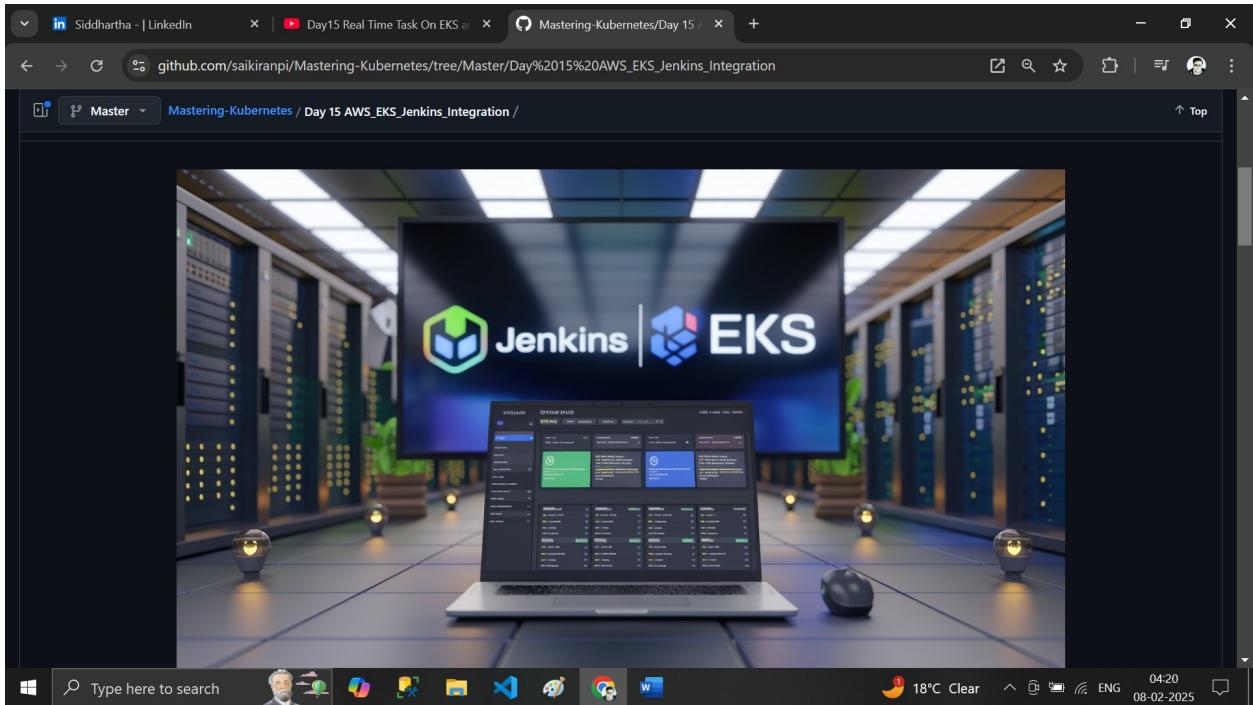


15 Real Time Task On EKS and Jenkins End to End Pipeline Integration

how to set up an Amazon EKS cluster and connect it with Jenkins for automated deployments. We'll start by launching an EC2 instance, setting up the necessary tools, and then deploying an EKS cluster. Finally, I'll walk you through integrating Jenkins with GitHub to automatically trigger deployments based on branch changes. Whether you're new to DevOps or just looking to automate your workflow, this step-by-step guide will help you get started quickly!

- 1. Intro**
- 2. Creating T2 Large Instance with 20GB**
- 3. Creating IAM User**
- 4. Installing Unzip and AwsCli**
- 5. Installing Kubectl and Ekctl**
- 6. Installing Control plan**



The screenshot shows a Microsoft Edge browser window with the following details:

- Tab 1: Siddhartha - LinkedIn
- Tab 2: Day15 Real Time Task On EKS
- Tab 3: Mastering-Kubernetes/Day 15 /
- Current Tab: Mastering-Kubernetes / Day 15 AWS_EKS_Jenkins_Integration /

The main content is titled "EKS Cluster Deployment and Jenkins Integration". It includes sections for "Overview" and "Prerequisites".

Overview

This repository contains scripts and instructions for automating the deployment of an EKS Cluster and its integration with Jenkins for continuous deployment. The setup includes creating an EKS control plane, configuring OIDC, deploying node groups, and setting up a Jenkins pipeline for automated deployments.

Prerequisites

Before you begin, ensure you have the following:

- An AWS account with the necessary IAM permissions:
 - AmazonEKSClusterPolicy
 - AmazonEKSWorkerNodePolicy
 - AmazonEKSServicePolicy
- An EC2 instance with:
 - t2.large type
 - 20GB storage
- SSH Key pair for EC2 access.
- Jenkins server installed on the EC2 instance.

At the bottom, there is a Windows taskbar with various icons and a system tray showing the date and time (08-02-2025).

The screenshot shows a Microsoft Edge browser window with the following details:

- Tab 1: Siddhartha - LinkedIn
- Tab 2: Day15 Real Time Task On EKS
- Tab 3: Mastering-Kubernetes/Day 15 /
- Current Tab: Mastering-Kubernetes / Day 15 AWS_EKS_Jenkins_Integration /

The main content is titled "Installation". It includes three numbered steps: 1. EC2 Instance Setup, 2. Install kubectl, and 3. Install eksctl.

1. EC2 Instance Setup

Launch a t2.large EC2 instance with 20GB storage.
SSH into your instance and install the required dependencies:

```
sudo apt update  
sudo apt install -y unzip awscli openjdk-11-jdk
```

2. Install kubectl

```
curl -LO "https://storage.googleapis.com/kubernetes-release/release/$(curl -s https://storage.googleapis.com/kubernetes-release/index.yaml | grep -o 'kubernetes-.*\.tgz' | tail -1)"  
curl -LO "https://storage.googleapis.com/kubernetes-release/release/$(curl -s https://storage.googleapis.com/kubernetes-release/index.yaml | grep -o 'kubernetes-.*\.tgz' | tail -1)"  
sudo mv ./kubectl /usr/local/bin/kubectl  
chmod 777 /usr/local/bin/kubectl  
kubectl version --short
```

3. Install eksctl

```
curl --silent --location "https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_$(uname -s)_amd64.tar.gz" | tar -x  
curl --silent --location "https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_$(uname -s)_amd64.tar.gz" | tar -x  
sudo mv /tmp/eksctl /usr/local/bin  
sudo chmod 700 /usr/local/bin/eksctl  
eksctl version
```

At the bottom, there is a Windows taskbar with various icons and a system tray showing the date and time (08-02-2025).

Siddhartha - | LinkedIn | Day15 Real Time Task On EKS | Mastering-Kubernetes/Day 15 / +

github.com/saikiranpi/Mastering-Kubernetes/tree/Master/Day%2015%20AWS_EKS_Jenkins_Integration

Master | Master -> Mastering-Kubernetes / Day 15 AWS_EKS_Jenkins_Integration / ↑ Top

Cluster Deployment

1. Create EKS Control Plane

Run the following command to create the EKS cluster control plane:

```
eksctl create cluster \
--name eks-cluster-1 \
--version 1.29 \
--zones=us-east-1a,us-east-1b,us-east-1c \
--without-nodegroup
```

2. Associate OIDC Provider

```
eksctl utils associate-iam-oidc-provider \
--region us-east-1 \
--cluster eks-cluster-1 \
--approve
```

3. Create Node Group

Siddhartha - | LinkedIn | Day15 Real Time Task On EKS | Mastering-Kubernetes/Day 15 / +

github.com/saikiranpi/Mastering-Kubernetes/tree/Master/Day%2015%20AWS_EKS_Jenkins_Integration

Master | Mastering-Kubernetes / Day 15 AWS_EKS_Jenkins_Integration / ↑ Top

```
eksctl create nodegroup --cluster=eks-cluster-1 \
--region=us-east-1 \
--name=eks-cluster-1-node-1 \
--node-type=t3.medium \
--nodes=2 \
--nodes-min=2 \
--nodes-max=4 \
--node-volume-size=20 \
--ssh-access \
--ssh-public-key=YourKeyPair \
--managed \
--asg-access \
--external-dns-access \
--full-ecr-access \
--appmesh-access \
--alb-ingress-access
```

Jenkins Integration

1. Install Jenkins Plugins

Ensure you have installed the following plugins:

- AWS CLI Plugin
- Kustomization Plugin

Type here to search  18°C Clear 04:21 08-02-2025 ENG

2. Configure Jenkins Credentials

- Add your AWS credentials to Jenkins under `Manage Jenkins > Credentials`.
- Set up an SSH key pair for Jenkins to access the GitHub repository.

3. Create Multibranch Pipeline

- Set up a Multibranch Pipeline in Jenkins.
- Configure a GitHub webhook to trigger builds automatically.

Testing

- Run `kubectl get pods -A` to ensure your cluster is up and running.
- Test the Jenkins pipeline by pushing code to your GitHub repository. The pipeline should automatically trigger and deploy to the appropriate namespace (development or production).

Cleanup

To delete the EKS cluster:

```
eksctl delete cluster --name eks-cluster-1
```

Practical

steps

```
# Create 1 t2-Large Instance with 20gb
# Create IAM USER with Administrator access or individual access anything is fine
# AmazonEKSClusterPolicy
# AmazonEKSWorkerNodePolicy
# AmazonEKSServicePolicy
```

EC 2 instance

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IP
EKS-Jenkins	i-0cc39b1ffde18a629	Running	t2.xlarge	Initializing	View alarms +	us-east-1a	ec2-3-23

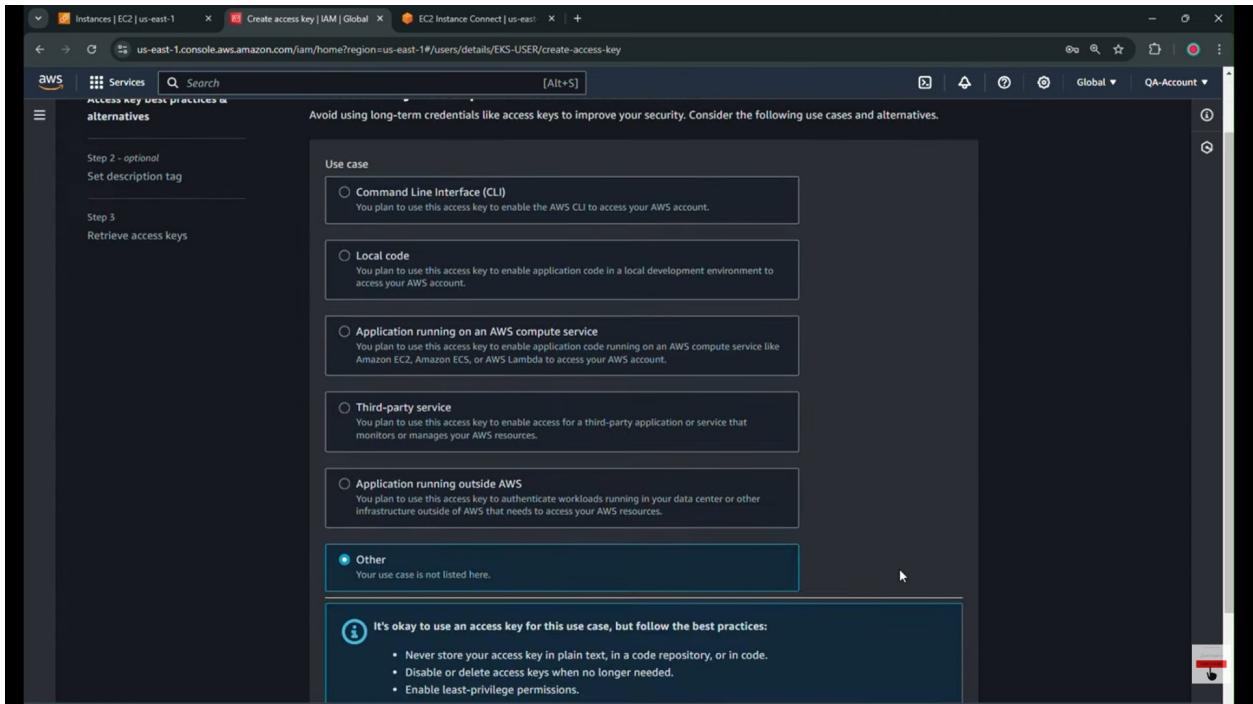
IAM user

The screenshot shows the AWS IAM Dashboard. On the left, a sidebar lists navigation options: Dashboard, Access management (with sub-options like User groups, Users, Roles, Policies, Identity providers, Account settings), Access reports (with sub-options like Access Analyzer, External access, Unused access, Analyzer settings, Credential report, Organization activity, Service control policies), and Related consoles. The main area is titled "IAM Dashboard" and contains three main sections: "Security recommendations" (0 notifications), "AWS Account" (status bar), and "Quick Links" (My security credentials, Policy simulator, Additional information). A central "What's new" section lists recent updates:

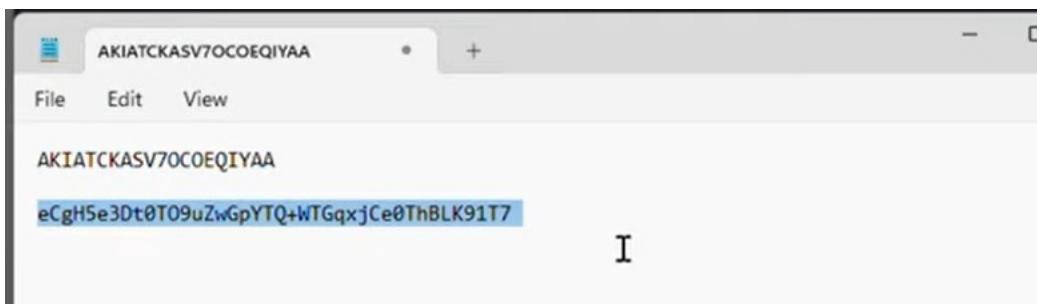
- AWS IAM Access Analyzer now offers policy checks for public and critical resource access. 3 months ago
- AWS IAM Access Analyzer now offers recommendations to refine unused access. 3 months ago
- AWS Launches Console-based Bulk Policy Migration for Billing and Cost Management Console Access. 3 months ago
- IAM Roles Anywhere now supports modifying the mapping of certificate attributes. 5 months ago

The screenshot shows the "EKS-USER" details page under the "Users" section of the IAM service. The left sidebar is identical to the one in the first screenshot. The main content area is titled "EKS-USER" and includes a "Summary" section with ARN (arn:aws:iam::211125710812:user/EKS-USER), Created date (September 01, 2024, 12:17 (UTC+05:30)), Console access status (Enabled without MFA), and two Access key entries. Below this is a "Security credentials" tab, which is currently selected, showing "Console sign-in" (Console sign-in link: https://211125710812.signin.aws.amazon.com/console) and "Multi-factor authentication (MFA) (0)". A note at the bottom states: "Use MFA to increase the security of your AWS environment. Signing in with MFA requires an authentication code from an MFA device. Each user can have a maximum of 8 MFA devices assigned." Buttons for "Remove", "Resync", and "Assign MFA device" are also present.

security credentials



access+ secret key



step

```
# Install unzip and AWSCLI  
  
#sudo apt update && sudo apt install unzip -y  
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"  
unzip awscliv2.zip  
sudo ./aws/install
```

```
# aws configure and pass the keys.
```

```
root@ip-10-0-9-146:~# aws configure
AWS Access Key ID [*****IYAA]:
AWS Secret Access Key [*****91T7]:
Default region name [us-east-1]:
Default output format [json]:
```

```
root@ip-10-0-9-146:~# aws s3 ls
2024-07-23 17:38:35 cloudvishwakarma.in
root@ip-10-0-9-146:~#
```

step

```
# install kubectl
curl -LO "https://storage.googleapis.com/kubernetes-release/release/$(curl -s https://storage.googleapis.com/kubernetes-release/release/ | grep "Kubernetes v" | awk '{print $2}')/bin/linux/amd64/kubectl"
sudo mv ./kubectl /usr/local/bin/kubectl
chmod 777 /usr/local/bin/kubectl
kubectl version -short

# 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
sudo chmod 700 /usr/local/bin/eksctl
eksctl version

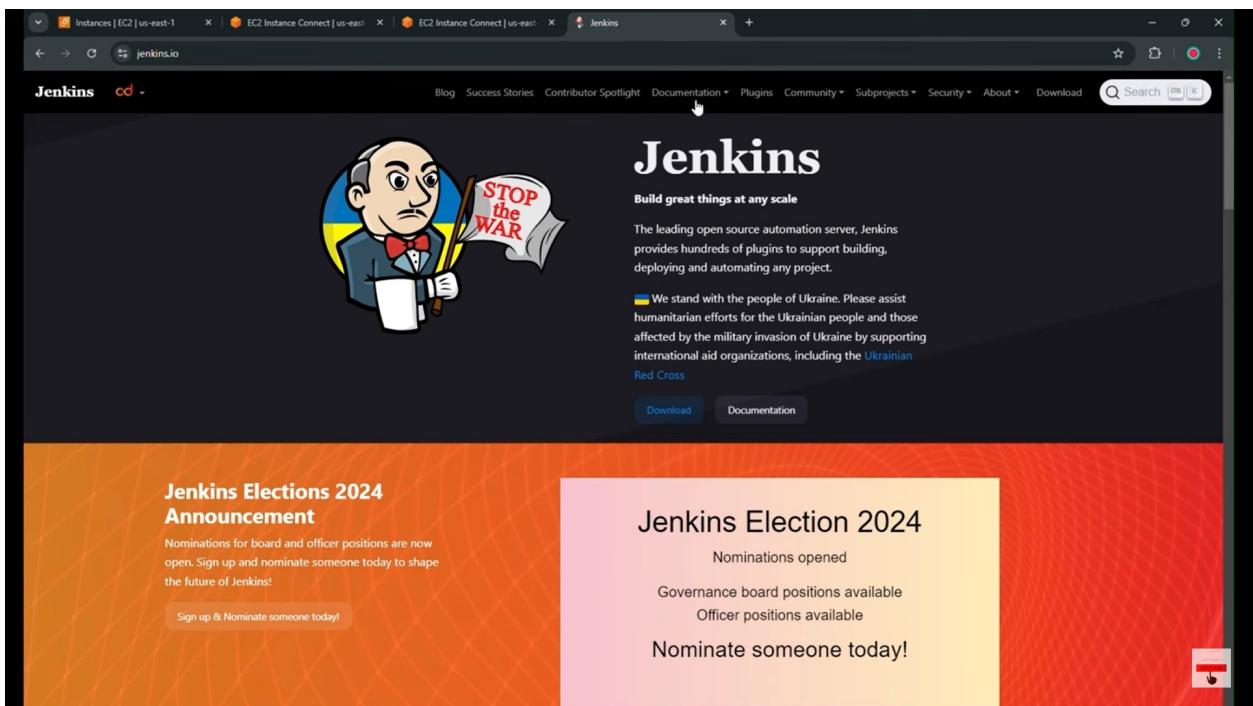
#deploy control plane to delete ""eksctl delete cluster --name eks-cluster-1"""

eksctl create cluster
--name eks-cluster-1 \
--version 1.29 \
--zones=us-east-1a,us-east-1b,us-east-1c \
--without-nodegroup
```

#Install java & jenkins

```
sudo apt install -y openjdk-11-jdk
```

```
root@ip-10-0-9-146:~# sudo apt install -y openjdk-11-jdk
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
alsa-topology-conf alsamixer-conf at-spi2-core ca-certificates-java dconf-gsettings-backend fontconfig-config fonts-dejavu-core fonts-dejavu-extra
gsettings-desktop-schemas java-common libasound2 libasound2-data libatk-bridge2.0-0 libatk-wrapper-java libatk-wrapper-java-jni libatk1.0-0 libatk1.0-data
libatspi2.0-0 libavahi-client3 libavahi-common-data libavahi-common3 libcurl3 libdconf1 libdrm-amdgpu libdrm-intel libdrm-nouveau2 libdrm-radeon1
libfontconfig1 libfontenc1 libgif7 libgl1-libgl1-amber-dri libgl1-mesa-dri libglapi-mesa libglvnd0 libglx-mesa0 libglx0 libgraphite2-3 libharfbuzz0b libice-dev
libice6 libjpeg-turbo8 libjpeg8 libicms2-2 libl1vml5 libpciaccess0 libpcsc-lite1 libpthread-stubs0-dev libsensors-config libsensors5 libsm-dev libsm6 libx11-dev
libx11-xcb1 libxau-dev libxaw7 libxcb-dri2-0 libxcb-dri3-0 libxcb-glx0 libxcb-present0 libxcb-randr0 libxcb-shape0 libxcb-sim0 libxcb-sync1 libxcb-xfixes0
libxcb1-dev libxcomposite1 libxdmcp-dev libxfixes3 libxtf2 libx16 libxinerama1 libxkbfile1 libxmu6 libxpm4 libxrandr2 libxrender1 libxshmfence1 libxt-dev
libxt6 libxtst6 libxv1 libxxf86gal libxxf86ml openjdk-11-jdk-headless openjdk-11-jre openjdk-11-jre-headless session-migration x11-common x11-utils
x11proto-dev xorg-sgml-doctools xtrans-dev
```



documentation

A screenshot of the Jenkins User Handbook documentation page for "Installing Jenkins". The page is part of the "User Handbook" section. It contains a sidebar with links to "User Handbook Overview", "Installing Jenkins" (which is currently selected), and other topics like Docker, Kubernetes, Linux, macOS, Windows, Other Systems, WAR file, Other Servlet Containers, Offline Installations, Initial Settings, Platform Information, Using Jenkins, Pipeline, Blue Ocean, Managing Jenkins, Securing Jenkins, System Administration, Scaling Jenkins, Troubleshooting Jenkins, and Glossary. The main content area has a heading "Installing Jenkins" and a paragraph explaining the installation process. A sidebar on the right lists "Chapter Sub-Sections" such as Docker, Kubernetes, Linux, macOS, Windows, Other Systems, WAR file, Other Servlet Containers, Offline Installations, and Initial Settings.

To exit full screen, press Esc

Debian/Ubuntu

On Debian and Debian-based distributions like Ubuntu you can install Jenkins through [apt](#).

Long Term Support release

A LTS (Long-Term Support) release is chosen every 12 weeks from the stream of regular releases as the stable release for that time period. It can be installed from the [debian-stable apt repository](#).

```
sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \
https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key
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
sudo apt-get update
sudo apt-get install jenkins
```

Weekly release

A new release is produced weekly to deliver bug fixes and features to users and plugin developers. It can be installed from the [debian apt repository](#).

```
sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \
https://pkg.jenkins.io/debian/jenkins.io-2023.key
echo "deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] " \
https://pkg.jenkins.io/debian binary/ | sudo tee \
/etc/apt/sources.list.d/jenkins.list > /dev/null
sudo apt-get update
sudo apt-get install jenkins
```

Beginning with Jenkins 2.335 and Jenkins 2.332.1, the package is configured with `systemd` rather than the older System V `init`. See the [migration guide](#).

now install Jenkins

```
sudo apt-get update
sudo apt-get install jenkins
--2024-09-04 09:42:20-- https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key
Resolving pkg.jenkins.io (pkg.jenkins.io)... 146.75.30.133, 2a04:4e42:78::645
Connecting to pkg.jenkins.io (pkg.jenkins.io)|146.75.30.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3175 (3.1K) [application/pgp-keys]
Saving to: '/usr/share/keyrings/jenkins-keyring.asc'

/usr/share/keyrings/jenkins-keyring.asc 100%[=====] 3.10K --.-KB/s in 0s

2024-09-04 09:42:20 (69.9 MB/s) - '/usr/share/keyrings/jenkins-keyring.asc' saved [3175/3175]

Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease
Ign:4 https://pkg.jenkins.io/debian-stable binary/ InRelease
Get:5 https://pkg.jenkins.io/debian-stable binary/ Release [2044 B]
Get:6 https://pkg.jenkins.io/debian-stable binary/ Release.gpg [833 B]
Hit:7 http://security.ubuntu.com/ubuntu jammy-security InRelease
Get:8 https://pkg.jenkins.io/debian-stable binary/ Packages [27.6 kB]
Fetched 30.5 kB in 0s (66.5 kB/s)
```

```
root@ip-10-0-9-146:~# jenkins --version
2.462.2
root@ip-10-0-9-146:~# 

i-0cc39b1ffde18a629 (EKS-Jenkins)

Public IPs: 3.231.161.7 Private IPs: 10.0.9.146
```

Jenkins up + running

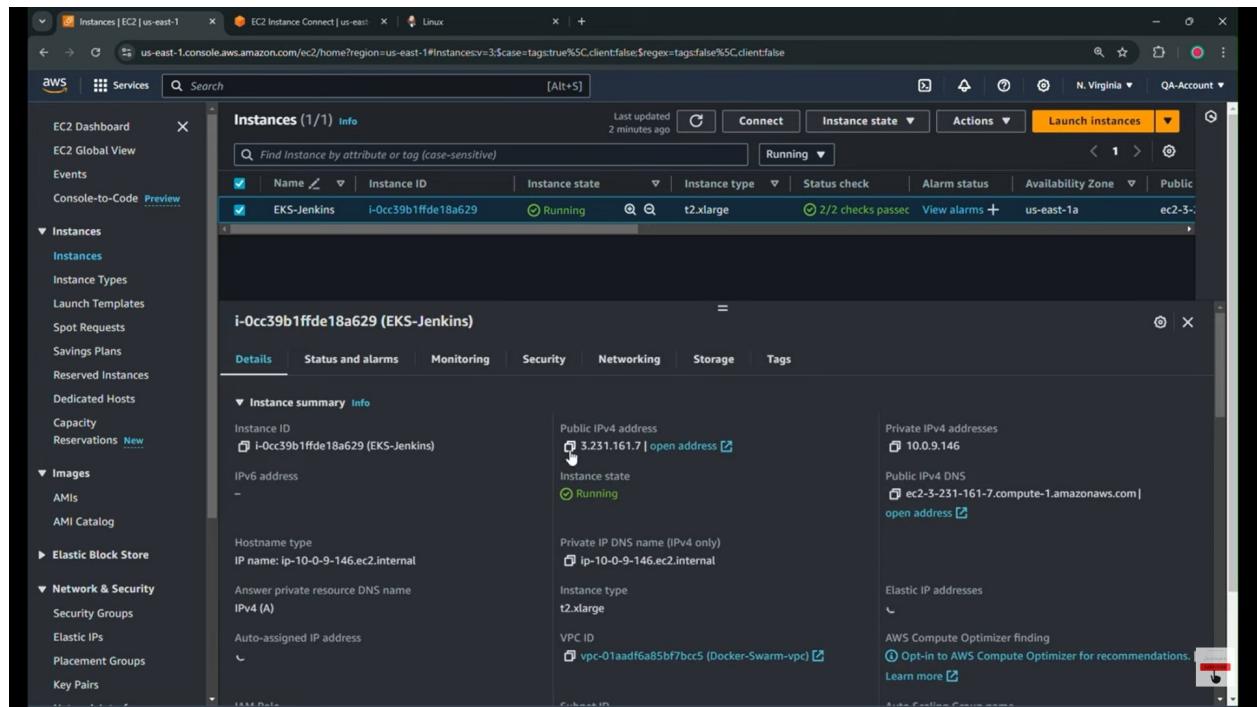
```

root@ip-10-0-9-146:~# systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
  Loaded: loaded (/lib/systemd/system/jenkins.service; enabled; vendor preset: enabled)
  Active: active (running) since Wed 2024-09-04 09:42:37 UTC; 30s ago
    Main PID: 6126 (java)
      Tasks: 52 (limit: 19159)
     Memory: 665.8M
        CPU: 17.454s
       CGroup: /system.slice/jenkins.service
               └─6126 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort=8080

Sep 04 09:42:35 ip-10-0-9-146 jenkins[6126]: f1255038fda046bb97920cd9ae3b378e
Sep 04 09:42:35 ip-10-0-9-146 jenkins[6126]: This may also be found at: /var/lib/jenkins/secrets/initialAdminPassword
Sep 04 09:42:35 ip-10-0-9-146 jenkins[6126]: ****
Sep 04 09:42:35 ip-10-0-9-146 jenkins[6126]: 2024-09-04 09:42:37.640+0000 [id=36]      INFO      jenkins.InitReactorRunner$1#onAttained: Completed initialization
Sep 04 09:42:37 ip-10-0-9-146 jenkins[6126]: 2024-09-04 09:42:37.663+0000 [id=23]      INFO      hudson.lifecycle.Lifecycle$onReady: Jenkins is fully up and running
Sep 04 09:42:37 ip-10-0-9-146 systemd[1]: Started Jenkins Continuous Integration Server.
Sep 04 09:42:37 ip-10-0-9-146 jenkins[6126]: 2024-09-04 09:42:37.684+0000 [id=52]      INFO      h.m.DownloadService$Downloadable#load: Obtained the updated descriptor
Sep 04 09:42:37 ip-10-0-9-146 jenkins[6126]: 2024-09-04 09:42:37.685+0000 [id=52]      INFO      hudson.util.Retriger#start: Performed the action check updates
lines 1-20/20 (END)

```

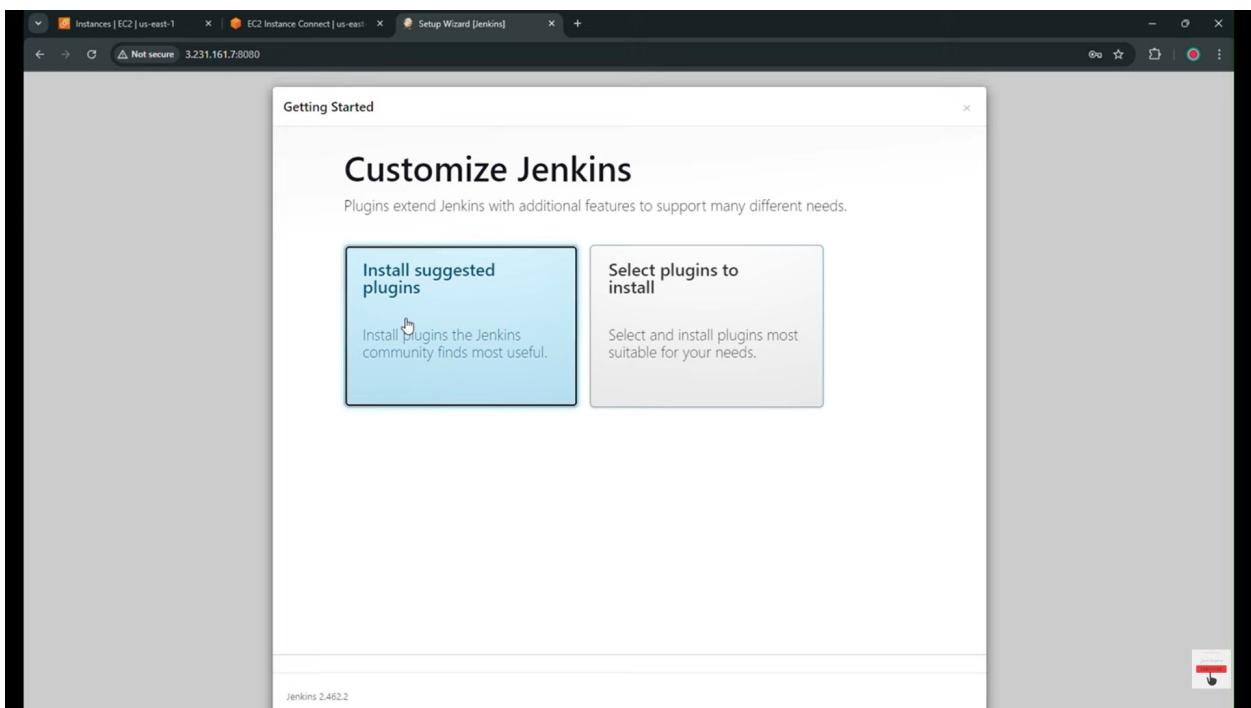
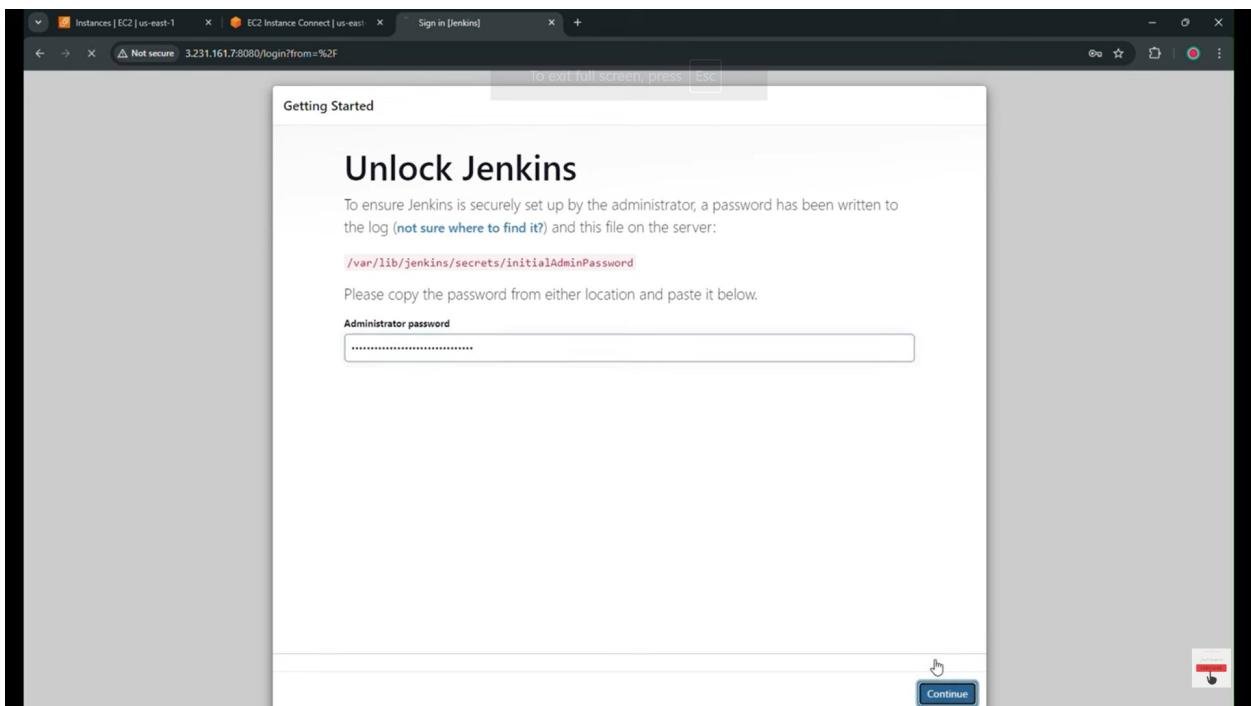
copy public ip of the instance to access jenking port :8080

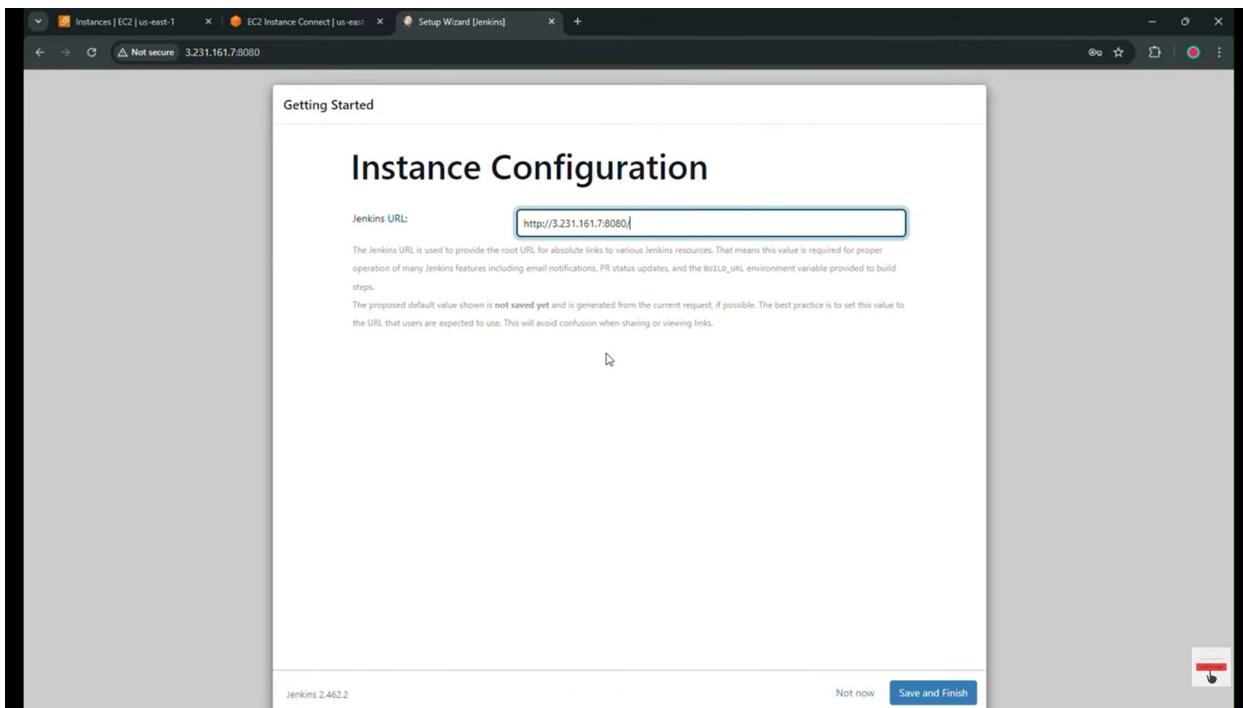
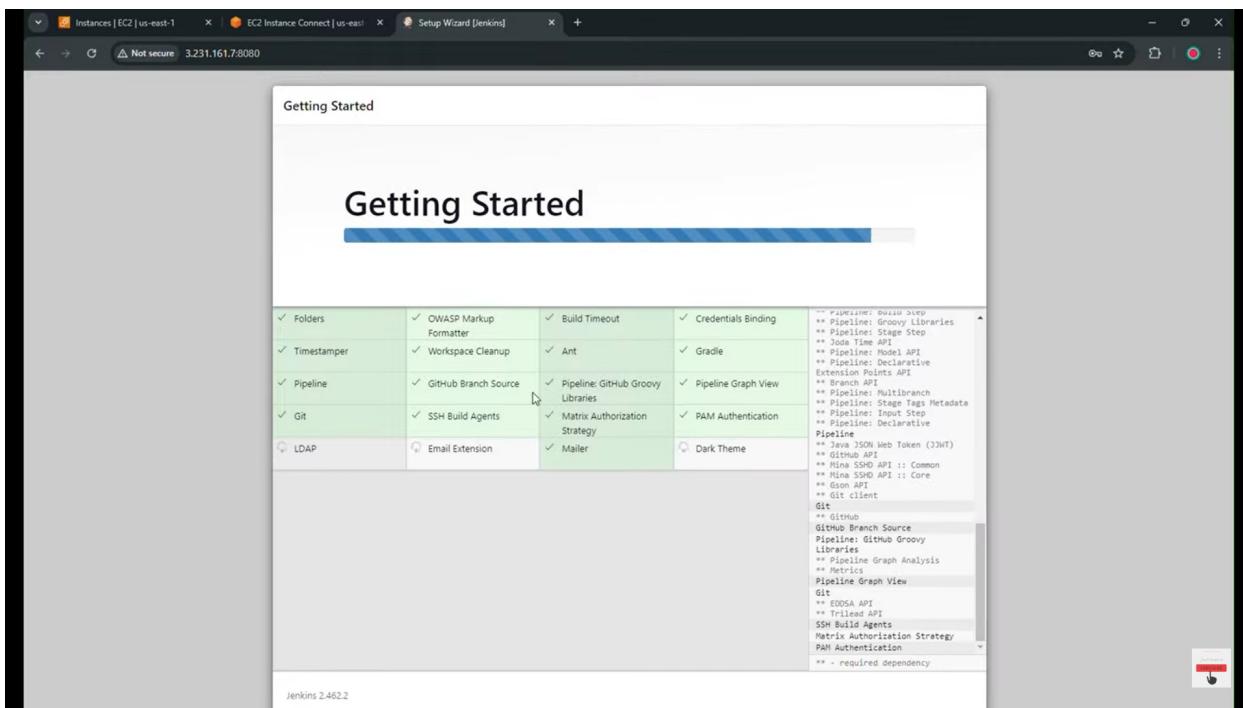


```

root@ip-10-0-9-146:~# jenkins --version
2.46.2
root@ip-10-0-9-146:~# cat /var/lib/jenkins/secrets/initialAdminPassword
f1255038fda046bb97920cd9ae3b378e
root@ip-10-0-9-146:~# 

```





Getting Started

Jenkins is ready!

You have skipped the **setup of an admin user**.

To log in, use the username: "admin" and the administrator password you used to access the setup wizard.

Your Jenkins setup is complete.

[Start using Jenkins](#)

manage Jenkins

The screenshot shows the Jenkins dashboard. On the left, there's a sidebar with links: 'New Item' (highlighted with a cursor), 'Build History', 'Manage Jenkins', and 'My Views'. Below these are two dropdown menus: 'Build Queue' (showing 'No builds in the queue.') and 'Build Executor Status' (showing '1 Idle' and '2 Idle'). The main content area has a title 'Welcome to Jenkins!'. It includes a search bar, a user menu for 'admin', and a link to 'Logout'. Below the title, it says 'This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.' A 'Start building your software project' button is present. To the right, there are three cards: 'Create a job' (with a '+' icon), 'Set up a distributed build' (with a computer icon), 'Set up an agent' (with a monitor icon), 'Configure a cloud' (with a cloud icon), and a link 'Learn more about distributed builds' with a help icon.

#

The screenshot shows the Jenkins Manage Jenkins page. On the left, there's a sidebar with links like 'New Item', 'Build History', 'Manage Jenkins' (which is selected), 'My Views', 'Build Queue' (with 'No builds in the queue.' message), 'Build Executor Status' (showing 1 idle and 2 idle), and 'Security'. The main content area is titled 'Manage Jenkins' and has a 'System Configuration' section. It includes sections for 'System' (Configure global settings and paths), 'Nodes' (Add, remove, control and monitor the various nodes that Jenkins runs jobs on), 'Clouds' (Add, remove, and configure cloud instances to provision agents on-demand), 'Tools' (Configure tools, their locations and automatic installers), 'Plugins' (Add, move, disable or enable plugins that can extend the functionality of Jenkins), 'Appearance' (Configure the look and feel of Jenkins), and 'Credential Providers' (Configure the credential providers and types). Below this is a 'Status Information' section with 'System Information' (Displays various environmental information to), 'System Log' (System log captures output from), and 'Load Statistics' (Check your resource utilization and see if you).

select plugins – AWS

The screenshot shows the Jenkins Available plugins - Plugins page. The sidebar on the left has links for 'Updates', 'Available plugins' (which is selected), 'Installed plugins', 'Advanced settings', and 'Download progress'. The main content area is titled 'Plugins' and has a search bar with 'aws'. A list of available AWS-related plugins is shown, each with a checkbox, name, version, description, and a timestamp. Plugins listed include:

- Amazon Web Services SDK : kinesis 1.12.767-467.vb_e93f0c614b_6 Library plugins (for use by other plugins) aws Kinesis module for the AWS SDK for Java. 22 days ago
- Amazon Web Services SDK : Secrets Manager 1.12.767-467.vb_e93f0c614b_6 Library plugins (for use by other plugins) aws Secrets Manager module for the AWS SDK for Java. 22 days ago
- Pipeline: AWS Steps 1.45 pipeline aws This plugin adds Jenkins pipeline steps to interact with the AWS API. 5 mo 9 days ago
- Amazon EC2 1688.v8c07e01d657f Cloud Providers Cluster Management Agent Management spotinst aws This plugin integrates Jenkins with Amazon EC2 or anything implementing the EC2 API's such as an Ubuntu. 3 mo 17 days ago
- Amazon Web Services SDK : Lambda 1.12.767-467.vb_e93f0c614b_6 Library plugins (for use by other plugins) aws Lambda module for the AWS SDK for Java. 22 days ago
- Amazon Web Services SDK : Organizations 1.12.767-467.vb_e93f0c614b_6 Library plugins (for use by other plugins) aws Organizations module for the AWS SDK for Java. 22 days ago
- Amazon Web Services SDK : Api Gateway 1.12.767-467.vb_e93f0c614b_6 Library plugins (for use by other plugins) aws API Gateway module for the AWS SDK for Java. 22 days ago
- Amazon Web Services SDK : CloudFront 1.12.767-467.vb_e93f0c614b_6 Library plugins (for use by other plugins) aws 22 days ago

```
#Once Cluster is up
export KUBECONFIG=/root/.kube/config
```

+

```
#Smoketest
ku get pods -A
ku get ns
ku get all -n kube-system
```

```
root@ip-10-0-9-146:~# export KUBECONFIG=/root/.kube/config
root@ip-10-0-9-146:~# kubectl get pods -A
NAMESPACE     NAME        READY   STATUS    RESTARTS   AGE
kube-system   coredns-54d6f577c6-5mmhj   0/1     Pending   0          7m5s
kube-system   coredns-54d6f577c6-rdftn   0/1     Pending   0          7m5s
root@ip-10-0-9-146:~# alias ku=kubectl
root@ip-10-0-9-146:~# ku cluster-info
Kubernetes control plane is running at https://0F0680CC3DD41D8681A1D1A8FA19E3C8.gr7.us-east-1.eks.amazonaws.com
CoreDNS is running at https://0F0680CC3DD41D8681A1D1A8FA19E3C8.gr7.us-east-1.eks.amazonaws.com/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy
To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
root@ip-10-0-9-146:~# ku get ns   I
NAME        STATUS   AGE
default     Active   9m39s
kube-node-lease Active   9m39s
kube-public  Active   9m39s
kube-system  Active   9m39s
root@ip-10-0-9-146:~#
```

```
# Now we need to install nodePools before that we need to do OIDC authentication.

eksctl utils associate-iam-oidc-provider \
--region us-east-1 \
--cluster eks-cluster-1 \
--approve
```

nodepools means---- worker node only .

```
root@ip-10-0-9-146:~# eksctl utils associate-iam-oidc-provider \
--region us-east-1 \
--cluster eks-cluster-1 \
--approve
2024-09-04 09:46:06 [i] will create IAM Open ID Connect provider for cluster "eks-cluster-1" in "us-east-1"
2024-09-04 09:46:06 [v] created IAM Open ID Connect provider for cluster "eks-cluster-1" in "us-east-1"
root@ip-10-0-9-146:~# I
```

create nodegroup, If production go with the private . We using local so lets o with the Public only.

#For Node Group In Public Subnet

```
eksctl create nodegroup --cluster=eks-cluster-1 \
--region=us-east-1 \
--name=eks-cluster-1-cluster-ng-1 \
--node-type=t3.medium \
--nodes=2 \
--nodes-min=2 \
--nodes-max=4 \
--node-volume-size=20 \
--ssh-access \
--ssh-public-key=QA-Account \
--managed \
--asg-access \
--external-dns-access \
--full-ecr-access \
--appmesh-access \
--alb-ingress-access
```

```

2024-09-04 09:46:06 [✓] created IAM OpenID Connect provider for cluster "eks-cluster-1" in "us-east-1"
root@ip-10-0-9-146:~# eksctl create nodegroup --cluster=eks-cluster-1 \
--region=us-east-1 \
--name=eks-cluster-1-cluster-ng-1 \
--node-type=t3.medium \
--nodes=2 \
--nodes-min=2 \
--nodes-max=4 \
--node-volume-size=20 \
--ssh-access \
--ssh-public-key=QA-Account \
--managed \
--asg-access \
--external-dns-access \
--full-ecr-access \
--appmesh-access \
--alb-ingress-access
2024-09-04 09:46:52 [i] will use version 1.29 for new nodegroup(s) based on control plane version
2024-09-04 09:46:53 [i] nodegroup "eks-cluster-1-cluster-ng-1" will use "" [AmazonLinux2/1.29]
2024-09-04 09:46:53 [i] using EC2 key pair "QA-Account"
2024-09-04 09:46:53 [i] 1 nodegroup (eks-cluster-1-cluster-ng-1) was included (based on the include/exclude rules)
2024-09-04 09:46:53 [i] will create a CloudFormation stack for each of 1 managed nodegroups in cluster "eks-cluster-1"
2024-09-04 09:46:53 [i] 2 sequential tasks: ( fix cluster compatibility, 1 task: ( create managed nodegroup "eks-cluster-1-cluster-n

```

2 instances created

The screenshot shows the AWS EC2 Instances page. The sidebar is expanded to show the 'Instances' section. The main table lists three instances:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
eks-cluster-1-eks-cluster-1...	i-0e90abf9051ef47f0	Running	t3.medium	OK	View alarms +	us-east-1a
eks-cluster-1-eks-cluster-1...	i-01efd3d090356bf75	Running	t3.medium	OK	View alarms +	us-east-1b
EKS-Jenkins	i-0cc39b1ffde18a629	Running	t2.xlarge	OK 2/2 checks passed	View alarms +	us-east-1a

Below the table, the details for the instance 'i-0cc39b1ffde18a629 (EKS-Jenkins)' are shown, including its Public IPv4 address (5.251.161.7) and Private IPv4 addresses (10.0.9.146).

handshake B/w EKS + jenkins

```

# NOW THE TASK IS OUR JENKINS SERVER SHOULD BE INTEGRATED WITH THE EKS
# WE WILL DO IT IN A CREDENTIAL PLUGIN WAY

# jenkins > credentials > system > global > Awscred > Global > aws-access-creds - copy paste the keys

```

manage Jenkins credentials

The screenshot shows the Jenkins Manage Jenkins interface. On the left, there's a sidebar with links like 'New Item', 'Build History', 'Manage Jenkins' (which is selected), and 'My Views'. The main area is titled 'Manage Jenkins' and has a 'System Configuration' section. It includes sections for 'System' (configure global settings and paths), 'Nodes' (add, remove, control and monitor various nodes), 'Tools' (configure tools, their locations and automatic installers), 'Clouds' (add, remove, and configure cloud instances to provision agents on-demand), 'Plugins' (add, remove, disable or enable plugins that can extend the functionality of Jenkins), 'Appearance' (configure the look and feel of Jenkins), and 'Security' (secure Jenkins; define who is allowed to access/use the system). Below these are 'Credentials' (configure credentials) and 'Credential Providers' (configure the credential providers and types). At the bottom, there are sections for 'Status Information' (System Information, System Log, Load Statistics), a 'Logs' link, and a 'Feedback' link.

Credentials – system

The screenshot shows the Jenkins Credentials page. The top navigation bar includes 'Dashboard', 'Manage Jenkins', 'Credentials', 'Search (CTRL+K)', and 'admin'. The main content area is titled 'Credentials' and shows a table with columns: T, P, Store, Domain, ID, and Name. Below this is a section titled 'Stores scoped to Jenkins' which lists 'System' under 'Store' and '(global)' under 'Domains'. There are also tabs for 'Linux', 'C', 'Bash', and 'Groovy'.

Global credentials + add Credentials

The screenshot shows the Jenkins Global credentials (unrestricted) page. The top navigation bar includes 'Dashboard', 'Manage Jenkins', 'Credentials', 'System', 'Global credentials (unrestricted)', 'Search (CTRL+K)', and 'admin'. The main content area is titled 'Global credentials (unrestricted)' and shows a table with columns: ID, Name, Kind, and Description. A message at the bottom of the table says 'This credential domain is empty. How about adding some credentials?'. There is a blue button labeled '+ Add Credentials' in the top right corner.

The screenshot shows the Jenkins 'New credentials' creation page for AWS Credentials. The 'Kind' dropdown is set to 'AWS Credentials'. The 'Scope' dropdown is set to 'Global (Jenkins, nodes, items, all child items, etc)'. The 'ID' field contains 'aws-creds'. The 'Description' field contains 'aws-creds'. The 'Access Key ID' field is empty. The 'Secret Access Key' field is empty. The 'IAM Role Support' section has an 'Advanced' dropdown. A 'Create' button is at the bottom.

The screenshot shows the Jenkins 'Global credentials (unrestricted)' list page. It displays a single credential entry:

ID	Name	Kind	Description
aws-creds	AKIATCKASV7OCOEQIYAA (aws-creds)	AWS Credentials	aws-creds

Below the table are icons for sorting by 'Icon', 'S' (Size), 'M' (Modified), and 'L' (Last modified).

Create a Jenkins Pipeline

The screenshot shows the Jenkins dashboard. On the left, there are links for 'New Item', 'Build History', 'Manage Jenkins', and 'My Views'. Below these are two dropdown menus: 'Build Queue' (No builds in the queue) and 'Build Executor Status' (1 Idle, 2 Idle). On the right, there are several cards: 'Welcome to Jenkins!', 'Start building your software project', 'Set up a distributed build', 'Set up an agent', 'Configure a cloud', and 'Learn more about distributed builds'.

New Item

Enter an item name

testing

Select an item

Testing01

Freestyle project
Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.

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.

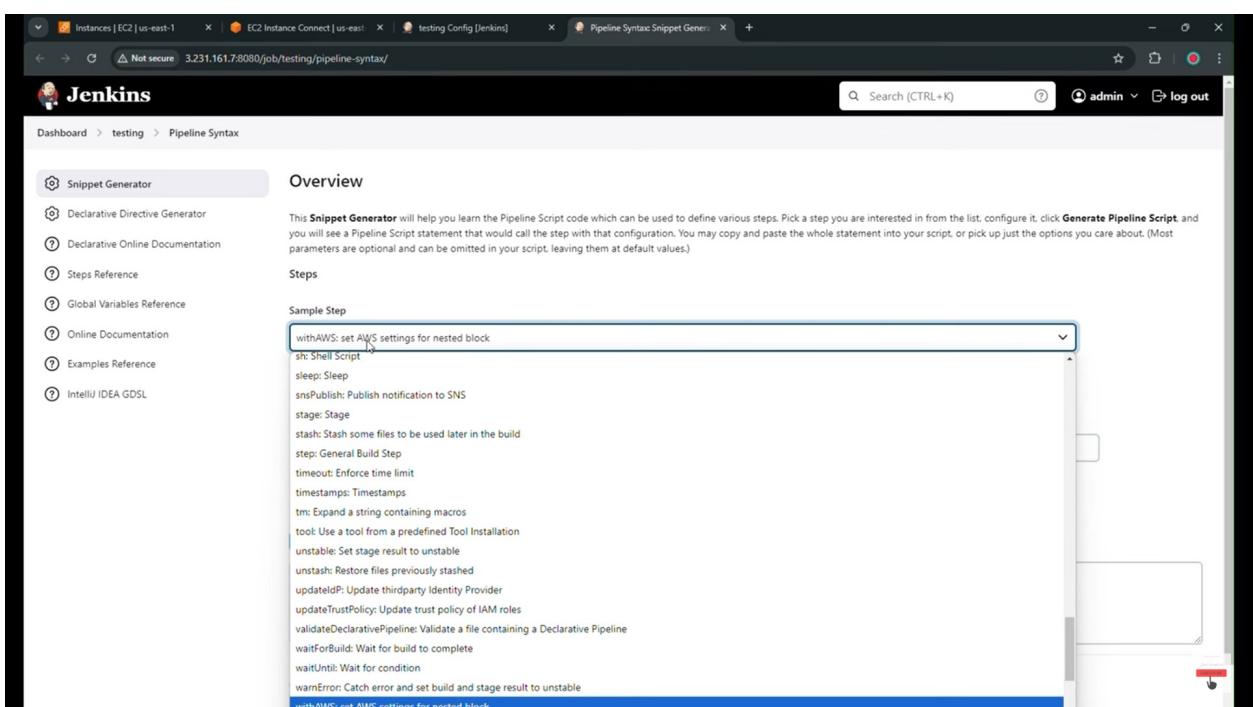
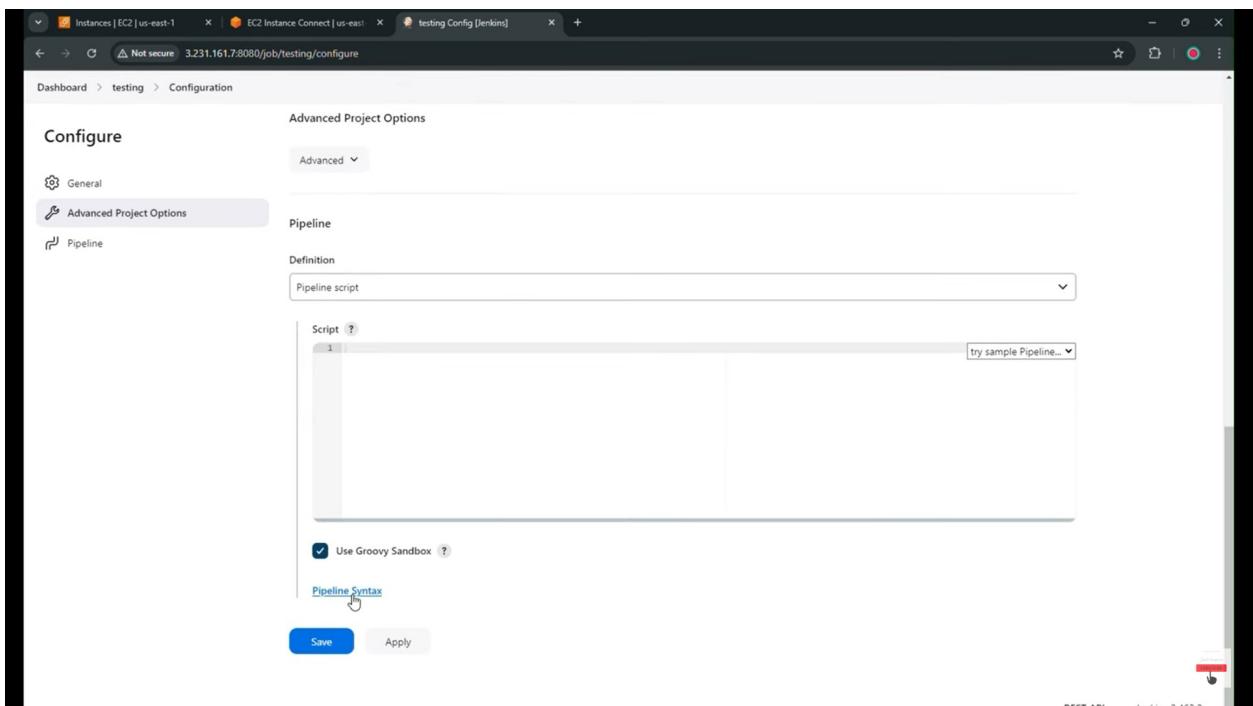
Multibranch Pipeline
Creates a set of Pipeline projects according to detected branches in one SCM repository.

Organization Folder
Creates a set of multibranch project subfolders by scanning for repositories.

OK

click pipeline syntax

<img alt="Screenshot of the VS Code code editor showing a Jenkinsfile. The file contains pipeline syntax with code completion highlights. The code is as follows: pipeline { agent any stages { stage ('Checking EKS Access') { steps [}] } }</pre>



Instances [EC2 | us-east-1] | EC2 Instance Connect [us-east-1] | testing Config [Jenkins] | Pipeline Syntax Snippet Generator

Not secure 3.231.161.7:8080/job/testing/pipeline-syntax/

Dashboard > testing > Pipeline Syntax

withAWS: set AWS settings for nested block

⑦ Online Documentation
⑦ Examples Reference
⑦ IntelliJ IDEA GDSL

withAWS ?

Region ?

Endpoint URL ?

Credentials ?

AKIATCKASV7OCOEQIYAA (aws-creds)

+ Add ▾

Profile ?

Role ?

Role Account ?

SAML Assertion ?

Federated User ID ?

Generate Pipeline Script

```
withAWS(credentials:'aws-creds') {  
    // some block  
}
```

Global Variables

```
1 pipeline {
2     agent any
3     stages {
4         stage ('Checking EKS Access') {
5             steps {
6                 withAWS(credentials: 'aws-creds') [
7                     // Stage steps here
8                 ]
9             }
10        }
11    }
12 }
13 }
```

```
# sh 'aws eks --region us-east-1 update-kubeconfig --name eks-cluster-1'
# sh 'kubectl get pods -A'
```

```
1 pipeline {
2     agent any
3     stages {
4         stage ('Checking EKS Access') {
5             steps {
6                 withAWS(credentials: 'aws-creds') {
7                     sh 'aws eks --region us-east-1 update-kubeconfig --name eks-cluster-1'
8                     sh 'kubectl get pods -A'
9                 }
10            }
11        }
12    }
13 }
14 }
```

```
# now go back to Jenkins pipeline .. & check configuration status + proper integration with cluster ??
```

This screenshot shows the Jenkins Pipeline Syntax Snippet Generator interface. The left sidebar has a 'Snippet Generator' section with links to Declarative Directive Generator, Declarative Online Documentation, Steps Reference, Global Variables Reference, Online Documentation, Examples Reference, and IntelliJ IDEA GDSL. The main content area is titled 'Overview' and contains a paragraph about the Snippet Generator's purpose. Below this is a 'Sample Step' section with a dropdown menu showing 'withAWS: set AWS settings for nested block'. A detailed configuration form follows, with fields for 'withAWS' (selected), 'Region' (empty), 'Endpoint URL' (empty), 'Credentials' (set to 'AKIATCKASV7OCOEQIYAA (aws-creds)'), and 'Profile' (empty). A red cursor arrow points to the 'Generate Pipeline Script' button in the bottom right corner.

```
# click testing - click configure
```

This screenshot shows the Jenkins Pipeline configuration page for the 'testing' pipeline. The left sidebar lists pipeline management options: Status, Changes, Build Now, Configure (which is selected and highlighted in grey), Delete Pipeline, Stages, Rename, and Pipeline Syntax. The main content area displays the pipeline configuration with sections for 'Permalinks' and 'Build History'. The 'Build History' section shows 'No builds' and includes a 'trend' dropdown menu. A red cursor arrow points to the 'Configure' link in the sidebar.

The screenshot shows the Jenkins Pipeline configuration page. The pipeline script is defined as follows:

```
1 pipeline {
2   agent any
3   stages {
4     stage ('Checking EKS Access') {
5       steps {
6         withAWS(credentials: 'aws-creds') {
7           sh 'aws eks --region us-east-1 update-kubeconfig --name eks-cluster-1'
8           sh 'kubectl get pods -A'
9         }
10      }
11    }
12  }
13}
14 }
```

The "Use Groovy Sandbox" checkbox is checked. At the bottom, there are "Save" and "Apply" buttons.

```
# error . rename .. kubectl
```

--- testing- configure --- Buildnow

The screenshot shows the Jenkins Pipeline configuration page. The pipeline script is identical to the one in the previous screenshot:

```
1 pipeline {
2   agent any
3   stages {
4     stage ('Checking EKS Access') {
5       steps {
6         withAWS(credentials: 'aws-creds') {
7           sh 'aws eks --region us-east-1 update-kubeconfig --name eks-cluster-1'
8           sh 'kubectl get pods -A'
9         }
10      }
11    }
12  }
13}
14 }
```

The "Use Groovy Sandbox" checkbox is checked. At the bottom, there are "Save" and "Apply" buttons.

The screenshot shows the Jenkins interface with the 'Console Output' tab selected. The output window displays the log of a pipeline job named 'testing #2'. The log shows the pipeline starting, connecting to an EKS cluster, and listing pods in the 'kube-system' namespace. It concludes with a 'Finished: SUCCESS' message.

```

Started by user admin
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/testing
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Checking EKS Access)
[Pipeline] withAWS
Constructing AWS Credentials[Pipeline]
[Pipeline] sh
+ aws eks --region us-east-1 update-kubeconfig --name eks-cluster-1
Updated context arn:aws:eks:us-east-1:211125710812:cluster:eks-cluster-1 in /var/lib/jenkins/.kube/config
[Pipeline] sh
+ kubectl get pods -A
NAMESPACE     NAME          READY   STATUS    RESTARTS   AGE
kube-system   aws-node-jqj27   2/2     Running   0          9m3s
kube-system   aws-node-s6pvj   2/2     Running   0          9m4s
kube-system   coredns-54d6f577c6-5mhj   1/1     Running   0          19m
kube-system   coredns-54d6f577c6-qdfn   1/1     Running   0          19m
kube-system   kube-proxy-4dnuq   1/1     Running   0          9m4s
kube-system   kube-proxy-r6bh5   1/1     Running   0          9m3s
[Pipeline] }
[Pipeline] // withAWS
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] end of Pipeline
Finished: SUCCESS

```

now Jenkins able to access eks cluster + aws credentials as I have allotted admin access

2nd .. create namespace -ns

The screenshot shows the VS Code editor with the Jenkinsfile open. The code defines a pipeline with two stages: 'Checking EKS Access' and 'Create namespace'. The 'Checking EKS Access' stage uses the 'withAWS' step to run commands like 'aws eks update-kubeconfig' and 'kubectl get pods'. The 'Create namespace' stage also uses 'withAWS' to run 'kubectl create ns development' and 'kubectl create ns production'.

```

1 pipeline {
2     agent any
3     stages {
4         stage ('Checking EKS Access') {
5             steps {
6                 withAWS(credentials: 'aws-creds') {
7                     sh 'aws eks --region us-east-1 update-kubeconfig --name eks-cluster-1'
8                     sh 'kubectl get pods -A'
9                 }
10            }
11        }
12        stage ('Create namespace ') {
13            steps [
14                withAWS(credentials: 'aws-creds') {
15                    sh 'aws eks --region us-east-1 update-kubeconfig --name eks-cluster-1'
16                    sh 'kubectl create ns development'
17                    sh 'kubectl create ns production'
18                }
19            ]
20        }
21    }
22 }
23

```

go back to Jenkins- testing- configure- Groovy script

The screenshot shows the Jenkins Pipeline configuration page for a job named "testing". The "Advanced Project Options" section is selected. Under the "Pipeline" tab, the "Definition" is set to "Pipeline script". The script content is as follows:

```

5+    withAWS(credentials: 'aws-creds') {
6+        sh 'aws eks --region us-east-1 update-kubeconfig --name eks-cluster-1'
7+        sh 'kubectl get pods -A'
8+    }
9+
10}
11
12}
13stage ('Create namespace ') {
14    steps {
15        withAWS(credentials: 'aws-creds') {
16            sh 'aws eks --region us-east-1 update-kubeconfig --name eks-cluster-1'
17            sh 'kubectl create ns development'
18            sh 'kubectl create ns production'
19        }
20    }
21}
22}
23}

```

Use Groovy Sandbox

Save Apply

Buildnow –

The screenshot shows the Jenkins Pipeline status and history pages for the "testing" pipeline.

Status:

- Changes: Build scheduled
- Build Now
- Configure
- Delete Pipeline
- Stages
- Rename
- Pipeline Syntax

Permalinks:

- Last build (#2), 1 min 36 sec ago
- Last stable build (#2), 1 min 36 sec ago
- Last successful build (#2), 1 min 36 sec ago
- Last failed build (#1), 2 min 24 sec ago
- Last unsuccessful build (#1), 2 min 24 sec ago
- Last completed build (#2), 1 min 36 sec ago

Build History:

Build #	Date
#2	Sep 4, 2024, 9:57 AM

```

root@ip-10-0-9-146:~# ku get ns
NAME        STATUS   AGE
default     Active   23m
development Active   0s
kube-node-lease Active   23m
kube-public  Active   23m
kube-system  Active   23m
root@ip-10-0-9-146:~# ku get ns
NAME        STATUS   AGE
default     Active   23m
development Active   22s
kube-node-lease Active   23m
kube-public  Active   23m
kube-system  Active   23m
production   Active   21s
root@ip-10-0-9-146:~#

```

The screenshot shows a Jenkins job named 'testing' with a build number of '#3'. The 'Console Output' tab is selected. The output log is displayed, showing the execution of a pipeline script. The log includes commands like 'aws eks --region us-east-1 update-kubeconfig --name eks-cluster-1' and 'kubectl get pods -A', along with a table of pod status information. The Jenkins interface also shows other tabs like 'Status', 'Changes', and 'Pipeline Overview'.

```

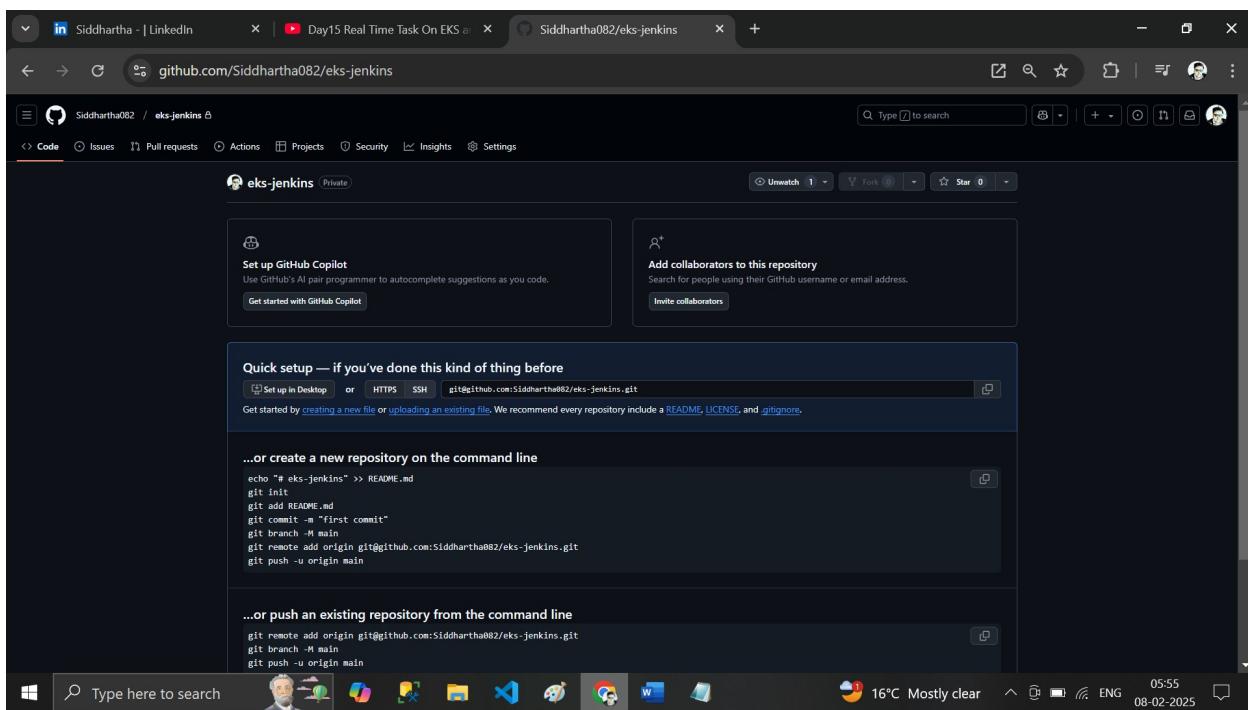
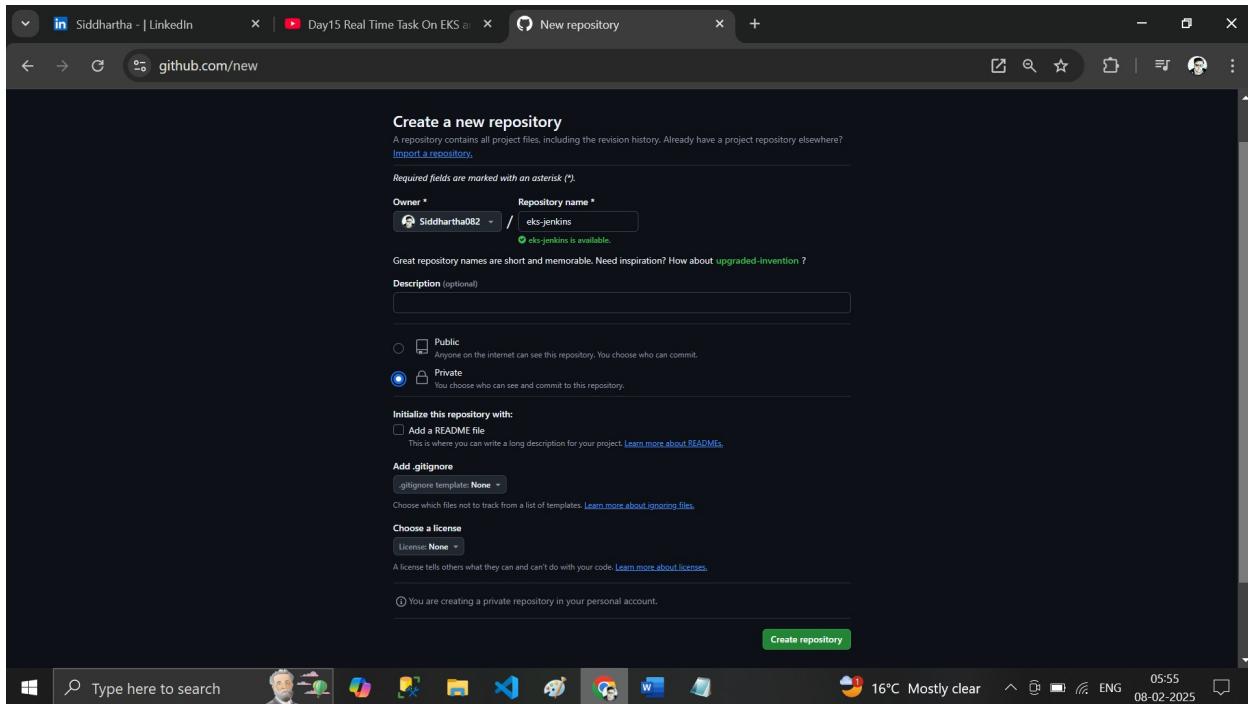
Started by user admin
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/testing
[Pipeline] {
[Pipeline] stage
[Pipeline] { ((Checking EKS Access))
[Pipeline] withAWS
Constructing AWS Credentials[Pipeline] {
[Pipeline] sh
+ aws eks --region us-east-1 update-kubeconfig --name eks-cluster-1
Updated context arn:aws:eks:us-east-1:211125710812:cluster/eks-cluster-1 in /var/lib/jenkins/.kube/config
[Pipeline] sh
+ kubectl get pods -A
NAMESPACE      NAME          READY   STATUS    RESTARTS   AGE
kube-system    aws-node-jqj27  2/2     Running   0          10m
kube-system    aws-node-s6pvv  2/2     Running   0          10m
kube-system    coredns-54d6f577c6-5mhbj  1/1   Running   0          20m
kube-system    coredns-54d6f577c6-rdftn  1/1   Running   0          20m
kube-system    kube-proxy-4duqq  1/1     Running   0          10m
kube-system    kube-proxy-r6h5h  1/1     Running   0          10m
[Pipeline] }
[Pipeline] // withAWS
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { ((Create namespace ))
[Pipeline] withAWS
Constructing AWS Credentials[Pipeline] {
[Pipeline] sh
+ aws eks --region us-east-1 update-kubeconfig --name eks-cluster-1

```

```

# LETS CREATE A PRIVATE GITHUB REPOSITORY AND MAKE A HANDSHAKE METHOD BETWEEN GITHUB AND JENKINS

```



#

```
root@ip-10-0-9-146:~# su - jenkins
jenkins@ip-10-0-9-146:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/var/lib/jenkins/.ssh/id_rsa):
Created directory '/var/lib/jenkins/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /var/lib/jenkins/.ssh/id_rsa
Your public key has been saved in /var/lib/jenkins/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:ZAwcoxjnU56NypbK2bHTVuXGKISK0AQyqdz4IyzKQHg jenkins@ip-10-0-9-146
The key's randomart image is:
+---[RSA 3072]----+
|+oo ..=.
|o+ = .B
|= = = = .
|+=E+ = o =
|+.o * . S +
|o+ B + o .
|= = = o
|.. o
|
+---[SHA256]----+
jenkins@ip-10-0-9-146:~$
```

i-0cc39b1ffde18a629 (EKS-Jenkins)

PublicIPs: 3.231.161.7 PrivateIPs: 10.0.9.146

```
su - jenkins
ssh-keygen #generating ket for authentication.
```

cp the Pvt Key .paste it in Jenkins credentials

```
jenkins@ip-10-0-9-146:~$ cat /var/lib/jenkins/.ssh/id_rsa
```

i-0cc39b1ffde18a629 (EKS-Jenkins)

PublicIPs: 3.231.161.7 PrivateIPs: 10.0.9.146

```
# come back to Jenkins dashboard
```



Jenkins

Dashboard >

- [+ New Item](#)
- [Build History](#)
- [Manage Jenkins](#)
- [My Views](#)

All+

S	W	Name	Last Success	Last Failure	Last Duration
		testing	2 min 44 sec	5 min 11 sec	6.1 sec

Build Queue▼...

No builds in the queue.

Build Executor Status▼

1 Idle
2 Idle

```
add these keys to your jenkins SERVER
Jenkins > manage Jenkins > credentials > system > global
ssh-username-with-private-key
github-access
jenkins > add-directly
paste the private key
```

```
# manage jenkins- credentials-system
```

The screenshot shows the Jenkins 'Credentials' management interface. At the top, there are several tabs: 'Instances | EC2 | us-east-1', 'EC2 Instance Connect | us-east-1', 'testing Config [jenkins]', 'Jenkins > Credentials [jenkins]', and 'saikiran/eks-jenkins'. The main title is 'Credentials'. Below it, a table lists a single credential:

T	P	Store	Domain	ID	Name
File	System	System	(global)	aws-creds	AKIATCKASV7OCOEQIYAA (aws-creds)

Below the table, a section titled 'Stores scoped to Jenkins' shows a table with one entry:

P	Store	Domains
System	System	(global)

At the bottom right, there are links for 'REST API' and 'jenkins 2.462.2'.

global credentials- Add Credentials

The screenshot shows the 'New credentials' form under 'Global credentials (unrestricted)'. The 'Kind' dropdown is set to 'Username with password'. The form fields are as follows:

- Scope**: A dropdown menu currently showing '(global)'.
- Username**: An input field with no value.
- Treat username as secret**: A checkbox that is unchecked.
- Password**: An input field with no value.
- ID**: An input field with no value.

Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) >

New credentials

Kind: SSH Username with private key

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

ID: github-access

Description: (empty)

Instances | EC2 | us-east-1 | EC2 Instance Connect | us-east-1 | testing Config [Jenkins] | New credentials [Jenkins] | saikiranpi/eks-jenkins | +

Not secure 3.231.161.7:8080/manage/credentials/store/system/domain/_/newCredentials

Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) >

SSH Username with private key

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

ID: github-access

Description: github-access

Username: jenkins

Treat username as secret

Private Key

Enter directly

Key

```
-----BEGIN OPENSSH PRIVATE KEY-----
b3B1bmNzC1+ZktdjfAAAAABG5vbmlAAAABEbm9vZQAAAAAAAABAA81uA4AAAdzc2gtcn
NHAAAAAAwEAQAAAAYEA1ekfh1ab915gZp39D9IvONuY/IwsD0hoDX5i4Oevgk@uar+lhs+
```

Enter New Secret Below

Passphrase

Create

Jenkins

Search (CTRL+K)

admin

Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) >

Global credentials (unrestricted)

+ Add Credentials

Credentials that should be available irrespective of domain specification to requirements matching.

ID	Name	Kind	Description
aws-creds	AKIATCKASV7OCOEQIYAA (aws-creds)	AWS Credentials	aws-creds
github-access	jenkins (github-access)	SSH Username with private key	github-access

Icon: S M L

do it in GitHub repo -- go to server ssh . cp the public key & paste it in github credentials

-NOW ON GITHUB PRIVATE REPO WITH PUBLIC key CAT PUBLICKEY

```
jenkins@ip-10-0-9-146:~$ cat /var/lib/jenkins/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAQABAAQgQCy4qOCNpv3XqBmf0P0i8425j8jCwMM2gNfnLg56+CTS5qv6WGs51vhKm/v3+caxPtYAAzAzh54AYuc8T5T7dutSa72RRshahgn0i0+3azqjLrUFceQ5FYQbb/5
6834+rq2MpzLHCosp2SY/CHkxOkGODhfleHJN/9e01huY221giq9owE8STWF462Jng4U1XY28Ldi17k56sdRxvJokd1OP7SLIJ0ud8xZ3YQ9a2g0Amb2R8uf6yNkyWneF2phSd82H3SAZCE/7eTeMcCo8te9q
Q/KLiw21+v8J7tBuIt?/osya.fMa/Mt:fSoEq/dcIax97ATfYQ7gmPp1HkrwyAIT+Pgs5aYCezcyWxom/sztqlxCrmJTx4aygwS8tBta+5FI4EXP3HFeVraYGiIublvFF1Ryw5pvxxHibXMbznaL15xtGwpYu6VJ2
2r/uDaJ8xtdfYfHAjAbhl4pzCHif5gp28K+zTEteklyH54+w0+aln0uZ94DPyk= jenkins@ip-10-0-9-146:~$
```

#github repo - setting – deploy keys

The screenshot shows the GitHub settings page for a repository named 'eks-jenkins'. The 'Deploy keys' tab is selected. On the left, there's a sidebar with options like General, Access, Collaborators, Code and automation (Branches, Tags, Rules, Actions, Webhooks, Codespaces, Pages), Security, and Deploy keys (which is currently selected). The main content area displays a message: 'There are no deploy keys for this repository. Check out our [guide on deploy keys](#) to learn more.' A green 'Add deploy key' button is located at the top right of this section.

This screenshot shows the 'Deploy keys / Add new' form. The 'Title' field contains 'jenkins@ip-10-0-9-146'. The 'Key' field contains the public SSH key previously copied from the Jenkins terminal. Below the key, a note states: 'Begins with 'ssh-rsa', 'ecdsa-sha2-nistp256', 'ecdsa-sha2-nistp384', 'ecdsa-sha2-nistp521', 'ssh-ed25519', 'sk-ecdsa-sha2-nistp256@openssh.com', or 'sk-ssh-ed25519@openssh.com''. There is also a checked checkbox for 'Allow write access' with the note: 'Can this key be used to push to this repository? Deploy keys always have pull access.' At the bottom right of the form is a green 'Add key' button.



```
File Edit Selection View Go Run Terminal ... ← → 🔍 Jenkinsfile X EKS-JEN... OPEN EDITORS ... Steps Jenkinsfile 1 pipeline { 2     stages { 3         stage('Creating EKS Namespaces') { 4             steps { 5                 withAWS(credentials: 'aws-creds') { 6                     sh 'kubectl create ns development || exit 0' 7                 } 8             } 9         } 10    } 11 12    stage('Deploy To Dev Namespace') { 13        when { 14            branch 'development' 15        } 16        steps { 17            withAWS(credentials: 'aws-creds') { 18                sh 'aws eks update-kubeconfig --region us-east-1 --name eks-cluster' 19                sh 'ls -al' 20                sh 'kustomize build kustomize/overlays/development' 21                sh 'kubectl apply -k kustomize/overlays/development' 22                sh 'kubectl get pods,deploy,svc -n development' 23            } 24        } 25    } 26 27    stage('Destroy App In Dev Namespace') { 28        when { 29            expression { 30                "${env.PROD_DESTROY}" == 'YES' && "$BRANCH_NAME" == 'development' 31            } 32        } 33    } 34 35 }
```

```
File Edit Selection View Go Run Terminal ... ← → 🔍 kustomization.yaml X EKS-JEN... OPEN EDITORS ... Steps Jenkinsfile kustomize > overlays > development > ! kustomization.yaml [ ]resources kustomization.yaml - Kubernetes native configuration management (kustomization.json) 1 apiVersion: kustomize.config.k8s.io/v1beta1 2 kind: Kustomization 3 namespace: development 4 namePrefix: dev- 5 nameSuffix: -env 6 7 resources: 8 - ../../base 9 10 replicas: 11 - count: 4 12 name: nginx001 13 14 images: 15 - newName: kiran2361993/kubegame 16 newTag: v2 17 name: kiran2361993/kubegame 18
```

The screenshot shows the VS Code interface with the title bar "Eks-jenkins". The left sidebar has "OPEN EDITORS" expanded, showing "EKS-JENKINS" and "kustomize". Inside "kustomize", "base" is expanded, showing "deployment.yaml", "kustomization.yaml", and "service.yaml". The "deployment.yaml" file is selected and open in the main editor area. The code is a Kubernetes Deployment manifest:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: nginx001
    name: nginx001
    namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx001
  template:
    metadata:
      labels:
        app: nginx001
    spec:
      containers:
        - image: kiran2361993/kubegame:v2
          name: appcontainer
```

The screenshot shows the VS Code interface with the title bar "Eks-jenkins". The left sidebar has "OPEN EDITORS" expanded, showing "EKS-JENKINS" and "kustomize". Inside "kustomize", "base" is expanded, showing "deployment.yaml", "kustomization.yaml", and "rbac.yaml". The "rbac.yaml" file is selected and open in the main editor area. The code is a Kubernetes ClusterRole manifest:

```
---  
apiVersion: v1  
kind: ServiceAccount  
metadata:  
  labels:  
    k8s-app: metrics-server  
    name: metrics-server  
---  
apiVersion: rbac.authorization.k8s.io/v1  
kind: ClusterRole  
metadata:  
  labels:  
    k8s-app: metrics-server  
    rbac.authorization.k8s.io/aggregate-to-admin: "true"  
    rbac.authorization.k8s.io/aggregate-to-edit: "true"  
    rbac.authorization.k8s.io/aggregate-to-view: "true"  
    name: system:aggregated-metrics-reader  
rules:  
  - apiGroups:  
    - metrics.k8s.io  
    resources:  
      - pods  
      - nodes  
    verbs:  
      - get  
      - list  
      - watch  
  ---  
apiVersion: rbac.authorization.k8s.io/v1  
kind: ClusterRole  
metadata:
```

A screenshot of the Visual Studio Code interface showing the Jenkinsfile for an EKS Jenkins pipeline. The code defines a pipeline with stages for destroying and deploying an application in both Dev and Prod namespaces, utilizing AWS credentials and kustomize overlays.

```
1 pipeline {
2     stages {
3         stage('Destroy App In Dev Namespace') {
4             steps {
5                 withAWS(credentials: 'aws-creds') {
6                     sh 'aws eks update-kubeconfig --region us-east-1 --name eks-cluster'
7                     sh 'ls -al'
8                     sh 'kustomize build kustomize/overlays/production'
9                     sh 'kubectl apply -k kustomize/overlays/production'
10                    sh 'kubectl get pods,deploy,svc -n production'
11                }
12            }
13        }
14        stage('Deploy To Prod Namespace') {
15            when {
16                branch 'production'
17            }
18            steps {
19                withAWS(credentials: 'aws-creds') {
20                    sh 'aws eks update-kubeconfig --region us-east-1 --name eks-cluster'
21                }
22            }
23        }
24    }
25    stage('Destroy App In Prod Namespace') {
26        when {
27            expression {
28                "${env.PROD_DESTROY}" == 'YES' && "$BRANCH_NAME" == 'production'
29            }
30        }
31        steps {
32            withAWS(credentials: 'aws-creds') {
33                sh 'aws eks update-kubeconfig --region us-east-1 --name eks-cluster'
34            }
35        }
36    }
37}
```

A screenshot of the Visual Studio Code interface showing the Jenkinsfile for an EKS Jenkins pipeline. The code has been modified to include additional steps in the 'Destroy App In Prod Namespace' stage, specifically adding 'kubectl delete' and 'get' commands for the production namespace.

```
1 pipeline {
2     stages {
3         stage('Destroy App In Dev Namespace') {
4             steps {
5                 withAWS(credentials: 'aws-creds') {
6                     sh 'aws eks update-kubeconfig --region us-east-1 --name eks-cluster'
7                     sh 'ls -al'
8                     sh 'kustomize build kustomize/overlays/production'
9                     sh 'kubectl apply -k kustomize/overlays/production'
10                    sh 'kubectl get pods,deploy,svc -n production'
11                }
12            }
13        }
14        stage('Deploy To Prod Namespace') {
15            when {
16                branch 'production'
17            }
18            steps {
19                withAWS(credentials: 'aws-creds') {
20                    sh 'aws eks update-kubeconfig --region us-east-1 --name eks-cluster'
21                }
22            }
23        }
24    }
25    stage('Destroy App In Prod Namespace') {
26        when {
27            expression {
28                "${env.PROD_DESTROY}" == 'YES' && "$BRANCH_NAME" == 'production'
29            }
30        }
31        steps {
32            withAWS(credentials: 'aws-creds') {
33                sh 'aws eks update-kubeconfig --region us-east-1 --name eks-cluster'
34                sh 'ls -al'
35                sh 'kustomize build kustomize/overlays/production'
36                sh 'kubectl delete -k kustomize/overlays/production'
37                sh 'kubectl get pods,deploy,svc -n production'
38            }
39        }
40    }
41}
```

#push repo in GitHub

```
$ MINGW64/c/Users/saiki/OneDrive/Desktop/EKs-jenkins
$ git init
Initialized empty Git repository in c:/Users/saiki/OneDrive/Desktop/EKs-jenkins/.git/
saiki@Saikiran236 MINGW64 ~/OneDrive/Desktop/EKs-jenkins (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    Jenkinsfile
    Steps
    kustomize/
nothing added to commit but untracked files present (use "git add" to track)

saiki@Saikiran236 MINGW64 ~/OneDrive/Desktop/EKs-jenkins (master)
$ git add . && git commit -m "added all integration files of Jenkins and EKS-v01"
warning: in the working copy of 'Jenkinsfile', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'kustomize/base/deployment.yaml', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'kustomize/base/kustomization.yaml', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'kustomize/base/rbac.yaml', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'kustomize/base/service.yaml', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'kustomize/overlays/development/kustomization.yaml', LF will be replaced by CRLF the next time Git touches it
it
warning: in the working copy of 'kustomize/overlays/production/kustomization.yaml', LF will be replaced by CRLF the next time Git touches it
[master (root-commit) 1b1c2ba] added all integration files of Jenkins and EKS-v01
  10 files changed, 445 insertions(+)
   Create mode 100644 Jenkinsfile
   Create mode 100644 Steps
   Create mode 100644 kustomize/Elements.png
   Create mode 100644 kustomize/base/deployment.yaml
   Create mode 100644 kustomize/base/kustomization.yaml
   Create mode 100644 kustomize/base/rbac.yaml
   Create mode 100644 kustomize/base/service.yaml
   Create mode 100644 kustomize/overlays/development/kustomization.yaml
   Create mode 100644 kustomize/overlays/production/Element
   Create mode 100644 kustomize/overlays/production/kustomization.yaml

saiki@Saikiran236 MINGW64 ~/OneDrive/Desktop/EKs-jenkins (master)
```

```
saiki@Saikiran236 MINGW64 ~/OneDrive/Desktop/EKs-jenkins (master)
$ 

saiki@Saikiran236 MINGW64 ~/OneDrive/Desktop/EKs-jenkins (master)
$ 

saiki@Saikiran236 MINGW64 ~/OneDrive/Desktop/EKs-jenkins (master)
$ git status
On branch master
nothing to commit, working tree clean

saiki@Saikiran236 MINGW64 ~/OneDrive/Desktop/EKs-jenkins (master)
$ git branch
* master

saiki@Saikiran236 MINGW64 ~/OneDrive/Desktop/EKs-jenkins (master)
$ git branch master -m development

saiki@Saikiran236 MINGW64 ~/OneDrive/Desktop/EKs-jenkins (development)
$ git branch
* development

saiki@Saikiran236 MINGW64 ~/OneDrive/Desktop/EKs-jenkins (development)
$ git |
```

```

saiki@Saikiran236 MINGW64 ~/OneDrive/Desktop/EKs-jenkins (master)
$ git branch master -m development

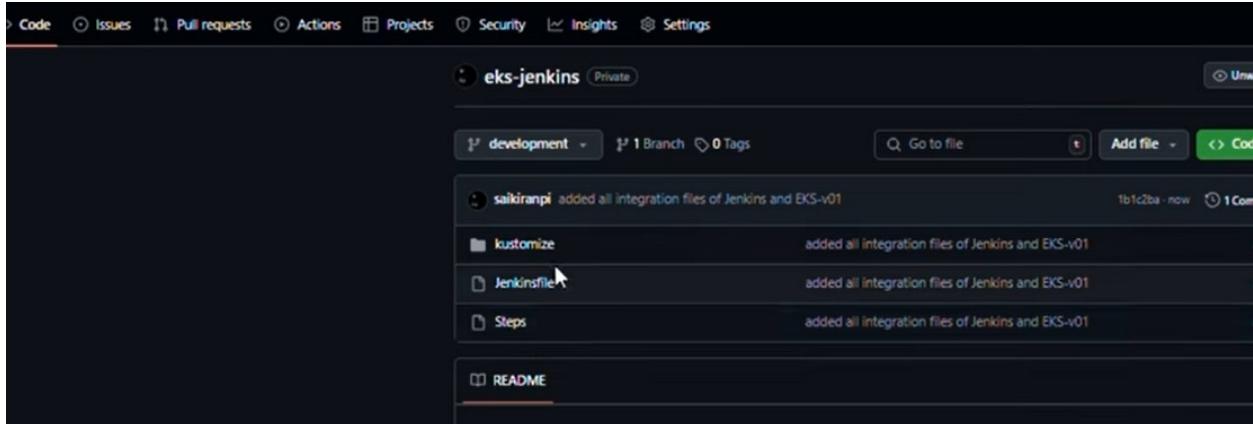
saiki@Saikiran236 MINGW64 ~/OneDrive/Desktop/EKs-jenkins (development)
$ git branch
* development

saiki@Saikiran236 MINGW64 ~/OneDrive/Desktop/EKs-jenkins (development)
$ git remote add origin https://github.com/saikiranpi/eks-jenkins.git

saiki@Saikiran236 MINGW64 ~/OneDrive/Desktop/EKs-jenkins (development)
$ git push origin development
Enumerating objects: 17, done.
Counting objects: 100% (17/17), done.
Delta compression using up to 12 threads
Compressing objects: 100% (15/15), done.
Writing objects: 100% (17/17), 101.40 KiB | 14.48 MiB/s, done.
Total 17 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/saikiranpi/eks-jenkins.git
 * [new branch]      development -> development

saiki@Saikiran236 MINGW64 ~/OneDrive/Desktop/EKs-jenkins (development)

```



step

```

for automatic trigger need to add a plugin on jenins server called 'multi branch scan webhook trigger'

```

Come back to Jenkins- manage Jenkins-plugins – available plugins

The screenshot shows the Jenkins plugin manager interface. On the left, a sidebar menu includes 'Updates', 'Available plugins' (which is selected), 'Installed plugins', 'Advanced settings', and 'Download progress'. The main area displays a list of available plugins:

Name	Description	Last Updated
SonarQube Scanner 2.17.2	This plugin allows an easy integration of SonarQube, the open source platform for Continuous Inspection of code quality.	6 mo 18 days ago
Build Blocker 166.vc82fc20b_a_ed6	This plugin blocks a build if one of the given jobs is running. The blocking behaviour can be configured to block builds on node or global level and to scan the queue and block if blocking jobs are about to run.	3 mo 16 days ago
Build Failure Analyzer 2.5.2	Jenkins Build Failure Analyzer. This plugin scans build logs and other files in the workspace for recognised patterns of known causes to build failures, and displays them on the build page for quicker recognition of why the build failed.	3 mo 3 days ago
Multibranch Scan Webhook Trigger 1.0.11	Trigger that can receive any HTTP request and trigger a multibranch job scan when token matched.	8 mo 22 days ago
Checkmarx 2024.2.3	This plugin allows scanning the source code in Checkmarx static code analysis engine. See https://checkmarx.atlassian.net/wiki/spaces/SD/pages/133913010/Jenkins+Plugin .	4 mo 15 days ago
Snyk Security 4.1.0	Add the ability to test your code dependencies for vulnerabilities against Snyk database	28 days ago
JFrog 1.5.1	The Jenkins JFrog Plugin allows for easy integration between Jenkins and the JFrog Platform. This integration allows your build jobs to deploy artifacts and resolve dependencies to and from Artifactory, and then have them linked to the build job that created them. It also allows you to scan your artifacts and builds with JFrog Xray and distribute your software package to remote locations using JFrog.	1 mo 13 days ago

The screenshot shows the Jenkins plugin manager interface. On the left, a sidebar menu includes 'Updates', 'Available plugins' (selected), 'Installed plugins', 'Advanced settings', and 'Download progress' (selected). The main area displays a 'Download progress' section:

Item	Status
Preparation	• Checking internet connectivity • Checking update center connectivity • Success
Ionicons API	Success
Folders	Success
OWASP Markup Formatter	Success
ASM API	Success
JSON Path API	Success
Structs	Success
Pipeline: Step API	Success
Token Macro	Success
Build Timeout	Success
bouncycastle API	Success
Credentials	Success
Plain Credentials	Success
Variant	Success
SSH Credentials	Success
Credentials Binding	Success
SCM API	Success
Pipeline: API	Success
commons-lang3 v3.x Jenkins API	Success
Timestamper	Success
Caffeine API	Success
Script Security	Success

Create a new pipeline with Multibranch and select git and give the credentials accordingly .

Jenkins- new item

New Item

Enter an item name

EKS-JENKINS

Select an item type

**Freestyle project**

Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.

**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

The screenshot shows the Jenkins 'New Item' creation interface. At the top, there's a breadcrumb navigation: Dashboard > All > New Item. Below that is a title 'New Item'. A text input field is labeled 'Enter an item name' with the value 'EKS-JENKINS'. A section titled 'Select an item type' lists several options:

- Freestyle project**: Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.
- 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.

Below the item type list, there's a note: 'If you want to create a new item from other existing, you can use this option:'. A 'Copy from' button is present, with a text input field below it labeled 'Type to autocomplete'. At the bottom is a blue 'OK' button.

Instances | EC2 | us-east-1 | EC2 Instance Connect | us-east-1 | EKS-JENKINS Config [jenkins] | Download progress - Plugins | saikranti/eks-jenkins | +

Not secure 3.231.161.7:8080/job/EKS-JENKINS/configure

Jenkins

Search (CTRL+K) admin log out

Dashboard > EKS-JENKINS > Configuration

Configuration

General

Enabled

Display Name

Description

Plain text [Preview](#)

Branch Sources

Add source

Mode by Jenkinsfile

Script Path Jenkinsfile

[Save](#) [Discard](#)

Properties

Branch Sources

Git Project Repository git@github.com:Siddhartha082/eks-jenkins.git

Credentials jenkins (github-access)

+ Add

Behaviors Discover branches

Add

Property strategy

The screenshot shows the Jenkins configuration interface for a job named 'EKS-JENKINS'. The 'Pipeline Libraries' section is visible, with a note stating 'Sharable libraries available to any Pipeline jobs inside this folder. These libraries will be untrusted, meaning their code runs in the Groovy sandbox.' A 'Add' button is present. Below the main configuration area are 'Save' and 'Apply' buttons.

error

The screenshot shows the Jenkins dashboard for the 'EKS-JENKINS' project. The 'Scan Multibranch Pipeline Log' section is selected. The log output shows an error related to host key verification:

```

Started by user admin
[Wed Sep 04 10:15:13 UTC 2024] Starting branch indexing...
> git --version # timeout=10
> git --version # 'git' version 2.34.1'
using GIT_SSH to set credentials github-access
Verifying host key using known hosts file
You're using 'Known hosts file' strategy to verify ssh host keys, but your known_hosts file does not exist, please go to 'Manage Jenkins' -> 'Security' -> 'Git Host Key Verification Configuration' and configure host key verification.
> git ls-remote --symref -- git@github.com:sakirani/eks-jenkins.git # timeout=10
ERROR: [Wed Sep 04 10:15:13 UTC 2024] Could not update older level actions from source 7fb6744c-6386-4bda-b041-4d5383eee3fb
[Wed Sep 04 10:15:13 UTC 2024] Finished branch indexing. Indexing took 0.13 sec
FATAL: Failed to recompute children of EKS-JENKINS
hudson.plugins.git.GitException: Command "git ls-remote --symref -- git@github.com:sakirani/eks-jenkins.git" returned status code 128:
stderr: No ED25519 host key is known for github.com and you have requested strict checking.
Host key verification failed.
fatal: Could not read from remote repository.

Please make sure you have the correct access rights
and the repository exists.

```

You will get error which took 2 hours to resolve got to manage jenkins > Security > turnoff host key verification

Jenkins – manage Jenkins- security

The screenshot shows the Jenkins Manage Jenkins interface. On the left sidebar, under 'Manage Jenkins', there are links for 'New Item', 'Build History', 'My Views', 'Build Queue' (No builds in the queue), and 'Build Executor Status' (1 Idle, 2 Idle). The main content area is titled 'Manage Jenkins' and contains three main sections: 'System Configuration', 'Security', and 'Status Information'.
System Configuration: Contains links for 'Tools', 'Clouds', and 'Appearance'.
Security: Contains links for 'Security' (Secure Jenkins: define who is allowed to access/use the system), 'Credentials', 'Users', and 'Credential Providers'.
Status Information: Contains links for 'System Information' (Displays various environmental information to), 'System Log' (System log captures output from), and 'Load Statistics' (Check your resource utilization and see if you).

The screenshot shows the Jenkins Security configuration page. The top navigation bar includes 'Instances | EC2 | us-east-1', 'EC2 Instance Connect | us-east-1', 'Manage Jenkins [Jenkins]', 'Download progress - Plugins []', 'saikiranpi/eks-jenkins', and 'admin log out'. The main content area is titled 'Security' and contains several configuration sections:
Authentication: Includes an option to 'Disable "Keep me signed in"' with a checkbox.
Security Realm: Set to 'Jenkins' own user database.
Authorization: Set to 'Logged-in users can do anything'.
Markup Formatter: Set to 'Plain text'. A note below states: '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.'
At the bottom, there are 'Save' and 'Cancel' buttons.

The screenshot shows the Jenkins Security configuration page. Under the "SSH Server" section, the "SSHD Port" dropdown is set to "Random". In the "Git Host Key Verification Configuration" section, the "Host Key Verification Strategy" dropdown is set to "No verification", which is highlighted with a red border and accompanied by a warning message: "⚠ This option is generally insecure. Host key will not be verified during SSH connection". Below this, under "Pipeline Sandbox", there is a checkbox for "Hide Sandbox checkbox in pipeline jobs". At the bottom of the page are "Save" and "Apply" buttons.

The screenshot shows the Jenkins EKS-JENKINS dashboard. The left sidebar includes options like Status, Configure, Scan Multibranch Pipeline Now, Scan Multibranch Pipeline Log, Multibranch Pipeline Events, Delete Multibranch Pipeline, Build History, Rename, Pipeline Syntax, and Credentials. The main content area displays a message: "This folder is empty" and "There are no branches found that contain buildable projects. Jenkins automatically builds and manages projects in branches that contain recognizable projects. Pipeline Branch projects support building branches within a repository containing a pipeline script. By default it uses a file named Jenkinsfile in the root directory." It also features links for "Configure the project", "Re-index branches", "Creating a Jenkins Pipeline", and "Creating Multibranch Projects".

```
Once done Install kustomize
cd /usr/local/bin
https://github.com/kubernetes-sigs/kustomize/releases/download/kustomize%2Fv5.2.1/kustomize_v5.2.1_linux_amd64.tar.gz
```

```
jenkins@ip-10-0-9-146:~$ exit  
logout  
root@ip-10-0-9-146:~#
```

```
root@ip-10-0-9-146:~# cd /usr/local/bin/  
root@ip-10-0-9-146:/usr/local/bin# wget https://github.com/kubernetes-sigs/kustomize/releases/download/kustomize%2Fv5.2.1/kustomize_v5.2.1_linux_amd64.tar.gz  
--2024-09-04 10:16:44-- https://github.com/kubernetes-sigs/kustomize/releases/download/kustomize%2Fv5.2.1/kustomize_v5.2.1_linux_amd64.tar.gz  
Resolving github.com (github.com)... 140.82.114.3  
Connecting to github.com (github.com)|140.82.114.3|:443... connected.  
HTTP request sent, awaiting response... 302 Found  
Location: https://objects.githubusercontent.com/github-production-release-asset-2e65be/133067498/88dbd3c1-b048-4327-a04f-834d97f5acb67X-Amz-Algorithm=AWS4-HMAC-S  
256X-Amz-Credential=releaseassetproduction%2F20240904%2Fus-east-1%2Fs3%2Faws4-requestX-Amz-Date=20240904T010644Zx-Amz-Expires=3004X-Amz-Signature=d81f97c2cef4  
6d1d46e2013427af79496b5bbcfd1d632ba6377684b19925bf36X-Amz-SignedHeaders=host&actor_id=0&key_id=0&repo_id=133067498&response-content-disposition=attachment%3B%20fi  
lename%3Dkustomize_v5.2.1_linux_amd64.tar.gz&response-content-type=application%2Foctet-stream [following]  
--2024-09-04 10:16:44-- https://objects.githubusercontent.com/github-production-release-asset-2e65be/133067498/88dbd3c1-b048-4327-a04f-834d97f5acb67X-Amz-Algorithm=AWS4-HMAC-S  
256X-Amz-Credential=releaseassetproduction%2F20240904%2Fus-east-1%2Fs3%2Faws4-requestX-Amz-Date=20240904T010644Zx-Amz-Expires=3004X-Amz-Signature=d81f97c2cef4  
6d1d46e2013427af79496b5bbcfd1d632ba6377684b19925bf36X-Amz-SignedHeaders=host&actor_id=0&key_id=0&repo_id=133067498&response-content-disposition=att  
achment%3B%20filename%3Dkustomize_v5.2.1_linux_amd64.tar.gz&response-content-type=application%2Foctet-stream  
Resolving objects.githubusercontent.com (objects.githubusercontent.com)... 185.199.108.133, 185.199.109.133, 185.199.110.133, ...  
Connecting to objects.githubusercontent.com (objects.githubusercontent.com)|185.199.108.133|:443... connected.  
HTTP request sent, awaiting response... 200 OK  
Length: 5946650 (5.7M) [application/octet-stream]  
Saving to: 'kustomize_v5.2.1_linux_amd64.tar.gz'  
  
kustomize_v5.2.1_linux_amd64.tar.gz      100%[=====]   5.67M  --.-KB/s   in 0.05s  
2024-09-04 10:16:44 (104 MB/s) - 'kustomize_v5.2.1_linux_amd64.tar.gz' saved [5946650/5946650]
```

```
root@ip-10-0-9-146:/usr/local/bin# tar zxvf kustomize_v5.2.1_linux_amd64.tar.gz  
kustomize  
root@ip-10-0-9-146:/usr/local/bin#
```

```
root@ip-10-0-9-146:/usr/local/bin#  
root@ip-10-0-9-146:/usr/local/bin# ls -al  
total 215644  
drwxr-xr-x  2 root root    4096 Sep  4 10:16 .  
drwxr-xr-x 11 root root    4096 Sep  4 09:28 ..  
lwxrwxrwx  1 root root     37 Sep  4 09:28 aws -> /usr/local/aws-cli/v2/current/bin/aws  
lwxrwxrwx  1 root root    47 Sep  4 09:28 aws_completer -> /usr/local/aws-cli/v2/current/bin/aws_completer  
-rwx----- 1 1001 127 143462400 Aug 19 11:52 eksctl  
-rwxrwxrwx  1 root root  56381592 Sep  4 09:30 kubectl  
-rwxr-xr-x  1 1001 127 15015936 Oct 19 2023 kustomize  
-rw-r--r--  1 root root  5946650 Oct 19 2023 kustomize_v5.2.1_linux_amd64.tar.gz  
root@ip-10-0-9-146:/usr/local/bin# rm -rf kustomize_v5.2.1_linux_amd64.tar.gz  
root@ip-10-0-9-146:/usr/local/bin#
```

```
# do dummy testing by creating the file and push , Automatically a build should be starated when you select scan multibranch pipeline
```

click on scan mulibranch pipeline

The screenshot shows the Jenkins EKS-JENKINS dashboard. On the left, a sidebar lists options like Status, Configure, Scan Multibranch Pipeline Now, Scan Multibranch Pipeline Log, Multibranch Pipeline Events, Delete Multibranch Pipeline, Build History, Rename, Pipeline Syntax, and Credentials. The main area displays a message: "This folder is empty" with a note about Jenkins automatically building and managing projects. It includes links for "Configure the project", "Re-index branches", "Creating a Jenkins Pipeline", and "Creating Multibranch Projects". Below this, the "Build Queue" section shows "No builds in the queue." The "Build Executor Status" section indicates 1 idle executor. At the bottom, a progress bar shows a build named "Building EKS-JENKINS Multibranch Pipeline" is in progress.

job started

The screenshot shows the "Scan Multibranch Pipeline" page under the EKS-JENKINS job. The sidebar on the left is identical to the dashboard. The main content shows a green checkmark icon and the text "Scan Multibranch Pipeline (Sep 4, 2024, 10:18:29 AM)". It details a successful scan with a duration of 4.3 seconds. A link to the log is provided. The "Build Queue (1)" section at the bottom is highlighted with a red box, showing the entry "EKS-JENKINS » development".

 Status

 Changes

 Build Now

 View Configuration

 Stages

 Pipeline Syntax

 Build History

 Filter... /

 Sep 4, 2024, 10:18 AM

 Atom feed for all  Atom feed for failures

 development

Full project name: EKS-JENKINS/development

Permalinks

- Last build (#1), 1.4 sec ago

For status click above

 Status

 Changes

 Console Output

 Edit Build Information

 Delete build '#1'

 Timings

 Git Build Data

 Pipeline Overview

 Pipeline Console

 Restart from Stage

 Replay

 Pipeline Steps

 Workspaces

 Build #1 (Sep 4, 2024, 10:18:39 AM)

 Branch indexing

 This run spent:

- 5.4 sec waiting;
- 5.1 sec build duration;
- 10 sec total from scheduled to completion.

 Revision: aabd1fcf41aa6c6335f300171a0f402bff4429f
Repository: git@github.com:sairiranpi/eks-jenkins.git

- development

console Output – error

```
Branch indexing
> git rev-parse --resolve-git-dir /var/lib/jenkins/caches/git-lee21d608bcfaefcbe5146/.git # timeout=10
Setting origin to git@github.com:saikirangi/eks-jenkins.git
> git config remote.origin.url git@github.com:saikirangi/eks-jenkins.git # timeout=10
Fetching origin...
Fetching upstream changes from origin
> git --version # timeout=10
> git --version # 'git version 2.34.1'
> git config --get remote.origin.url # timeout=10
using GIT_SSH to set credentials github-access
> git fetch --tags --force --progress -- origin +refs/heads/*:refs/remotes/origin/* # timeout=10
Seen branch in repository origin/development
Seen 1 remote branch
Obtained Jenkinsfile from aabd1ccf41aa6c6335f300171a0f402bff4429f
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/EKS-JENKINS_development
[Pipeline] {
[Pipeline] stage
[Pipeline] {
  (Declarative: Checkout SCM)
[Pipeline] checkout
Selected Git installation does not exist. Using Default
The recommended git tool is: NONE
using credential github-access
Cloning the remote Git repository
Cloning with configured refspecs honoured and without tags
Cloning repository git@github.com:saikirangi/eks-jenkins.git
> git init /var/lib/jenkins/workspace/EKS-JENKINS_development # timeout=10
Fetching upstream changes from git@github.com:saikirangi/eks-jenkins.git
> git --version # timeout=10
> git --version # 'git version 2.34.1'
```

```
> git checkout -f aabd1ccf41aa6c6335f300171a0f402bff4429f # timeout=10
Commit message: "added the dummy file to test the integration"
First time build. Skipping changelog.
[Pipeline]
[Pipeline] // stage
[Pipeline] withEnv
[Pipeline] {
[Pipeline] withEnv
[Pipeline] {
[Pipeline] stage
[Pipeline] {
  (Checking EKS Access)
[Pipeline] withAWS
Constructing AWS Credentials[Pipeline] {
[Pipeline] sh
+ aws eks update-kubeconfig --region us-east-1 --name eks-cluster
An error occurred (ResourceNotFoundException) when calling the DescribeCluster operation: No cluster found for name: eks-cluster.
[Pipeline]
[Pipeline] // withAWS
[Pipeline]
[Pipeline] // stage
[Pipeline] stage
[Pipeline] {
  (Creating EKS Namespaces)
Stage "Creating EKS Namespaces" skipped due to earlier failure(s)
[Pipeline] getContext
[Pipeline]
[Pipeline] // stage
[Pipeline] stage
[Pipeline] {
  (Deploy To Dev Namespace)
Stage "Deploy To Dev Namespace" skipped due to earlier failure(s)
[Pipeline] getContext
[Pipeline]
[Pipeline] // stage
[Pipeline] stage
[Pipeline] {
  (Destroy App In Dev Namespace)
Stage "Destroy App In Dev Namespace" skipped due to earlier failure(s)
[Pipeline] getContext
[Pipeline]
```

rename eks-cluster to eks-cluster1

The screenshot shows a terminal window with the following Jenkinsfile content:

```
1 pipeline {
2     agent any
3     environment {
4         DEV_DESTROY = "YES"
5         PROD_DESTROY = "YES"
6     }
7
8     stages {
9         stage('Checking EKS Access') {
10            steps {
11                withAWS(credentials: 'aws-creds') {
12                    sh 'aws eks update-kubeconfig --region us-east-1 --name eks-cluster-1'
13                    sh 'kubectl get pods -A'
14                }
15            }
16        }
17
18        stage('Creating EKS Namespaces') {
19            steps {
20                withAWS(credentials: 'aws-creds') {
21                    sh 'aws eks update-kubeconfig --region us-east-1 --name eks-cluster-1'
22                    sh 'kubectl create ns development || exit 0'
23                    sh 'kubectl create ns production || exit 0'
24                }
25            }
26        }
27
28        stage('Deploy To Dev Namespace') {
29            when {
30                branch 'development'
31            }
32            steps {
33                withAWS(credentials: 'aws-creds') {
34                    sh 'aws eks update-kubeconfig --region us-east-1 --name eks-cluster-1'
35                    sh 'ls -al'
36                }
37            }
38        }
39    }
40 }
```

To the right of the terminal, there is a separate window titled "eks-cluster" showing the details of "eks-cluster-1".

```
root@ip-10-0-9-146:/usr/local/bin# cd
root@ip-10-0-9-146:~# ku get ns
NAME          STATUS   AGE
default       Active   44m
development   Active   21m
kube-node-lease Active   44m
kube-public   Active   44m
kube-system   Active   44m
production    Active   21m
root@ip-10-0-9-146:~# ku delete ns development production
namespace "development" deleted
namespace "production" deleted
^L
```

```

MINGW64:/c/Users/saikirana/OneDrive/Desktop/EKS-jenkins
$ git status
On branch development
nothing to commit, working tree clean

saiki@Saikiran236 MINGW64 ~/OneDrive/Desktop/EKS-jenkins (development)
$ git status
On branch development
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
    (use "git restore <file>..." to discard changes in working directory)
      modified: Jenkinsfile

no changes added to commit (use "git add" and/or "git commit -a")

saiki@Saikiran236 MINGW64 ~/OneDrive/Desktop/EKS-jenkins (development)
$ git add .
$ git commit -m "added the cluster name as eks-cluster-1"
warning: in the working copy of 'Jenkinsfile', LF will be replaced by CRLF the next time Git touches it
[development abad1cf] added the cluster name as eks-cluster-1
 1 file changed, 6 insertions(+), 6 deletions(-)

saiki@Saikiran236 MINGW64 ~/OneDrive/Desktop/EKS-jenkins (development)
$ git push origin development
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 463 bytes | 463.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object
To https://github.com/saikiranpi/eks-jenkins.git
  abad1cf..abad1cf [development] -> development

```

development

1 Branch 0 Tags

Go to file Add file Code

Author	Message	Time	Commits
saikiranpi	added the cluster name as eks-cluster-1	aba2692 · now	3 Commits
kustomize	added all integration files of Jenkins and EKS-v01	8 minutes ago	
Jenkinsfile	added the cluster name as eks-cluster-1	now	
README.md	added the dummy file to test the integration	3 minutes ago	
Steps	added all integration files of Jenkins and EKS-v01	8 minutes ago	
README			

eks Jenkins .. scan multi-branch pipeline now

Dashboard > EKS-JENKINS >

Status EKS-JENKINS

Configure Scan Multibranch Pipeline Now Scan Multibranch Pipeline Log Multibranch Pipeline Events Delete Multibranch Pipeline Build History Project Relationship Check File Fingerprint Rename Pipeline Syntax

Disable Multibranch Pipeline

S	W	Name	Last Success	Last Failure	Last Duration
		development	N/A	3 min 8 sec #1	5.1 sec

Icon: S M L

[Status](#)[Changes](#)[Build Now](#)[View Configuration](#)[Stages](#)[Pipeline Syntax](#)

development

Full project name: EKS-JENKINS/development

Permalinks

- [Last build \(#1\), 3 min 16 sec ago](#)
- [Last failed build \(#1\), 3 min 16 sec ago](#)
- [Last unsuccessful build \(#1\), 3 min 16 sec ago](#)
- [Last completed build \(#1\), 3 min 16 sec ago](#)

Build History

trend

Filter...

#2 (pending—In the quiet period. Expires in 0.45 sec)

#1 Sep 4, 2024, 10:18 AM

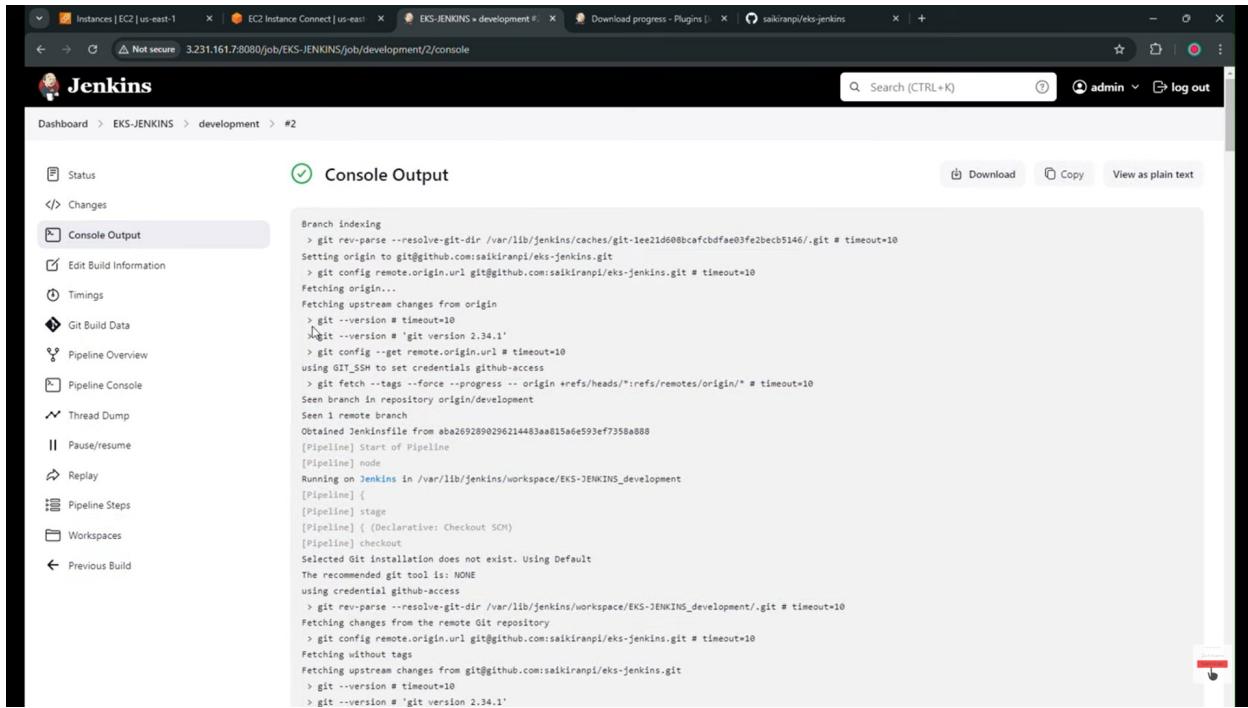
Atom feed for all Atom feed for failure

console O/p

```

[Pipeline] { ((Checking EKS Access))
[Pipeline] withAWS
Constructing AWS Credentials[Pipeline]
[Pipeline] sh
+ aws eks update-kubeconfig --region us-east-1 --name eks-cluster-1
Updated context arn:aws:eks:us-east-1:211125710812:cluster/eks-cluster-1 in /var/lib/jenkins/.kube/config
[Pipeline] sh
+ kubectl get pods -A
NAMESPACE     NAME           READY   STATUS    RESTARTS   AGE
kube-system   aws-node-jqj27   2/2     Running   0          33m
kube-system   aws-node-s6pvv   2/2     Running   0          33m
kube-system   coredns-54d6f577c6-5mhj   1/1     Running   0          43m
kube-system   coredns-54d6f577c6-rdftn  1/1     Running   0          43m
kube-system   kube-proxy-4duqq   1/1     Running   0          33m
kube-system   kube-proxy-r6bh5   1/1     Running   0          33m
[Pipeline] }
[Pipeline] // withAWS
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { ((Creating EKS Namespaces))
[Pipeline] withAWS
Constructing AWS Credentials[Pipeline]
[Pipeline] sh
+ aws eks update-kubeconfig --region us-east-1 --name eks-cluster-1
Updated context arn:aws:eks:us-east-1:211125710812:cluster/eks-cluster-1 in /var/lib/jenkins/.kube/config
[Pipeline] sh
+ kubectl create ns development
namespace/development created
[Pipeline] sh
+ kubectl create ns production

```



Final step automation on Webhook triggers

```

# Now i want auto commit not by just clicking on multibranch pipeline
multibranch.jenkins > configure > scan multibranch > Scan Multibranch Pipeline Triggers > scan by webhook > and carefully give the webhook accordingly...
http://ec2-3-231-161-7.compute-1.amazonaws.com:8080/multibranch-webhook-trigger/invoke?token=eks-cluster-1

ADD THE SAME ON GITHUB WEBHOOKS TOO

delete """eksctl delete cluster --name eks-cluster-1"""

```

go to instance copy DNS

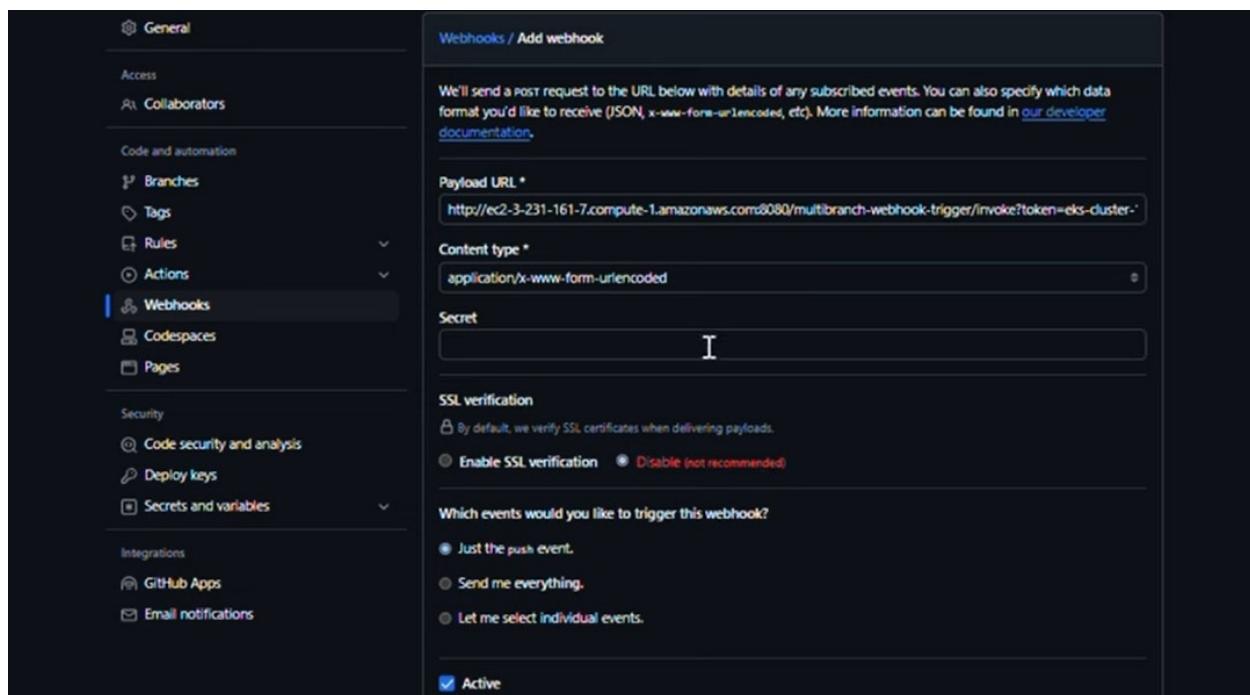
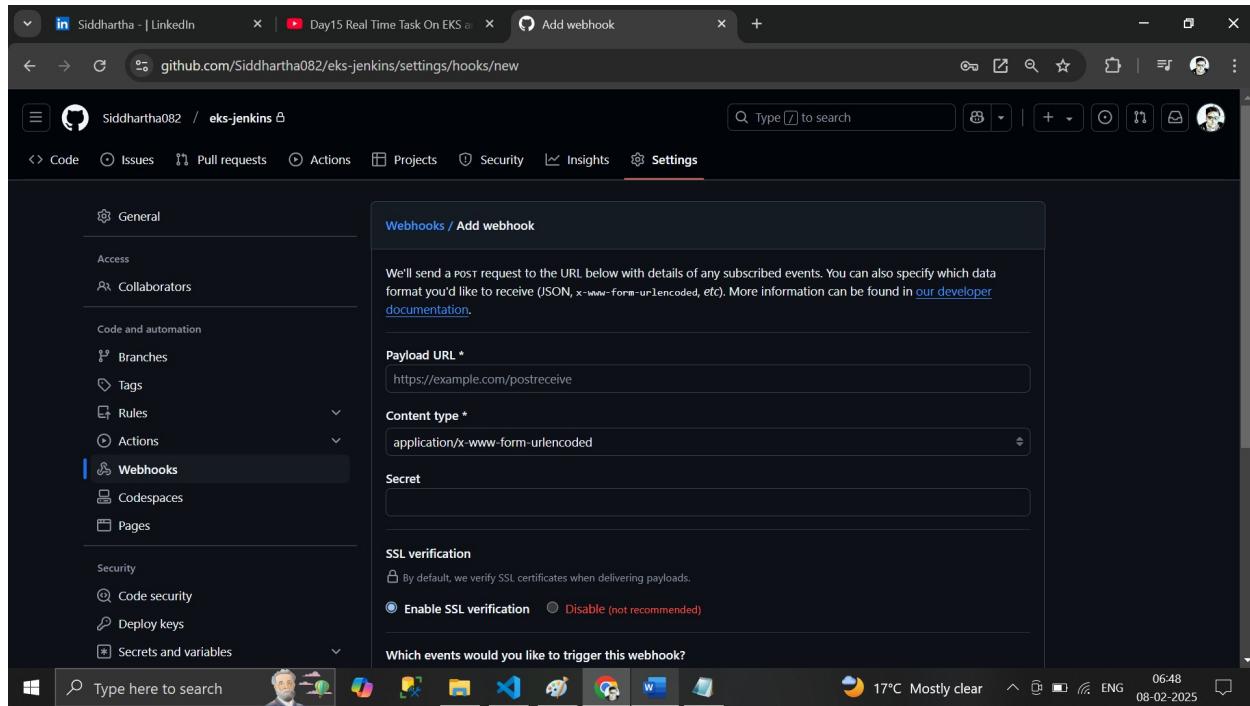
The screenshot shows the AWS EC2 Instances page. The left sidebar includes options like EC2 Dashboard, EC2 Global View, Events, Instances (selected), Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity, Reservations, Images (AMIs, AMI Catalog), and Elastic Block Store (Security Groups, Elastic IPs, Placement Groups, Key Pairs). The main content area displays a table of instances:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Pu...
eks-cluster-1-e...	i-0e90abf9051ef47f0	Running	t3.medium	3/3 checks passed	View alarms	us-east-1a	ec2
i-01efd3d0903368f75	i-01efd3d0903368f75	Running	t3.medium	0/3 checks passed	View alarms	us-east-1b	ec2
EKS-Jenkins	i-0cc39b1ffde18a629	Running	t2.xlarge	2/2 checks passed	View alarms	us-east-1a	ec2

Below the table, the details for the instance **i-0cc39b1ffde18a629 (EKS-Jenkins)** are shown. The **Details** tab is selected, displaying information such as Instance ID (i-0cc39b1ffde18a629 (EKS-Jenkins)), Public IPv4 address (3.231.161.7), Private IPv4 addresses (eg2-3-231-161-7.compute-1.amazonaws.com), Instance state (Running), Hostname type (IP name: ip-10-0-9-146.ec2.internal), Private IP DNS name (ip-10-0-9-146.ec2.internal), Instance type (t2.xlarge), VPC ID (vpc-01aadfa85bf7bcc5 (Docker-Swarm-vpc)), and Elastic IP addresses (none listed).



```
# go to Github repo- settings- Webhooks- add
```



multibranch jenkins > configure > scan multibranch > Scan Multibranch Pipeline Triggers > scan by webhook > and carefully give the webhook accordingly..

Dashboard > EKS-JENKINS >

EKS-JENKINS

Status Config

Scan Multibranch Pipeline Now

Scan Multibranch Pipeline Log

Multibranch Pipeline Events

Delete Multibranch Pipeline

Build History

Project Relationship

Check File Fingerprint

Rename

Pipeline Syntax

Branches (1)

S	W	Name	Last Success	Last Failure	Last Duration
		development	2 min 5 sec #2	5 min 25 sec #1	15 sec

Icon: S M L

Disable Multibranch Pipeline

click configure

Instances | EC2 | us-east-1 | EC2 Instance Connect | us-east-1 | EKS-JENKINS Config [jenkins] | Download progress - Plugins | Webhooks · Settings · saikiranpi | + | admin | log out

Jenkins

Dashboard > EKS-JENKINS > Configuration

Configuration

General

Enabled

Display Name

Description

Plain text [Preview](#)

Branch Sources

Git

Project Repository git@github.com:saikirangi/eks-jenkins.git

Credentials jenkins (github-access)

+ Add

Behaviors

Scan Abort

scan multibranch pipeline triggers

The screenshot shows the Jenkins configuration page for the 'EKS-JENKINS' job. Under the 'Mode' section, 'by Jenkinsfile' is selected. In the 'Scan Multibranch Pipeline Triggers' section, 'Scan by webhook' is checked, and the 'Trigger token' dropdown is set to 'eks-cluster-1'. The 'Orphaned Item Strategy' section includes options for 'Abort builds' and 'Discard old items', with 'Days to keep old items' set to 0. A 'Save' button is visible at the bottom.

The screenshot shows the Jenkins configuration page for the 'EKS-JENKINS' job. Under the 'Properties' section, 'Max # of old items to keep' is set to 0. Other sections like 'Appearance' and 'Health metrics' are also visible. A 'Save' button is highlighted with a mouse cursor.

Automatically Job running

The screenshot shows the Jenkins EKS-JENKINS dashboard. On the left, there's a sidebar with various Jenkins management options like Status, Configure, Scan Multibranch Pipeline Now, Scan Multibranch Pipeline Log, Multibranch Pipeline Events, Delete Multibranch Pipeline, Build History, Project Relationship, Check File Fingerprint, Rename, Pipeline Syntax, and Credentials. The main area is titled "EKS-JENKINS" and shows a table for "Branches (1)". The table has columns for S (Status), W (Webhook), Name, Last Success, Last Failure, and Last Duration. There is one entry for "development". A "Disable Multibranch Pipeline" button is located in the top right corner of the branches table. Below the branches table, there are sections for "Build Queue" (No builds in the queue) and "Build Executor Status" (1 Idle, 2 Idle).

GitHub Webhook

The screenshot shows the GitHub Settings page with the "Webhooks / Manage webhook" section selected. The left sidebar includes "Pull requests", "Actions", "Projects", "Security", "Insights", and "Settings". Under "Settings", the "Webhooks" option is highlighted. The main area shows a "Recent Deliveries" tab with a list of recent webhook deliveries. Each delivery entry includes a status icon, the delivery ID, the event type (e.g., push, ping), and the timestamp (e.g., 2024-09-04 15:53:52). The first entry is a "ping" event from a repository named "c7e45f5b-6aa7-11ef-95c7-5534091301ae".

This screenshot is similar to the previous one, showing the GitHub Settings page with the "Webhooks / Manage webhook" section. The "Recent Deliveries" tab is selected, and the list of deliveries has grown. It now includes five entries, each with a green checkmark icon, a delivery ID, an event type, and a timestamp. The latest entry is a "push" event from a repository named "a5e30fc-6aa8-11ef-9288-18a0b13024b" at 2024-09-04 16:02:45.

