**Deploying Apache Project using Ansible via Jenkins**

**-Pulkit Mathur**

**Aim :** The project aims to Deploy Apache project on Node EC2’s using Ansible which will be triggered via Jenkins pipeline

**Git Hub repo to fork :**

<https://github.com/akshu20791/apachewebsite>

**Git Hub repo for reference :** <https://github.com/pulkit2003/apachewebsite>

**Base plan :**

* Create a Master EC2 with – Jenkins, Ansible, docker installed
* Create Cluster and nodes
* Connect the Node to master
* Make a Jenkins pipeline and deploy Apache project on all nodes by running Ansible playbook

**Note :**

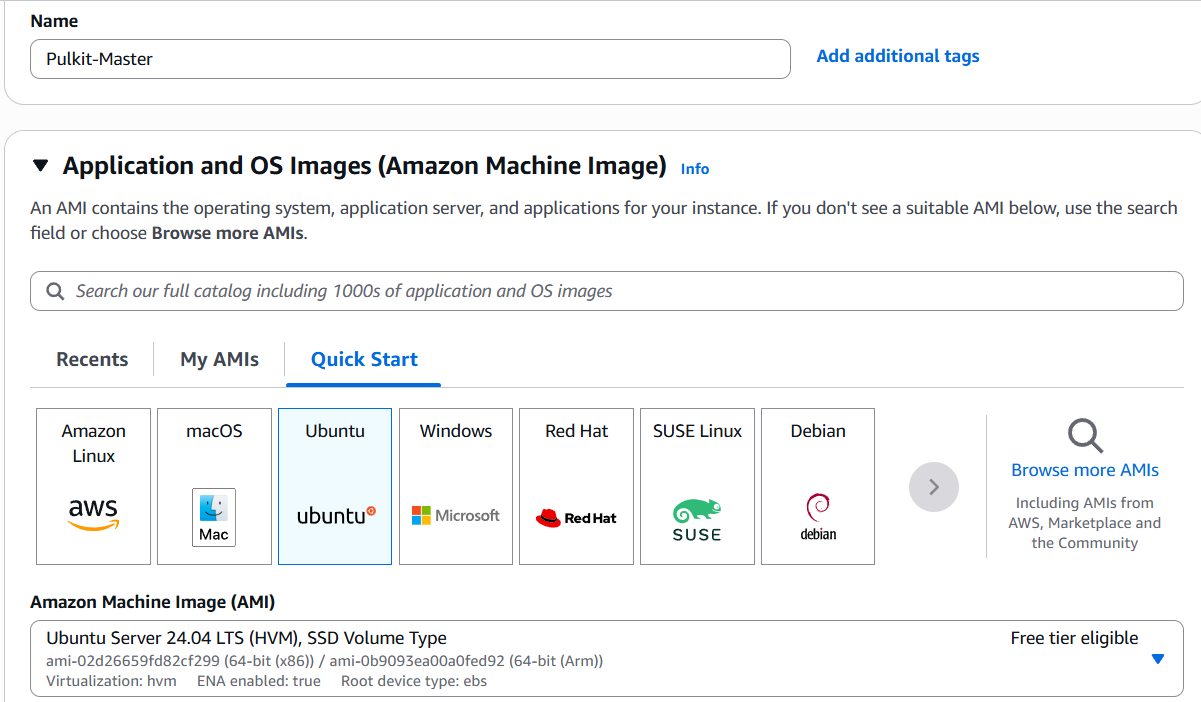
Replace all the Blue Highlighted data with your own data

AL the Yellow Highlighted data is code

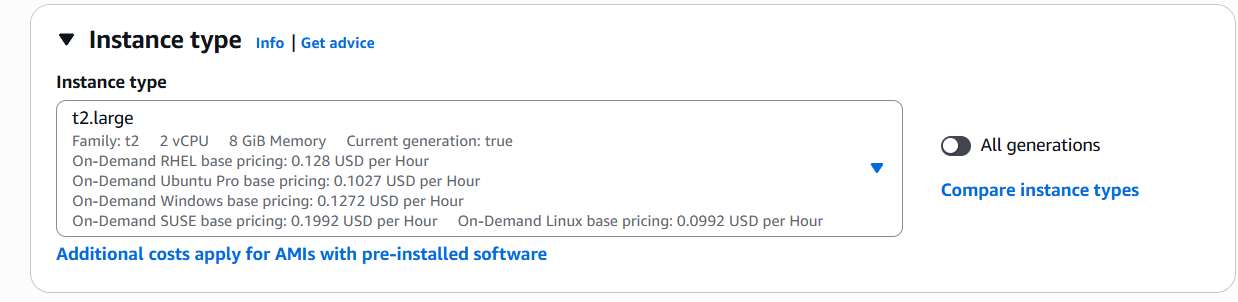
**Steps to follow :**

1. Create a Master EC2
2. Create IAM User
3. Installations
4. Create EKS Cluster
5. EstablishingJenkins Plugins
6. Creating Dynamic Inventory
7. Connecting Ansible Master – Node
8. Final Deployment Stage
9. Jenkins pipeline
10. Confirmation

**Step 1. Create a Master EC2**

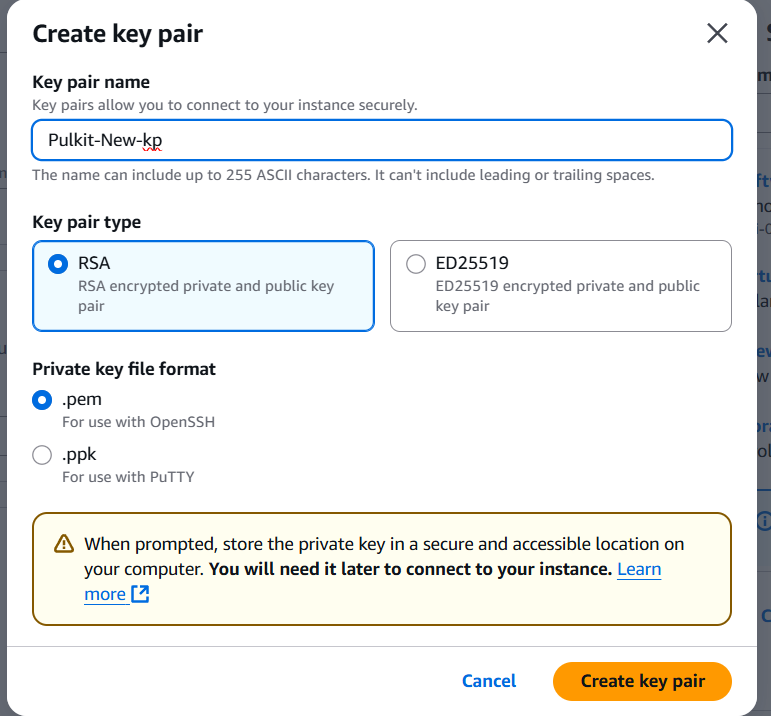
a) Give name to EC2 and select AMI as Ubuntu 24.04LTS ****

b) Select instance type as t2.large ( since this is master EC2 it will require large size )



c) Create a Key pair ( Save it we will use this key pair throughout the project )

Give it name -> key pair type = RSA -> Key file format = .pem -> Create key pair



d) Create Security Group

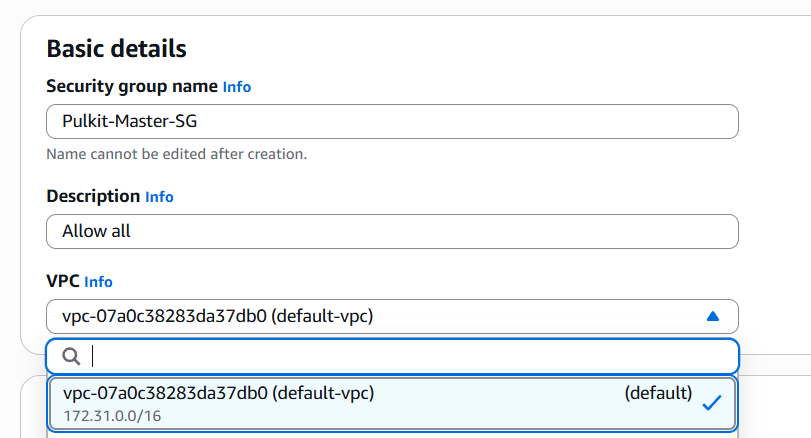
Click on Create Security Group  
Give name -> Give description -> Select default VPC -> Give Inbound rules -> Click Create Security Group

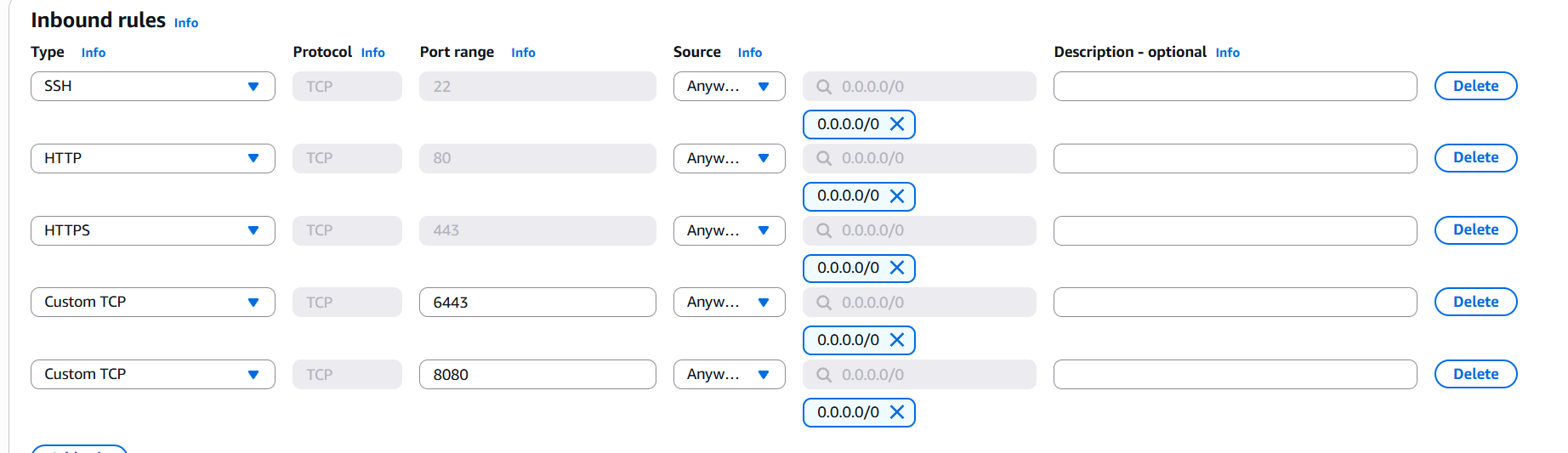
* SSH, Port =22, Source = Anywhere 0.0.0.0/0
* HTTP , Port = 80, Source = Anywhere 0.0.0.0/0
* HTTPS, Port = 443, Source = Anywhere 0.0.0.0/0
* TCP, Port = 6443, Source = Anywhere 0.0.0.0/0

(for Kubernetes)

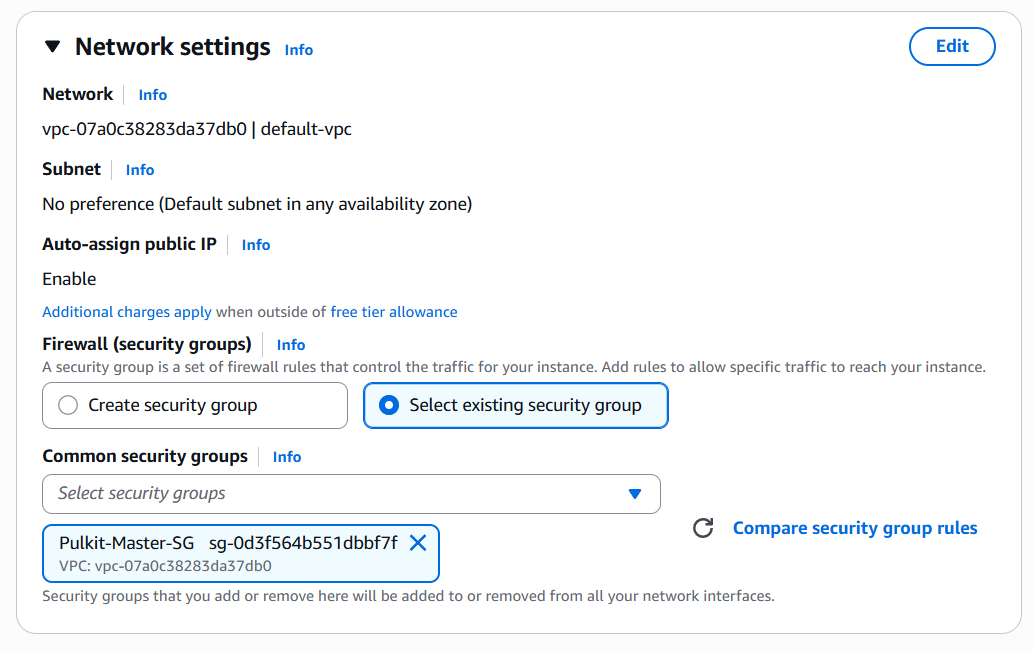
* TCP, Port = 8080, Source = Anywhere 0.0.0.0/0

(for Jenkins)



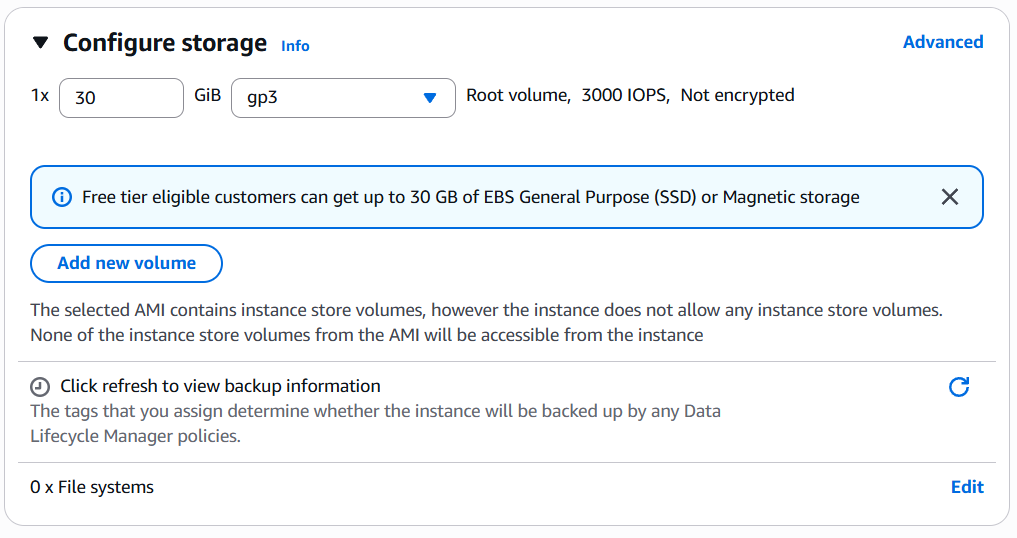


Select created security group and make sure Auto-assign public IP is enabled



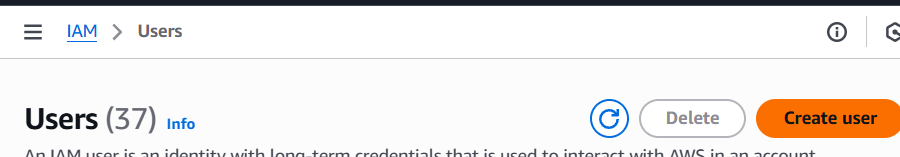
e) Configure Storage as 1x30 GiB gp3

then click on Launch Instance



**Step 2 . Create IAM User**

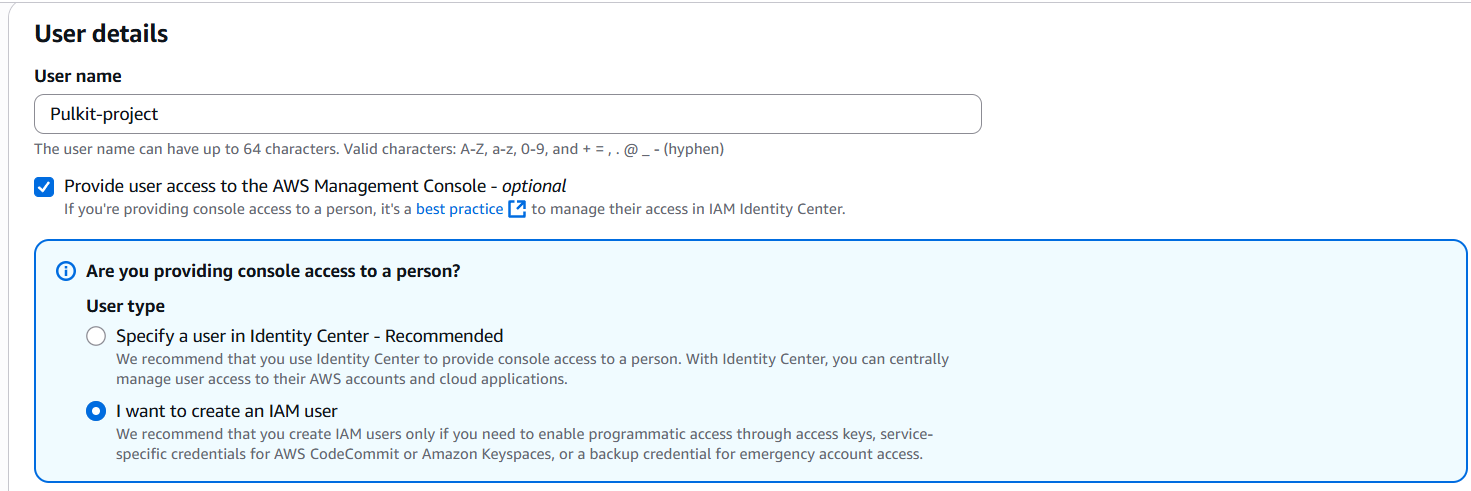
a) Open IAM in AWS console and click on create user

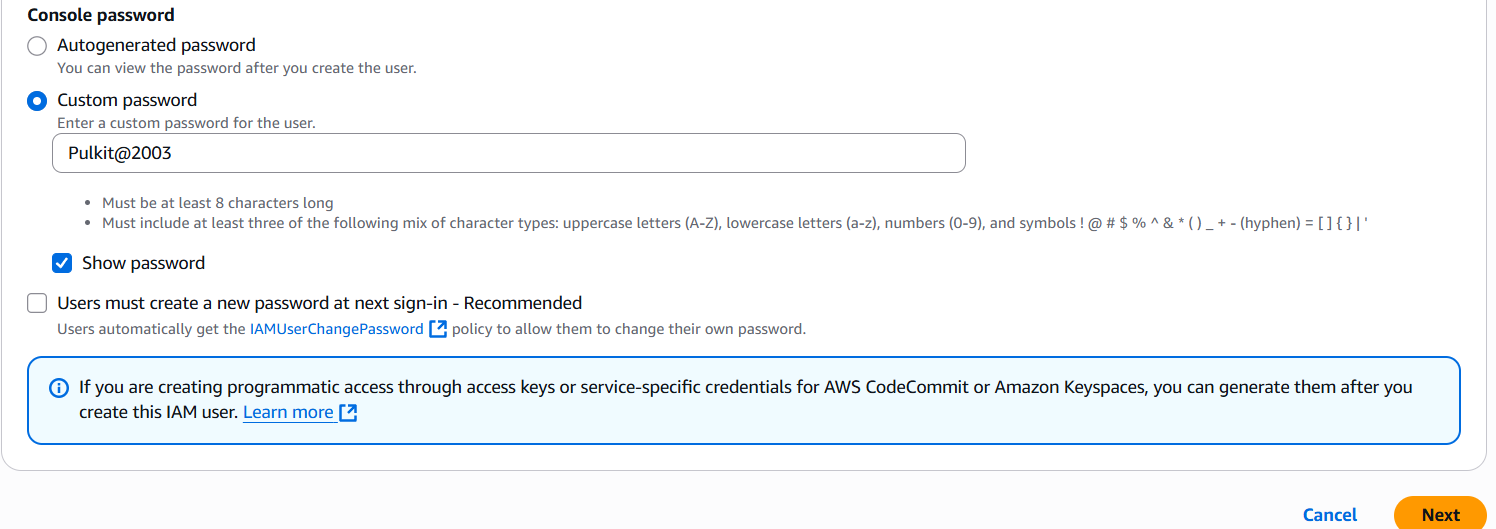


b) Give name then

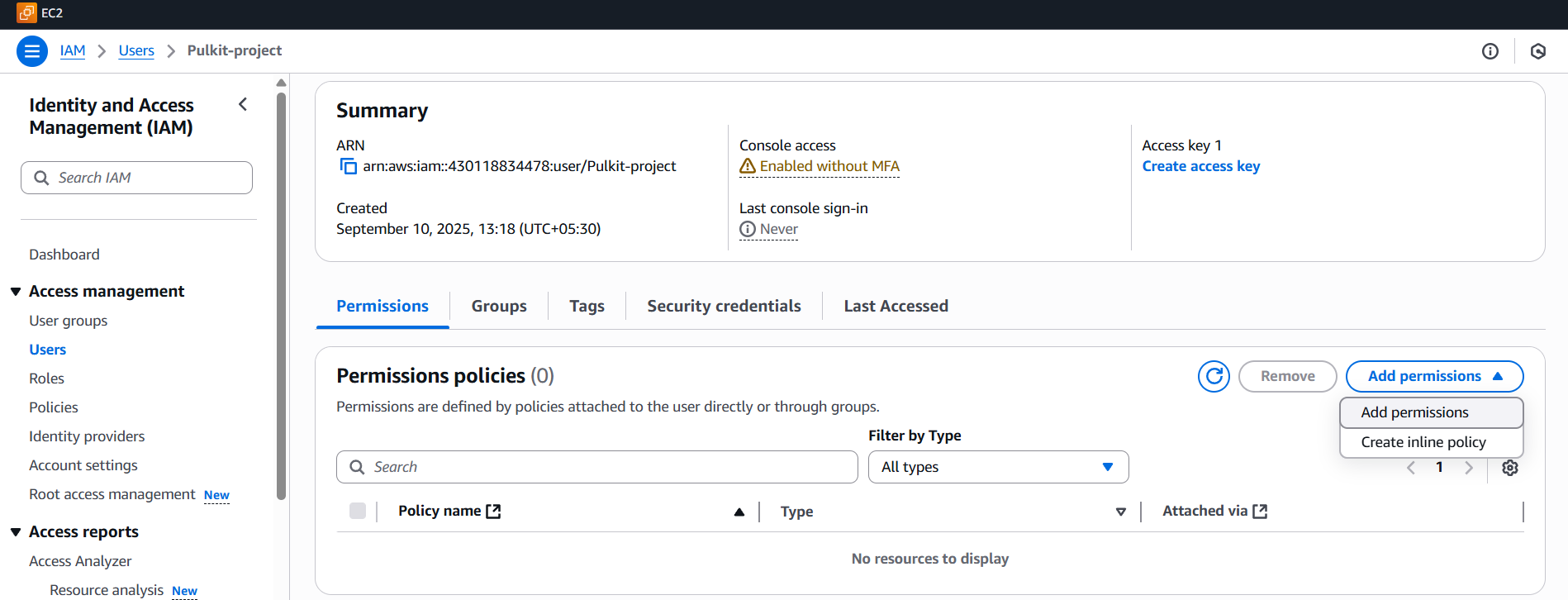
Check Provide user access -> Select I want to create an IAM user -> Click Custom Password and enter Password -> Unckeck User must create a new password….. option -> Click next

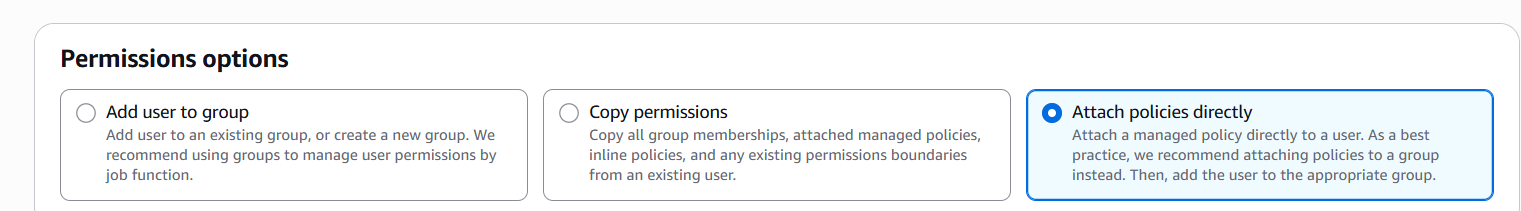
Skip everything -> Click next -> Click Create user



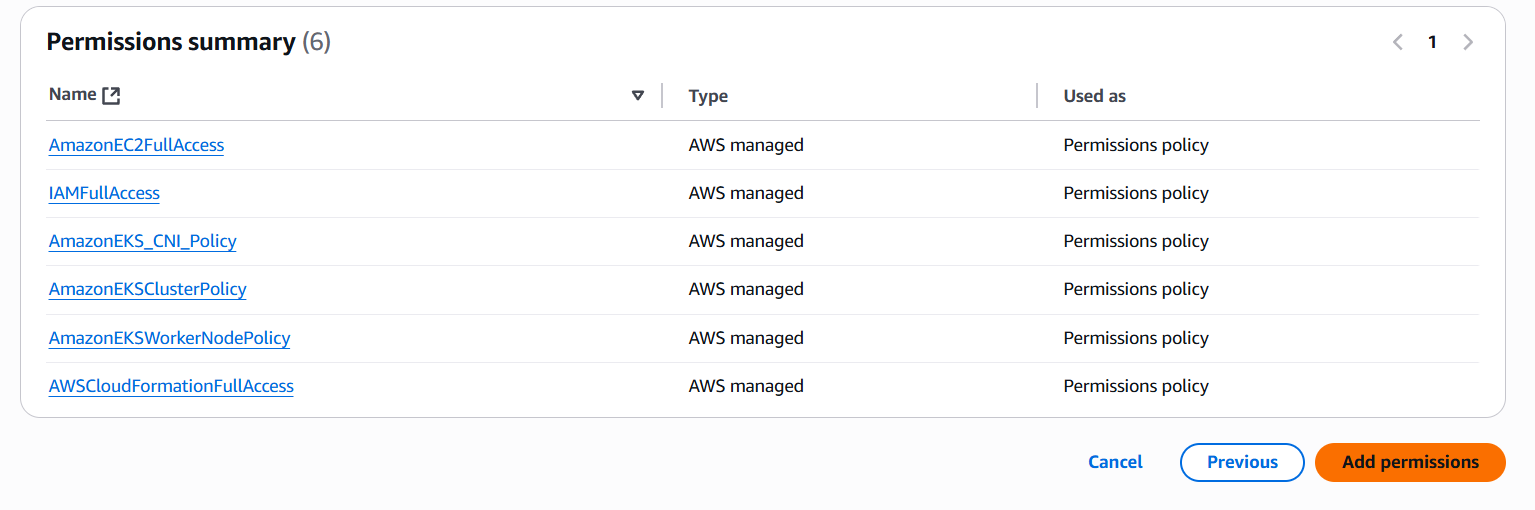


c) Move back to user list and select your user -> click Add Permissions -> Click on Attach Policies Directly

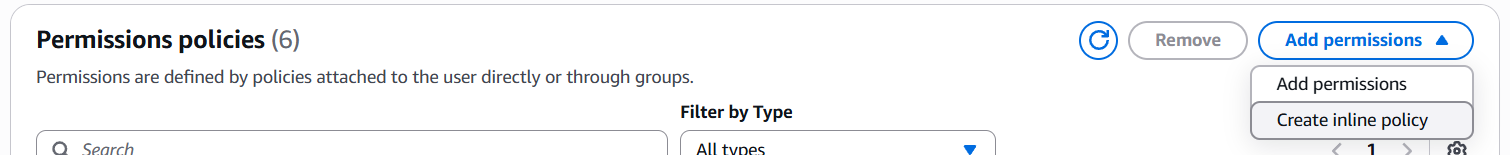




Add the 6 policies listed below



Click on Add Permission -> Create inline policy



Click jSON and replace the editor code with this -> Click next

{

    "Version": "2012-10-17",

    "Statement": [

        {

            "Sid": "VisualEditor0",

            "Effect": "Allow",

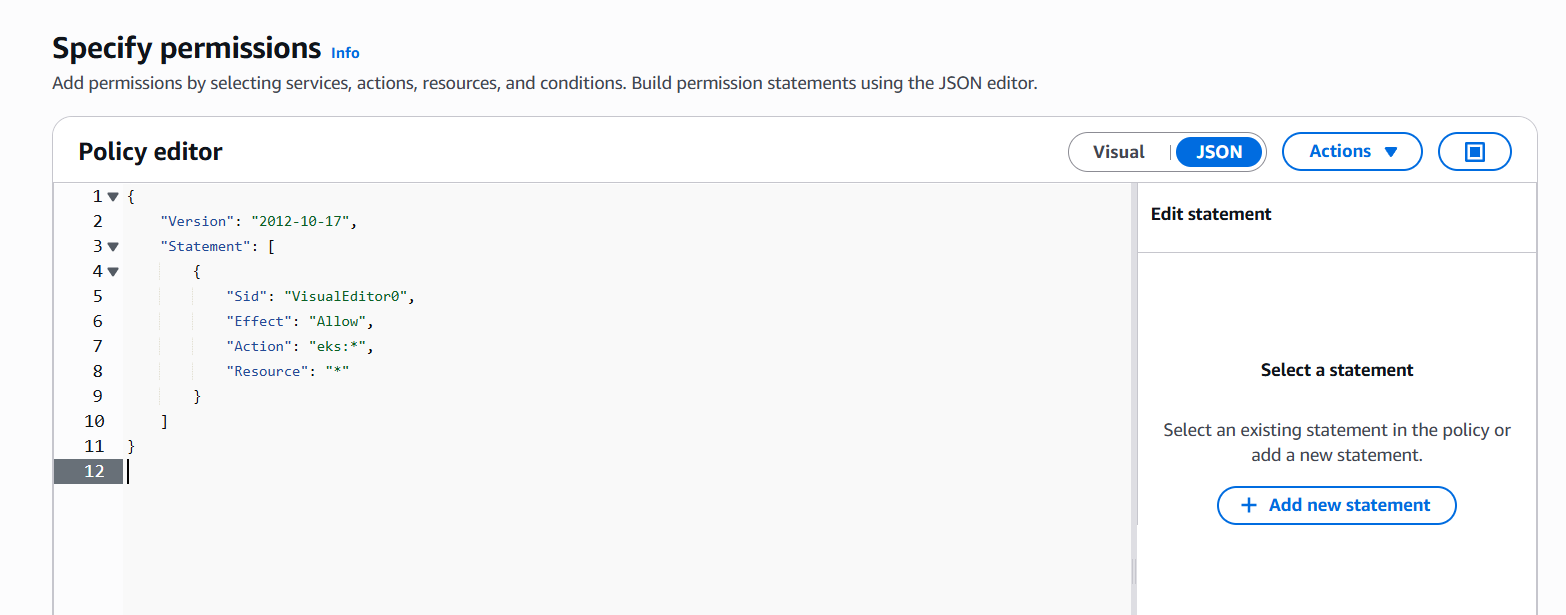
            "Action": "eks:\*",

            "Resource": "\*"

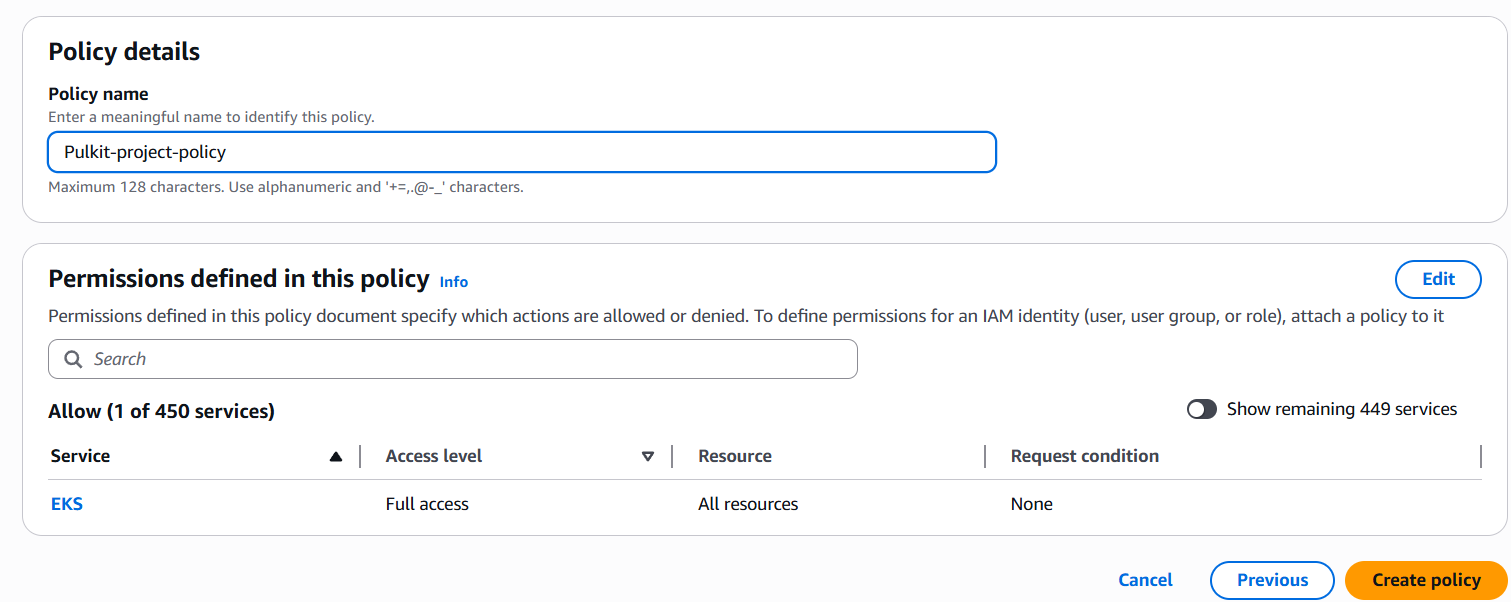
        }

    ]

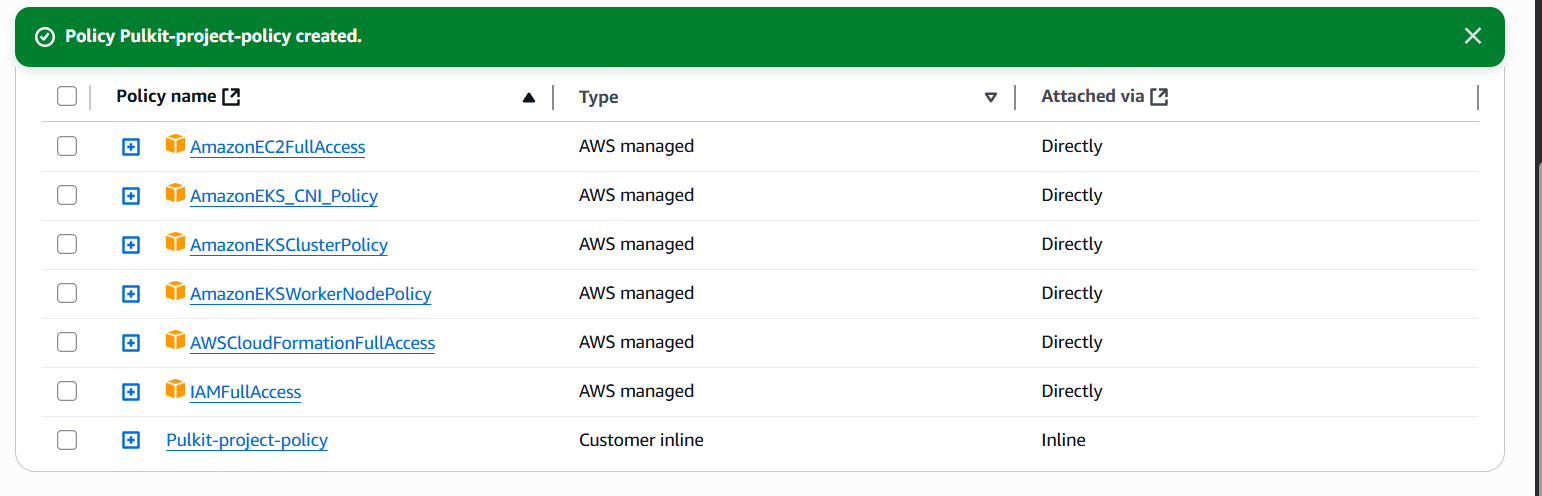
}



Give name and click on Create policy

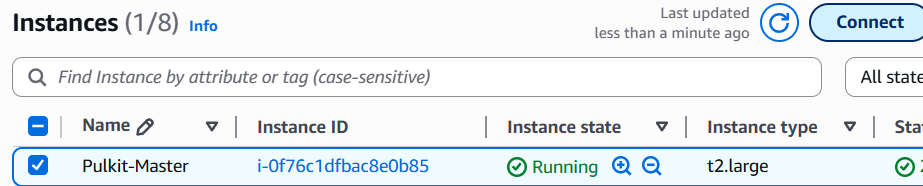


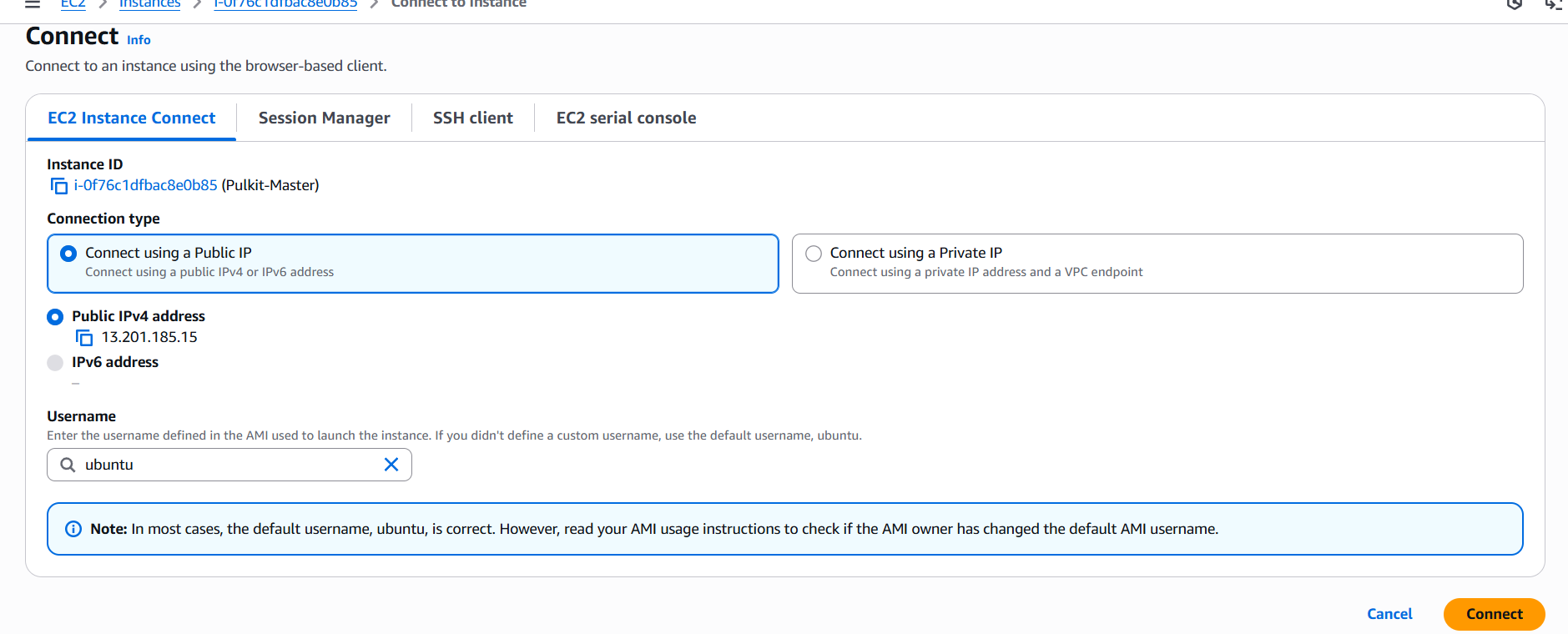
Now you will have 7 policies attached to user



**Step 3. Installations**

a) Connect EC2 to ssh





b) Run basic commands

sudo su

apt update -y



c) Jenkins

vi Jenkins.sh

#!/bin/bash

# Install OpenJDK 17 JRE Headless

sudo apt install openjdk-17-jre-headless -y

# Download Jenkins GPG key

sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \

  https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key

# Add Jenkins repository to package manager sources

echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \

  https://pkg.jenkins.io/debian-stable binary/ | sudo tee \

  /etc/apt/sources.list.d/jenkins.list > /dev/null

# Update package manager repositories

sudo apt-get update

# Install Jenkins

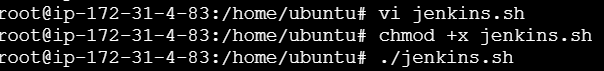
sudo apt-get install jenkins -y

press - esc :wq

give permissions and run the file

chmod +x jenkin.sh

./jenkins.sh

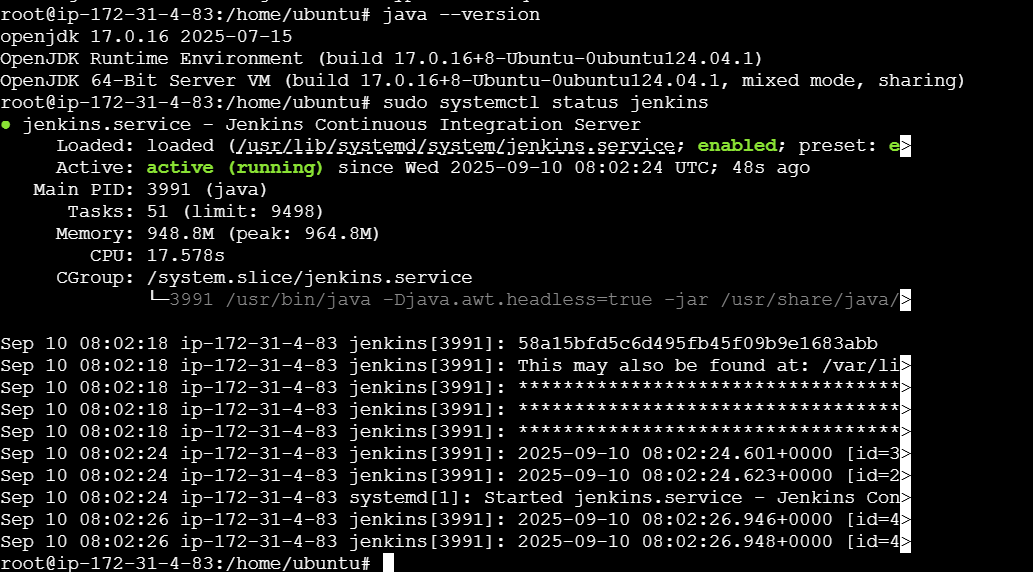


Check for java and Jenkins working

java –version

systemctl status Jenkins

press q to exit

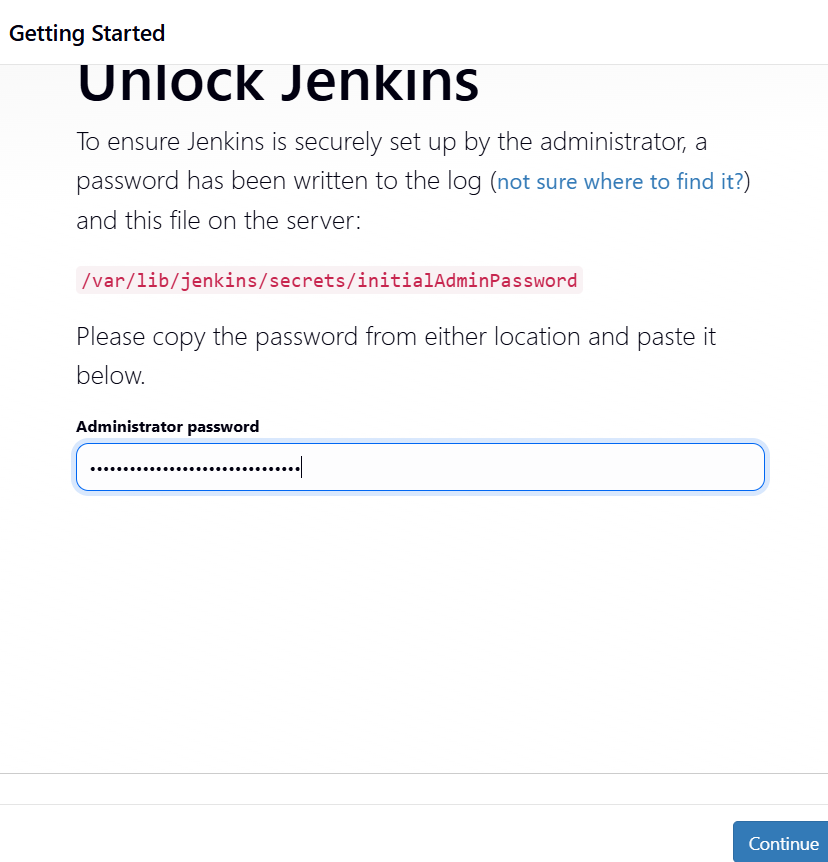


Steup Jenkins Open Port 8080 in Jenkins server

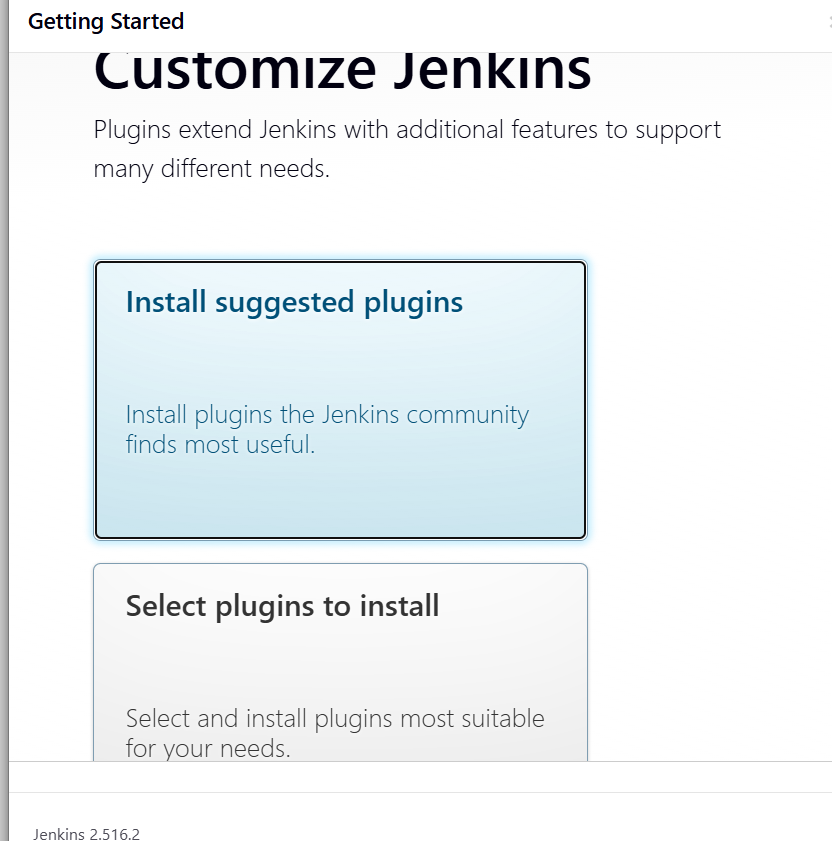
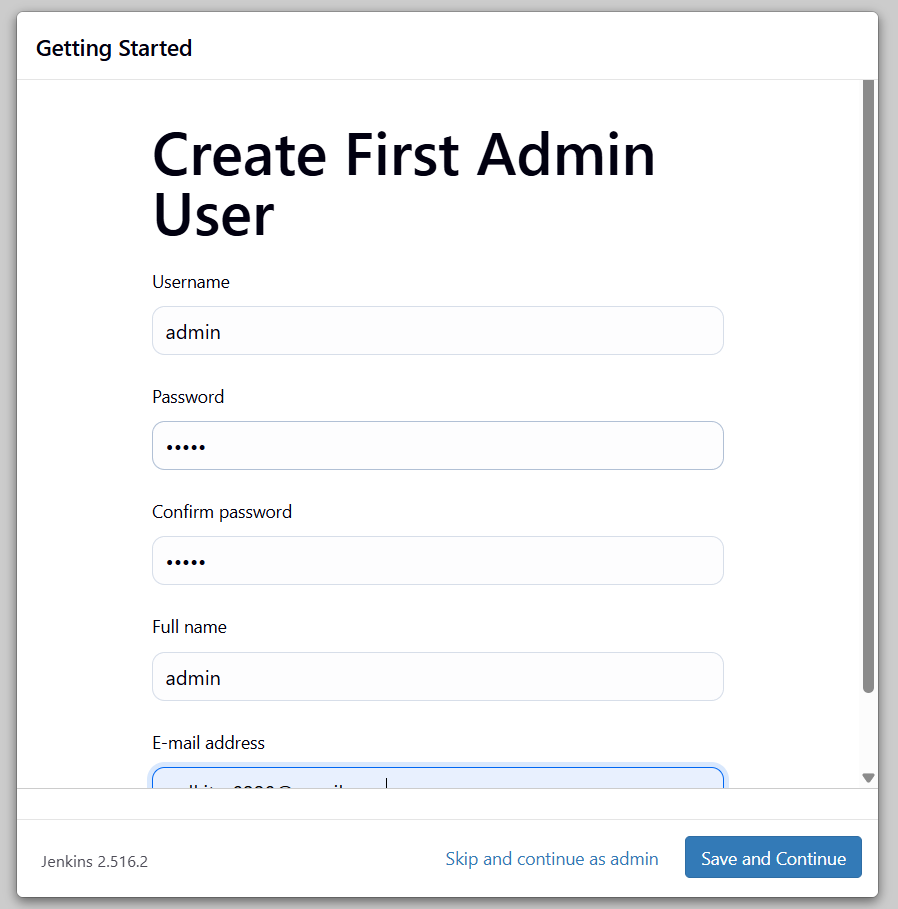
<ip\_address>:8080

cat /var/lib/jenkins/secrets/initialAdminPassword

copy and paste in Administration password



Click on install all suggested plugins

d) Docker

vi docker.sh

#!/bin/bash

# Update package manager repositories

sudo apt-get update

# Install necessary dependencies

sudo apt-get install -y ca-certificates curl

# Create directory for Docker GPG key

sudo install -m 0755 -d /etc/apt/keyrings

# Download Docker's GPG key

sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc

# Ensure proper permissions for the key

sudo chmod a+r /etc/apt/keyrings/docker.asc

# Add Docker repository to Apt sources

echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu \

$(. /etc/os-release && echo "$VERSION\_CODENAME") stable" | \

sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

# Update package manager repositories

sudo apt-get update

sudo apt-get install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin

press – esc :wq

give permission and run the file

chmod +x docker.sh

./docker.sh



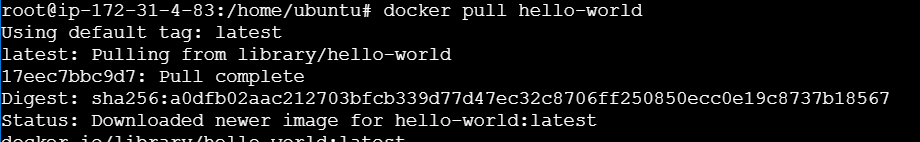
Check docker version

docker –version



lets pull a default imagedocker pull hello-world   
  
if it shows permission denied execute this command   
  
sudo chmod 666 /var/run/docker.sock

docker pull hello-world



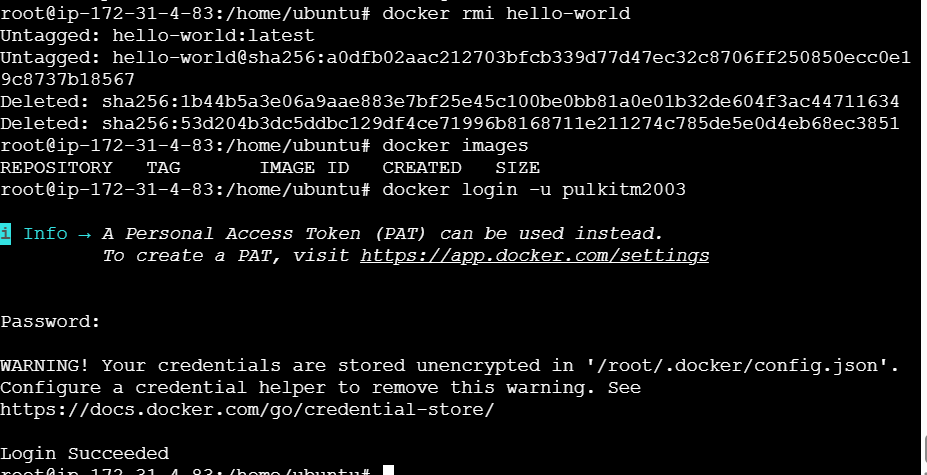
Remove the image

docker rmi hello-world:latest  
docker images

no images should be there after this open docker hub on browser and login there otherwise next step might show error

docker login -u <DockerHubUserName>

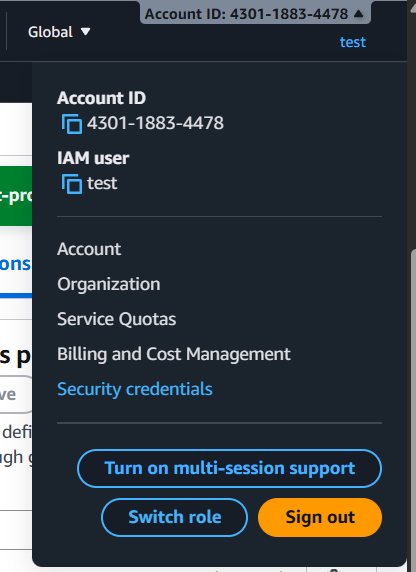
Click Enter -> Enter the password of your DockerHub



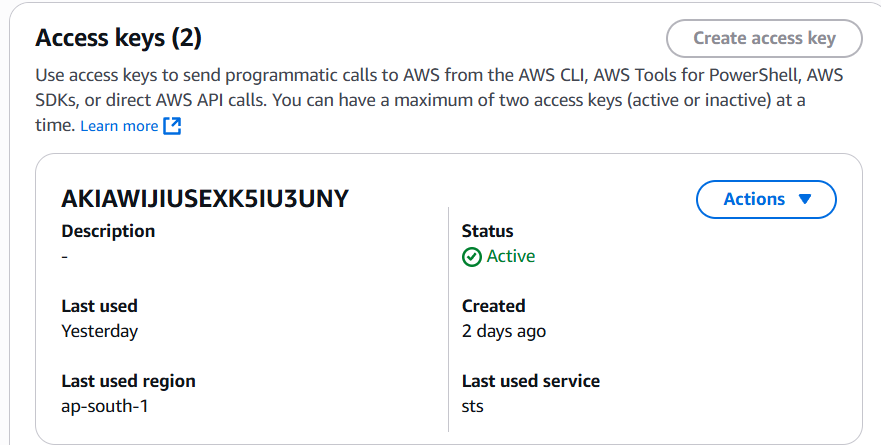
e) AWS cli

First create Access key and Secret Access key

Go to security credentials



Scroll down and click on create access key and create one   
 since I am using sir’s account thus I will use the given access key



Now run below commands in ssh

vi aws.sh  
  
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86\_64.zip" -o "awscliv2.zip"

sudo apt install unzip

unzip awscliv2.zip

sudo ./aws/install

press- esc :wq

give permission and run file

chmod +x aws.sh

./aws.sh



Check for aws version

aws –version



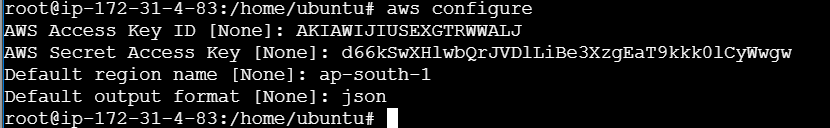
Lets configure aws run below command

aws configure

give your access keys and secret keys

region: <your\_working\_region>

output fromat : json



f) Kubeclt

vi kubectl.sh

curl -o kubectl https://amazon-eks.s3.us-west-2.amazonaws.com/1.19.6/2021-01-05/bin/linux/amd64/kubectl

chmod +x ./kubectl

sudo mv ./kubectl /usr/local/bin

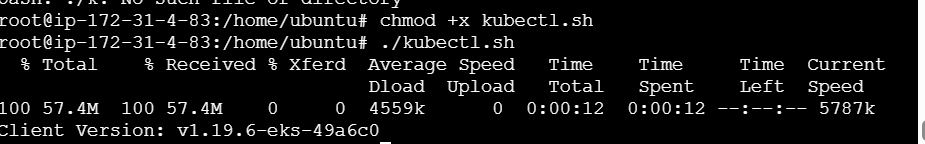
kubectl version --short --client

press – esc :wq

give permission and runn the file

chmod +x kubectl.sh

./kubectl.sh



g) eksctl

vi eksctl.sh

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

eksctl version

give permission and run file

chmod +x eksctl.sh

./eksctl.sh

  
  
  
 h) Ansible

vi ansible.sh

add-apt-repository --yes --update ppa:ansible/ansible

apt install ansible -y

give permission and run file

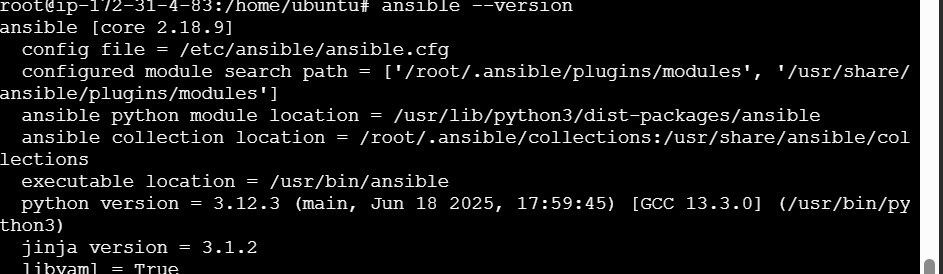
chmod +x ansible.sh

./ansible.sh



Check ansible version

ansible --version



**Step 4. Create EKS Cluster**

a) Create EKS Cluster ( I am using pre existing VPC and subnet )

vi cluster.sh

#!/bin/bash

# --- Set your subnets ---

SUBNETS="<your\_subnet\_a>,<your\_subnet\_b>"

# --- Create EKS cluster using existing VPC ---

eksctl create cluster --name <your\_cluster\_name>\

--region <your\_region\_name> \

--version 1.30 \

--without-nodegroup \

--vpc-private-subnets $SUBNETS \

Give permission and run file

chmod +x cluster.sh

./cluster.sh

b) Association of eks

vi ekassociation.sh

eksctl utils associate-iam-oidc-provider \

--region <your\_region\_name> \

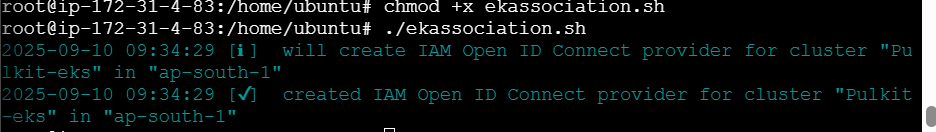
--cluster <your\_cluster\_name>\

--approve

Give permission and run file

chmod +x ekassociation.sh

./ekassociation.sh



c) Node group and Node Creation   
vi createnode.sh

eksctl create nodegroup \

--cluster <your\_cluster\_name> \

--region <your\_region> \

--name node2 \

--node-type t3.medium \

--nodes 2 \

--nodes-min 1 \

--nodes-max 2 \

--node-volume-size 20 \

--ssh-access \

--ssh-public-key <your\_key\_name> \

--managed \

--asg-access \

--external-dns-access \

--full-ecr-access \

--appmesh-access \

--alb-ingress-access \

--node-private-networking \

--subnet-ids <your\_subnet\_id>,<your\_subnet\_id> \

--tags "Environment=dev"

Give permissions and run file

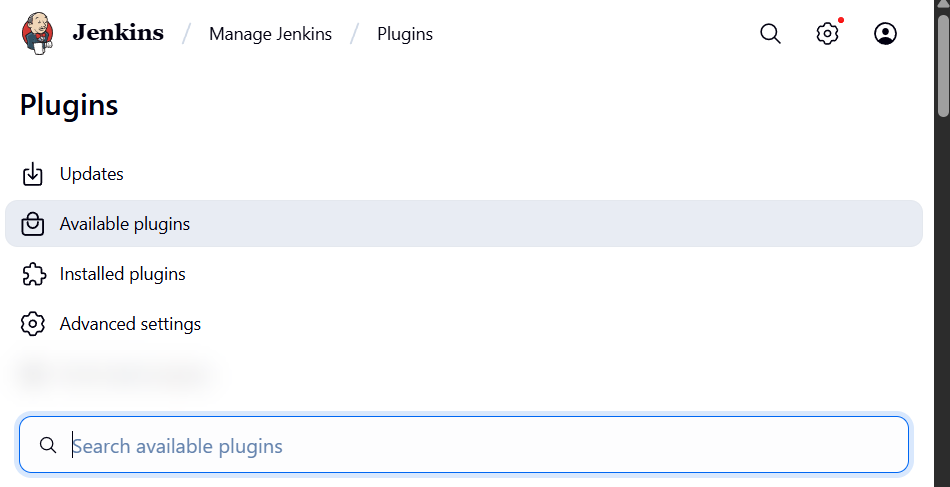
chmod +x createnode.sh

./createnode.sh



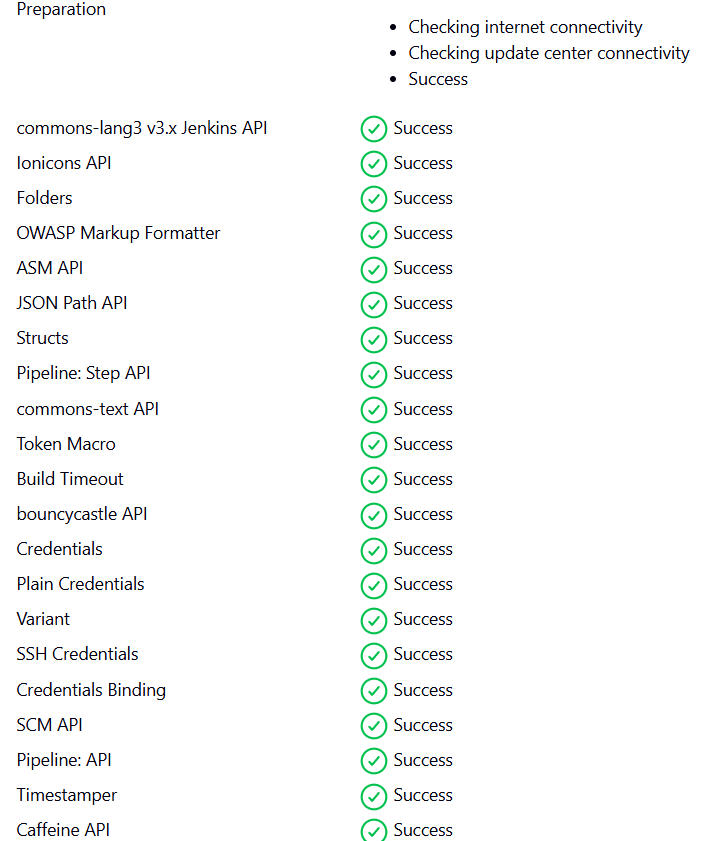
**Step 5. Establishing Jenkins Plugins**

a) Open Jenkins -> Manage Jenkins -> Plugins -> Available Plugins

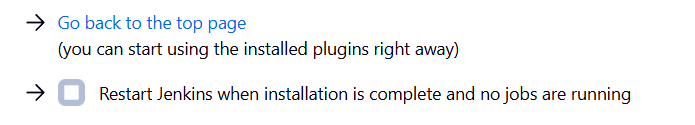


Install all 13 Plugins listed below and after that restart Jenkins

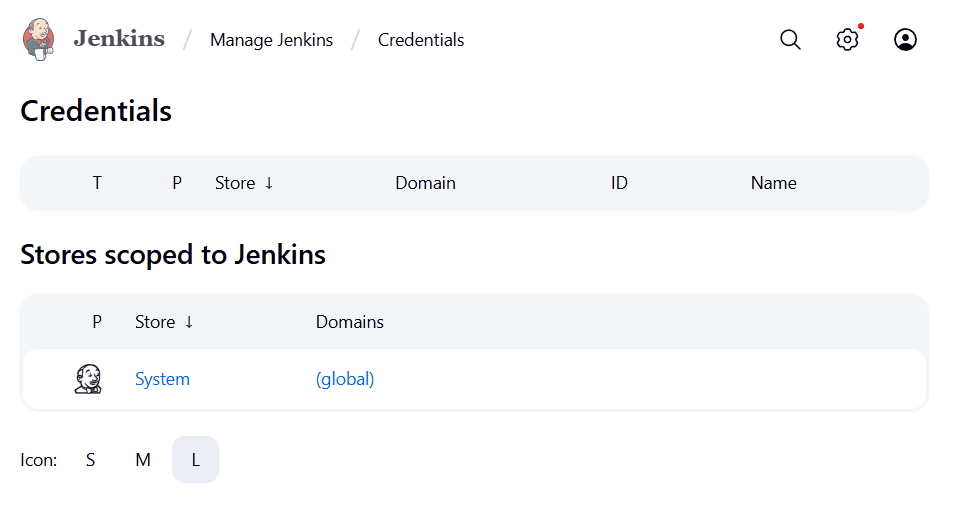
1. Pipeline: Stage View
2. Docker
3. Docker Commons
4. Docker Pipeline
5. Docker API
6. docker-build-step
7. Kubernetes
8. Kubernetes Client API
9. Kubernetes Credentials
10. Kubernetes CLI
11. Config File Provider
12. Eclipse Temurin Installer (for Java)
13. Ansible

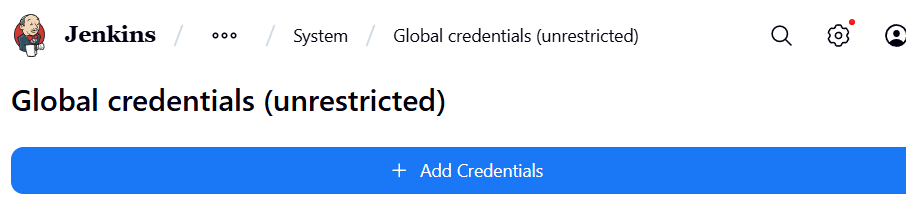


after installation check on Restart option



b) Go to Manage jenikns -> Credentials (Under security) ->click global -> Add credentials





Make below changes

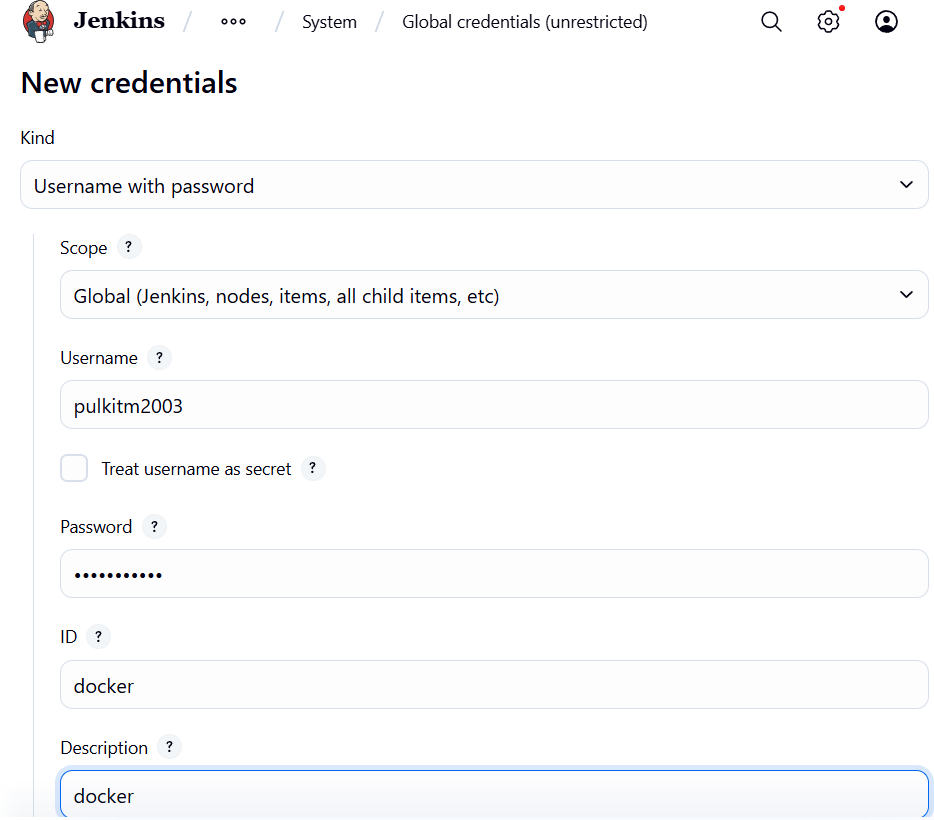
User name = your docker username

Password = your docker hub password

Id = docker

Description = docker

Click create



Add global credentials for ansible too

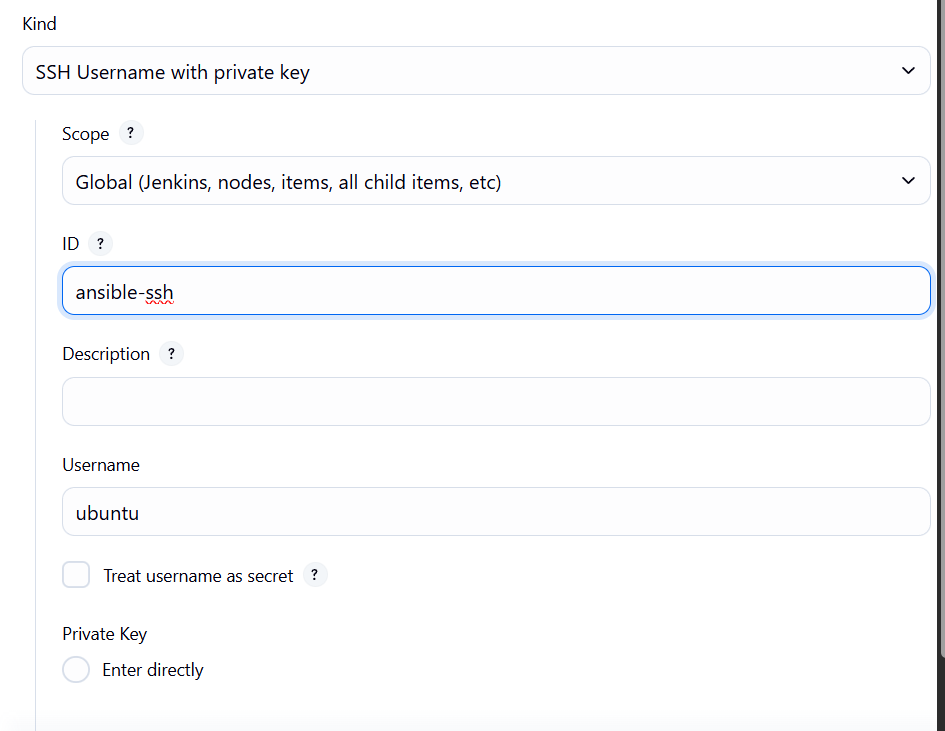
Kind : SSH Username with private key

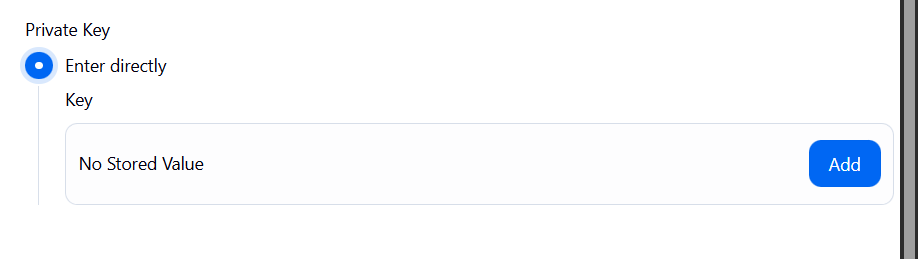
Scope : Global

ID : ansible-ssh

Username : ubuntu

Click on Private key enter directly and enter the private key used to make master ec2





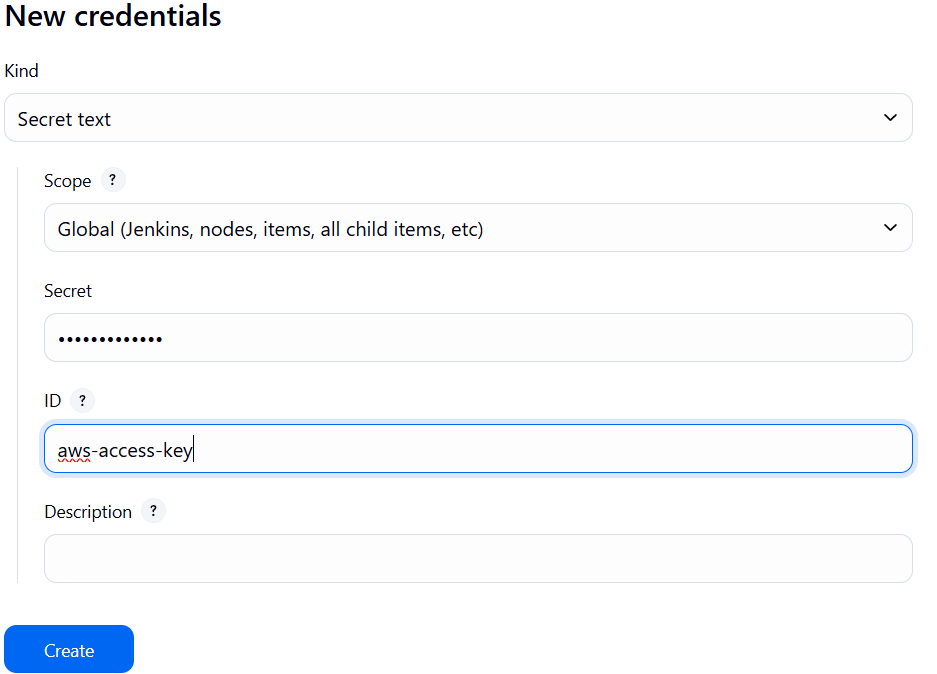
Add Credentials for AWS access key

Kind : Secret key

Scope : Global

Secret : <add\_accesskey>

ID : aws-access-key



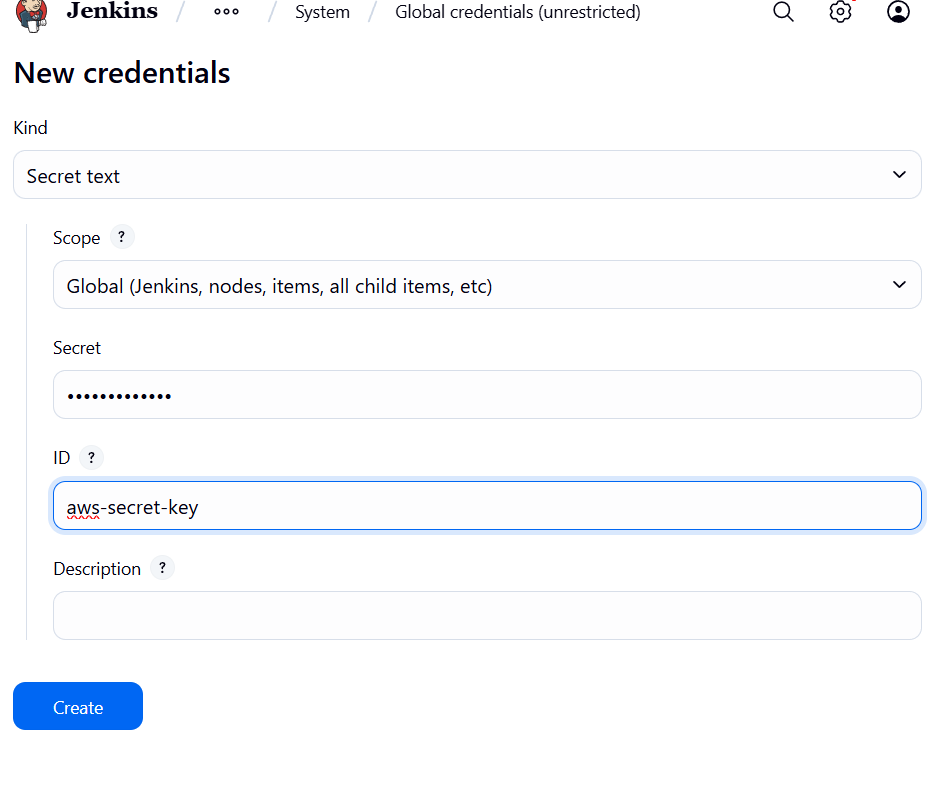
Add Credentials for AWS Secret access key

Kind : Secret key

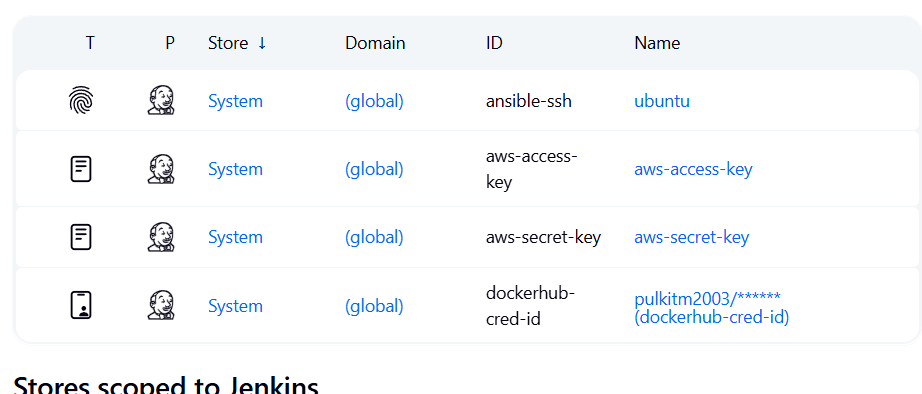
Scope : Global

Secret : <add\_Secret\_accesskey>

ID : aws-secret-key



Click on create



c) go to -> Manage Jenkins -> Tools (under system configuration) ->

click on Add jdk  
name: jdk17  
select install automatically

under installer dropdown  
select “install from adoptium.net”   
version? : select jdk-17.0.8.1+1 version  
-----------------------------------------------------------------------------  
add docker   
name: docker

select install automatically  
add installer  
Download from docker.com  
--------------------------------------------------------------------------

add ansible  
name: ansible

leave as it is  
--------------------------------------------------------------------------  
  
click on apply and save

**Step 6. Creating Dynamic Inventory**

a) Give different names

since all our nodes have same names thus we need to give them different names to differentiate here we will use the tag we gave earlier Environment = dev

vi tags.sh

# Fetch instance IDs that match Environment=dev and Role=web

instance\_ids=$(aws ec2 describe-instances \

--filters "Name=tag:Environment,Values=dev" "Name=instance-state-name,Values=running" \

--query 'Reservations[\*].Instances[\*].InstanceId' \

--output text)

# Sort instance IDs deterministically

sorted\_ids=($(echo "$instance\_ids" | tr '\t' '\n' | sort))

# Rename instances sequentially

counter=1

for id in "${sorted\_ids[@]}"; do

name="<name\_you\_want>-$(printf "%02d" $counter)"

echo "Tagging $id as $name"

aws ec2 create-tags --resources "$id" \

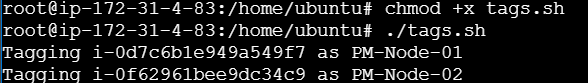
--tags Key=Name,Value="$name"

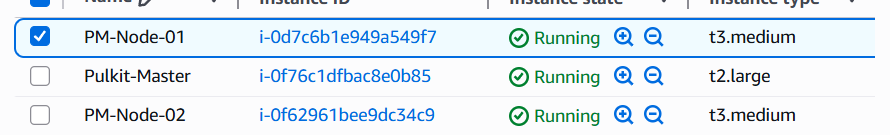
((counter++))

done

press esc:wq

give permission and run file

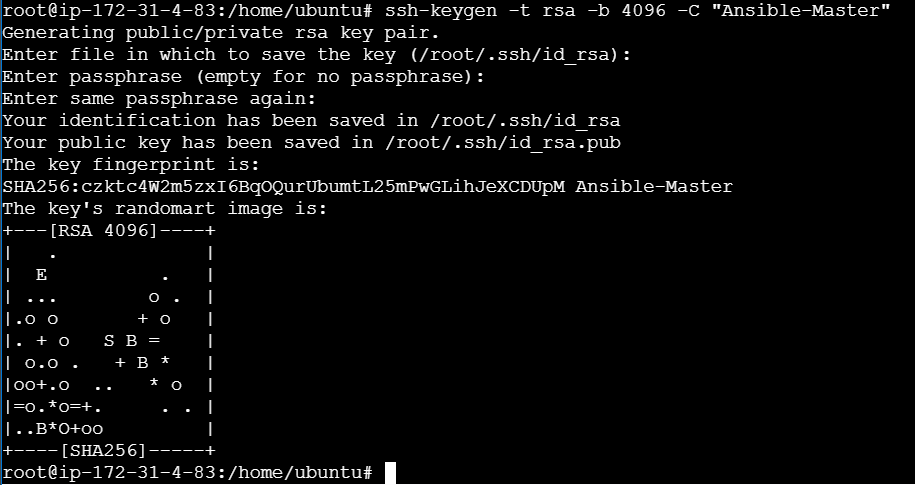




b) Create Ansible key

ssh-keygen -t rsa -b 4096 -C "Ansible-Master"

press enter 3 times



c) Creating Yaml file that will hold IP address of node

mkdir inventory

vi inventory/aws\_ec2.yaml

plugin: amazon.aws.aws\_ec2

regions:

* <your\_region\_name>

filters:

tag:Environment: dev

instance-state-name: running

compose:

ansible\_host: public\_ip\_address

keyed\_groups:

- key: tags.Name

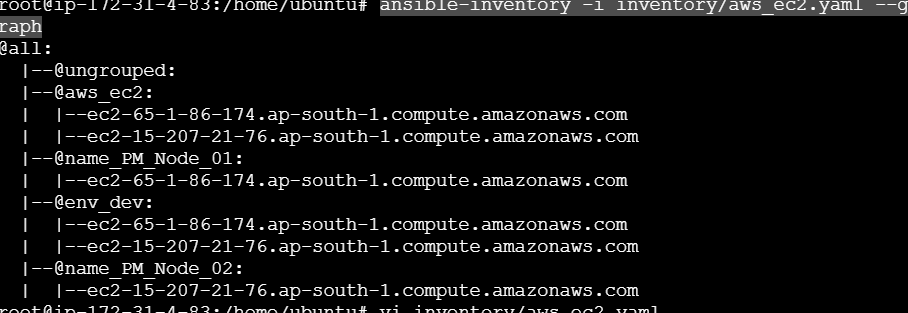
prefix: name

- key: tags.Environment

prefix: env

to test run this command

ansible-inventory -i inventory/aws\_ec2.yaml –graph



**Step 7. Connecting Ansible Master - Node**

a) Creating Ansible Environment

We are setting up separate environment to isolate Ansible dependencies

Install venv module if not already present :

sudo apt install python3-venv -y

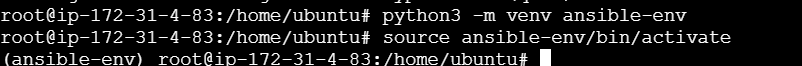
Create a virtual environment :

python3 -m venv ansible-env

Activate it :

source ansible-env/bin/activate

we can see now we have entered into ansible environment

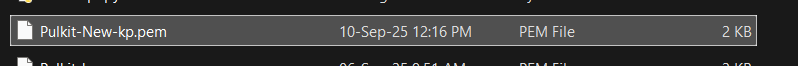


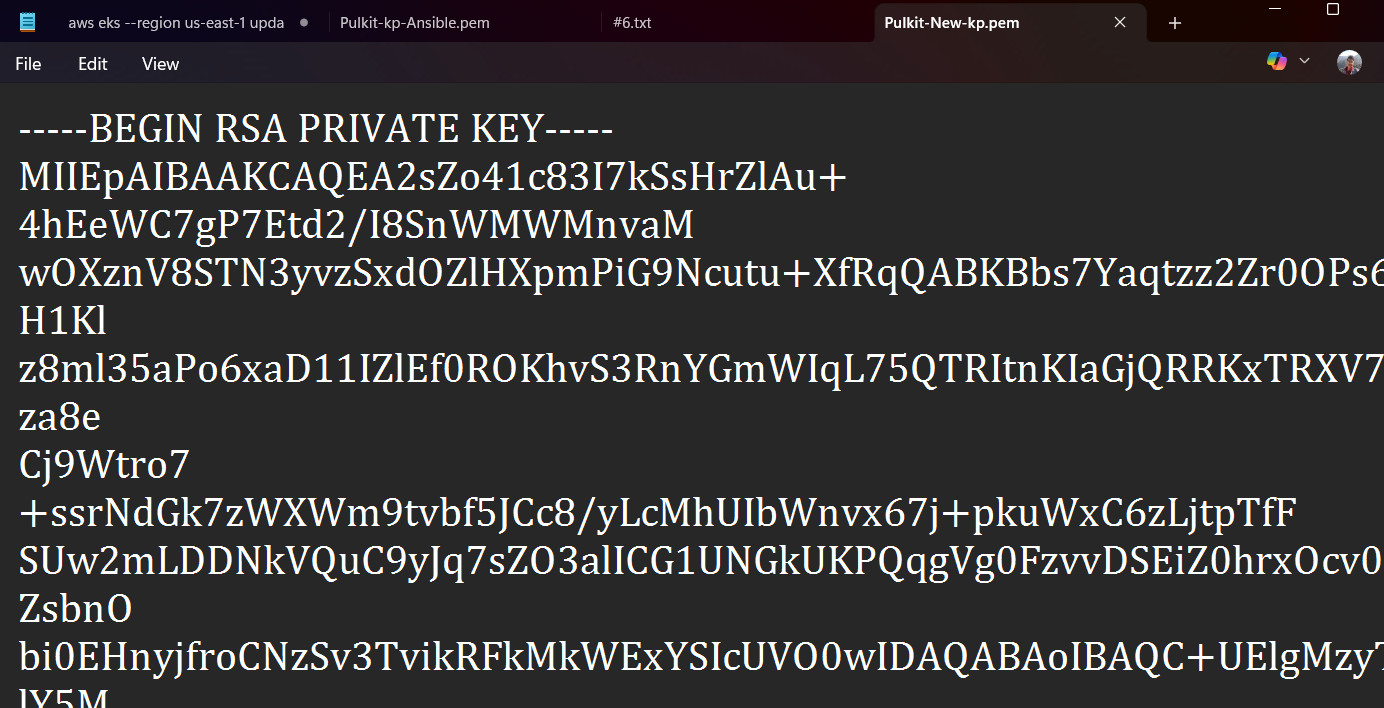
Install required Python packages :

pip install boto3 botocore docker

b) Creating Key file

Open the key pair file in notepad and copy its content





Paste this content in file below

vi Pulkit-New-kp.pem

(name should be same as key pair used to create master and nodes )



After pasting key value here press – esc:wq

Give read only permission

Chmod 400 Pulkit-New-kp.pem



c) Give over ride permission

since while adding key in nodes nodes require permission as yes but we cannot open each node and give it permission as there are multiple nodes thus we will over ride those permissions

vi ansible.cfg

[defaults]

inventory = ./inventory/aws\_ec2.yaml

host\_key\_checking = False

[ssh\_connection]

ssh\_args = -o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null

d) Script to connect Nodes and Master

vi copy-public-key.sh

#!/bin/bash

# Define vars

PEM\_FILE="Pulkit-New-kp.pem"

PUB\_KEY=$(cat ~/.ssh/id\_rsa.pub)

USER="ec2-user" # Since nodegroup is Linux

INVENTORY\_FILE="inventory/aws\_ec2.yaml"

# Extract hostnames/IPs from dynamic inventory

HOSTS=$(ansible-inventory -i $INVENTORY\_FILE --list | jq -r '.\_meta.hostvars | keys[]')

for HOST in $HOSTS; do

  echo "Injecting key into $HOST"

  ssh -o StrictHostKeyChecking=no -i $PEM\_FILE $USER@$HOST "

    mkdir -p ~/.ssh && \

    echo \"$PUB\_KEY\" >> ~/.ssh/authorized\_keys && \

    chmod 700 ~/.ssh && \

    chmod 600 ~/.ssh/authorized\_keys

  "

done

give permission and run file

chmod +x copy-public-key.sh

./copy-public-key.sh

**Step 8. Final Deployment Stage**

a) Give Jenkins Permission to ansible environment

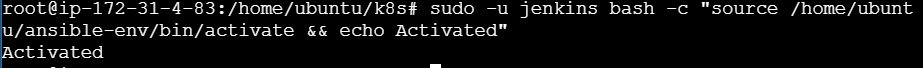
sudo chown -R jenkins:jenkins /home/ubuntu/ansible-env

sudo chmod -R 755 /home/ubuntu/ansible-env

sudo chmod 755 /home/ubuntu  
  
 verify

sudo -u jenkins ls -l /home/ubuntu/ansible-env/bin/activate

sudo -u jenkins bash -c "source/home/ubuntu/ansible-env/bin/activate && echo Activated"



a) Give docker permission in ansible environment

sudo usermod -aG docker jenkins

sudo systemctl restart Jenkins

b) Exiting Ansible environment after checking the requirements

# Inside ansible-env

source /home/ubuntu/ansible-env/bin/activate

pip install --upgrade pip

pip install ansible boto3 botocore kubernetes

ansible-galaxy collection install kubernetes.core

deactivate

c) Making Kubernetes deployment files

make the below files on both ssh and Github

On Github create a new k8s folder and makes files in it

Make files on forked repository

mkdir k8s

vi k8s/deployment.yaml

apiVersion: apps/v1

kind: Deployment

metadata:

name: apache-deployment

spec:

replicas: 2

selector:

matchLabels:

app: apache

template:

metadata:

labels:

app: apache

spec:

containers:

- name: apache

image: <your\_docker\_id>/apachewebsite:latest # Your Docker Hub image

ports:

- containerPort: 80

vi k8s/service.yaml

apiVersion: v1

kind: Service

metadata:

name: apache-service

spec:

type: LoadBalancer

selector:

app: apache

ports:

- protocol: TCP

port: 80

targetPort: 80

d) Ansible Deployment file make it on Github on Master branch

name of file deploy-k8s.yaml

- name: Deploy Apache website to EKS

hosts: localhost

connection: local

tasks:

- name: Ensure AWS CLI is configured for EKS

command: aws eks update-kubeconfig --name <your\_cluster\_name> --region <your\_region\_name>

- name: Deploy Apache Deployment

kubernetes.core.k8s:

state: present

namespace: default

src: k8s/deployment.yaml

- name: Deploy Apache Service

kubernetes.core.k8s:

state: present

namespace: default

src: k8s/service.yaml

- name: Verify deployment pods

command: kubectl get pods -o wide

register: pod\_status

- debug:

var: pod\_status.stdout\_lines

e) Make Jenkins file on Github

name of file Jenkinsfile

pipeline {

agent any

environment {

DOCKER\_IMAGE = "<docker\_id>/apachewebsite:latest"

}

stages {

stage('Checkout SCM') {

steps {

checkout scm

}

}

stage('Docker Build & Push') {

steps {

script {

withDockerRegistry([credentialsId: 'dockerhub-cred-id', url: 'https://index.docker.io/v1/']) {

sh '''

echo "Building Docker image..."

docker build -t $DOCKER\_IMAGE .

echo "Pushing Docker image to DockerHub..."

docker push $DOCKER\_IMAGE

'''

}

}

}

}

stage('Prepare Kubernetes Manifests') {

steps {

sh '''

echo "Copying Kubernetes manifests into Jenkins workspace..."

mkdir -p k8s

cp /home/ubuntu/k8s/service.yaml k8s/service.yaml

cp /home/ubuntu/k8s/deployment.yaml k8s/deployment.yaml

'''

}

}

stage('Deploy to EKS via Ansible') {

steps {

withCredentials([

string(credentialsId: 'aws-access-key', variable: 'AWS\_ACCESS\_KEY\_ID'),

string(credentialsId: 'aws-secret-key', variable: 'AWS\_SECRET\_ACCESS\_KEY')

]) {

sh '''

echo "Exporting AWS credentials..."

export AWS\_ACCESS\_KEY\_ID=$AWS\_ACCESS\_KEY\_ID

export AWS\_SECRET\_ACCESS\_KEY=$AWS\_SECRET\_ACCESS\_KEY

export AWS\_DEFAULT\_REGION=ap-south-1

echo "Activating Ansible virtual environment..."

. /home/ubuntu/ansible-env/bin/activate

echo "Running Ansible playbook for Kubernetes deployment..."

ansible-playbook deploy-k8s.yaml

'''

}

}

}

stage('Verify Deployment') {

steps {

echo "Stage skipped if deployment failed"

}

}

}

post {

success {

echo "✅ Deployment succeeded!"

}

failure {

echo "❌ Deployment failed!"

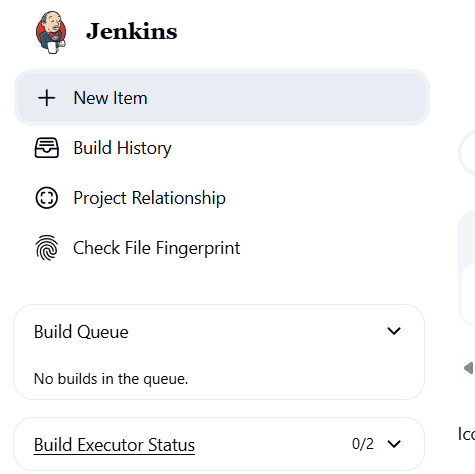
}

}

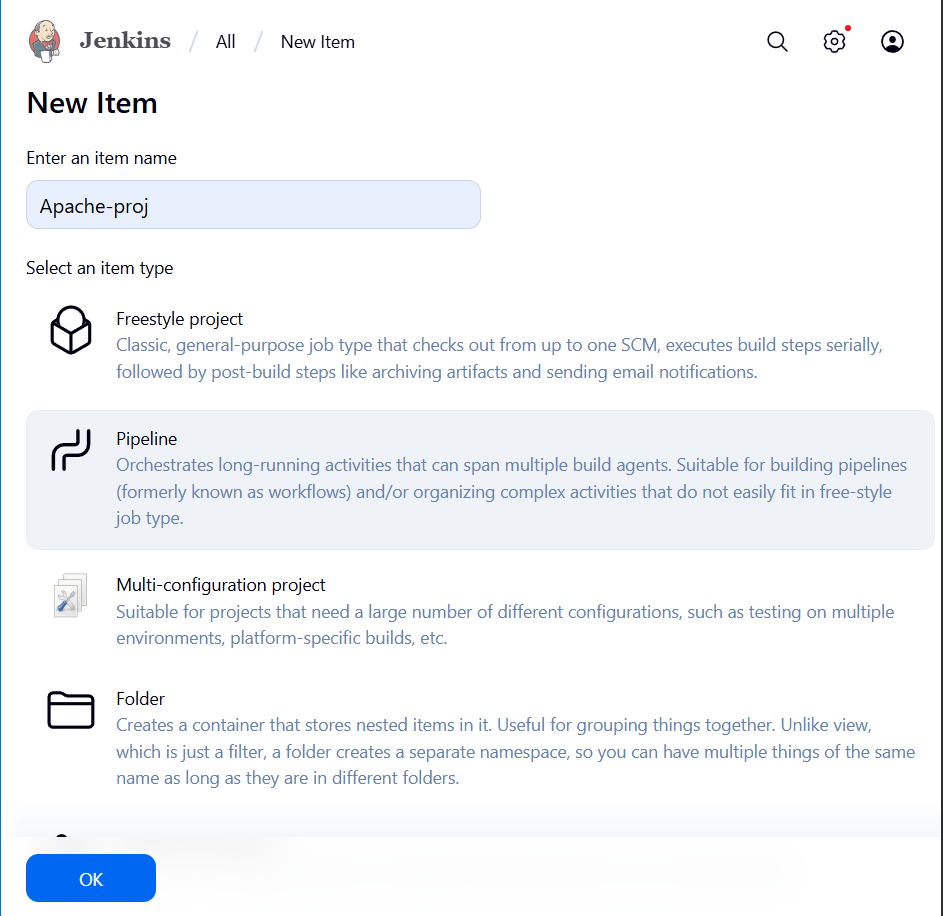
}

**Step 9. Jenkins pipeline**

a) Click New Item



Enter name -> Select Pipeline option -> click on OK

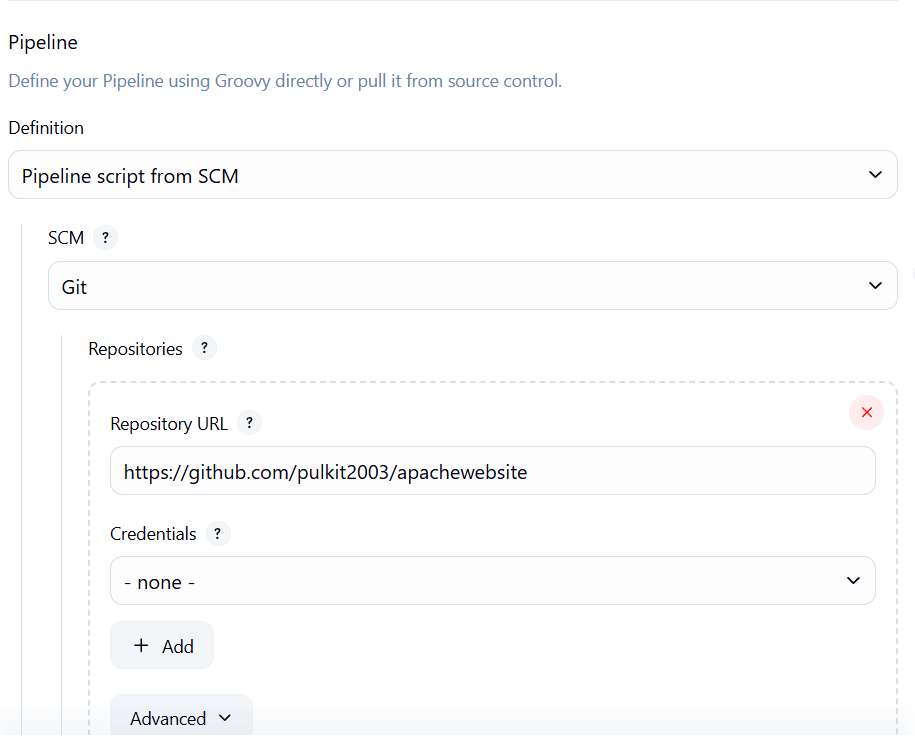
****

Scroll down to Pipeline section

Select Defination -> Pipeline script from SCM

Select SCM as Git

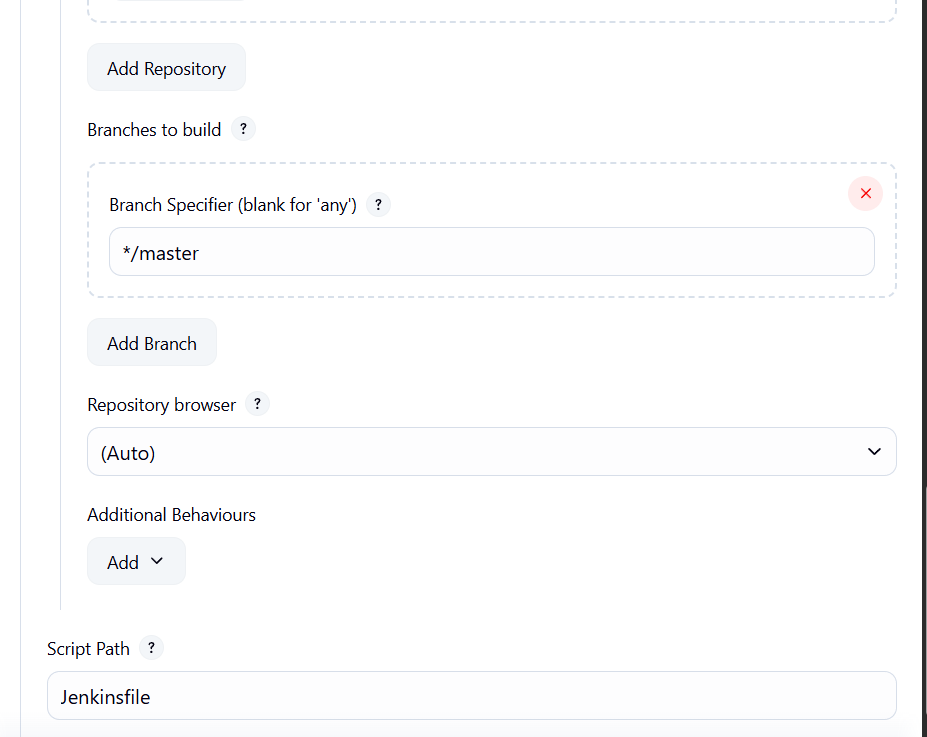
Under Repositories Add <your repository url>



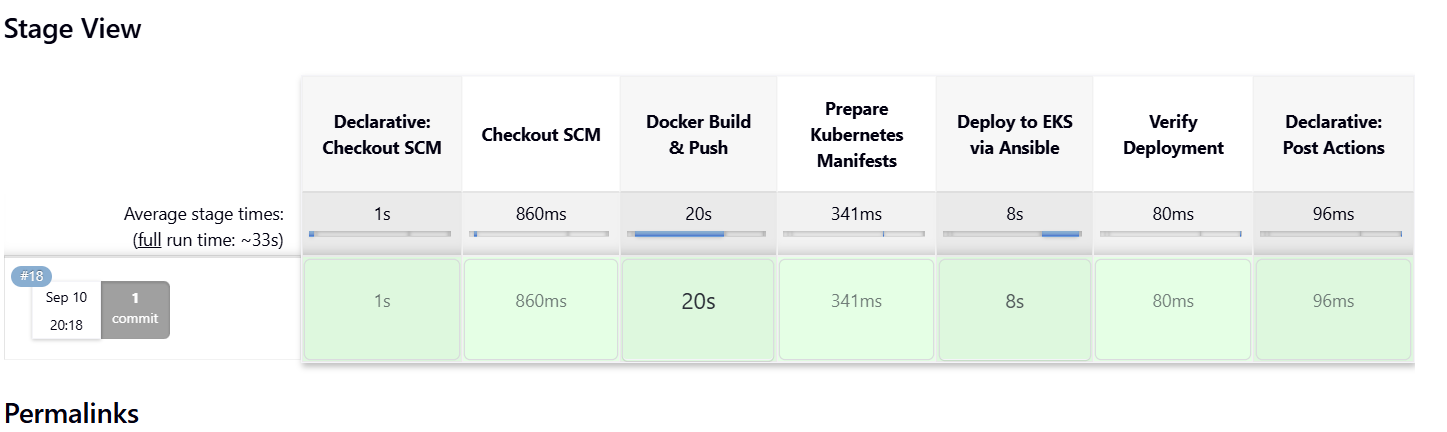
Select Master as Branch

Script path as Jenkinfile

Then click Apply and Save

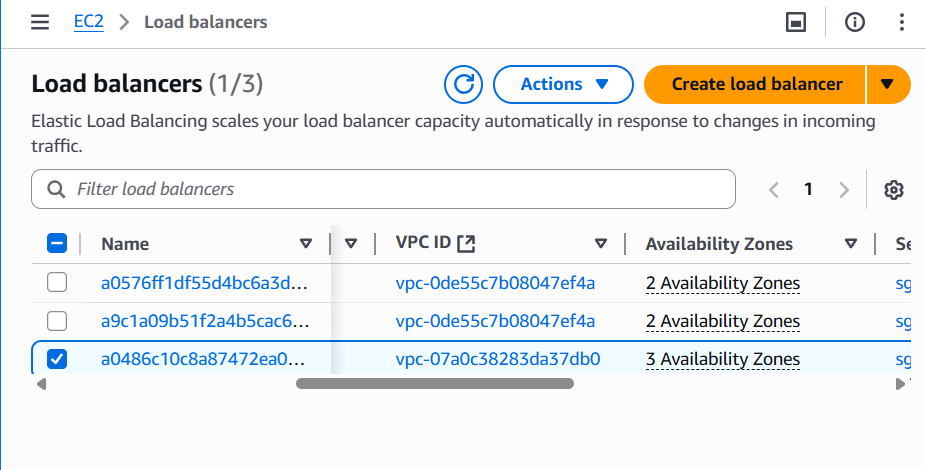


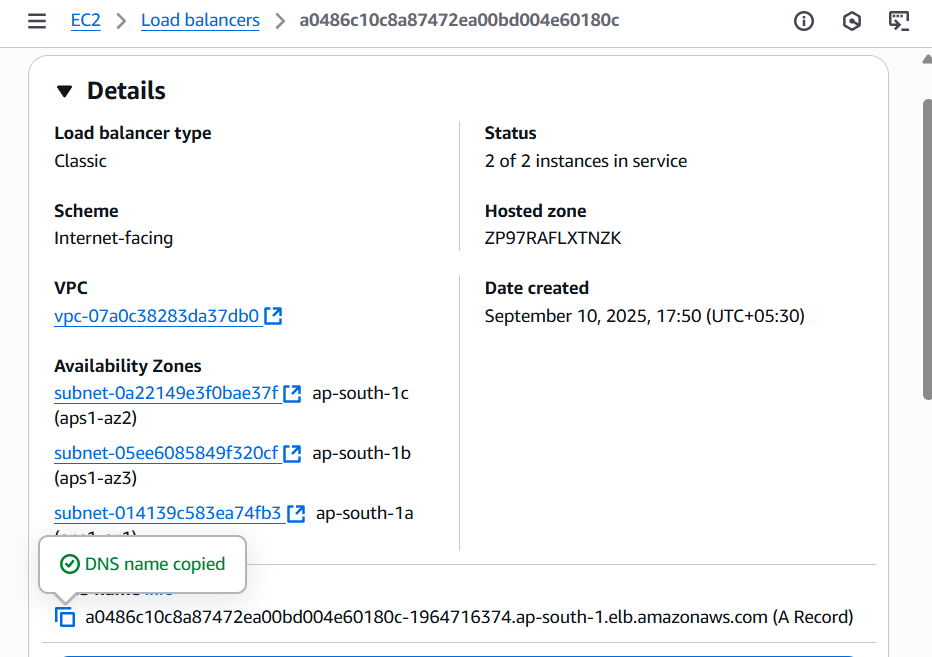
b)Run pipeline

****

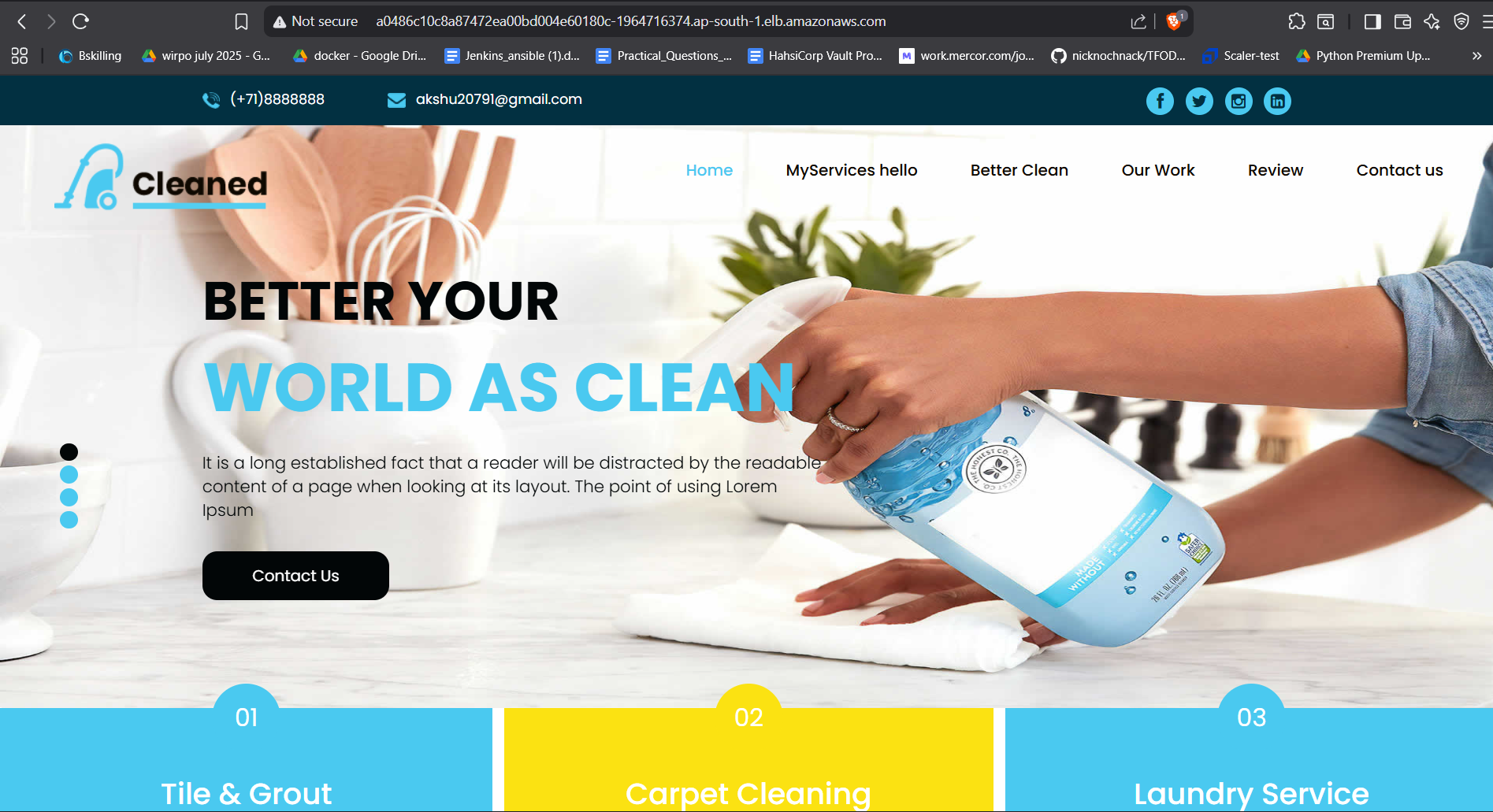
**Step 10. Confirmation**

a) Open Load Balancer and select your Load Balancer that has been created Copy the DNS link and open on Browser

****



**The site is running**

****