



INDIAN INSTITUTE OF  
INFORMATION  
TECHNOLOGY

CS457

# DevOps Final Assignment -1

Submitted to:

Dr. Uma S

Submitted by:

S Namratha  
(18BCS083)

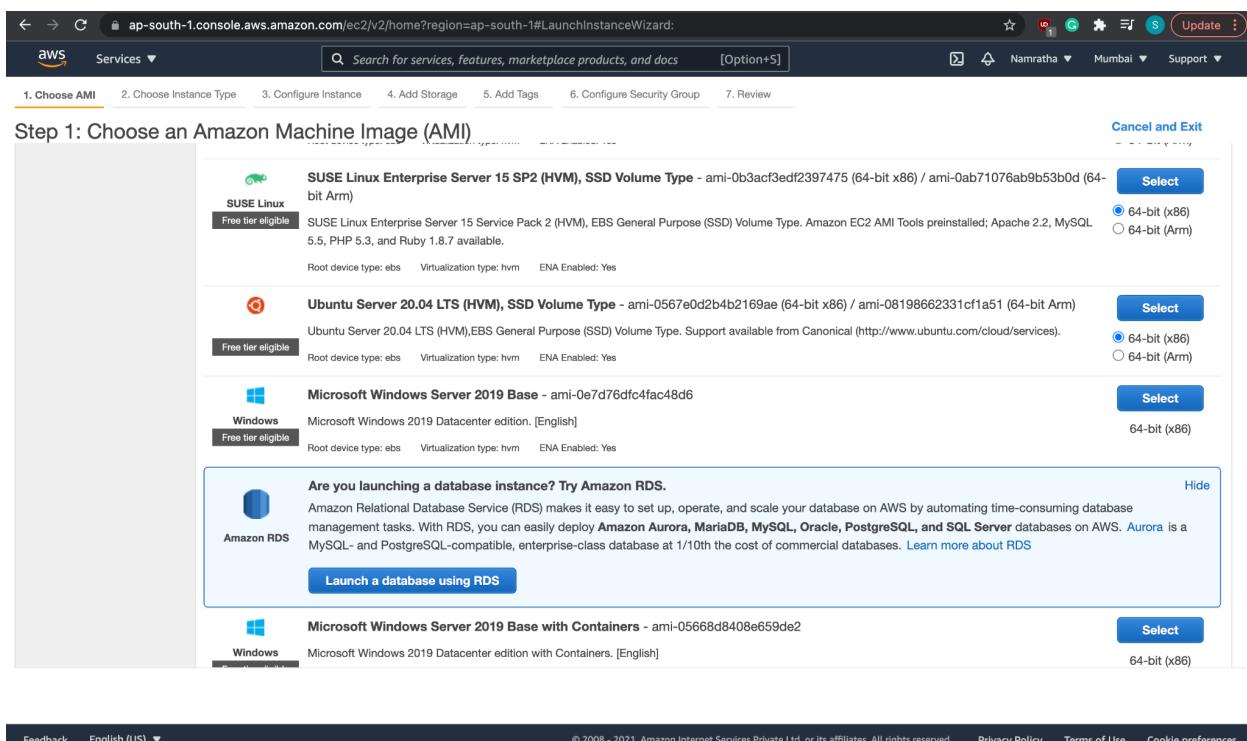
# DevOps Assignment 1

Set up complete CI/CD jenkins pipeline for kubernetes (AWS)

Note- Github repo containing files of this assignment :  
<https://github.com/Namratha-shivaraju/k8s-jenkins-aws>

## 1. Set up and connect an AWS EC2 instance

Select Ubuntu server



The screenshot shows the AWS Launch Instance Wizard Step 1: Choose an Amazon Machine Image (AMI). The page has a header with tabs: 1. Choose AMI, 2. Choose Instance Type, 3. Configure Instance, 4. Add Storage, 5. Add Tags, 6. Configure Security Group, and 7. Review. A search bar at the top says "Search for services, features, marketplace products, and docs". On the left, there's a sidebar with "Services" and "Namratha Mumbai Support". The main content area is titled "Step 1: Choose an Amazon Machine Image (AMI)". It lists several AMI options:

- SUSE Linux Enterprise Server 15 SP2 (HVM), SSD Volume Type** - ami-0b3acf3edf2397475 (64-bit x86) / ami-0ab71076ab9b53b0d (64-bit Arm)  
Free tier eligible  
Select button  
Radio buttons: 64-bit (x86) (selected), 64-bit (Arm)  
Details: Root device type: ebs, Virtualization type: hvm, ENA Enabled: Yes
- Ubuntu Server 20.04 LTS (HVM), SSD Volume Type** - ami-0567e0d2b4b2169ae (64-bit x86) / ami-08198662331cf1a51 (64-bit Arm)  
Free tier eligible  
Select button  
Radio buttons: 64-bit (x86) (selected), 64-bit (Arm)  
Details: Root device type: ebs, Virtualization type: hvm, ENA Enabled: Yes
- Microsoft Windows Server 2019 Base** - ami-0e7d76dfc4fac48d6  
Windows  
Free tier eligible  
Select button  
Radio button: 64-bit (x86)  
Details: Microsoft Windows 2019 Datacenter edition, [English]  
Root device type: ebs, Virtualization type: hvm, ENA Enabled: Yes
- Amazon RDS**  
Are you launching a database instance? Try Amazon RDS.  
Amazon Relational Database Service (RDS) makes it easy to set up, operate, and scale your database on AWS by automating time-consuming database management tasks. With RDS, you can easily deploy Amazon Aurora, MariaDB, MySQL, Oracle, PostgreSQL, and SQL Server databases on AWS. Aurora is a MySQL- and PostgreSQL-compatible, enterprise-class database at 1/10th the cost of commercial databases. [Learn more about RDS](#)  
Launch a database using RDS button  
Hide button
- Microsoft Windows Server 2019 Base with Containers** - ami-05668d8408e659de2  
Windows  
Free tier eligible  
Select button  
Radio button: 64-bit (x86)  
Details: Microsoft Windows 2019 Datacenter edition with Containers, [English]

At the bottom, there are links for Feedback, English (US), © 2008 - 2021, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved., Privacy Policy, Terms of Use, and Cookie preferences.

**Step 2: Choose an Instance Type**

Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They have varying combinations of CPU, memory, storage, and networking capacity, and give you the flexibility to choose the appropriate mix of resources for your applications. [Learn more](#) about instance types and how they can meet your computing needs.

Filter by: All instance families ▾ Current generation ▾ Show/Hide Columns

Currently selected: t2.medium (~ ECUs, 2 vCPUs, 2.3 GHz, ~ 4 GiB memory, EBS only)

	Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance	IPv6 Support
	t2	t2.nano	1	0.5	EBS only	-	Low to Moderate	Yes
	t2	t2.micro <small>Free tier eligible</small>	1	1	EBS only	-	Low to Moderate	Yes
	t2	t2.small	1	2	EBS only	-	Low to Moderate	Yes
<input checked="" type="checkbox"/>	t2	<b>t2.medium</b>	2	4	<b>EBS only</b>	-	<b>Low to Moderate</b>	<b>Yes</b>
	t2	t2.large	2	8	EBS only	-	Low to Moderate	Yes
	t2	t2.xlarge	4	16	EBS only	-	Moderate	Yes
	t2	t2.2xlarge	8	32	EBS only	-	Moderate	Yes
	t3	t3.nano	2	0.5	EBS only	Yes	Up to 5 Gigabit	Yes
	t3	t3.micro	2	1	EBS only	Yes	Up to 5 Gigabit	Yes
	t3	t3.small	2	2	EBS only	Yes	Up to 5 Gigabit	Yes

Cancel Previous Review and Launch Next: Configure Instance Details

**Step 6: Configure Security Group**

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a web server and allow Internet traffic to reach your instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select from an existing one below. [Learn more](#) about Amazon EC2 security groups.

Assign a security group:  Create a new security group  Select an existing security group

Security group name: launch-wizard-2

Description: launch-wizard-2 created 2021-11-11T10:38:10.696+05:30

Type	Protocol	Port Range	Source	Description
SSH	TCP	22	Custom 0.0.0.0/0	e.g. SSH for Admin Desktop
Custom TCP F	TCP	8080	Custom 0.0.0.0/0	e.g. SSH for Admin Desktop

Add Rule

**⚠ Warning**  
Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

Cancel Previous Review and Launch

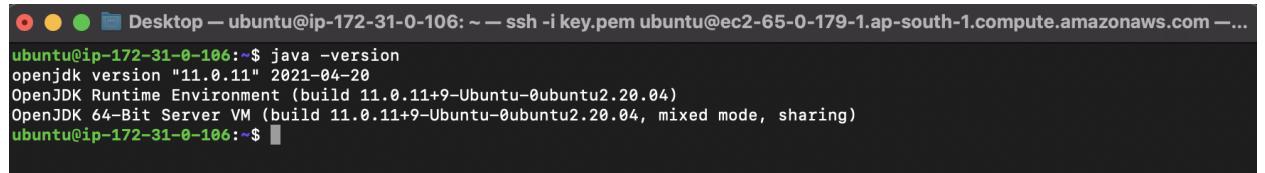
## 2. Install JDK on AWS EC2 Instance

Run the below commands to install JDK

```
sudo apt-get update
```

```
sudo apt install openjdk-11-jre-headless
```

```
java -version
```



A screenshot of a terminal window titled "Desktop — ubuntu@ip-172-31-0-106: ~ — ssh -i key.pem ubuntu@ec2-65-0-179-1.ap-south-1.compute.amazonaws.com —...". The terminal shows the command "java -version" being run, which outputs:

```
ubuntu@ip-172-31-0-106:~$ java -version
openjdk version "11.0.11" 2021-04-20
OpenJDK Runtime Environment (build 11.0.11+9-Ubuntu-0ubuntu2.20.04)
OpenJDK 64-Bit Server VM (build 11.0.11+9-Ubuntu-0ubuntu2.20.04, mixed mode, sharing)
ubuntu@ip-172-31-0-106:~$
```

### 3. Install and Setup Jenkins

Run the below commands to install Jenkins

```
wget -q -O -
https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo
apt-key add -
```

```
sudo sh -c 'echo deb https://pkg.jenkins.io/debian-stable
binary/ > /etc/apt/sources.list.d/jenkins.list'
```

```
sudo apt-get update
```

```
sudo apt-get install jenkins
```

```
sudo service jenkins status
```

```

Desktop — ubuntu@ip-172-31-0-106: ~ — ssh -i key.pem ubuntu@ec2-65-0-179-1.ap-south-1.compute.amazonaws.com —...
ubuntu@ip-172-31-0-106:~$ wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo apt-key add -
OK
ubuntu@ip-172-31-0-106:~$ sudo sh -c 'echo deb https://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'
ubuntu@ip-172-31-0-106:~$ sudo apt-get update
Ign:1 https://pkg.jenkins.io/debian-stable binary/ InRelease
Get:2 https://pkg.jenkins.io/debian-stable binary/ Release [2044 B]
Get:3 https://pkg.jenkins.io/debian-stable binary/ Release.gpg [833 B]
Hit:4 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu focal InRelease
Get:5 https://pkg.jenkins.io/debian-stable binary/ Packages [20.9 kB]
Hit:6 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu focal-updates InRelease
Hit:7 http://security.ubuntu.com/ubuntu focal-security InRelease
Hit:8 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu focal-backports InRelease
Fetched 23.8 kB in 1s (34.5 kB/s)
Reading package lists... Done
ubuntu@ip-172-31-0-106:~$ 

```

```

Desktop — ubuntu@ip-172-31-0-106: ~ — ssh -i key.pem ubuntu@ec2-65-0-179-1.ap-south-1.compute.amazonaws.com —...
The following NEW packages will be installed:
  daemon jenkins net-tools
0 upgraded, 3 newly installed, 0 to remove and 23 not upgraded.
Need to get 72.2 MB of archives.
After this operation, 73.5 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu focal/universe amd64 daemon amd64 0.6.4-1build2 [96.3 kB]
Get:3 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu focal/main amd64 net-tools amd64 1.60+git20180626.aebd88e-1ubuntu1 [196 kB]
Get:1 https://pkg.jenkins.io/debian-stable binary/ jenkins 2.303.3 [71.9 MB]
Fetched 72.2 MB in 38s (1912 kB/s)
Selecting previously unselected package daemon.
(Reading database ... 64281 files and directories currently installed.)
Preparing to unpack .../daemon_0.6.4-1build2_amd64.deb ...
Unpacking daemon (0.6.4-1build2) ...
Selecting previously unselected package net-tools.
Preparing to unpack .../net-tools_1.60+git20180626.aebd88e-1ubuntu1_amd64.deb ...
Unpacking net-tools (1.60+git20180626.aebd88e-1ubuntu1) ...
Selecting previously unselected package jenkins.
Preparing to unpack .../jenkins_2.303.3_all.deb ...
Unpacking jenkins (2.303.3) ...
Setting up net-tools (1.60+git20180626.aebd88e-1ubuntu1) ...
Setting up daemon (0.6.4-1build2) ...
Setting up jenkins (2.303.3) ...
Processing triggers for man-db (2.9.1-1) ...
Processing triggers for systemd (245.4-4ubuntu3.13) ...
ubuntu@ip-172-31-0-106:~$ sudo service jenkins status
● jenkins.service - LSB: Start Jenkins at boot time
  Loaded: loaded (/etc/init.d/jenkins; generated)
    Active: active (exited) since Thu 2021-11-11 06:00:38 UTC; 22s ago
      Docs: man:systemd-sysv-generator(8)
      Tasks: 0 (limit: 4700)
     Memory: 0B
        CGroup: /system.slice/jenkins.service

Nov 11 06:00:37 ip-172-31-0-106 systemd[1]: Starting LSB: Start Jenkins at boot time...
Nov 11 06:00:37 ip-172-31-0-106 jenkins[3981]: Correct java version found
Nov 11 06:00:37 ip-172-31-0-106 jenkins[3981]: * Starting Jenkins Automation Server jenkins
Nov 11 06:00:37 ip-172-31-0-106 su[4021]: (to jenkins) root on none
Nov 11 06:00:37 ip-172-31-0-106 su[4021]: pam_unix(su-1:session): session opened for user jenkins by (uid=0)
Nov 11 06:00:38 ip-172-31-0-106 su[4021]: pam_unix(su-1:session): session closed for user jenkins
Nov 11 06:00:38 ip-172-31-0-106 jenkins[3981]: ...done.
Nov 11 06:00:38 ip-172-31-0-106 systemd[1]: Started LSB: Start Jenkins at boot time.
ubuntu@ip-172-31-0-106:~$ 

```

Let's verify if jenkins is installed.

Go to EC2 instance -> copy public IP address -> open a new tab -> type "ip\_add:8080 and enter.

Spot Requests  
Savings Plans  
Reserved Instances [New](#)  
Dedicated Hosts  
Capacity Reservations

▼ Images  
AMIs

▼ Elastic Block Store  
Volumes [New](#)  
Snapshots  
Lifecycle Manager [New](#)

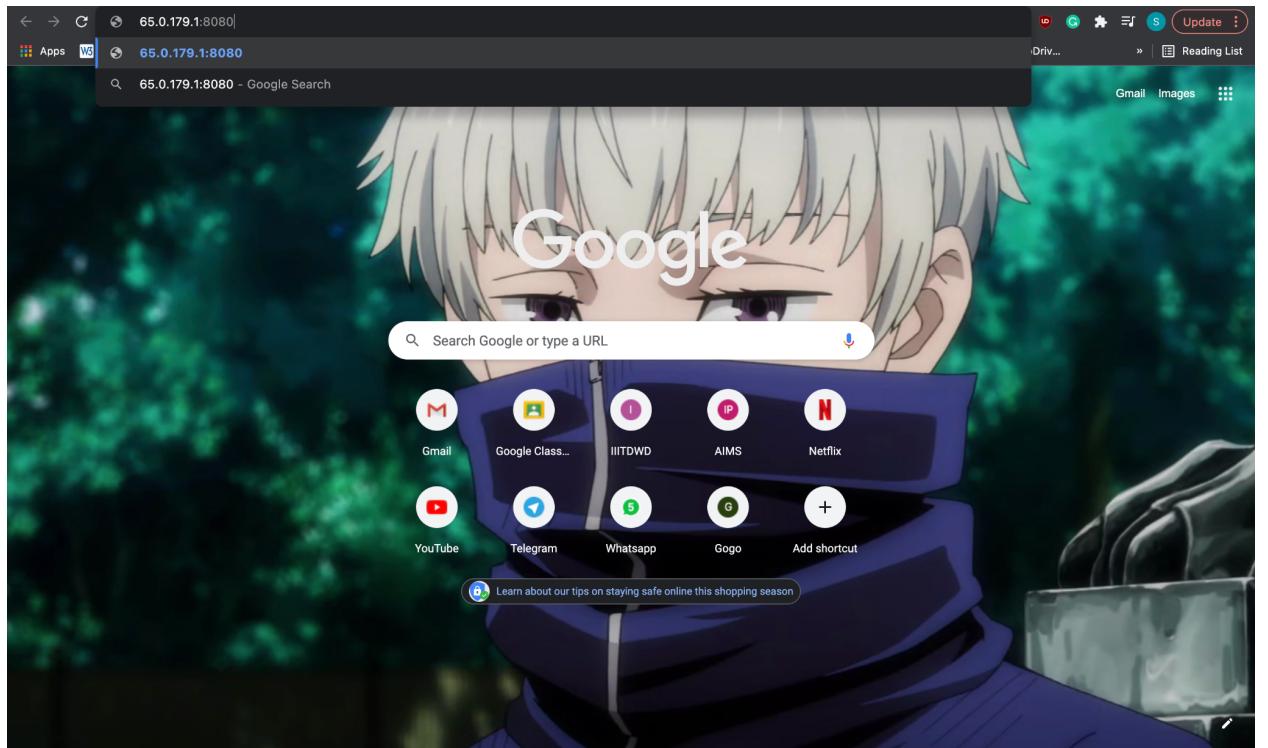
**Jenkins-1** i-0a7528a255f439f86 Running t2.medium 2/2 checks passed No alarms + ap-south-1b ec2-1

**Instance: i-0a7528a255f439f86 (Jenkins-1)**

**Details** Security Networking Storage Status checks Monitoring Tags

**Instance summary** Info

Instance ID <a href="#">i-0a7528a255f439f86 (Jenkins-1)</a>	Public IPv4 address <a href="#">65.0.179.1   open address</a>	Private IPv4 addresses <a href="#">172.31.0.106</a>
IPv6 address -	Instance state <span style="color: green;">Running</span>	Public IPv4 DNS <a href="#">ec2-65-0-179-1.ap-south-1.compute.amazonaws.com   open address</a>



Not Secure | 65.0.179.1:8080/login?from=%2F

## Getting Started

# Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log ([not sure where to find it?](#)) and this file on the server:

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

Please copy the password from either location and paste it below.

Administrator password



Continue

Not Secure | 65.0.179.1:8080

## Getting Started

# Customize Jenkins

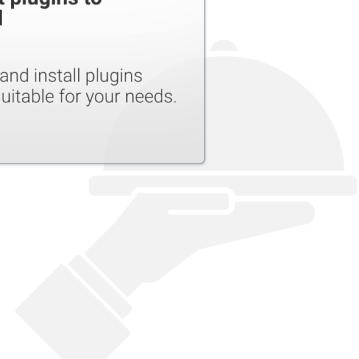
Plugins extend Jenkins with additional features to support many different needs.

**Install suggested plugins**

Install plugins the Jenkins community finds most useful.

**Select plugins to install**

Select and install plugins most suitable for your needs.



Jenkins 2.303.3

65.0.179.1:8080/#

Not Secure | 65.0.179.1:8080

## Getting Started

# Getting Started

✓ Folders	OWASP Markup Formatter	Build Timeout	Credentials Binding	** SSH server
Timestamper	Workspace Cleanup	Ant	Gradle	Folders
Pipeline	GitHub Branch Source	Pipeline: GitHub Groovy Libraries	Pipeline: Stage View	OWASP Markup Formatter
Git	SSH Build Agents	Matrix Authorization Strategy	PAM Authentication	
LDAP	Email Extension	Mailer		

\*\* - required dependency

Jenkins 2.303.3

This screenshot shows the Jenkins 'Getting Started' page. It displays a grid of available plugins. The first column contains 'Folders', 'Timestamper', 'Pipeline', 'Git', and 'LDAP'. The second column contains 'OWASP Markup Formatter', 'Workspace Cleanup', 'GitHub Branch Source', 'SSH Build Agents', and 'Email Extension'. The third column contains 'Build Timeout', 'Ant', 'Pipeline: GitHub Groovy Libraries', 'Matrix Authorization Strategy', and 'Mailer'. The fourth column contains 'Credentials Binding', 'Gradle', 'Pipeline: Stage View', 'PAM Authentication', and an empty slot. The fifth column contains '\*\* SSH server', 'Folders', 'OWASP Markup Formatter', and an empty slot. A note at the bottom indicates that the last two columns represent required dependencies. The Jenkins version '2.303.3' is also mentioned.

Not Secure | 65.0.179.1:8080

## Getting Started

# Create First Admin User

Username:

Password:

Confirm password:

Full name:

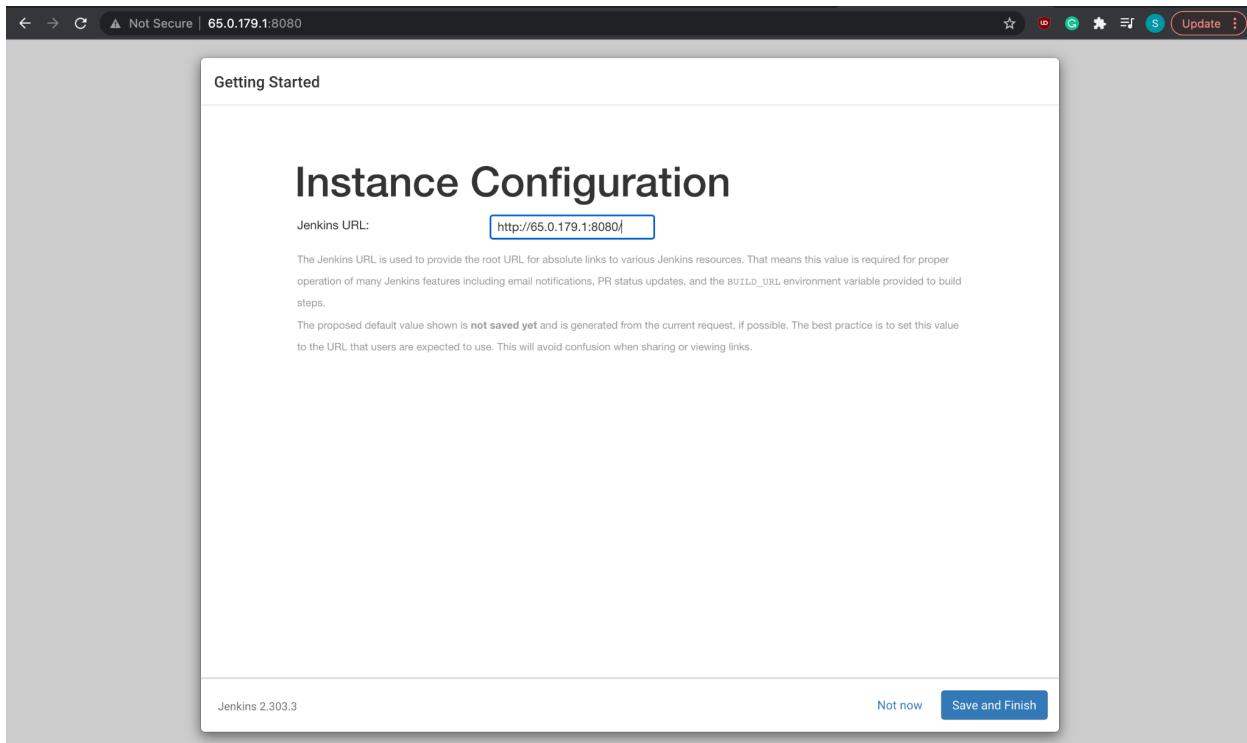
E-mail address:

Jenkins 2.303.3

Skip and continue as admin

Save and Continue

This screenshot shows the 'Create First Admin User' form. It includes fields for Username ('namratha'), Password ('\*\*\*\*\*'), Confirm password ('\*\*\*\*\*'), Full name ('namratha'), and E-mail address ('18bcs083@iitdwd.ac.in'). At the bottom, there are two buttons: 'Skip and continue as admin' and 'Save and Continue'. The Jenkins version '2.303.3' is also visible.



Goto -> Manage Jenkins -> Global Tool Configuration -> Gradle

The screenshot shows the Jenkins 'Manage Jenkins' page. On the left, there's a sidebar with options like 'New Item', 'People', 'Build History', 'Manage Jenkins' (which is selected), 'My Views', 'Lockable Resources', and 'New View'. Below that are sections for 'Build Queue' (empty) and 'Build Executor Status' (2 idle). The main content area has a title 'Manage Jenkins' with a note about building on the controller node being a security issue. It features several sections: 'System Configuration' (Configure System, Global Tool Configuration, Manage Nodes and Clouds), 'Security' (Configure Global Security, Manage Credentials, Manage Users), 'Status Information' (System Information, System Log, Load Statistics), and a 'Dismiss' button for the security note.

The screenshot shows the Jenkins Global Tool Configuration interface. At the top, there's a header with a back arrow, forward arrow, a 'Not Secure' warning, and the URL '65.0.179.1:8080/configureTools/'. Below the header, the 'Dashboard' and 'Global Tool Configuration' links are visible. The main content area is divided into sections for 'Gradle' and 'Ant'.  
  
The 'Gradle' section contains:

- A 'Gradle installations' heading with a 'Add Gradle' button.
- A 'Gradle' configuration block with a 'name' field set to 'default'.
- A checked checkbox for 'Install automatically'.
- An 'Install from Gradle.org' section with a 'Version' dropdown set to 'Gradle 7.1'.
- Buttons for 'Delete Installer' and 'Delete Gradle'.
- A 'Save' and 'Apply' button at the bottom.

  
The 'Ant' section contains:

- A 'List of Gradle installations on this system' heading.
- A 'Save' and 'Apply' button at the bottom.

#### 4. Update visudo and assign administration privileges to jenkins use

Add jenkins user as administrator - This step is not necessary but it avoids the process of entering password every time the pipeline runs.

```
sudo vi /etc/sudoers
```

```
jenkins ALL=(ALL) NOPASSWD: ALL
```

To use jenkins as root user, run the following

```
sudo su - jenkins
```

## 5. Install Docker

Run the below commands to install Docker

```
sudo apt install docker.io
```

```
docker --version
```

To add jenkins user to Docker group, run the following

```
sudo usermod -aG docker jenkins
```

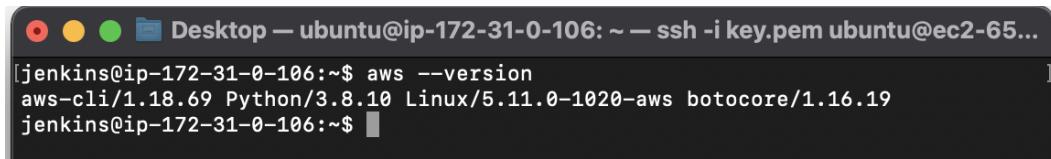
## 6. Install and Setup AWS CLI

To use eksctl we install AWS CLI.

Run the below commands to install AWS CLI

```
sudo apt install awscli
```

```
aws --version
```



A screenshot of a terminal window titled "Desktop — ubuntu@ip-172-31-0-106: ~ — ssh -i key.pem ubuntu@ec2-65...". The window shows the command "aws --version" being run, with the output "aws-cli/1.18.69 Python/3.8.10 Linux/5.11.0-1020-aws botocore/1.16.19". The terminal has a dark background with light-colored text and a black border.

## 7. Configure AWS CLI

Configure the AWS CLI to authenticate and communicate with AWS environment.

Go to AWS -> My Security Credentials -> Access keys -> Create New Access Key.

Screenshot of the AWS EC2 Instances page showing a single instance named Jenkins-1 (i-0a7528a255f439f86) in the running state (t2.medium).

The sidebar navigation includes:

- New EC2 Experience
- EC2 Dashboard
- EC2 Global View
- Events
- Tags
- Limits
- Instances
  - Instances New
  - Instance Types
  - Launch Templates
  - Spot Requests
  - Savings Plans
  - Reserved Instances New
  - Dedicated Hosts
  - Capacity Reservations
- Images
- AMIs
- Elastic Block Store
  - Volumes New
  - Snapshots
  - Lifecycle Manager New
- Network & Security

Bottom of the page shows the URL: [https://console.aws.amazon.com/iam/home?region=ap-south-1#security\\_credential](https://console.aws.amazon.com/iam/home?region=ap-south-1#security_credential)

**Access keys (access key ID and secret access key)**

Use access keys to make programmatic calls to AWS from the AWS CLI, Tools for PowerShell, AWS SDKs, or direct AWS API calls. You can have a maximum of two access keys (active or inactive) at a time.

For your protection, you should never share your secret keys with anyone. As a best practice, we recommend frequent key rotation.

If you lose or forget your secret key, you cannot retrieve it. Instead, create a new access key and make the old key inactive. [Learn more](#)

Created	Access Key ID	Last Used	Last Used Region	Last Used Service	Status	Actions
<a href="#">Create New Access Key</a>						

Root user access keys provide unrestricted access to your entire AWS account. If you need long-term access keys, we recommend creating a new IAM user with limited permissions and generating access keys for that user instead. [Learn more](#)

**Create Access Key**

**✓ Your access key (access key ID and secret access key) has been created successfully.**

Download your key file now, which contains your new access key ID and secret access key. If you do not download the key file now, you will not be able to retrieve your secret access key again.

To help protect your security, store your secret access key securely and do not share it.

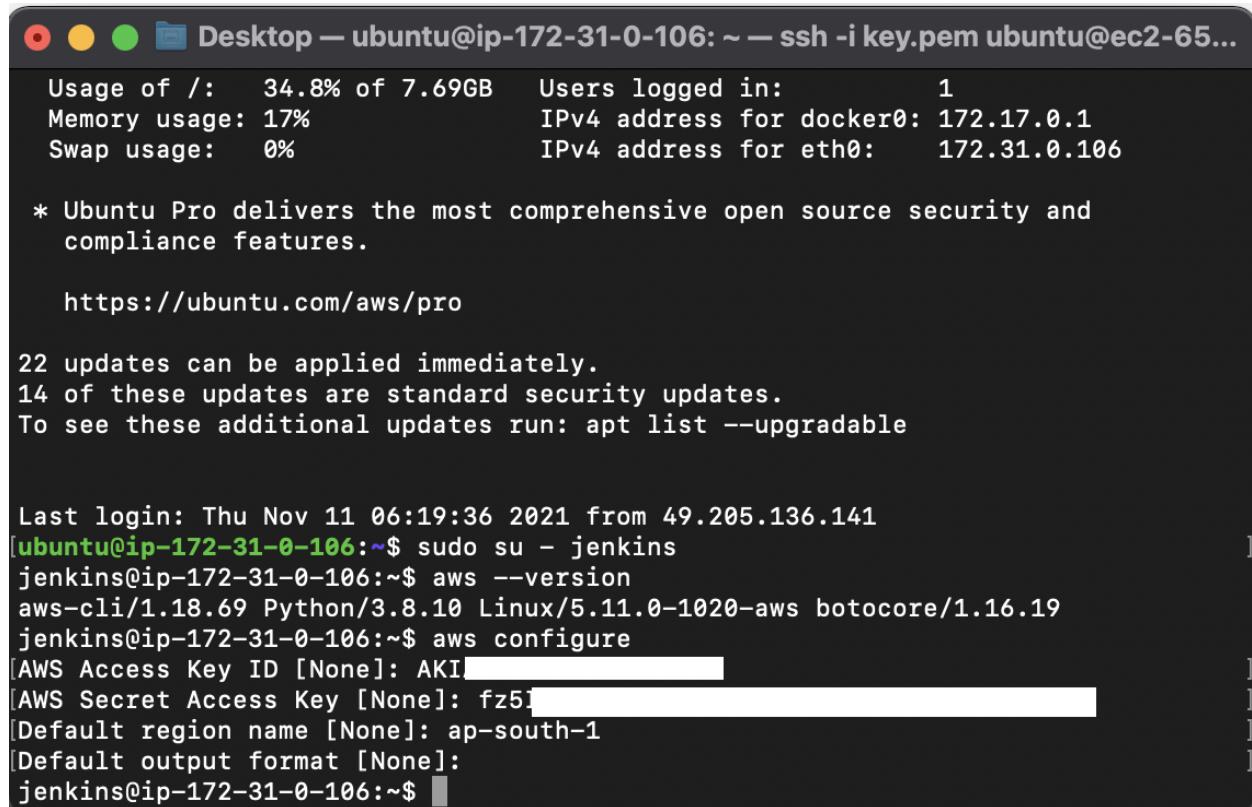
► [Show Access Key](#)

[Download Key File](#) [Close](#)

Run the below commands to configure.

```
aws configure
```

Enter the Access Key ID and Secret Access Key obtained after creation of access key.



```
Desktop — ubuntu@ip-172-31-0-106: ~ — ssh -i key.pem ubuntu@ec2-65...
Usage of /: 34.8% of 7.69GB Users logged in: 1
Memory usage: 17% IPv4 address for docker0: 172.17.0.1
Swap usage: 0% IPv4 address for eth0: 172.31.0.106

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

https://ubuntu.com/aws/pro

22 updates can be applied immediately.
14 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

Last login: Thu Nov 11 06:19:36 2021 from 49.205.136.141
[ubuntu@ip-172-31-0-106:~$ sudo su - jenkins
jenkins@ip-172-31-0-106:~$ aws --version
aws-cli/1.18.69 Python/3.8.10 Linux/5.11.0-1020-aws botocore/1.16.19
jenkins@ip-172-31-0-106:~$ aws configure
[AWS Access Key ID [None]: AKI[REDACTED]
[AWS Secret Access Key [None]: fz5i[REDACTED]
[Default region name [None]: ap-south-1
[Default output format [None]:
jenkins@ip-172-31-0-106:~$ ]
```

## 8. Install and Setup Kubectl

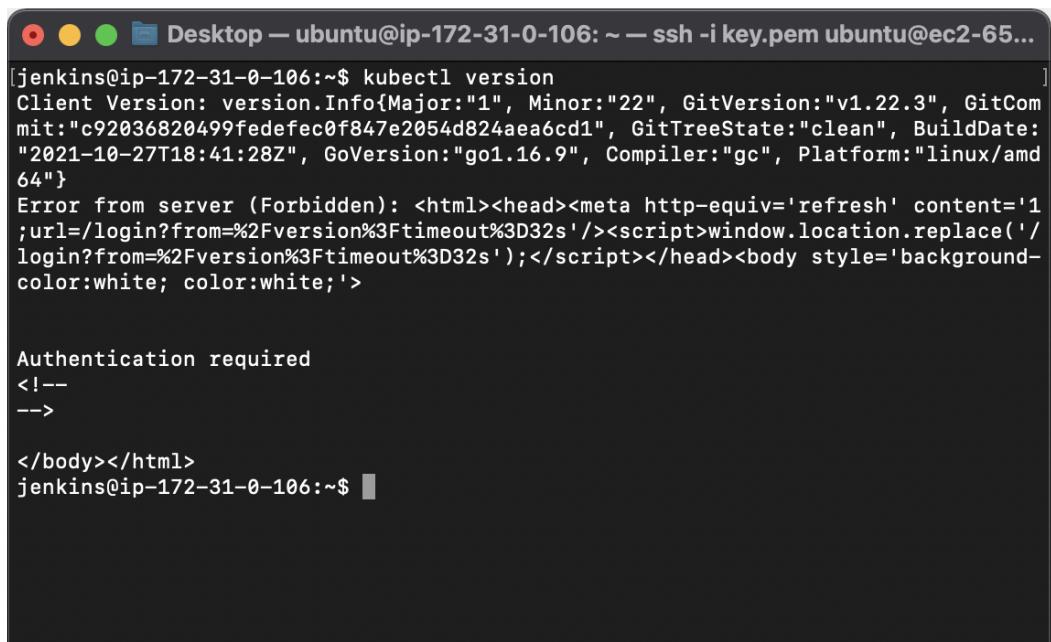
Run the below commands to install Kubectl

```
curl -LO
"https://storage.googleapis.com/kubernetes-release/release/$(curl -s
https://storage.googleapis.com/kubernetes-release/release/stable.txt)/bin/linux/amd64/kubectl"
chmod +x ./kubectl
```

```
sudo mv ./kubectl /usr/local/bin
```

To verify the installation, run the following

```
Kubectl version
```



```
[jenkins@ip-172-31-0-106:~$ kubectl version
Client Version: version.Info{Major:"1", Minor:"22", GitVersion:"v1.22.3", GitCommit:"c92036820499fedefec0f847e2054d824aea6cd1", GitTreeState:"clean", BuildDate:"2021-10-27T18:41:28Z", GoVersion:"go1.16.9", Compiler:"gc", Platform:"linux/amd64"}
Error from server (Forbidden): <html><head><meta http-equiv='refresh' content='1
;url=/login?from=%2Fversion%3Ftimeout%3D32s'/'><script>window.location.replace('/
login?from=%2Fversion%3Ftimeout%3D32s');</script></head><body style='background-
color:white; color:white;'>

Authentication required
<!--
-->

</body></html>
jenkins@ip-172-31-0-106:~$ ]
```

## 9. Install and Setup eksctl

Run the below commands to install eksctl

```
curl --silent --location
"https://github.com/weaveworks/eksctl/releases/latest/download/
eksctl_$(uname -s)_amd64.tar.gz" | tar xz -C /tmp
```

```
sudo mv /tmp/eksctl /usr/local/bin
```

To verify the installation, run the following

```
eksctl version
```

```

jenkins@ip-172-31-0-106:~$ curl --silent --location "https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_$(uname -s)_amd64.tar.gz" | tar xz -C /tmp
jenkins@ip-172-31-0-106:~$ sudo mv /tmp/eksctl /usr/local/bin
jenkins@ip-172-31-0-106:~$ eksctl version
0.73.0
jenkins@ip-172-31-0-106:~$ █

```

## 10. Create eks cluster using eksctl

To create a cluster, run the eksctl command

```

eksctl create cluster \
--name jhooq-test-cluster1 \
--version 1.17 \
--region ap-south-1b \
--nodegroup-name worker-node \
--node-type t2.micro \
--nodes 2

```

```

● ● ● Desktop — ubuntu@ip-172-31-0-106: ~ ssh -i key.pem ubuntu@ec2-65-0-179-1.ap-south-1.compute.amazonaws.com — 132x46
jenkins@ip-172-31-0-106:~$ eksctl create cluster --name jhooq-test-cluster1 --version 1.17 --region ap-south-1 --nodegroup-name worker-node --node-type t2.medium --nodes 2
2021-11-11 07:23:44 [i] eksctl version 0.73.0
2021-11-11 07:23:44 [i] using region ap-south-1
2021-11-11 07:23:44 [i] setting availability zones to [ap-south-1a ap-south-1c ap-south-1b]
2021-11-11 07:23:44 [i] subnets for ap-south-1a - public:192.168.0.0/19 private:192.168.96.0/19
2021-11-11 07:23:44 [i] subnets for ap-south-1c - public:192.168.32.0/19 private:192.168.128.0/19
2021-11-11 07:23:44 [i] subnets for ap-south-1b - public:192.168.64.0/19 private:192.168.160.0/19
2021-11-11 07:23:44 [i] nodegroup "worker-node" will use "" [AmazonLinux2/1.17]
2021-11-11 07:23:44 [i] using Kubernetes version 1.17
2021-11-11 07:23:44 [i] creating EKS cluster "jhooq-test-cluster1" in "ap-south-1" region with managed nodes
2021-11-11 07:23:44 [i] will create 2 separate CloudFormation stacks for cluster itself and the initial managed nodegroup
2021-11-11 07:23:44 [i] if you encounter any issues, check CloudFormation console or try 'eksctl utils describe-stacks --region=ap-south-1 --cluster=jhooq-test-cluster1'
2021-11-11 07:23:44 [i] CloudWatch logging will not be enabled for cluster "jhooq-test-cluster1" in "ap-south-1"
2021-11-11 07:23:44 [i] you can enable it with 'eksctl utils update-cluster-logging --enable-types={SPECIFY-YOUR-LOG-TYPES-HERE (e.g. all)} --region=ap-south-1 --cluster=jhooq-test-cluster1'
2021-11-11 07:23:44 [i] Kubernetes API endpoint access will use default of {publicAccess=true, privateAccess=false} for cluster "jhooq-test-cluster1" in "ap-south-1"
2021-11-11 07:23:44 [i]
2 sequential tasks: { create cluster control plane "jhooq-test-cluster1",
  2 sequential sub-tasks: {
    wait for control plane to become ready,
    create managed nodegroup "worker-node",
  }
}
2021-11-11 07:23:44 [i] building cluster stack "eksctl-jhooq-test-cluster1-cluster"
2021-11-11 07:23:44 [i] deploying stack "eksctl-jhooq-test-cluster1-cluster"
2021-11-11 07:24:14 [i] waiting for CloudFormation stack "eksctl-jhooq-test-cluster1-cluster"
2021-11-11 07:24:44 [i] waiting for CloudFormation stack "eksctl-jhooq-test-cluster1-cluster"
2021-11-11 07:25:45 [i] waiting for CloudFormation stack "eksctl-jhooq-test-cluster1-cluster"
2021-11-11 07:26:45 [i] waiting for CloudFormation stack "eksctl-jhooq-test-cluster1-cluster"
2021-11-11 07:27:45 [i] waiting for CloudFormation stack "eksctl-jhooq-test-cluster1-cluster"
2021-11-11 07:28:45 [i] waiting for CloudFormation stack "eksctl-jhooq-test-cluster1-cluster"
2021-11-11 07:29:45 [i] waiting for CloudFormation stack "eksctl-jhooq-test-cluster1-cluster"
2021-11-11 07:30:45 [i] waiting for CloudFormation stack "eksctl-jhooq-test-cluster1-cluster"
2021-11-11 07:31:45 [i] waiting for CloudFormation stack "eksctl-jhooq-test-cluster1-cluster"
2021-11-11 07:32:45 [i] waiting for CloudFormation stack "eksctl-jhooq-test-cluster1-cluster"
2021-11-11 07:33:45 [i] waiting for CloudFormation stack "eksctl-jhooq-test-cluster1-cluster"
2021-11-11 07:34:45 [i] waiting for CloudFormation stack "eksctl-jhooq-test-cluster1-cluster"
2021-11-11 07:35:45 [i] waiting for CloudFormation stack "eksctl-jhooq-test-cluster1-cluster"
2021-11-11 07:37:45 [i] building managed nodegroup stack "eksctl-jhooq-test-cluster1-nodegroup-worker-node"
2021-11-11 07:37:46 [i] deploying stack "eksctl-jhooq-test-cluster1-nodegroup-worker-node"
2021-11-11 07:37:46 [i] waiting for CloudFormation stack "eksctl-jhooq-test-cluster1-nodegroup-worker-node"
2021-11-11 07:38:02 [i] waiting for CloudFormation stack "eksctl-jhooq-test-cluster1-nodegroup-worker-node"
2021-11-11 07:38:19 [i] waiting for CloudFormation stack "eksctl-jhooq-test-cluster1-nodegroup-worker-node"

```

```

Desktop — ubuntu@ip-172-31-0-106: ~ — ssh -i key.pem ubuntu@ec2-65-0-179-1.ap-south-1.compute.amazonaws.com — 132x46
2021-11-11 07:23:44 [i] building cluster stack "eksctl-jhooc-test-cluster1-cluster"
2021-11-11 07:23:44 [i] deploying stack "eksctl-jhooc-test-cluster1-cluster"
2021-11-11 07:24:44 [i] waiting for CloudFormation stack "eksctl-jhooc-test-cluster1-cluster"
2021-11-11 07:24:44 [i] waiting for CloudFormation stack "eksctl-jhooc-test-cluster1-cluster"
2021-11-11 07:25:45 [i] waiting for CloudFormation stack "eksctl-jhooc-test-cluster1-cluster"
2021-11-11 07:26:45 [i] waiting for CloudFormation stack "eksctl-jhooc-test-cluster1-cluster"
2021-11-11 07:27:45 [i] waiting for CloudFormation stack "eksctl-jhooc-test-cluster1-cluster"
2021-11-11 07:28:45 [i] waiting for CloudFormation stack "eksctl-jhooc-test-cluster1-cluster"
2021-11-11 07:29:45 [i] waiting for CloudFormation stack "eksctl-jhooc-test-cluster1-cluster"
2021-11-11 07:30:45 [i] waiting for CloudFormation stack "eksctl-jhooc-test-cluster1-cluster"
2021-11-11 07:31:45 [i] waiting for CloudFormation stack "eksctl-jhooc-test-cluster1-cluster"
2021-11-11 07:32:45 [i] waiting for CloudFormation stack "eksctl-jhooc-test-cluster1-cluster"
2021-11-11 07:33:45 [i] waiting for CloudFormation stack "eksctl-jhooc-test-cluster1-cluster"
2021-11-11 07:34:45 [i] waiting for CloudFormation stack "eksctl-jhooc-test-cluster1-cluster"
2021-11-11 07:35:45 [i] waiting for CloudFormation stack "eksctl-jhooc-test-cluster1-cluster"
2021-11-11 07:37:45 [i] building managed nodegroup stack "eksctl-jhooc-test-cluster1-nodegroup-worker-node"
2021-11-11 07:37:46 [i] deploying stack "eksctl-jhooc-test-cluster1-nodegroup-worker-node"
2021-11-11 07:37:46 [i] waiting for CloudFormation stack "eksctl-jhooc-test-cluster1-nodegroup-worker-node"
2021-11-11 07:38:02 [i] waiting for CloudFormation stack "eksctl-jhooc-test-cluster1-nodegroup-worker-node"
2021-11-11 07:38:19 [i] waiting for CloudFormation stack "eksctl-jhooc-test-cluster1-nodegroup-worker-node"
2021-11-11 07:38:38 [i] waiting for CloudFormation stack "eksctl-jhooc-test-cluster1-nodegroup-worker-node"
2021-11-11 07:38:55 [i] waiting for CloudFormation stack "eksctl-jhooc-test-cluster1-nodegroup-worker-node"
2021-11-11 07:39:15 [i] waiting for CloudFormation stack "eksctl-jhooc-test-cluster1-nodegroup-worker-node"
2021-11-11 07:39:34 [i] waiting for CloudFormation stack "eksctl-jhooc-test-cluster1-nodegroup-worker-node"
2021-11-11 07:39:53 [i] waiting for CloudFormation stack "eksctl-jhooc-test-cluster1-nodegroup-worker-node"
2021-11-11 07:40:10 [i] waiting for CloudFormation stack "eksctl-jhooc-test-cluster1-nodegroup-worker-node"
2021-11-11 07:40:28 [i] waiting for CloudFormation stack "eksctl-jhooc-test-cluster1-nodegroup-worker-node"
2021-11-11 07:40:44 [i] waiting for CloudFormation stack "eksctl-jhooc-test-cluster1-nodegroup-worker-node"
2021-11-11 07:41:00 [i] waiting for CloudFormation stack "eksctl-jhooc-test-cluster1-nodegroup-worker-node"
2021-11-11 07:41:19 [i] waiting for CloudFormation stack "eksctl-jhooc-test-cluster1-nodegroup-worker-node"
2021-11-11 07:41:34 [i] waiting for CloudFormation stack "eksctl-jhooc-test-cluster1-nodegroup-worker-node"
2021-11-11 07:41:51 [i] waiting for CloudFormation stack "eksctl-jhooc-test-cluster1-nodegroup-worker-node"
2021-11-11 07:41:51 [i] waiting for the control plane availability...
2021-11-11 07:41:51 [✓] saved kubeconfig as "/var/lib/jenkins/.kube/config"
2021-11-11 07:41:51 [i] no tasks
2021-11-11 07:41:51 [✓] all EKS cluster resources for "jhooc-test-cluster1" have been created
2021-11-11 07:41:51 [i] nodegroup "worker-node" has 2 node(s)
2021-11-11 07:41:51 [i] node "ip-192-168-2-173.ap-south-1.compute.internal" is ready
2021-11-11 07:41:51 [i] node "ip-192-168-67-128.ap-south-1.compute.internal" is ready
2021-11-11 07:41:51 [i] waiting for at least 2 node(s) to become ready in "worker-node"
2021-11-11 07:41:51 [i] nodegroup "worker-node" has 2 node(s)
2021-11-11 07:41:51 [i] node "ip-192-168-2-173.ap-south-1.compute.internal" is ready
2021-11-11 07:41:51 [i] node "ip-192-168-67-128.ap-south-1.compute.internal" is ready
2021-11-11 07:41:52 [i] kubectl command should work with "/var/lib/jenkins/.kube/config", try 'kubectl get nodes'
2021-11-11 07:41:52 [✓] EKS cluster "jhooc-test-cluster1" in "ap-south-1" region is ready
jenkins@ip-172-31-0-106:~$ 

```

## Verify the EKS kubernetes cluster from AWS.

The screenshot shows the AWS EKS console interface. On the left, there's a sidebar with 'Amazon Container Services' and 'Amazon EKS' sections. The 'Clusters' section under EKS is selected, showing a single cluster named 'jhooc-test-cluster1'. The main area displays the cluster details with a table:

Cluster name	Status	Kubernetes version	Provider
jhooc-test-cluster1	Active	1.17 <a href="#">Update now</a>	EKS

A message at the top of the cluster view says: 'New Kubernetes versions are available for 1 cluster.'

The screenshot shows the AWS EKS console interface. On the left, there's a sidebar for Amazon Container Services with options for Amazon ECS (Clusters, Task definitions), Amazon EKS (Clusters, New), and Amazon ECR (Repositories). The main content area shows the 'jhooq-test-cluster1' cluster under the 'Clusters' section. At the top right of the cluster card, there are buttons for 'Active' (with a green circle icon), a copy icon, 'Delete cluster', and 'Update now'. Below the cluster card, there are tabs for 'Overview', 'Workloads', and 'Configuration', with 'Overview' selected. Under the 'Nodes (2)' section, there's a table with two rows:

Node name	Instance type	Node Group	Created	Status
ip-192-168-2-173.ap-south-1.compute.internal	t2.medium	worker-node	6 minutes ago	Ready
ip-192-168-67-128.ap-south-1.compute.internal	t2.medium	worker-node	6 minutes ago	Ready

At the bottom of the page, there are links for Feedback, English (US) (dropdown), Privacy Policy, Terms of Use, and Cookie preferences.

## 11. Add Docker and GitHub Credentials into Jenkins

Goto -> Jenkins -> Manage Jenkins -> Manage Credentials -> Stored scoped to jenkins -> global -> Add Credentials.

Not Secure | 65.0.179.1:8080/manage

# Jenkins

Dashboard >

- New Item
- People
- Build History
- Manage Jenkins**
- My Views
- Lockable Resources
- New View

**Build Queue**  
No builds in the queue.

**Build Executor Status**  
1 Idle  
2 Idle

## Manage Jenkins

Building on the controller node can be a security issue. You should set up distributed builds. See [the documentation](#).

**System Configuration**

- Configure System**  
Configure global settings and paths.
- Global Tool Configuration**  
Configure tools, their locations and automatic installers.
- Manage Nodes and Clouds**  
Add, remove, control and monitor the various nodes that Jenkins runs jobs on.
- Manage Plugins**  
Add, remove, disable or enable plugins that can extend the functionality of Jenkins.

## Security

- Configure Global Security**  
Secure Jenkins; define who is allowed to access/use the system.
- Manage Credentials**  
Configure credentials
- Manage Users**  
Create/delete/modify users that can log in to this Jenkins
- Configure Credential Providers**  
Configure the credential providers and types

## Status Information

**System Information**  
Displays various environmental information

**System Log**  
System log captures output from

**Load Statistics**  
Check your resource utilization and see if

Not Secure | 65.0.179.1:8080/credentials/

# Jenkins

Dashboard > Credentials

- New Item
- People
- Build History
- Manage Jenkins
- My Views
- Lockable Resources
- New View

**Build Queue**  
No builds in the queue.

**Build Executor Status**  
1 Idle  
2 Idle

## Credentials

Icon: S M L

### Stores scoped to Jenkins

P	Store ↓	Domains	ID	Name
	Jenkins	(global)		Add credentials

REST API Jenkins 2.303.3

Not Secure | 65.0.179.1:8080/credentials/store/system/domain/\_/credential/GIT\_HUB\_CREDENTIALS/update

Jenkins

Dashboard > Credentials > System > Global credentials (unrestricted) > Namratha-shivaraju/\*\*\*\*\* (Git Hub Credentials)

Back to Global credentials (unrestricted) | Update | Delete | Move

Scope: Global (Jenkins, nodes, items, all child items, etc)

Username: Namratha-shivaraju

Treat username as secret:

Password: Concealed | Change Password

ID: GIT\_HUB\_CREDENTIALS

Description: Git Hub Credentials

Save

REST API Jenkins 2.303.3

Not Secure | 65.0.179.1:8080/credentials/store/system/domain/\_/newCredentials

Jenkins

Dashboard > Credentials > System > Global credentials (unrestricted)

Back to credential domains | Add Credentials

Kind: Secret text

Scope: Global (Jenkins, nodes, items, all child items, etc)

Secret: ..... (redacted)

ID: DOCKER\_HUB\_PASSWORD

Description: Docker Hub Credentials

OK

REST API Jenkins 2.303.3

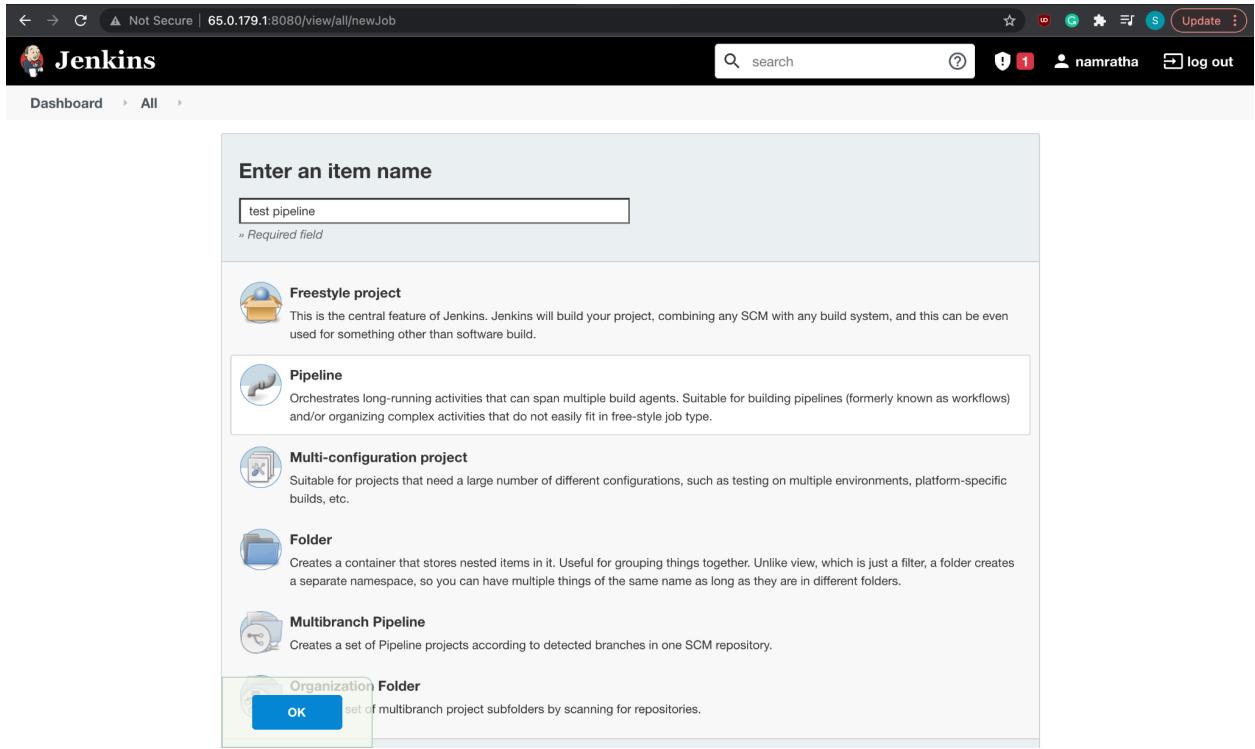
## 12. CI/CD pipeline

Go to jenkins -> new item-> enter the name of the item -> select “Pipeline”  
-> click Ok.

The screenshot shows the Jenkins 'New Item' creation interface. At the top, there is a search bar and a user profile for 'namratha'. Below the header, the URL is shown as 'Not Secure | 65.0.179.1:8080/view/all/newJob'. The main area has a title 'Enter an item name' with a text input field containing a single character. Below this, there is a list of project types:

- Freestyle project**: This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.
- Pipeline**: Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.
- Multi-configuration project**: Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.
- Folder**: Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.
- Multibranch Pipeline**: Creates a set of Pipeline projects according to detected branches in one SCM repository.
- Organization Folder**: Creates a set of multibranch project subfolders by scanning for repositories.

A blue 'OK' button is visible at the bottom of the list.



## 13. Add jenkins stages

We need to write jenkins pipeline to deploy our application.  
Combine the stages and add it in the pipeline script.  
Click “Apply” and “Save”.

### 13.1. Jenkins stage-1 : Checkout the GitHub Repository

```
stage("Git Clone") {  
  
    git branch: 'main', credentialsId: 'GIT_HUB_CREDENTIALS',  
    url:  
        'https://github.com/Namratha-shivaraju/k8s-jenkins-aws.git'  
    '  
}  
}
```

### 13.2. Jenkins stage-2 : Gradle compilation and build

```
stage('Gradle Build') {  
    sh './gradlew build'
```

```
}
```

### 13.3. Jenkins stage-3 : Create Docker Container and push to Docker Hub

```
stage("Docker build") {
    sh 'docker version'
    sh 'docker build -t jhooq-docker-demo .'
    sh 'docker image list'
    sh 'docker tag jhooq-docker-demo
nam07/jhooq-docker-demo:jhooq-docker-demo'
}
withCredentials([string(credentialsId:
'DOCKER_HUB_PASSWORD', variable: 'PASSWORD')]) {
    sh 'docker login -u nam07 -p $PASSWORD'
}
stage("Push Image to Docker Hub") {
    sh 'docker push
nam07/jhooq-docker-demo:jhooq-docker-demo'
}
```

### 13.4. Jenkins stage-4 : Kubernetes deployment

```
stage("kubernetes deployment") {
    sh 'kubectl apply -f
k8s-spring-boot-deployment.yml'
}
```

The screenshot shows the Jenkins Pipeline configuration page for a job named "test pipeline". The "Advanced Project Options" tab is selected. The "Pipeline" section contains a "Definition" dropdown set to "Pipeline script" and a code editor containing the following Groovy script:

```
node {
    stage("Git Clone"){
        git branch: 'main', credentialsId: 'GIT_HUB_CREDENTIALS', url: 'https://github.com/Namratha-shivaraju/k8s-jenkins'
    }
    stage('Gradle Build') {
        sh './gradlew build'
    }
    stage("Docker build"){
        sh 'docker version'
    }
}
```

A checkbox labeled "Use Groovy Sandbox" is checked. At the bottom are "Save" and "Apply" buttons.

The screenshot shows the Jenkins Pipeline configuration page for the same job. The "Advanced Project Options" tab is selected. The "Pipeline" section contains a "Definition" dropdown set to "Pipeline script" and a code editor containing the following Groovy script:

```
node {
    stage("Git Clone"){
        git branch: 'main', credentialsId: 'GIT_HUB_CREDENTIALS', url: 'https://github.com/Namratha-shivaraju/k8s-jenkins'
    }
    stage('Gradle Build') {
        sh './gradlew build'
    }
    stage("Docker build"){
        sh 'docker build -t jhooq-docker-demo .'
        sh 'docker image list'
        sh 'docker tag jhooq-docker-demo nam07/jhooq-docker-demo:jhooq-docker-demo'
    }
    withCredentials([string(credentialsId: 'DOCKER_HUB_PASSWORD', variable: 'PASSWORD')]) {
        sh 'docker login -u nam07 -p $PASSWORD'
    }
    stage("Push Image to Docker Hub"){
        sh 'docker push nam07/jhooq-docker-demo:jhooq-docker-demo'
    }
    stage("Kubernetes deployment"){
        sh 'kubectl apply -f k8s-spring-boot-deployment.yml'
    }
}
```

A checkbox labeled "Use Groovy Sandbox" is checked. At the bottom are "Save" and "Apply" buttons.

## 14. Build and run the pipeline

Go to the pipeline and click on “Build Now”.

The screenshot shows the Jenkins interface for a pipeline named "Pipeline test pipeline". On the left, there's a sidebar with various navigation links. The "Build Now" link is currently selected and highlighted in blue. The main content area displays the pipeline stages: Git Clone, Gradle Build, Docker build, Push Image to Docker Hub, and kubernetes deployment. Below the stages, there's a table showing average stage times and a recent changes log. The "Permalinks" section lists the last four builds, including the current one (#7) which is successful.

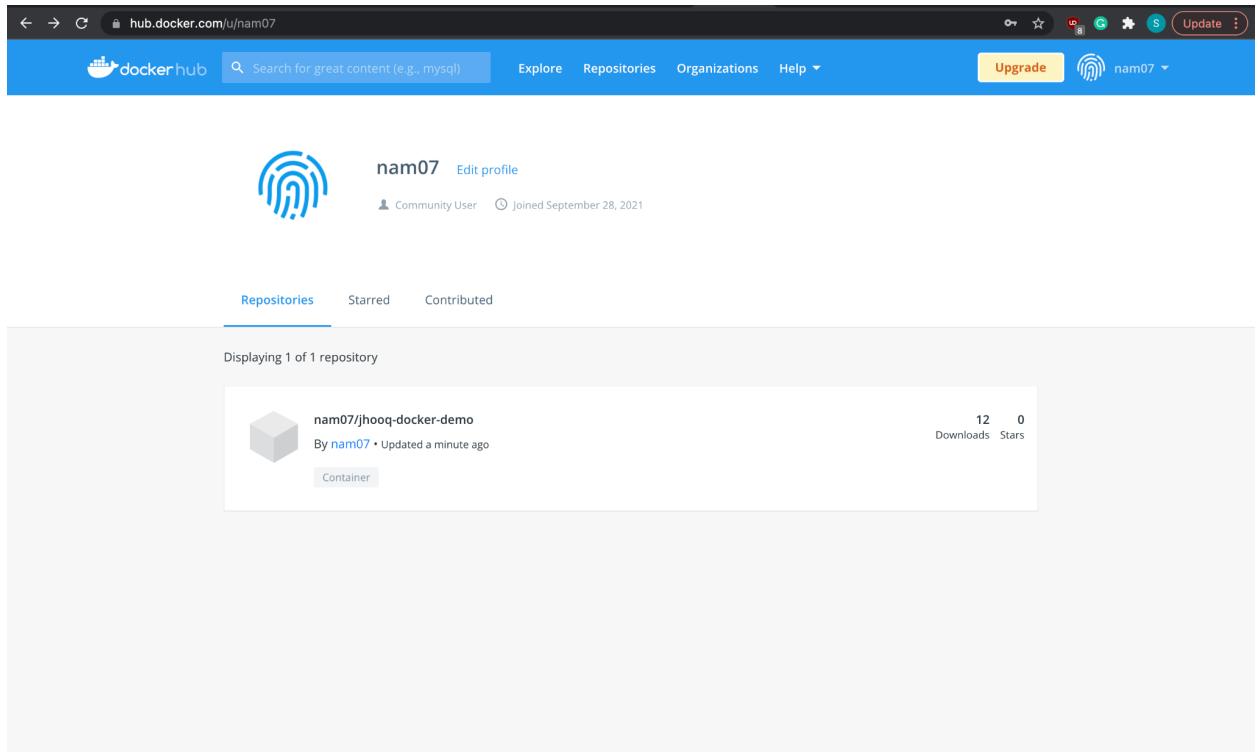
## 15. Verify using kubectl commands

Run the below commands to verify the deployment.

kubectl get deployments

kubectl get service

```
Desktop — ubuntu@ip-172-31-0-106: ~ ssh -i key.pem ubuntu@ec2-65-0-179-1.ap-south-1.compute.amazonaws.com — 153x39
jenkins@ip-172-31-0-106:~$ kubectl get deployments
NAME          READY   UP-TO-DATE   AVAILABLE   AGE
jhoog-springboot   3/3      3          3           4m47s
jenkins@ip-172-31-0-106:~$ kubectl get service
NAME          TYPE      CLUSTER-IP      EXTERNAL-IP
jhoog-springboot   LoadBalancer   10.100.115.204   a4b6b5d44b816472ba168fb0a8bc15ab-1578614707.ap-south-1.elb.amazonaws.com
kubernetes     ClusterIP    10.100.0.1       <none>
jenkins@ip-172-31-0-106:~$
```



## 16. Access the rest end point from browser

Copy the external IP address from terminal and paste it in the browser with /hello.

