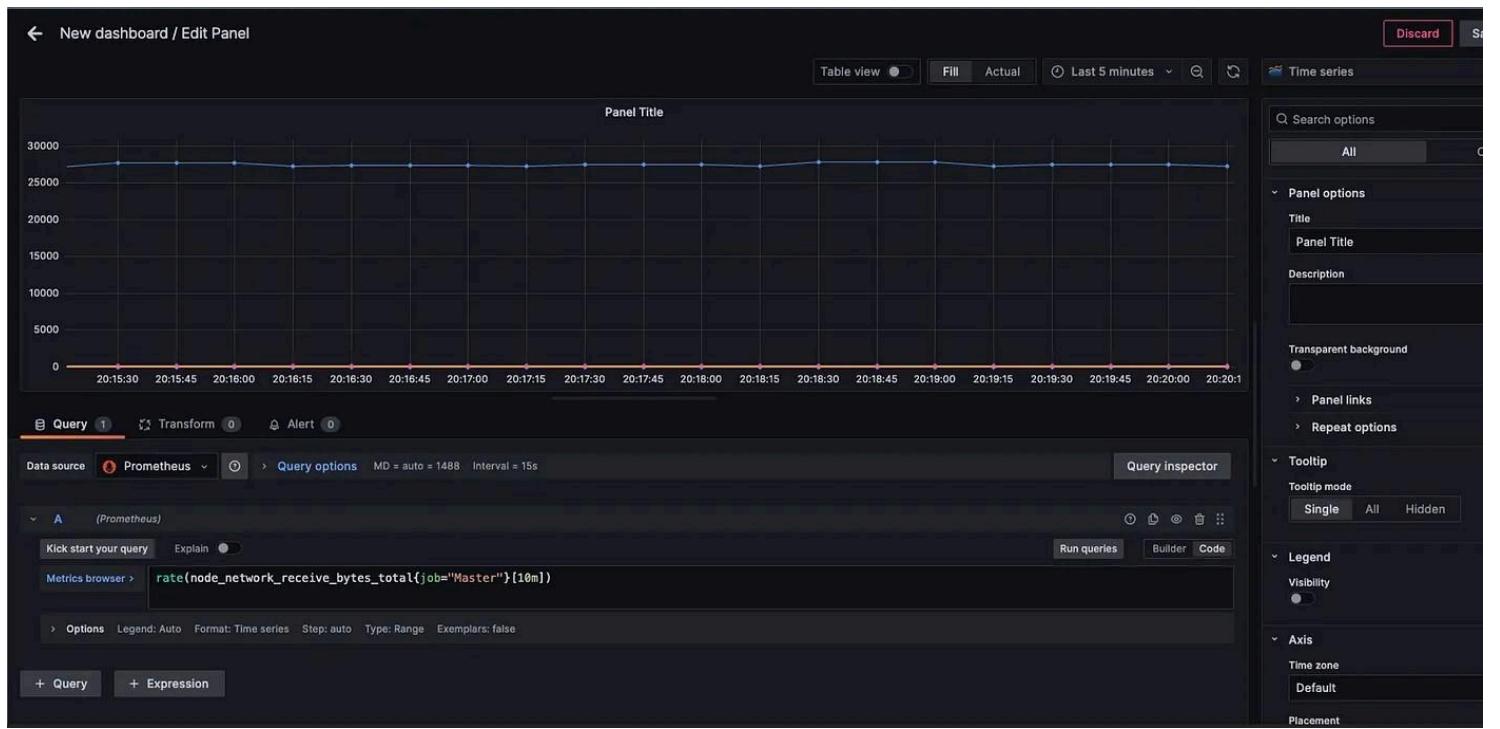


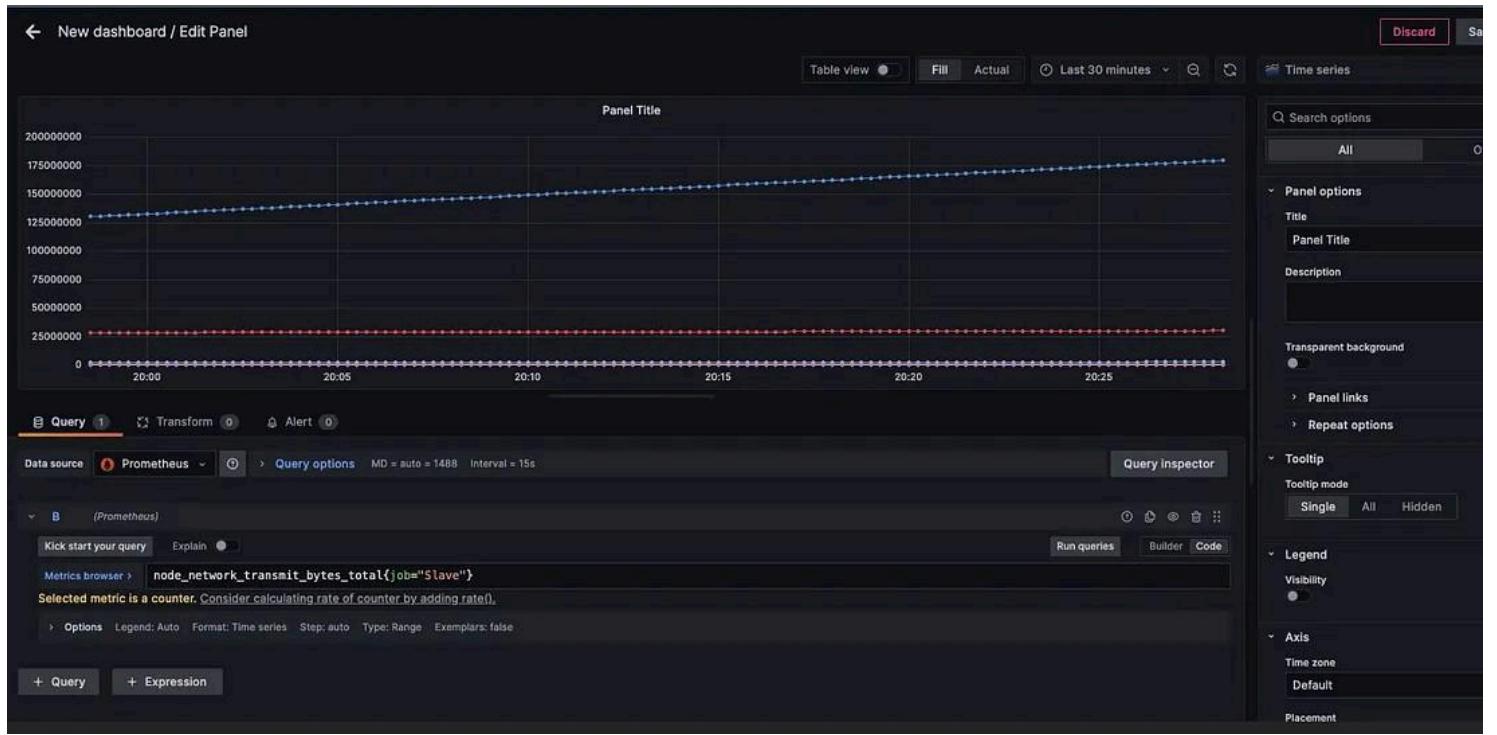


For Memory Usage



For Network





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

