CICD Integration Pipeline Using Jenkins, Docker, AWS ECR, ECS and Slack

Github :- For Repository

https://github.com/Hussain147/paac-with-ecs.git

Jenkins :- For CICD Integration

Docker :- To Containerize the App

AWS Services :- To continue the flow of execution

AWS ECR :- To Register our Image that we built

AWS ECS :- To Run on a Container

Slack :- To Get the Notifications of the Jobs(Execution)

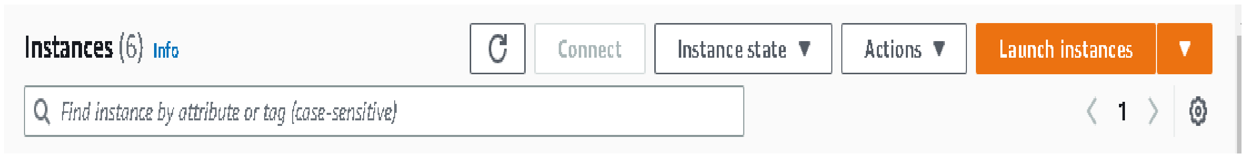
1. **Flow of Continuous Integration Pipeine:-**

$$\begin{tikzcd}
\text{Developer} \arrow[dd] & & \\
& \text{(Jenkins)} & \text{(Docker)} \\
\text{Github} \arrow[r] & \text{Fetch Code} \arrow[r] & \text{Build} \arrow[d] \\
& & \text{Registry (ECR)} \arrow[d] \\
& & \text{Container (ECS)}
\end{tikzcd}$$

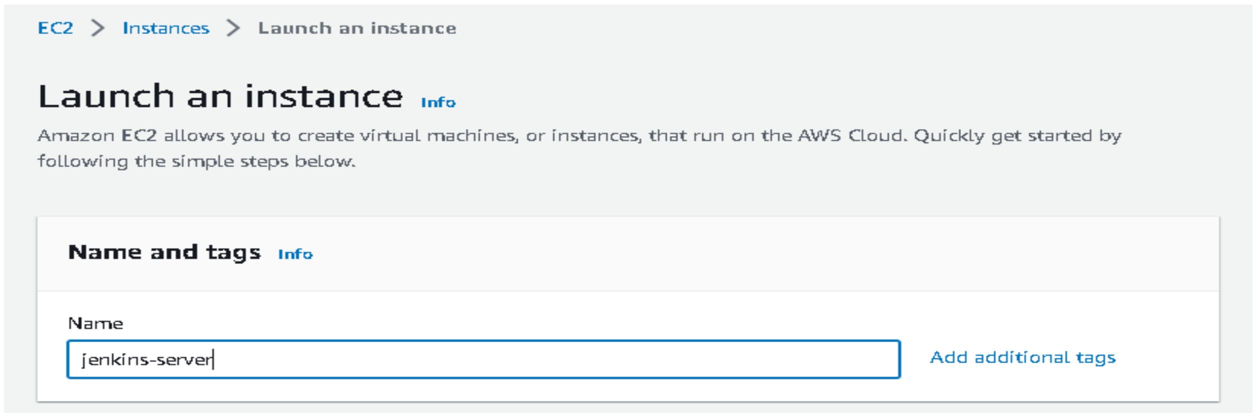
2. **<u>Installation of Jenkins</u>**:-

Let's install Jenkins on Amazon Linux 2 Server

Go to EC2 > Launch Instance >



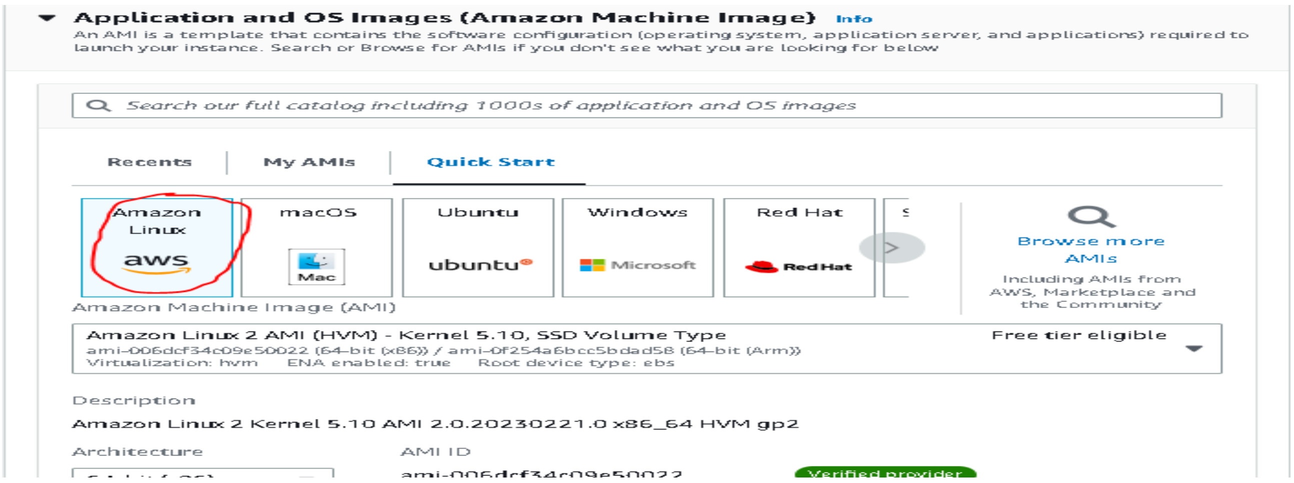
Name : Jenkins-server



Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

Name: jenkins-server

Select **Amazon Linux 2**


▼ Application and OS Images (Amazon Machine Image) Info

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Search our full catalog including 1000s of application and OS images
Recents
My AMIs
Quick Start
Amazon Linux
aws
macOS
Mac
Ubuntu
ubuntu
Windows
Microsoft
Red Hat
Red Hat
Browse more AMIs
Including AMIs from AWS, Marketplace and the Community

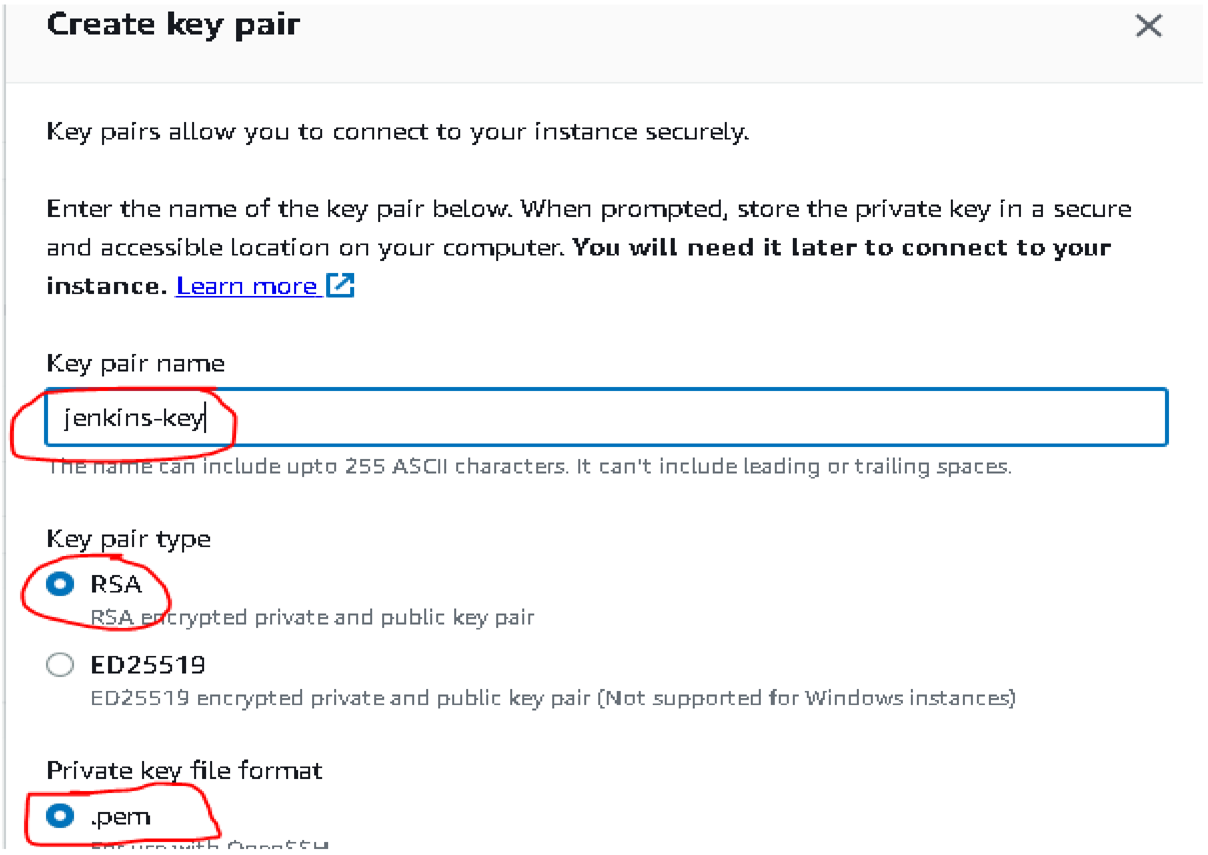
Amazon Machine Image (AMI)

Amazon Linux 2 AMI (HVM) - Kernel 5.10, SSD Volume Type — Free tier eligible
ami-006dcf34c09e50022 (64-bit (x86)) / ami-0f254a6bcc5bdad58 (64-bit (Arm))
Virtualization: hvm ENA enabled: true Root device type: ebs

Description
Amazon Linux 2 Kernel 5.10 AMI 2.0.20230221.0 x86_64 HVM gp2

Architecture
AMI ID
ami-006dcf34c09e50022

Create a **New Key Pair**: Jenkins-key


Create key pair

Key pairs allow you to connect to your instance securely.

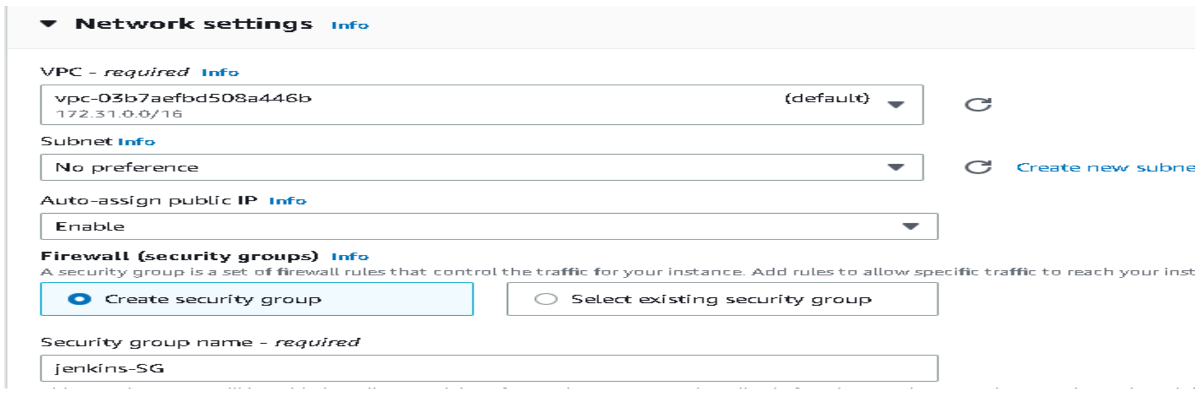
Enter the name of the key pair below. When prompted, store the private key in a secure and accessible location on your computer. **You will need it later to connect to your instance.** Learn more

Key pair name
jenkins-key
The name can include upto 255 ASCII characters. It can't include leading or trailing spaces.

Key pair type
[✓ RSA] RSA encrypted private and public key pair
ED25519 ED25519 encrypted private and public key pair (Not supported for Windows instances)

Private key file format
[✓ .pem] For use with OpenSSH

Create a New Security Group : Jenkins-SG



Network settings Info

VPC - *required* Info

vpc-03b7aefbd508a446b (default)
172.31.0.0/16

Subnet Info

No preference

Create new subne

Auto-assign public IP Info

Enable

Firewall (security groups) Info

A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your inst

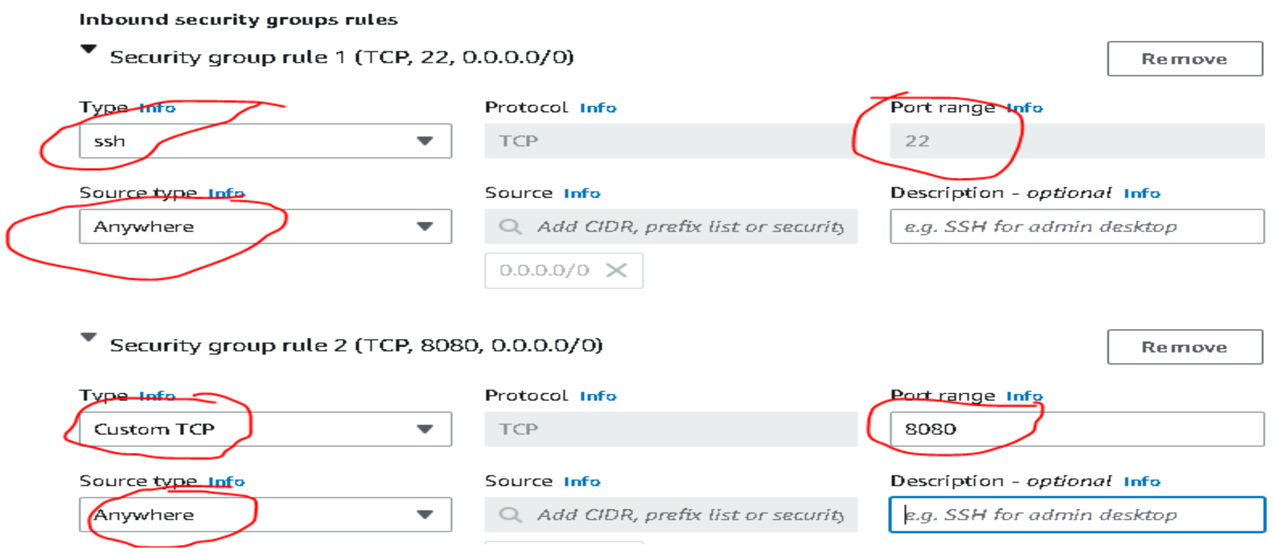
Create security group [✓] | Select existing security group

Security group name - *required*

jenkins-SG

Add these **Inbound Rules**:- SSH : 22 : Anywhere

Custom TCP : 8080 : Anywhere



Inbound security groups rules

Security group rule 1 (TCP, 22, 0.0.0.0/0) — Remove

Type Info: ssh

Protocol Info: TCP

Port range Info: 22

Source type Info: Anywhere

Source Info: *Add CIDR, prefix list or security* — 0.0.0.0/0

Description - *optional* Info: *e.g. SSH for admin desktop*

Security group rule 2 (TCP, 8080, 0.0.0.0/0) — Remove

Type Info: Custom TCP

Protocol Info: TCP

Port range Info: 8080

Source type Info: Anywhere

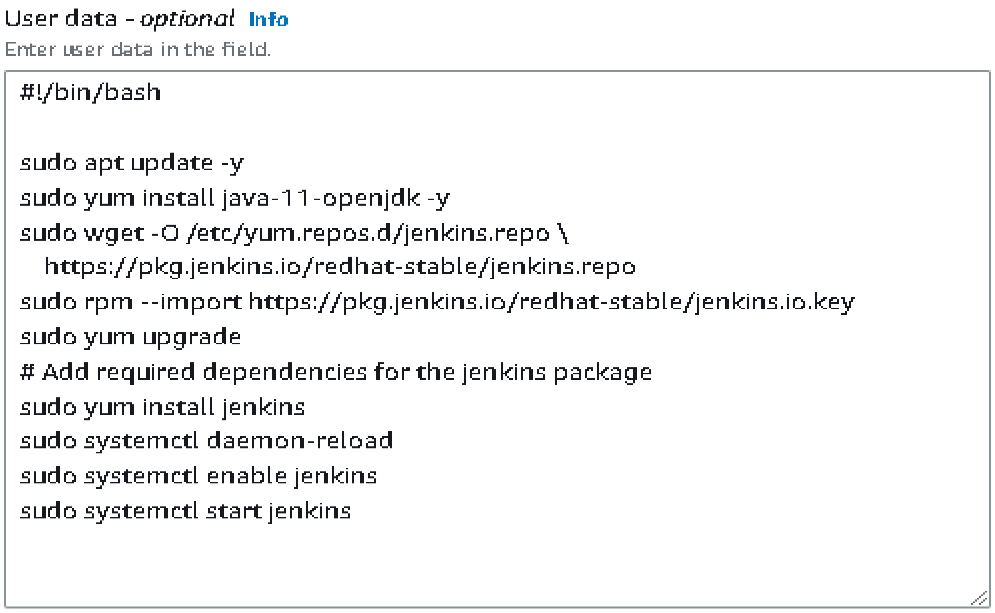
Source Info: *Add CIDR, prefix list or security*

Description - *optional* Info: *e.g. SSH for admin desktop*

Click On Advanced Details > Go to User Data > Paste the Jenkins Installation Scripts which is given in the Link :-

https://github.com/Hussain147/paac-with-ecs/blob/main/jenkins%20installation/jenkins-installation_on_Amazon_linux_2.txt

Note : This script will not only install Jenkins



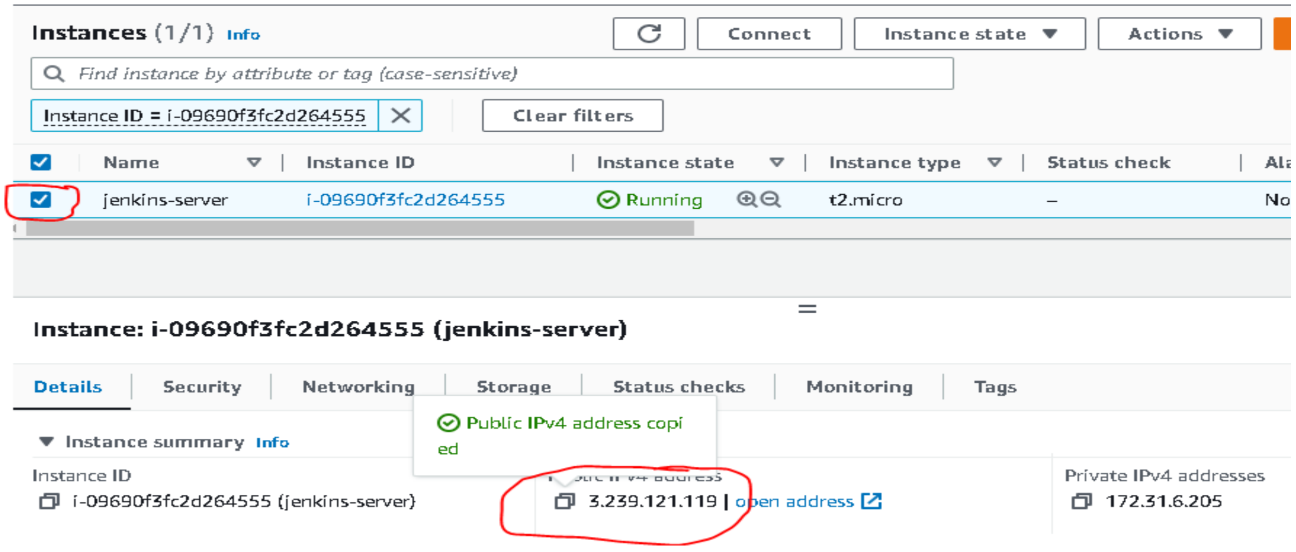
User data - *optional* Info

Enter user data in the field.

```
#!/bin/bash

sudo apt update -y
sudo yum install java-11-openjdk -y
sudo wget -O /etc/yum.repos.d/jenkins.repo \
  https://pkg.jenkins.io/redhat-stable/jenkins.repo
sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io.key
sudo yum upgrade
# Add required dependencies for the jenkins package
sudo yum install jenkins
sudo systemctl daemon-reload
sudo systemctl enable jenkins
sudo systemctl start jenkins
```

Now, Click on **Launch Instance**

Now, copy the Public Ipv4 Address & paste it in the URL with Port 8080:



Instances (1/1) Info
Connect
Instance state
Actions
Find instance by attribute or tag (case-sensitive)
Instance ID = i-09690f3fc2d264555
Clear filters

Name	Instance ID	Instance state	Instance type	Status check	Ala
jenkins-server	i-09690f3fc2d264555	Running	t2.micro	–	No

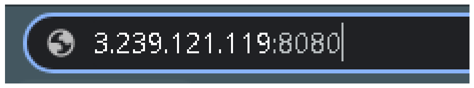
Instance: i-09690f3fc2d264555 (jenkins-server)

Public IPv4 address copied

Instance ID: i-09690f3fc2d264555 (jenkins-server)

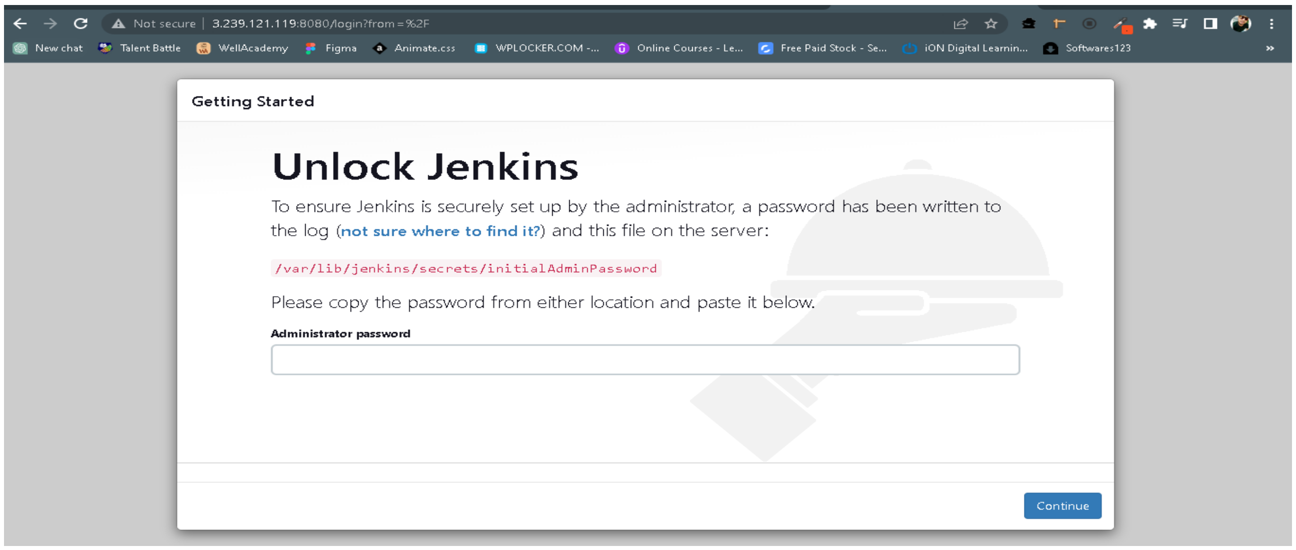
Public IPv4 address: 3.239.121.119 | open address

Private IPv4 addresses: 172.31.6.205



3.239.121.119:8080

You will get the Jenkins welcome page



Unlock Jenkins

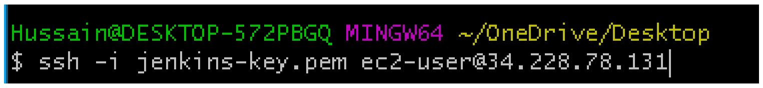
To ensure Jenkins is securely set up by the administrator, a password has been written to the log (not sure where to find it?) and this file on the server:

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

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

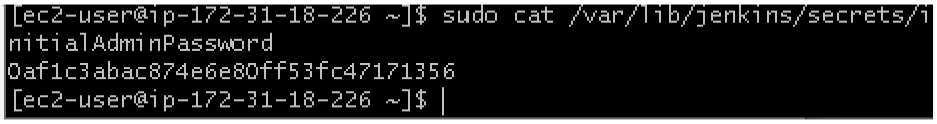
Administrator password

Now, Connect via SSH >



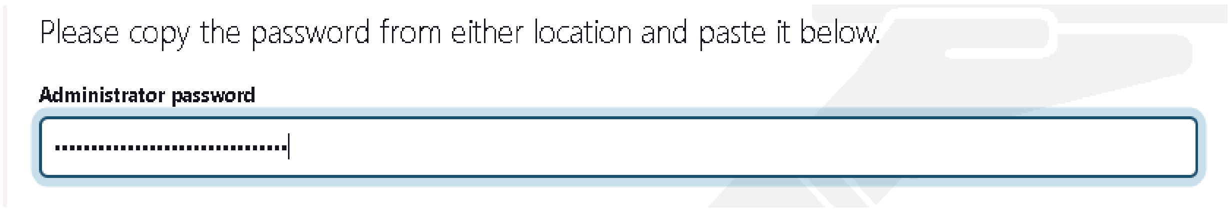
```
Hussain@DESKTOP-572PBGQ MINGW64 ~/OneDrive/Desktop
$ ssh -i jenkins-key.pem ec2-user@34.228.78.131
```

Read the content of this directory :- /var/lib/jenkins/secrets/initialAdminPassword



```
[ec2-user@ip-172-31-18-226 ~]$ sudo cat /var/lib/jenkins/secrets/i
nitialAdminPassword
0af1c3abac874e6e80ff53fc47171356
[ec2-user@ip-172-31-18-226 ~]$ |
```

Copy the Admin Password & paste it in the Jenkins Page



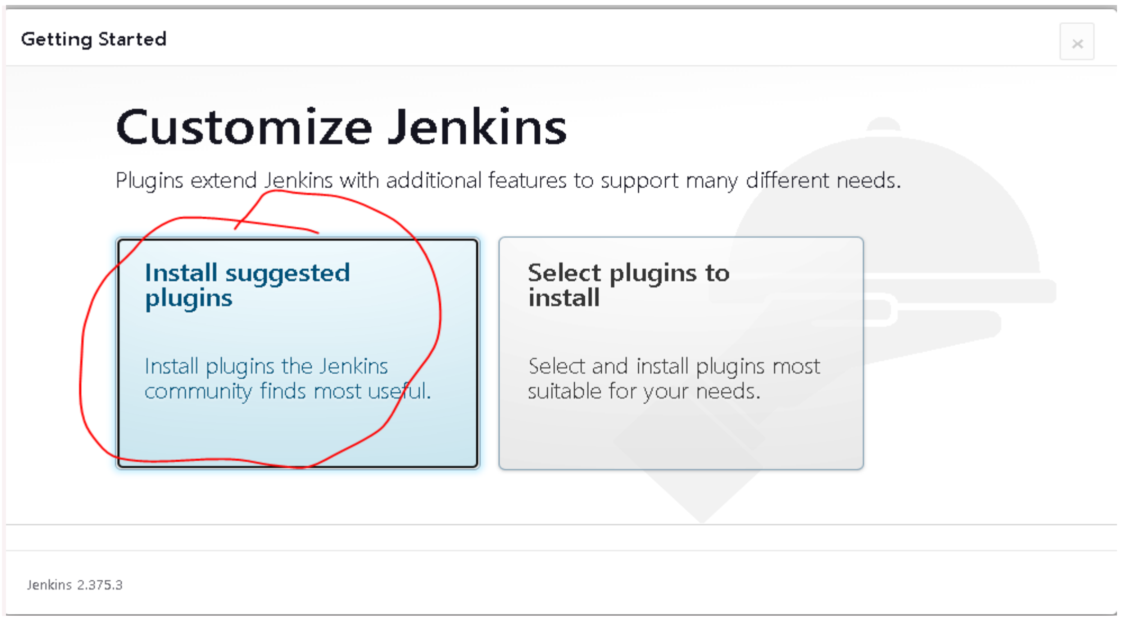
Please copy the password from either location and paste it below.

Administrator password

....

Click **Continue**

Click on **Install Suggested Plugins**



Getting Started

Customize Jenkins

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

Install suggested plugins

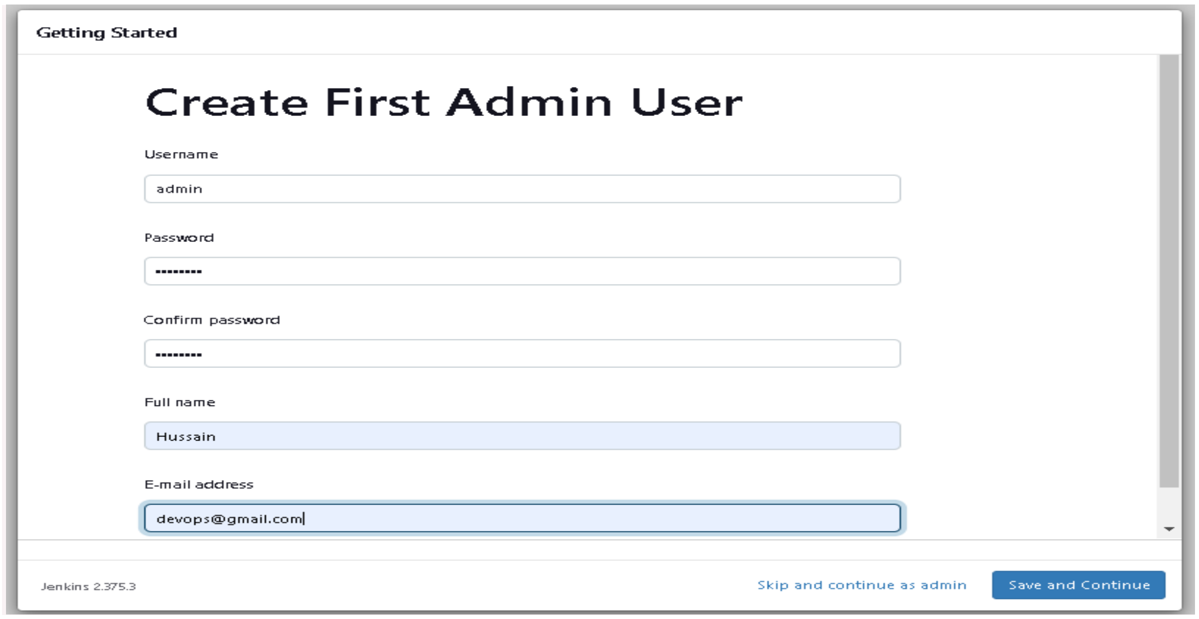
Install plugins the Jenkins community finds most useful.

Select plugins to install

Select and install plugins most suitable for your needs.

Jenkins 2.375.3

Setup the Credentials >



Getting Started

Create First Admin User

Username: admin

Password: ·········

Confirm password: ·········

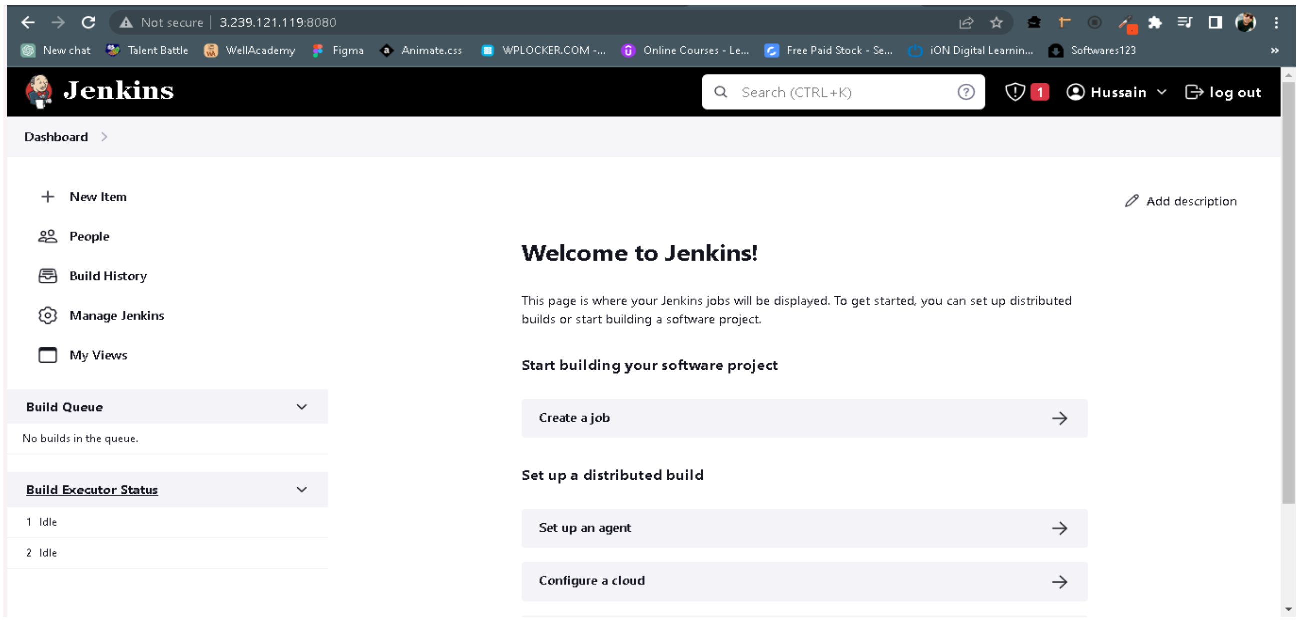
Full name: Hussain

E-mail address: devops@gmail.com

Jenkins 2.375.3

Click on **Save & Continue** > **Save & Finish** > **Start Using Jenkns**

You will see the UI as below:-



No builds in the queue.

1 Idle

2 Idle

Welcome to Jenkins!

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.

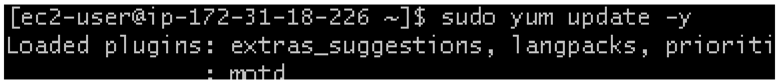
Start building your software project

Set up a distributed build

3. Docker Installation on Jenkins Server:-

Connect to Jenkins server via SSH and start installing docker

sudo yum update -y



```
[ec2-user@ip-172-31-18-226 ~]$ sudo yum update -y
Loaded plugins: extras_suggestions, langpacks, prioriti
              : motd
```

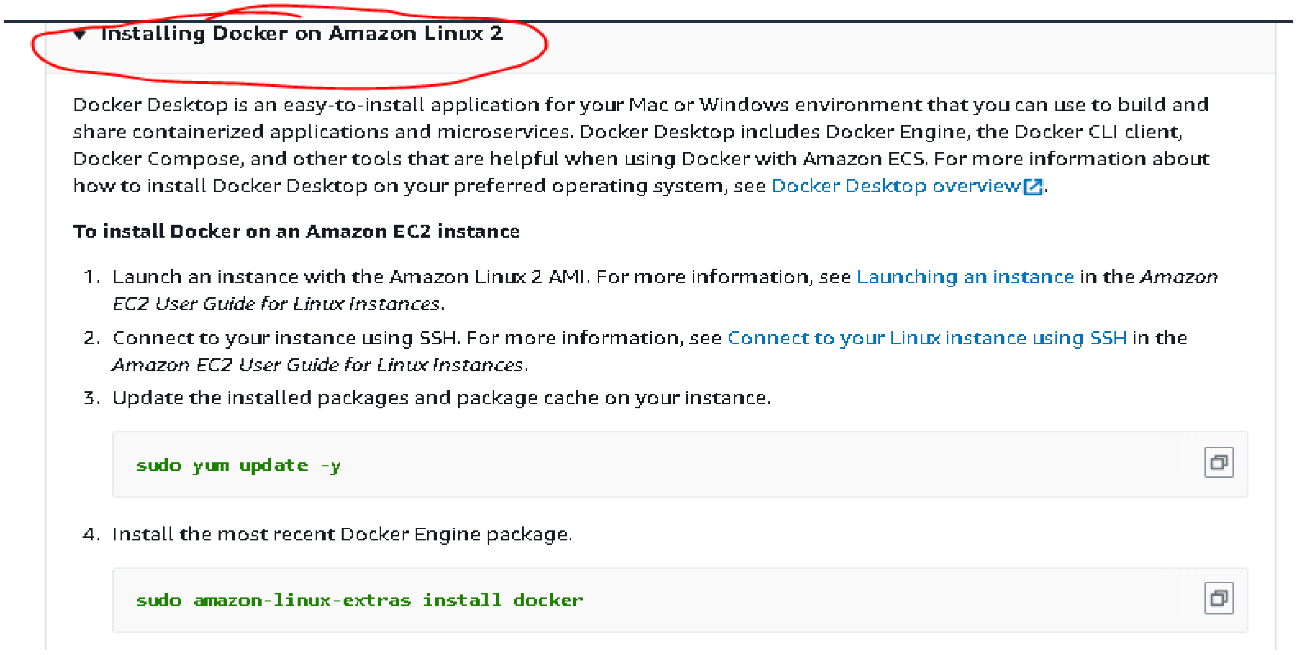
Now go to this link ->

https://github.com/Hussain147/paac-with-ecs/blob/main/docker%20installation/docker_install_Amazon_linux2.txt

(or)

https://docs.aws.amazon.com/AmazonECS/latest/developerguide/create-container-image.html

Install Docker as per the documentation:-



▼ Installing Docker on Amazon Linux 2

Docker Desktop is an easy-to-install application for your Mac or Windows environment that you can use to build and share containerized applications and microservices. Docker Desktop includes Docker Engine, the Docker CLI client, Docker Compose, and other tools that are helpful when using Docker with Amazon ECS. For more information about how to install Docker Desktop on your preferred operating system, see Docker Desktop overview.

To install Docker on an Amazon EC2 instance

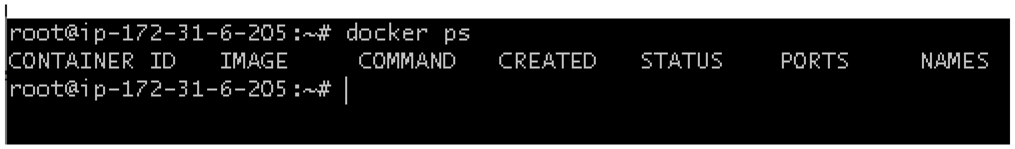
1. Launch an instance with the Amazon Linux 2 AMI. For more information, see Launching an instance in the *Amazon EC2 User Guide for Linux Instances*.
2. Connect to your instance using SSH. For more information, see Connect to your Linux instance using SSH in the *Amazon EC2 User Guide for Linux Instances*.
3. Update the installed packages and package cache on your instance.

```
sudo yum update -y
```

4. Install the most recent Docker Engine package.

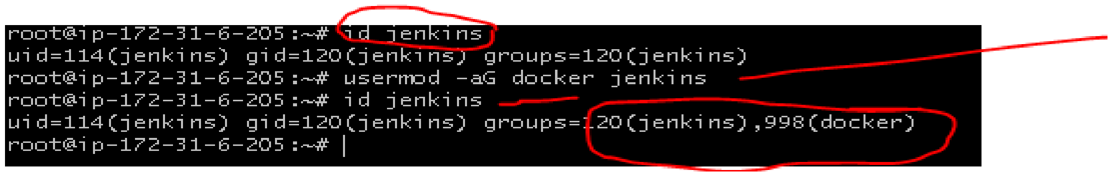
```
sudo amazon-linux-extras install docker
```

Once it is installed, validate the docker :- docker ps



```
root@ip-172-31-6-205:~# docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
root@ip-172-31-6-205:~# |
```

Now, add our **Jenkins user** to the **Docker Group**. So that the Jenkins will use the docker.



```
root@ip-172-31-6-205:~# id jenkins
uid=114(jenkins) gid=120(jenkins) groups=120(jenkins)
root@ip-172-31-6-205:~# usermod -aG docker jenkins
root@ip-172-31-6-205:~# id jenkins
uid=114(jenkins) gid=120(jenkins) groups=120(jenkins),998(docker)
root@ip-172-31-6-205:~# |
```

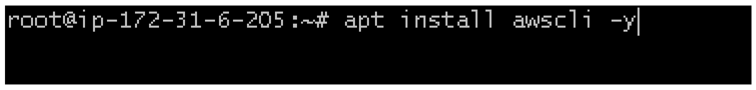
Run these commands :-

sudo service jenkins restart

sudo systemctl daemon reload

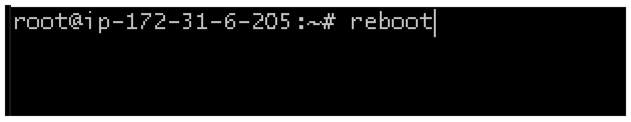
sudo service docker restart

Install AWSCLI for future purpose while delivering the artifact



```
root@ip-172-31-6-205:~# apt install awscli -y
```

Now **reboot** the server

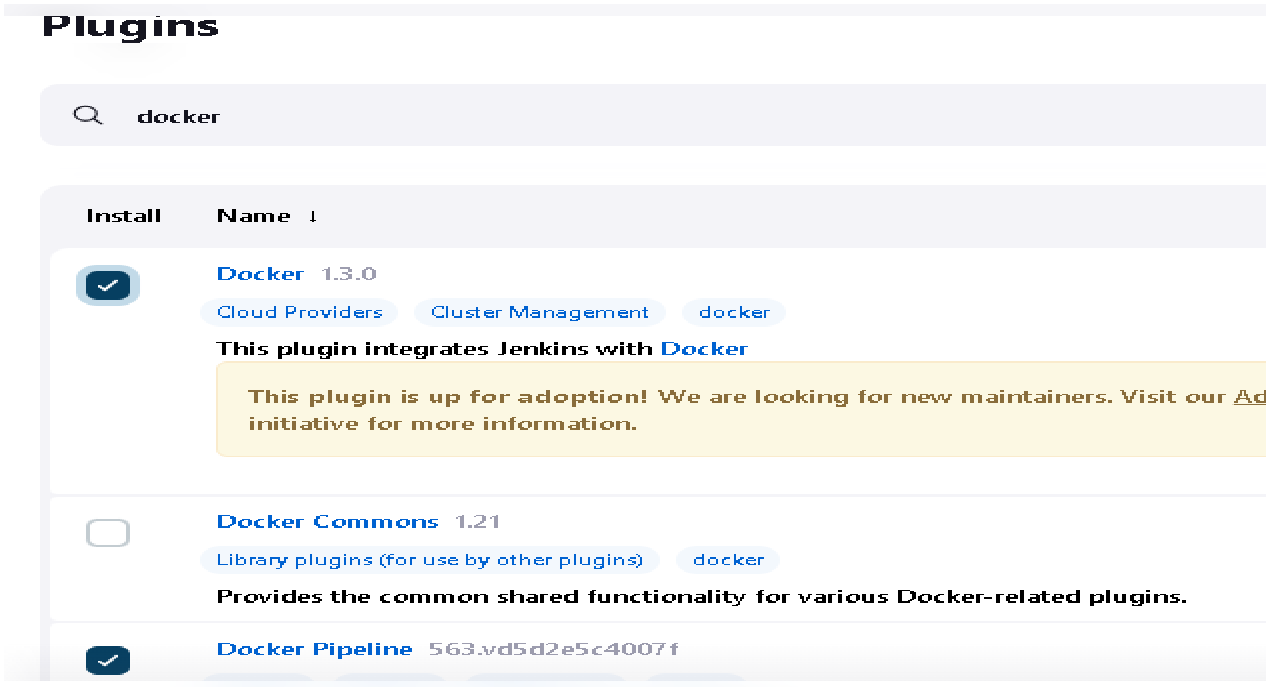


```
root@ip-172-31-6-205:~# reboot
```

Now, Go to Jenkins Dashboard > Manage Jenkins > manage plugins > Available :-

Install Plugins :-

- Docker
- Docker pipeline



Plugins

docker

Install Name ↓

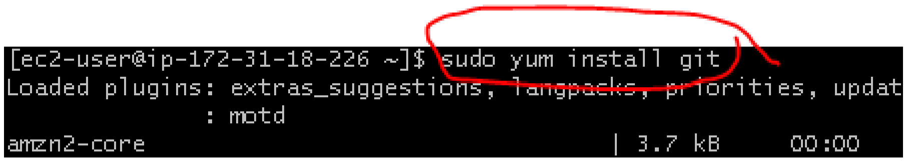
✓ Docker 1.3.0
Cloud Providers | Cluster Management | docker
This plugin integrates Jenkins with Docker
This plugin is up for adoption! We are looking for new maintainers. Visit our Ad
initiative for more information.

☐ Docker Commons 1.21
Library plugins (for use by other plugins) | docker
Provides the common shared functionality for various Docker-related plugins.

✓ Docker Pipeline 563.vd5d2e5c4007f

Click on **Install without Restart**

Now, go to ssh & **install git** :- yum install git



```
[ec2-user@ip-172-31-18-226 ~]$ sudo yum install git
Loaded plugins: extras_suggestions, langpacks, priorities, updat
              : motd
amzn2-core                                    | 3.7 kB     00:00
```

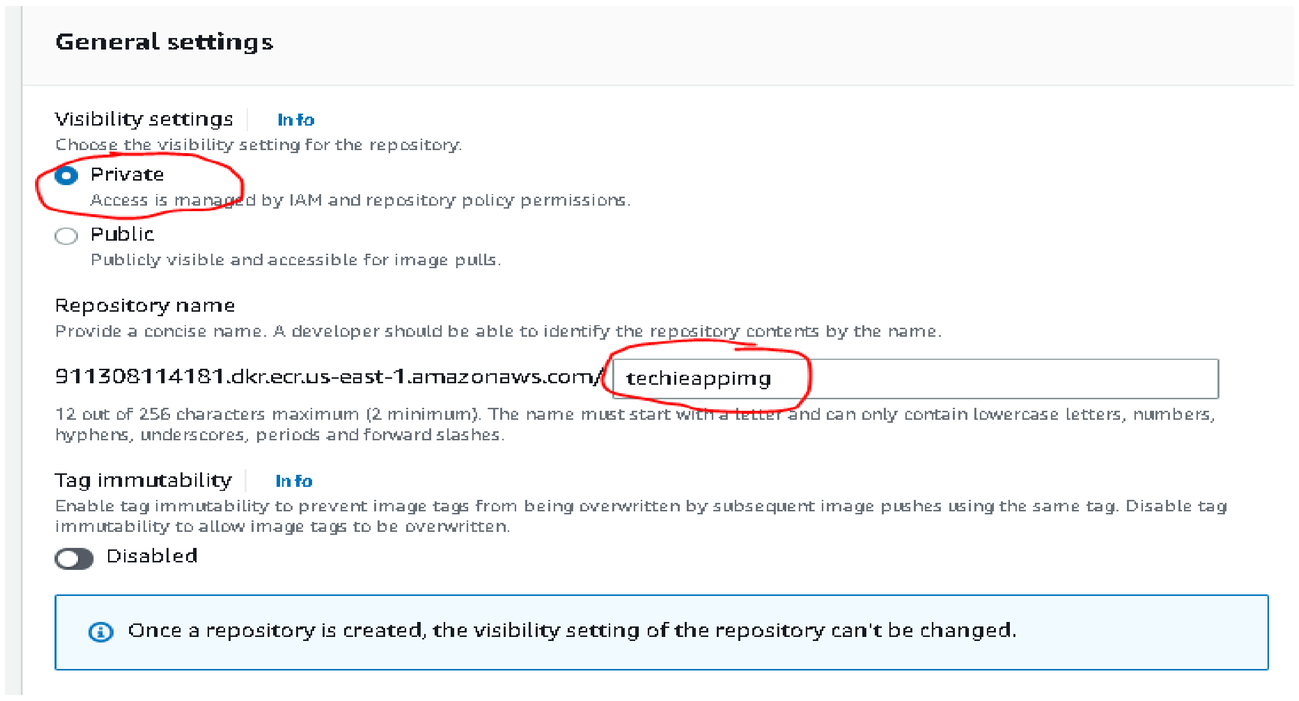
ECR SETUP:-

Now, Goto **AWS ECR**(Elastic Container Resgistry)

AWS > ECR > Get Started

Keep it private

Repo Name : techieappimg



General settings

Visibility settings | Info

Choose the visibility setting for the repository.

[✓ Private] Access is managed by IAM and repository policy permissions.

Public Publicly visible and accessible for image pulls.

Repository name

Provide a concise name. A developer should be able to identify the repository contents by the name.

911308114181.dkr.ecr.us-east-1.amazonaws.com/ techieappimg

12 out of 256 characters maximum (2 minimum). The name must start with a letter and can only contain lowercase letters, numbers, hyphens, underscores, periods and forward slashes.

Tag immutability | Info

Enable tag immutability to prevent image tags from being overwritten by subsequent image pushes using the same tag. Disable tag immutability to allow image tags to be overwritten.

Disabled

Once a repository is created, the visibility setting of the repository can't be changed.

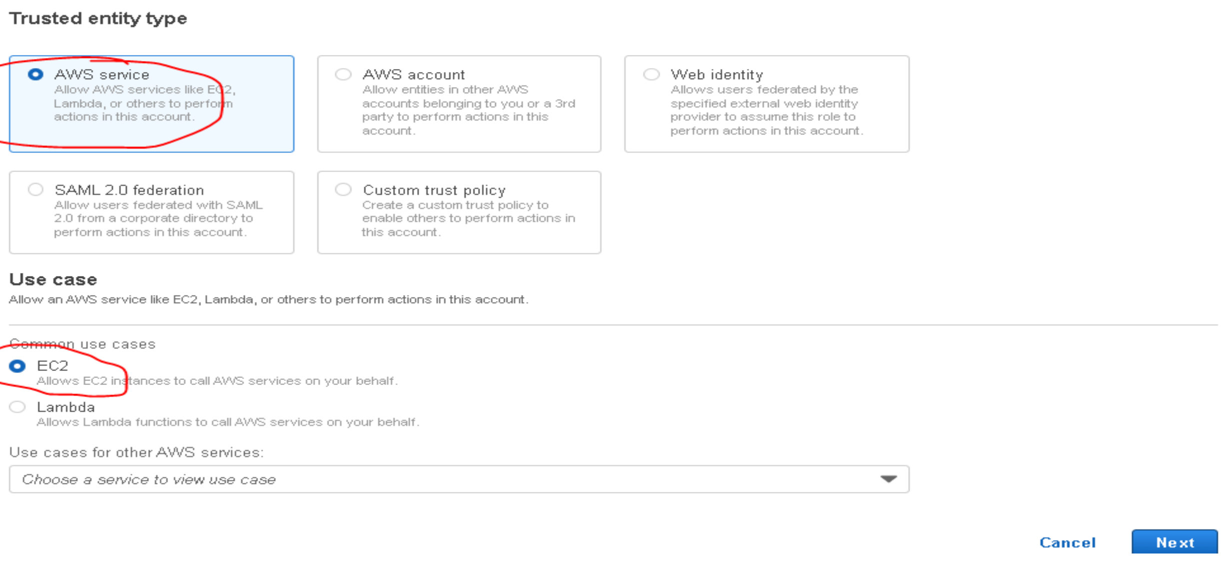
Click on **Create Repository**

IAM Role:-

Now, Go to AWS Console > IAM > Roles > Create Role

Select AWS Service

Select EC2



Trusted entity type

[✓ AWS service] Allow AWS services like EC2, Lambda, or others to perform actions in this account.

AWS account: Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.

Web identity: Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.

SAML 2.0 federation: Allow users federated with SAML 2.0 from a corporate directory to perform actions in this account.

Custom trust policy: Create a custom trust policy to enable others to perform actions in this account.

Use case

Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

Common use cases

[✓ EC2] Allows EC2 instances to call AWS services on your behalf.

Lambda: Allows Lambda functions to call AWS services on your behalf.

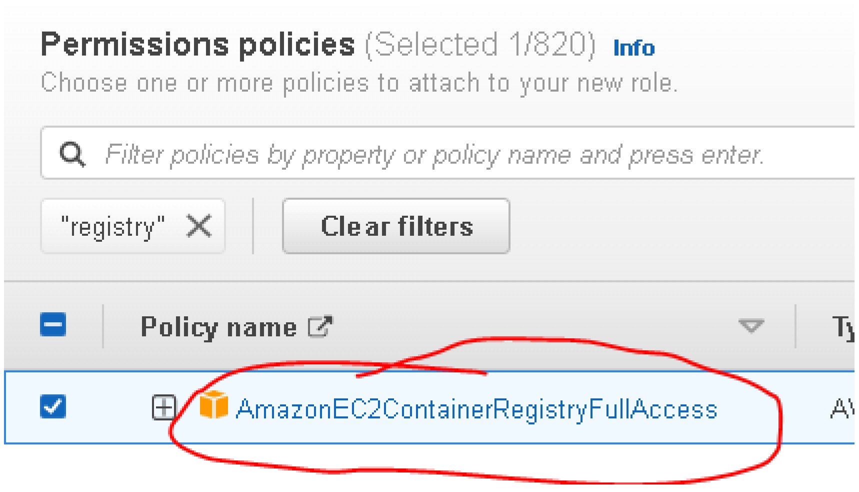
Use cases for other AWS services:

Choose a service to view use case

Click **Next**

Now add permissions:-

AmazonEC2ContainerRegistryFullAccess



Permissions policies (Selected 1/820) Info

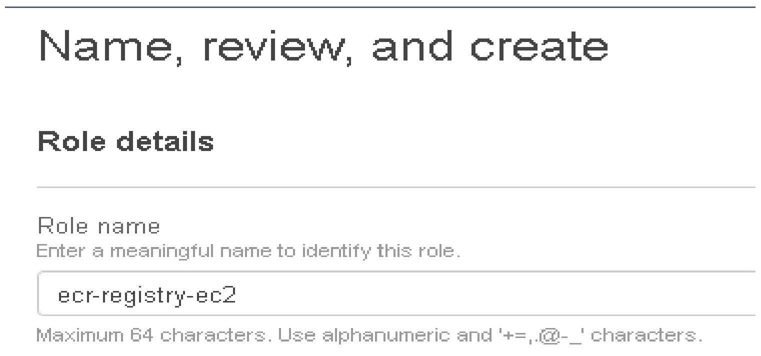
Choose one or more policies to attach to your new role.

Filter policies by property or policy name and press enter.

"registry"

[✓ AmazonEC2ContainerRegistryFullAccess]

Name : ecr-registry-ec2


Name, review, and create

Role details

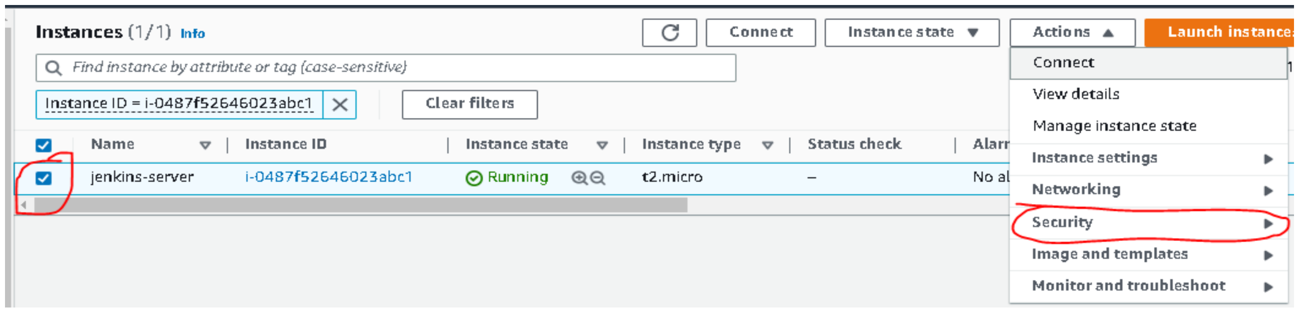
Role name
Enter a meaningful name to identify this role.

ecr-registry-ec2

Maximum 64 characters. Use alphanumeric and '+=,.@-_' characters.

Click **Create Role**

Now, Go to EC2 > select our Jenkins server > Actions > Security > Modify IAM Role

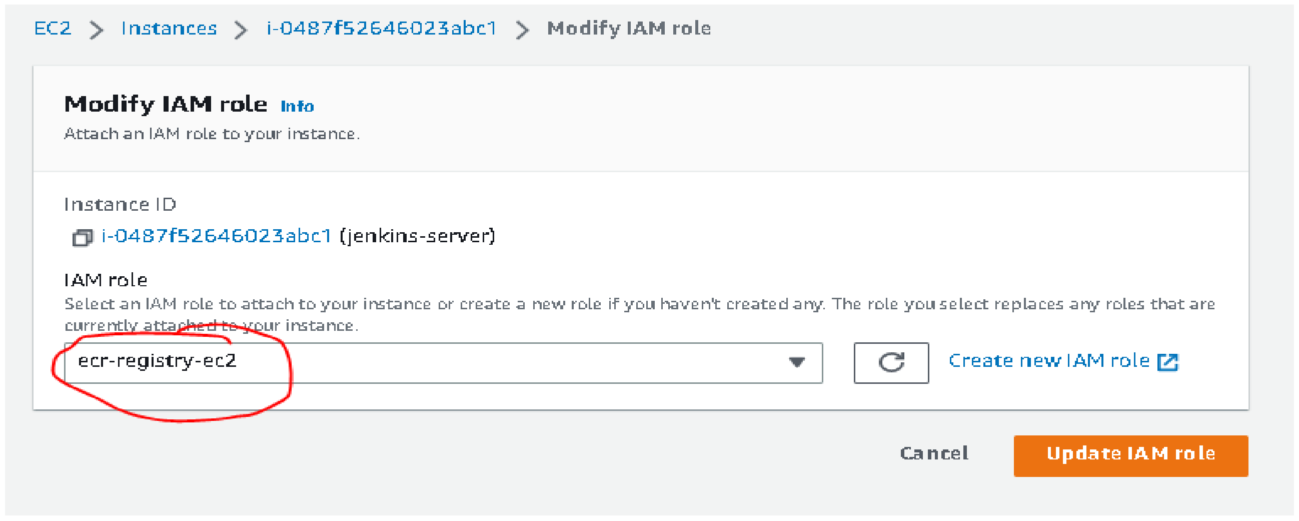

Instances (1/1) Info

Find instance by attribute or tag (case-sensitive)

Instance ID = i-0487f52646023abc1

Name	Instance ID	Instance state	Instance type	Status check	Alarr
jenkins-server	i-0487f52646023abc1	Running	t2.micro	–	No al

Select our role which we created


EC2 > Instances > i-0487f52646023abc1 > Modify IAM role

Modify IAM role Info

Attach an IAM role to your instance.

Instance ID

i-0487f52646023abc1 (jenkins-server)

IAM role

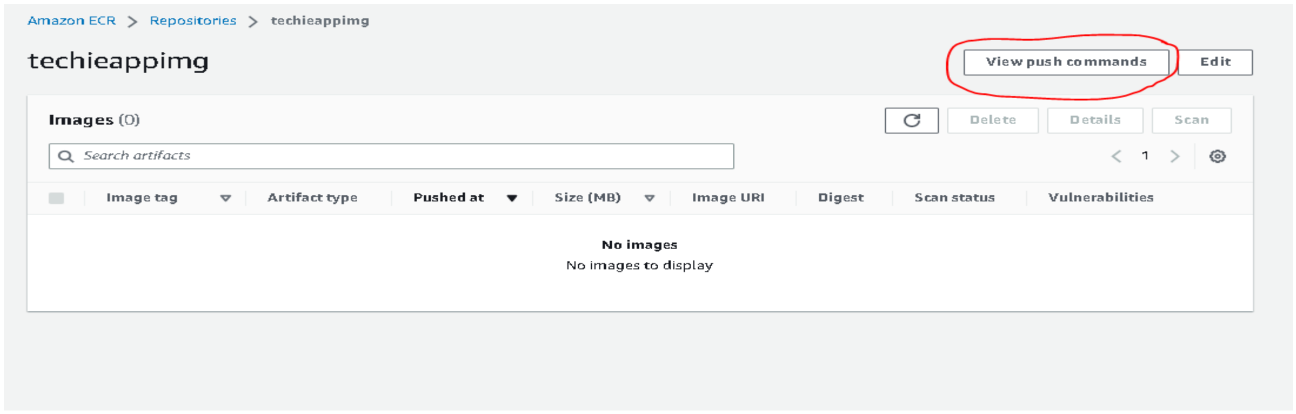
Select an IAM role to attach to your instance or create a new role if you haven't created any. The role you select replaces any roles that are currently attached to your instance.

ecr-registry-ec2

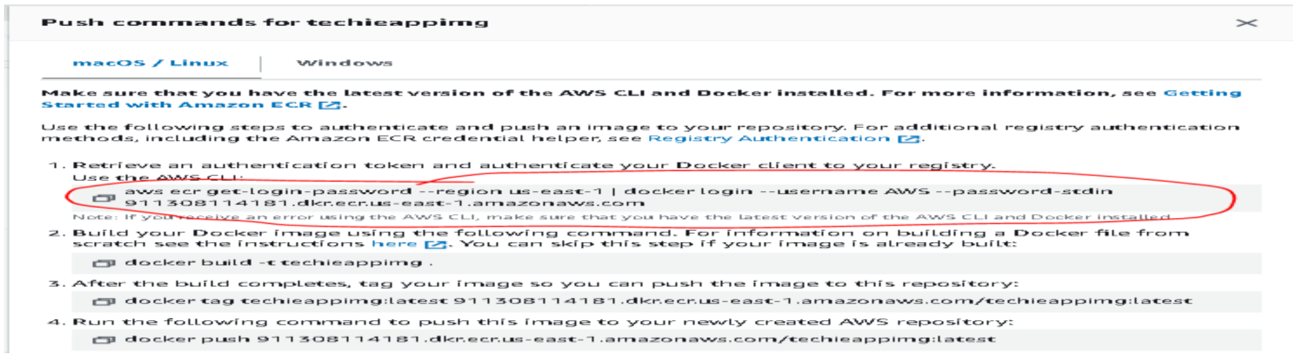
Click **Update IAM Role**

Now, go to **ECR** >

Click on **techieappimg** and then click on **View push commands**



Copy the 1st Command & paste it in Jenkins serverwith the sudo privilege to login



Push commands for techieappimg

macOS / Linux | Windows

Make sure that you have the latest version of the AWS CLI and Docker installed. For more information, see Getting Started with Amazon ECR.

Use the following steps to authenticate and push an image to your repository. For additional registry authentication methods, including the Amazon ECR credential helper, see Registry Authentication.

1. Retrieve an authentication token and authenticate your Docker client to your registry.
   Use the AWS CLI:

```
aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin 911308114181.dkr.ecr.us-east-1.amazonaws.com
```

   Note: If you receive an error using the AWS CLI, make sure that you have the latest version of the AWS CLI and Docker installed.

2. Build your Docker image using the following command. For information on building a Docker file from scratch see the instructions here. You can skip this step if your image is already built:

```
docker build -t techieappimg .
```

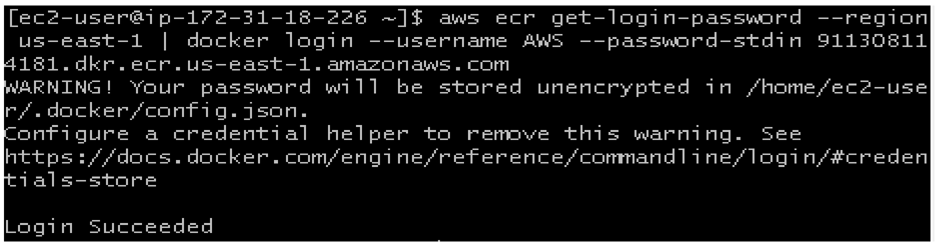
3. After the build completes, tag your image so you can push the image to this repository:

```
docker tag techieappimg:latest 911308114181.dkr.ecr.us-east-1.amazonaws.com/techieappimg:latest
```

4. Run the following command to push this image to your newly created AWS repository:

```
docker push 911308114181.dkr.ecr.us-east-1.amazonaws.com/techieappimg:latest
```

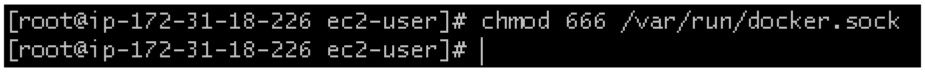
In ssh:-



```
[ec2-user@ip-172-31-18-226 ~]$ aws ecr get-login-password --region
 us-east-1 | docker login --username AWS --password-stdin 91130811
4181.dkr.ecr.us-east-1.amazonaws.com
WARNING! Your password will be stored unencrypted in /home/ec2-use
r/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#creden
tials-store

Login Succeeded
```

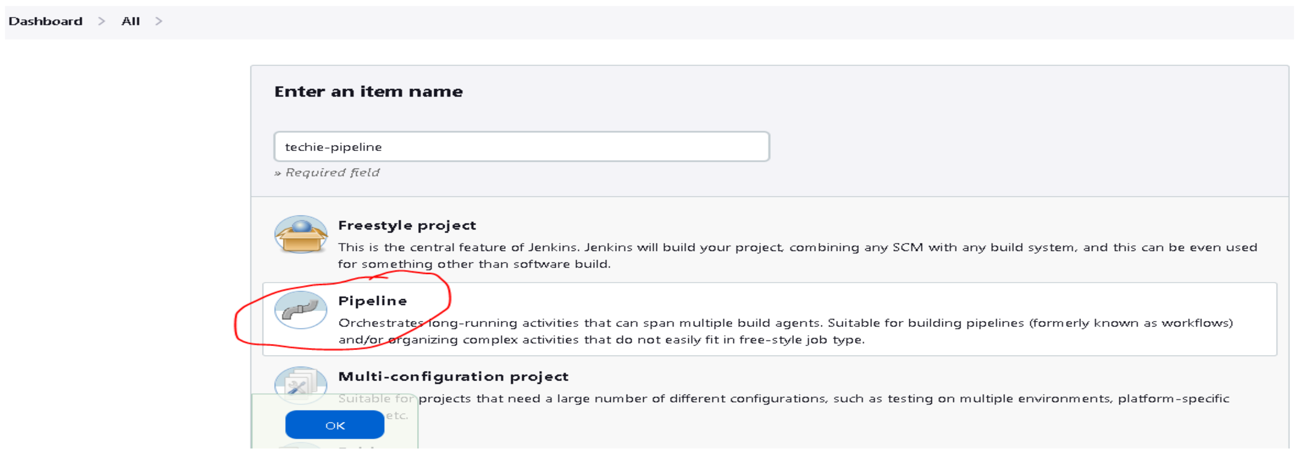
Give permission to docker sock :-

chmod 666 /var/run/docker.sock



```
[root@ip-172-31-18-226 ec2-user]# chmod 666 /var/run/docker.sock
[root@ip-172-31-18-226 ec2-user]# |
```

Now go to Jenkins Dashboard > New Item >



Dashboard > All >

Enter an item name

techie-pipeline

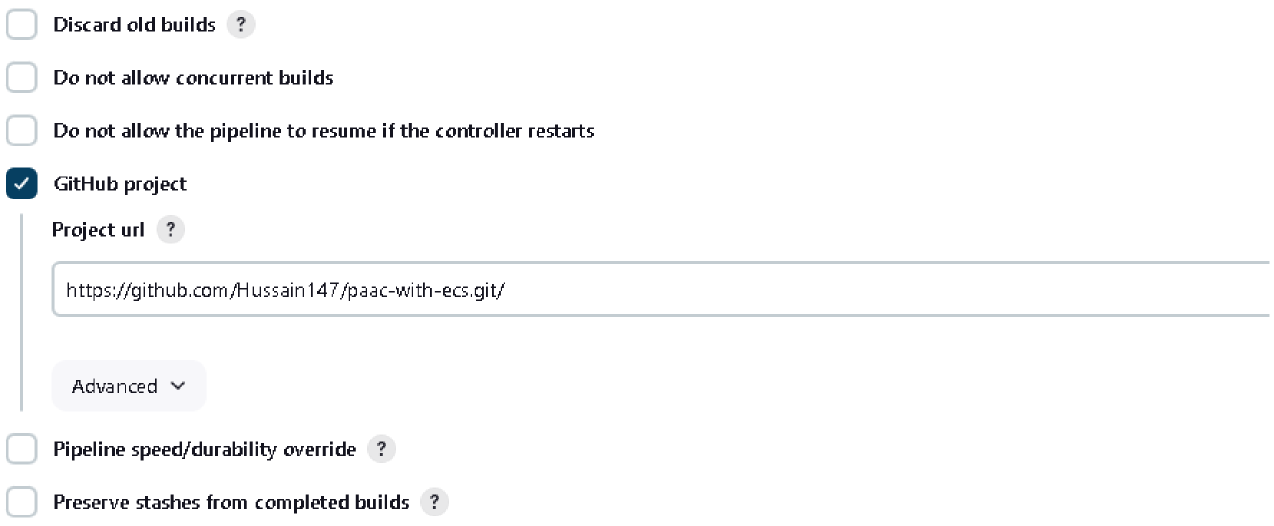
» Required field

Freestyle project
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific etc.

Give Github URL in Github Project



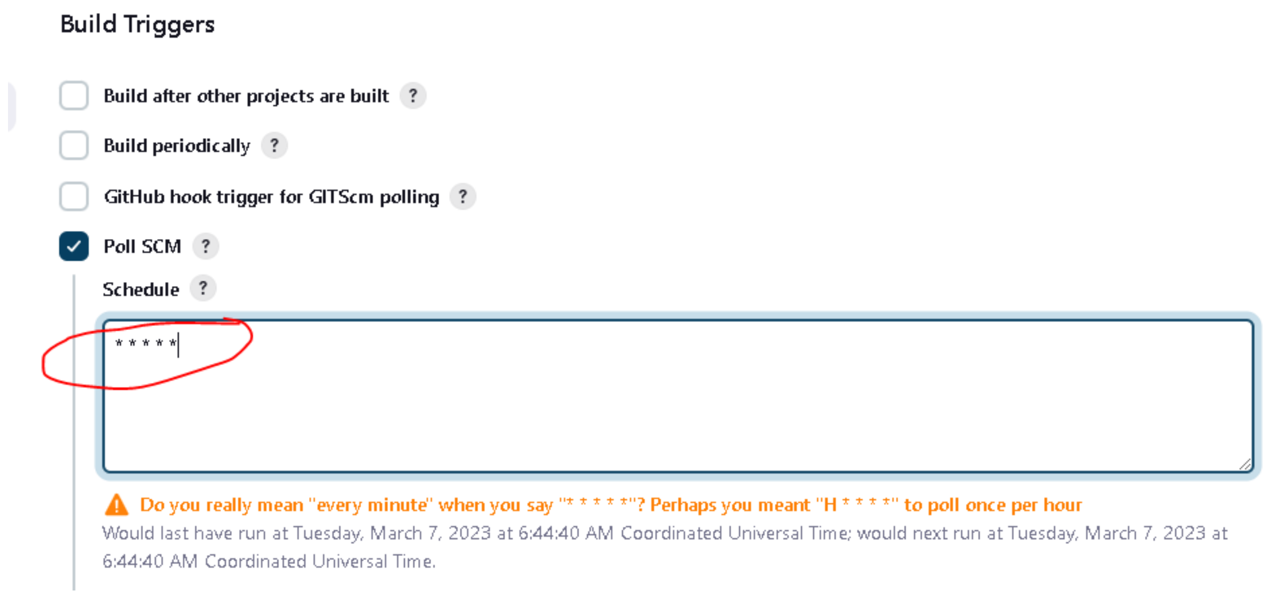
- [ ] Discard old builds ?
- [ ] Do not allow concurrent builds
- [ ] Do not allow the pipeline to resume if the controller restarts
- [x] GitHub project

Project url ?

https://github.com/Hussain147/paac-with-ecs.git/

- [ ] Pipeline speed/durability override ?
- [ ] Preserve stashes from completed builds ?

Give Poll SCM : * * * * * (which means, we're telling Jenkins to check every minute whether any changes are made in the github repository)



Build Triggers

- [ ] Build after other projects are built
- [ ] Build periodically
- [ ] GitHub hook trigger for GITScm polling
- [✓] Poll SCM

Schedule

```
* * * * *
```

⚠ Do you really mean "every minute" when you say "* * * * *"? Perhaps you meant "H * * * *" to poll once per hour

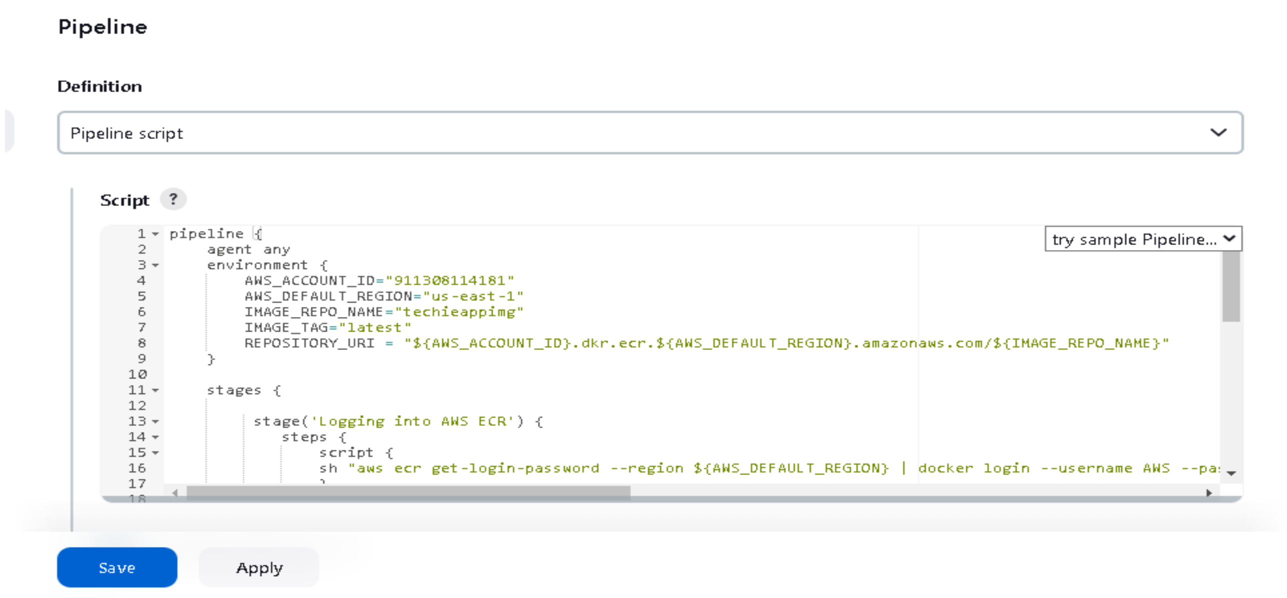
Would last have run at Tuesday, March 7, 2023 at 6:44:40 AM Coordinated Universal Time; would next run at Tuesday, March 7, 2023 at 6:44:40 AM Coordinated Universal Time.

Paste the script from the link given and make some changes in the environment like Account id, Region, Image Repo Name.

Link:- https://github.com/Hussain147/paac-with-ecs/blob/main/PAAC_CI_Docker_ECR_ECS_with_Slack.txt

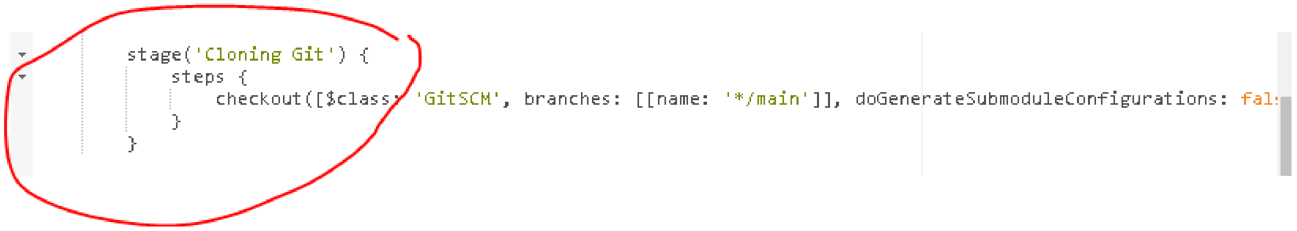
Without Slack :-

https://github.com/Hussain147/paac-with-ecs/blob/main/PAAC_CI_Docker_ECR_ECS.txt



```
pipeline {
    agent any
    environment {
        AWS_ACCOUNT_ID="911308114181"
        AWS_DEFAULT_REGION="us-east-1"
        IMAGE_REPO_NAME="techieappimg"
        IMAGE_TAG="latest"
        REPOSITORY_URI = "${AWS_ACCOUNT_ID}.dkr.ecr.${AWS_DEFAULT_REGION}.amazonaws.com/${IMAGE_REPO_NAME}"
    }

    stages {

        stage('Logging into AWS ECR') {
            steps {
                script {
                sh "aws ecr get-login-password --region ${AWS_DEFAULT_REGION} | docker login --username AWS --pa:
```

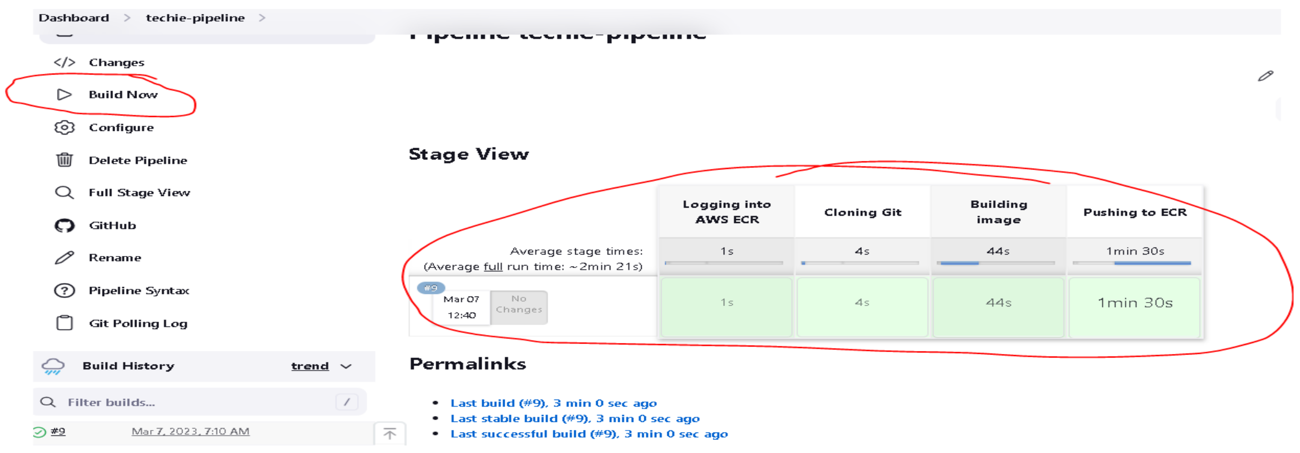
In cloning git stage, check the url of the github repo.



```
stage('Cloning Git') {
    steps {
        checkout([$class: 'GitSCM', branches: [[name: '*/main']], doGenerateSubmoduleConfigurations: fal:
    }
}
```

Then Click on **Save**

Now, Click **Build Now**

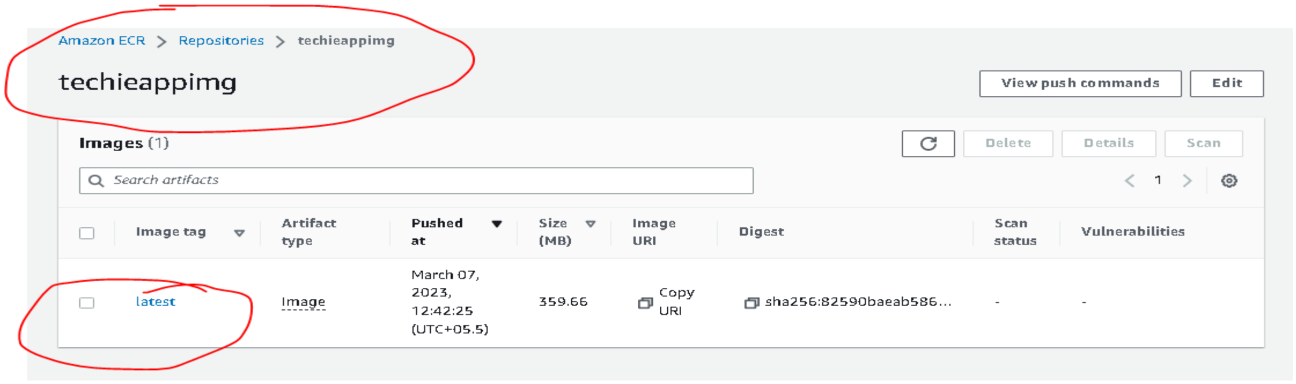


	Logging into AWS ECR	Cloning Git	Building image	Pushing to ECR
Average stage times: (Average full run time: ~2min 21s)	1s	4s	44s	1min 30s
#9 Mar 07 12:40 No Changes	1s	4s	44s	1min 30s

- Last build (#9), 3 min 0 sec ago
- Last stable build (#9), 3 min 0 sec ago
- Last successful build (#9), 3 min 0 sec ago

See Our Pipeline is Successfully Completed upto ECR

Now Goto ECR > select our registry > You will see the Image that we build by using docker and pushed to ECR



Amazon ECR > Repositories > techieappimg

techieappimg

View push commands | Edit

Images (1)

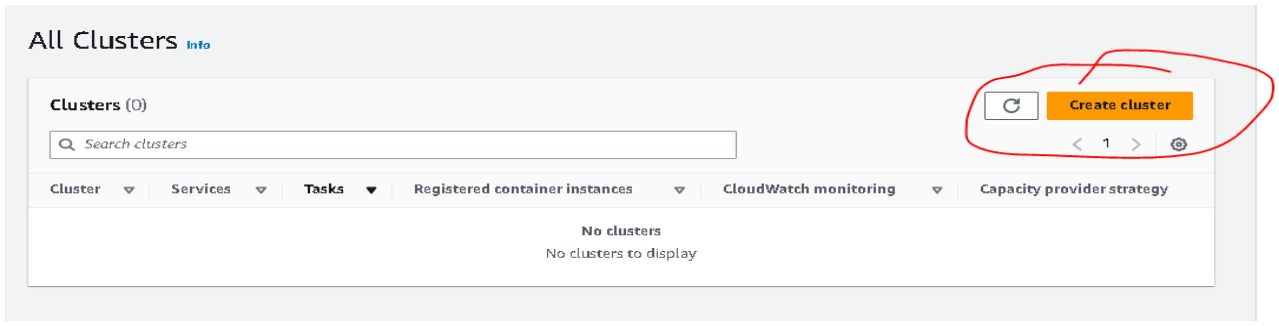
Delete | Details | Scan

Search artifacts

Image tag	Artifact type	Pushed at	Size (MB)	Image URI	Digest	Scan status	Vulnerabilities
latest	Image	March 07, 2023, 12:42:25 (UTC+05.5)	359.66	Copy URI	sha256:82590baeab586...	-	-

4. <u>AWS ECS Setup</u>:-

Now goto AWS > ECS > Get Started > Click Create Cluster



All Clusters Info

Clusters (0)

Create cluster

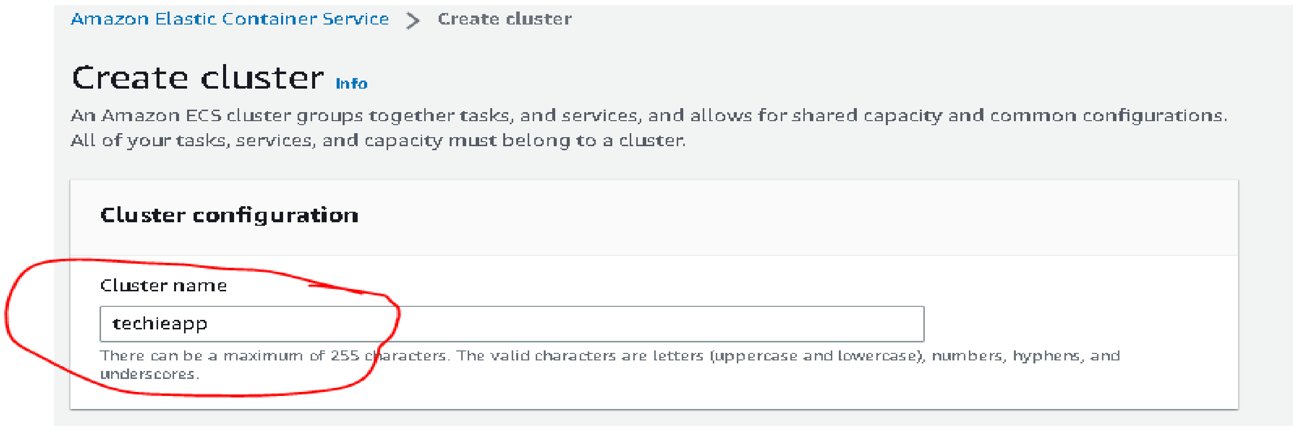
Search clusters

Cluster	Services	Tasks	Registered container instances	CloudWatch monitoring	Capacity provider strategy

No clusters

No clusters to display

Give a Name : techieapp



Amazon Elastic Container Service > Create cluster

Create cluster Info

An Amazon ECS cluster groups together tasks, and services, and allows for shared capacity and common configurations. All of your tasks, services, and capacity must belong to a cluster.

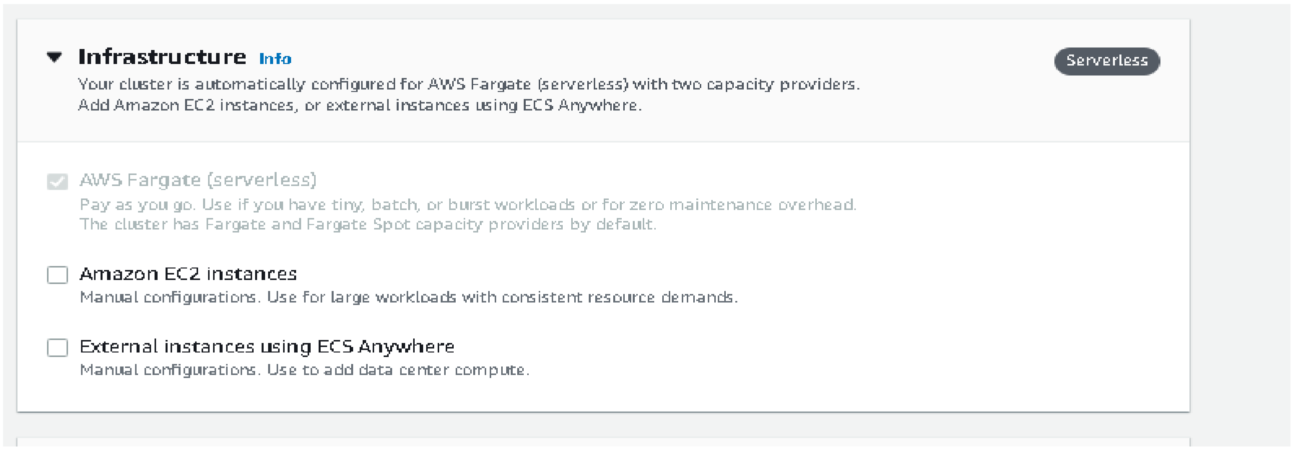
Cluster configuration

Cluster name

techieapp

There can be a maximum of 255 characters. The valid characters are letters (uppercase and lowercase), numbers, hyphens, and underscores.

Keep Infrastucture as default : AWS Fargate



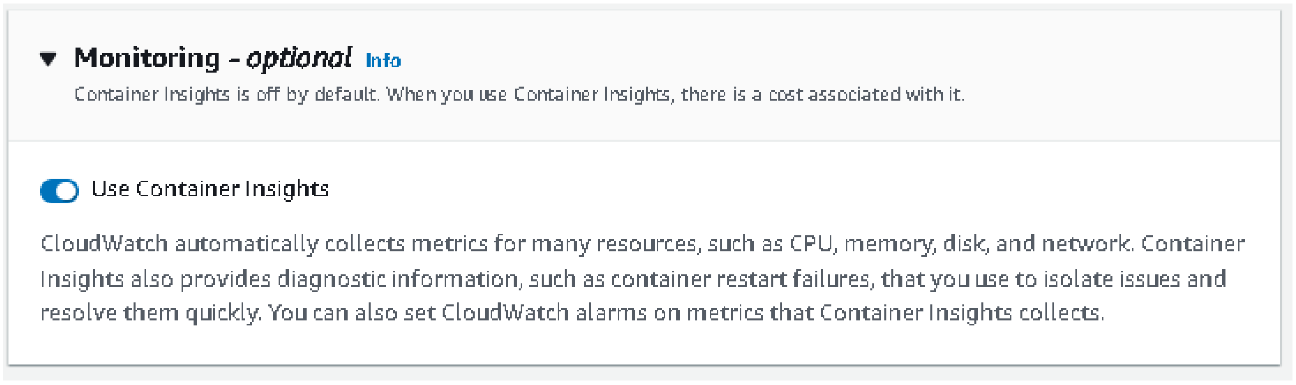
Infrastructure Info

Serverless

Your cluster is automatically configured for AWS Fargate (serverless) with two capacity providers. Add Amazon EC2 instances, or external instances using ECS Anywhere.

- [✓] AWS Fargate (serverless)
  Pay as you go. Use if you have tiny, batch, or burst workloads or for zero maintenance overhead. The cluster has Fargate and Fargate Spot capacity providers by default.
- [ ] Amazon EC2 instances
  Manual configurations. Use for large workloads with consistent resource demands.
- [ ] External instances using ECS Anywhere
  Manual configurations. Use to add data center compute.

Use Container Insights : enable



Monitoring - *optional* Info

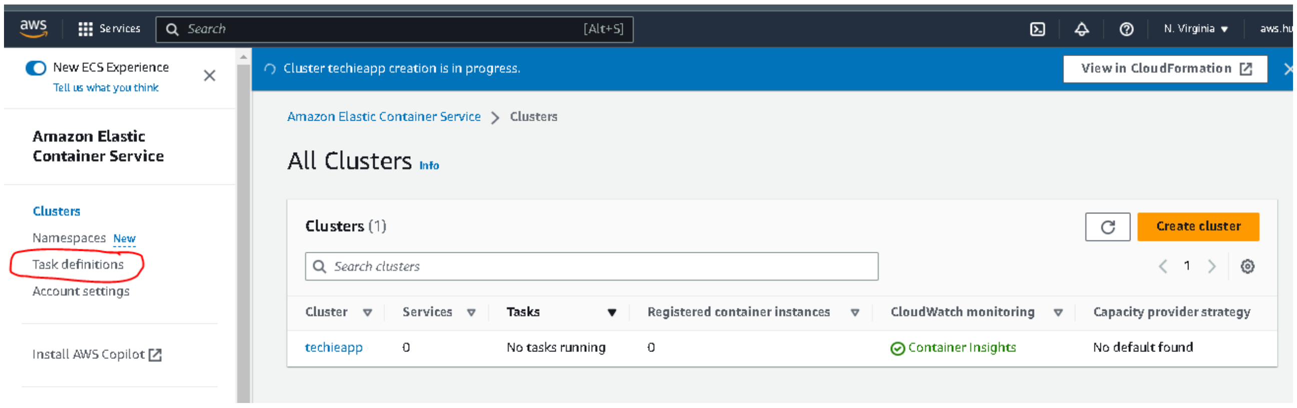
Container Insights is off by default. When you use Container Insights, there is a cost associated with it.

[✓] Use Container Insights

CloudWatch automatically collects metrics for many resources, such as CPU, memory, disk, and network. Container Insights also provides diagnostic information, such as container restart failures, that you use to isolate issues and resolve them quickly. You can also set CloudWatch alarms on metrics that Container Insights collects.

Click **Create**

Now, go to Task Definitions



Cluster techieapp creation is in progress.

View in CloudFormation

Amazon Elastic Container Service > Clusters

All Clusters Info

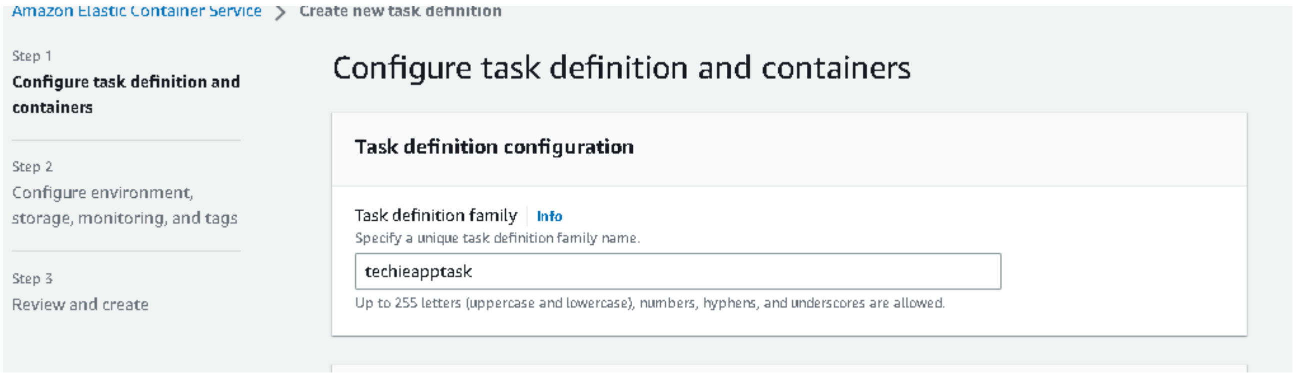
Clusters (1)

Create cluster

Search clusters

Cluster	Services	Tasks	Registered container instances	CloudWatch monitoring	Capacity provider strategy
techieapp	0	No tasks running	0	Container Insights	No default found

Give a Name : techieapptask


Amazon Elastic Container Service > Create new task definition

Step 1
Configure task definition and containers

Step 2
Configure environment, storage, monitoring, and tags

Step 3
Review and create

Configure task definition and containers

Task definition configuration

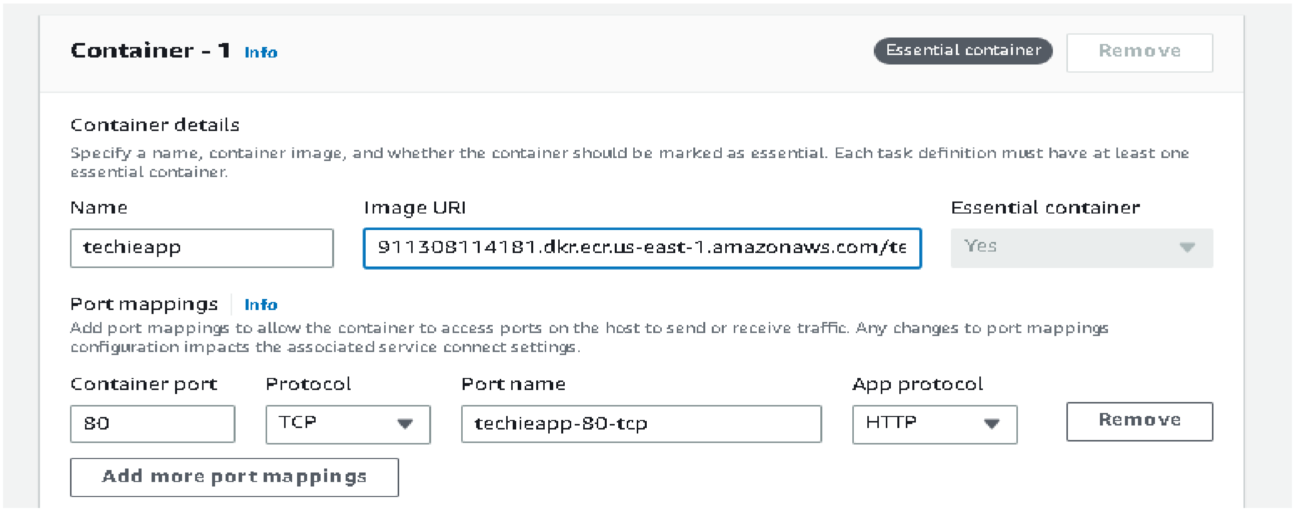
Task definition family Info
Specify a unique task definition family name.
techieapptask
Up to 255 letters (uppercase and lowercase), numbers, hyphens, and underscores are allowed.

Container Details:-

Name : techieapp

Image URI : copy the **techieapp** ECR repo & paste here

Port : 80


Container - 1 Info

Essential container | Remove

Container details
Specify a name, container image, and whether the container should be marked as essential. Each task definition must have at least one essential container.

Name	Image URI	Essential container
techieapp	911308114181.dkr.ecr.us-east-1.amazonaws.com/te	Yes

Port mappings Info
Add port mappings to allow the container to access ports on the host to send or receive traffic. Any changes to port mappings configuration impacts the associated service connect settings.

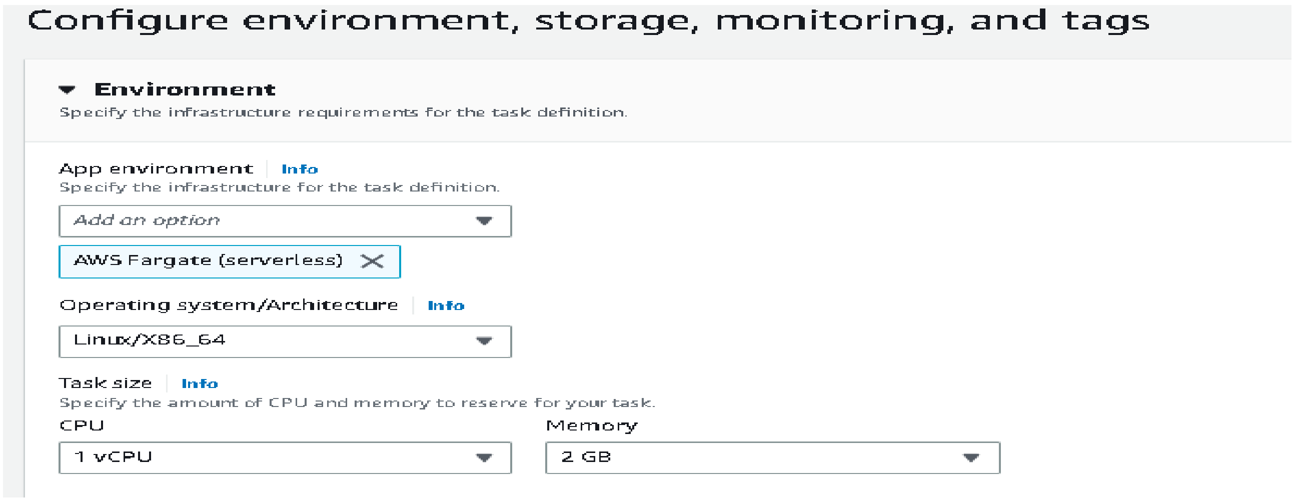
Container port	Protocol	Port name	App protocol
80	TCP	techieapp-80-tcp	HTTP

Remove

Add more port mappings

Click **Next**

Check these details > App Environment


Configure environment, storage, monitoring, and tags

Environment
Specify the infrastructure requirements for the task definition.

App environment Info
Specify the infrastructure for the task definition.
Add an option
AWS Fargate (serverless)

Operating system/Architecture Info
Linux/X86_64

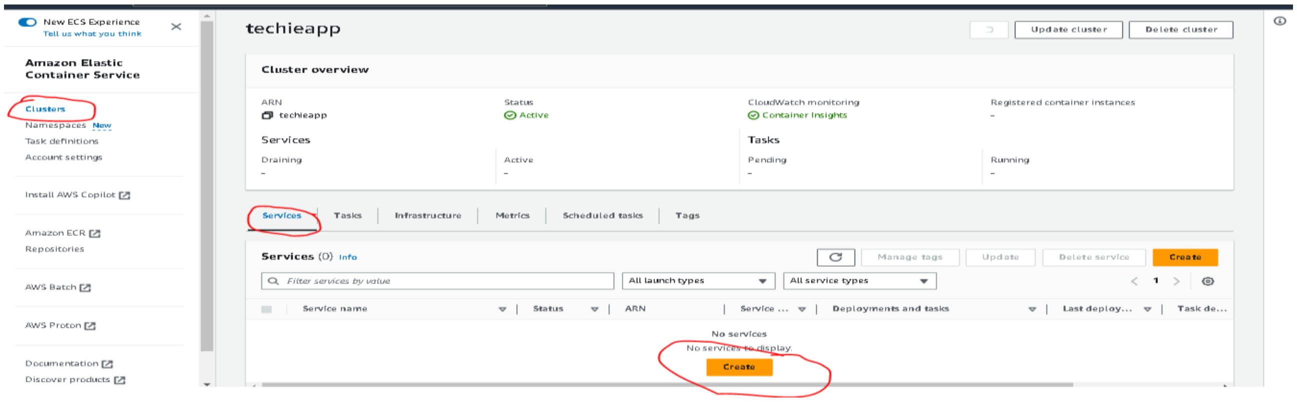
Task size Info
Specify the amount of CPU and memory to reserve for your task.

CPU	Memory
1 vCPU	2 GB

Click **Next** > Click **Create**

Now, we need to combine the task with our cluster.

Goto Clusters > Select our cluster **techieapp >** service **>** Click **Create**



techieapp

Cluster overview

ARN	Status	CloudWatch monitoring	Registered container instances
techieapp	Active	Container Insights	-

Services

Draining	Active
-	-

Tasks

Pending	Running
-	-

Services (0) Info

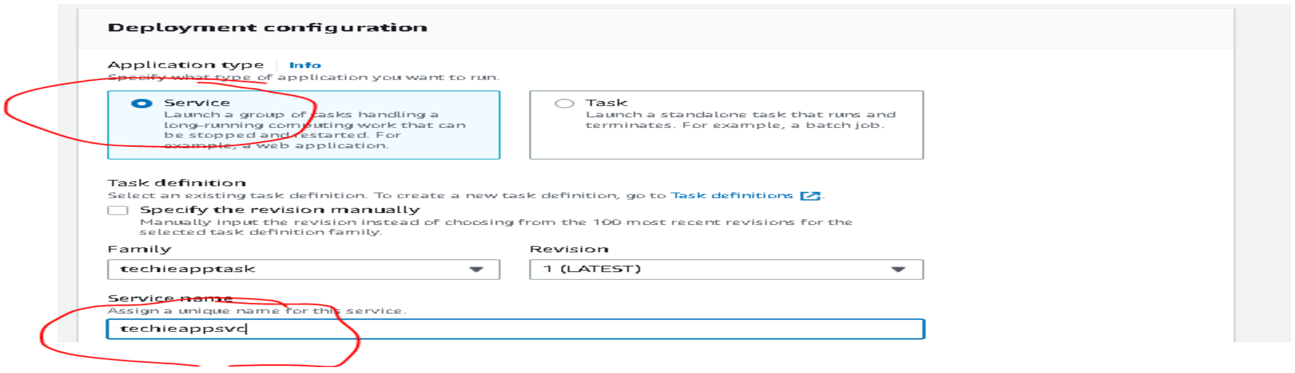
Filter services by value | All launch types | All service types

Service name	Status	ARN	Service ...	Deployments and tasks	Last deploy...	Task de...

No services
No services to display.

Application Type : Service

Service Name : techieappsvc



Deployment configuration

Application type | Info
Specify what type of application you want to run.

(●) Service
Launch a group of tasks handling a long-running computing work that can be stopped and restarted. For example, a web application.

( ) Task
Launch a standalone task that runs and terminates. For example, a batch job.

Task definition
Select an existing task definition. To create a new task definition, go to Task definitions.

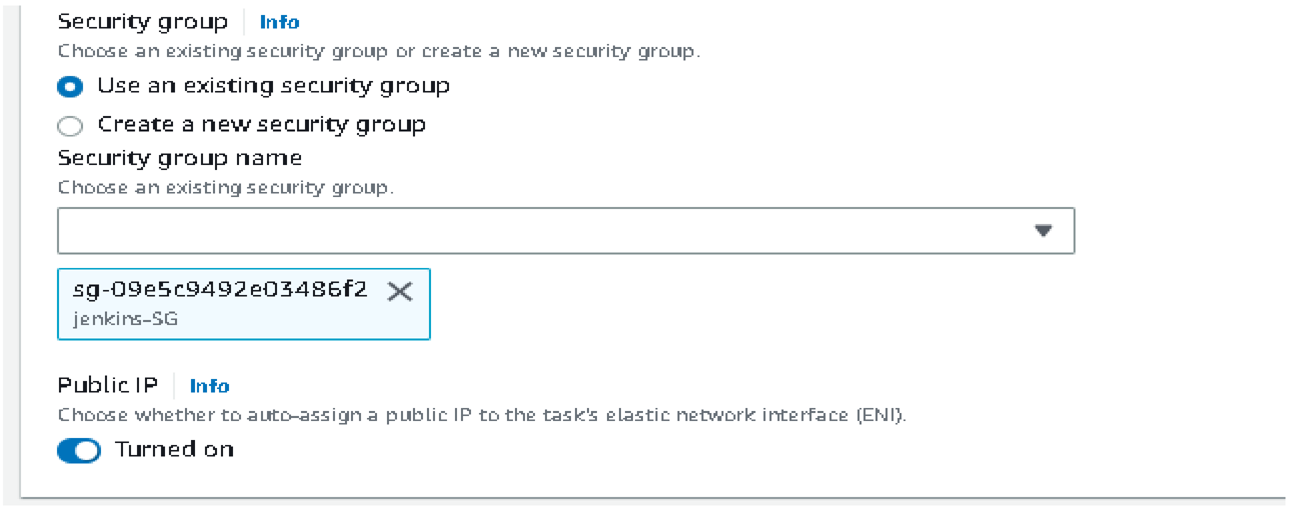
☐ Specify the revision manually
Manually input the revision instead of choosing from the 100 most recent revisions for the selected task definition family.

Family: techieapptask

Revision: 1 (LATEST)

Service name
Assign a unique name for this service.

techieappsvc



Security group | Info
Choose an existing security group or create a new security group.

(●) Use an existing security group
( ) Create a new security group

Security group name
Choose an existing security group.

sg-09e5c9492e03486f2
jenkins-SG

Public IP | Info
Choose whether to auto-assign a public IP to the task's elastic network interface (ENI).

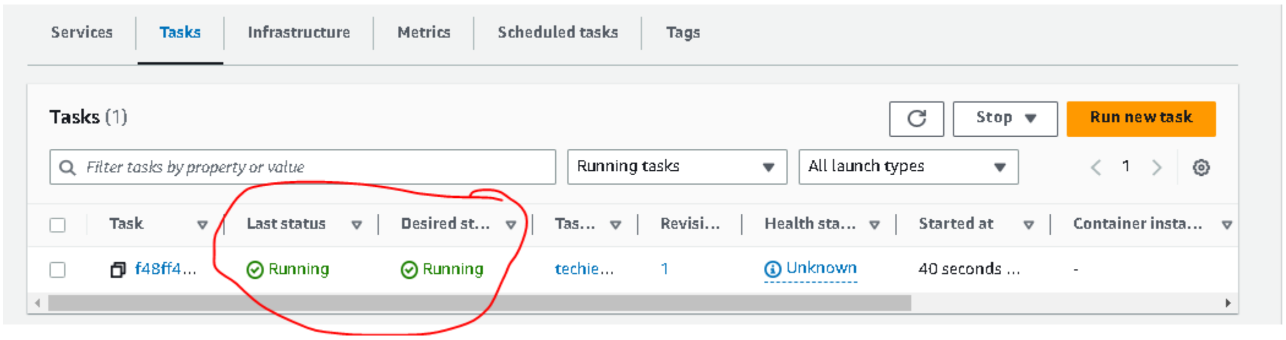
Turned on

Click **Create**

Now Keep Wait…

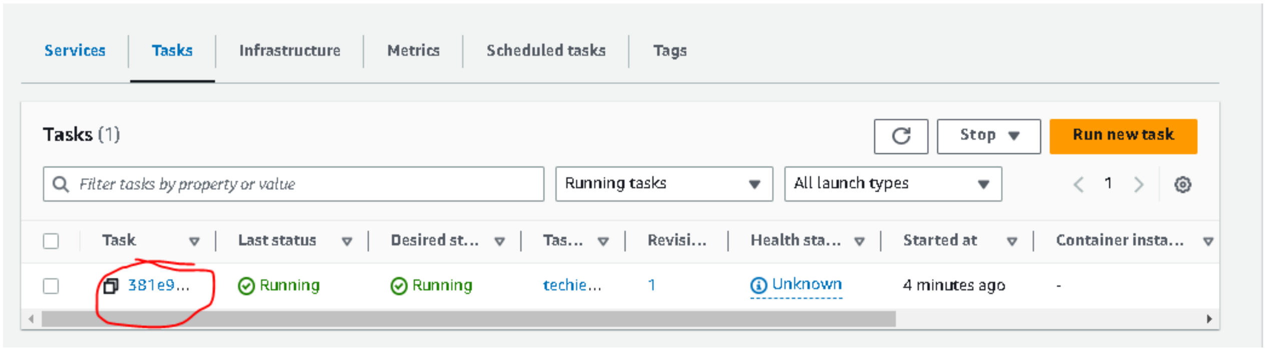
It will take some time to deploy

Check the status after sometime >



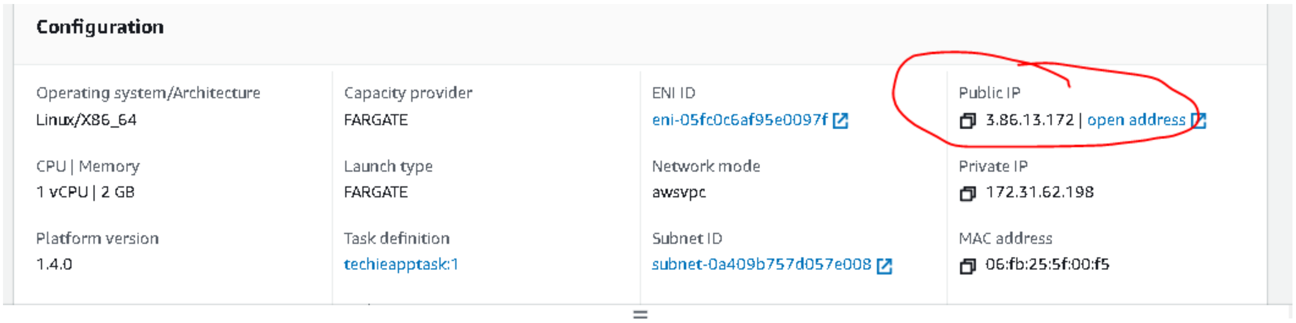
Task	Last status	Desired st...	Tas...	Revisi...	Health sta...	Started at	Container insta...
f48ff4...	Running	Running	techie...	1	Unknown	40 seconds ...	-

Now, Select Task



Task	Last status	Desired st...	Tas...	Revisi...	Health sta...	Started at	Container insta...
381e9...	Running	Running	techie...	1	Unknown	4 minutes ago	-

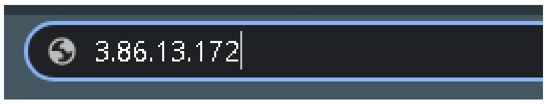
Scroll Down, Copy the Public Ip



Configuration

Operating system/Architecture Linux/X86_64	Capacity provider FARGATE	ENI ID eni-05fc0c6af95e0097f	Public IP 3.86.13.172 | open address
CPU | Memory 1 vCPU | 2 GB	Launch type FARGATE	Network mode awsvpc	Private IP 172.31.62.198
Platform version 1.4.0	Task definition techieapptask:1	Subnet ID subnet-0a409b757d057e008	MAC address 06:fb:25:5f:00:f5

Paste in the URL bar > Press Enter



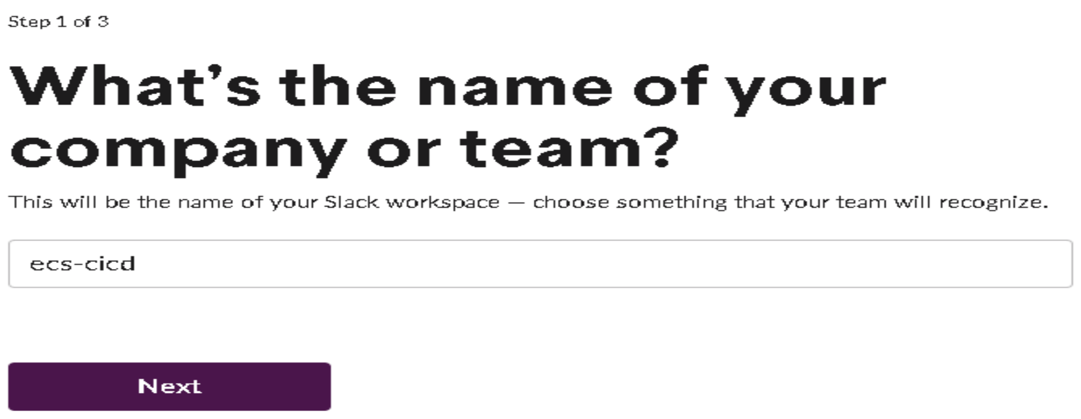
3.86.13.172

Congratulations…You have deployed the application successfully with ECS

If you push any changes to the Github, then CICD process will get starts.

If you push the code to the github, then our Jenkins server will fetch the code & the docker will build the code & push to the ECR & then from ECR to ECS…

~~~THANK YOU~~~

If You want the Notification to slack, Then Add these…

1. <u>Slack Setup</u>:-

Slack setup is used to get the notification whether our build job pass or fail.

Go to **Slack** in google or in application on your local machine **> Sign Up** with your gmail account.

Give a **Workspace Name** : ecs-cicd
~~~


Step 1 of 3

What's the name of your company or team?

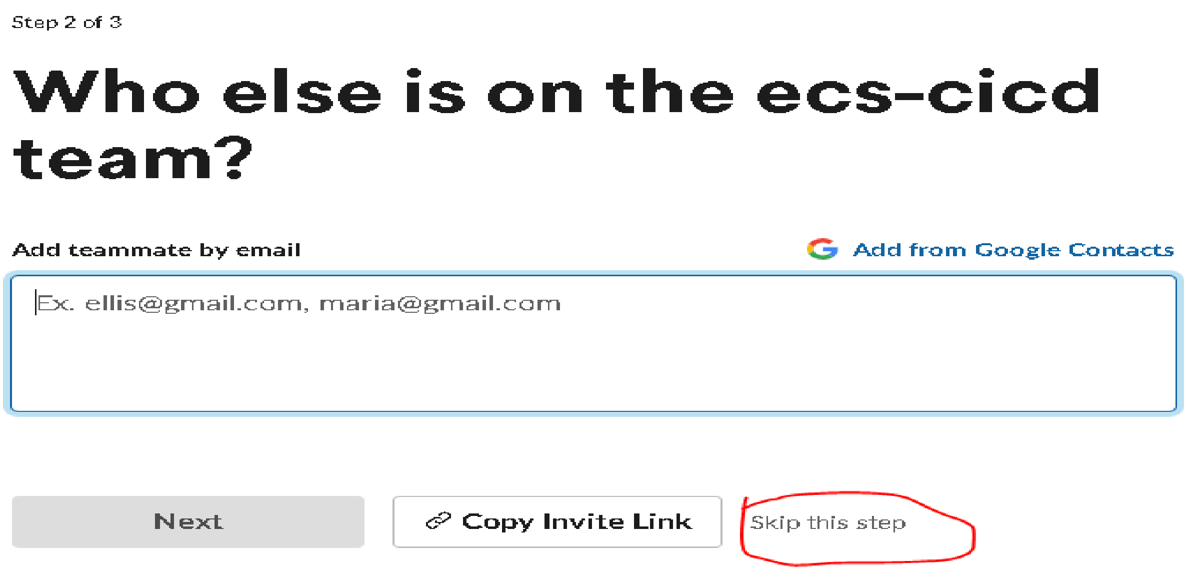
This will be the name of your Slack workspace — choose something that your team will recognize.

ecs-cicd

Next

Click Next

Add Teammates mail id's or skip this step (I'm skipping this step for now).


Step 2 of 3

Who else is on the ecs-cicd team?

Add teammate by email

Add from Google Contacts

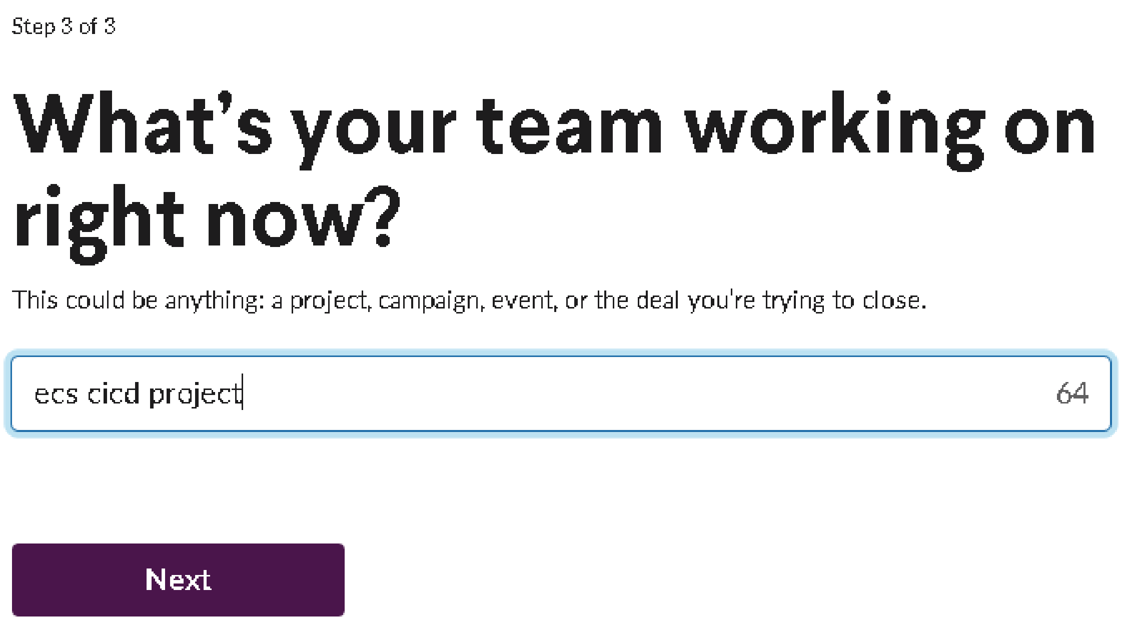
Ex. ellis@gmail.com, maria@gmail.com

Next

Copy Invite Link

Skip this step

Give a **channel name** : ecs-cicd-project



Step 3 of 3

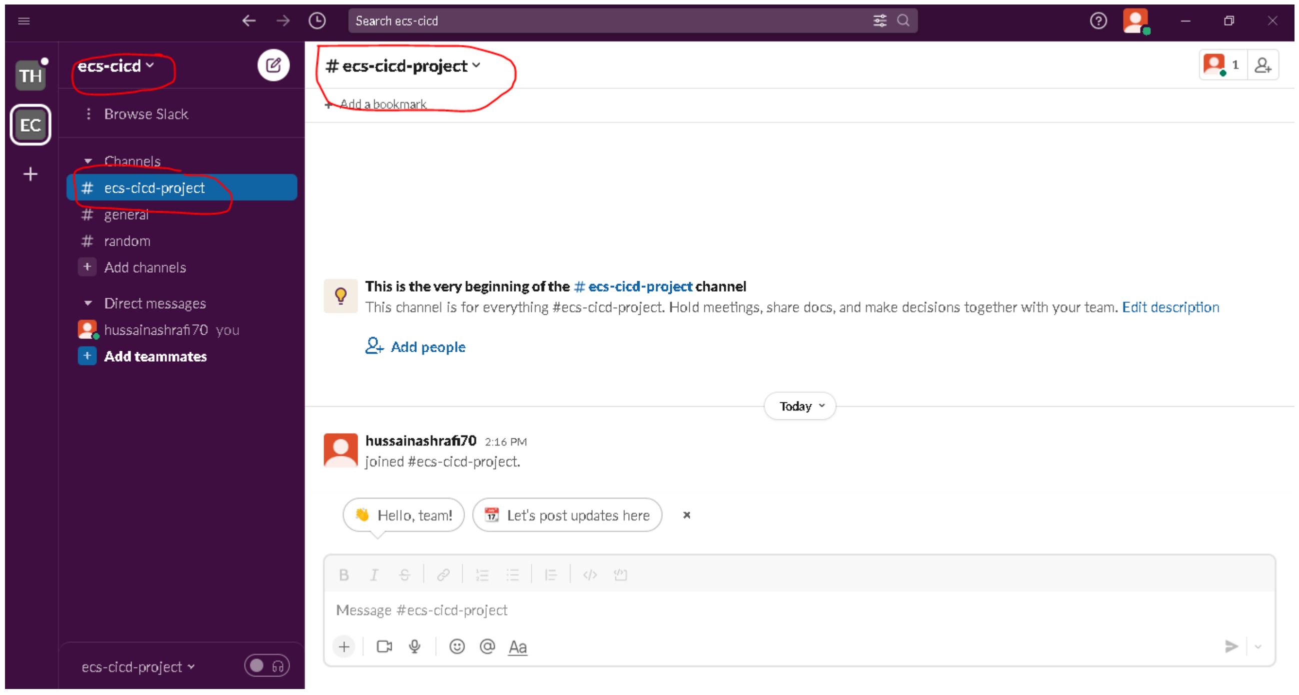
What's your team working on right now?

This could be anything: a project, campaign, event, or the deal you're trying to close.

ecs cicd project 64

Click **Next**

You will see the Dashboard like below:-



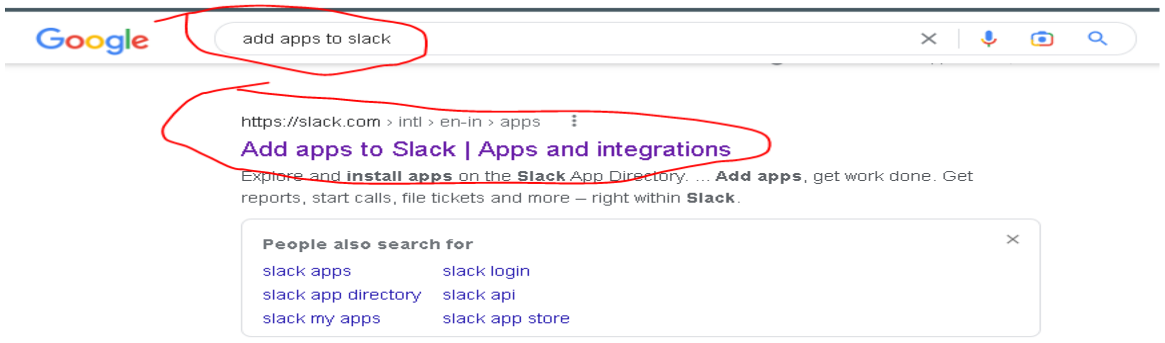
This is the very beginning of the # ecs-cicd-project channel

This channel is for everything #ecs-cicd-project. Hold meetings, share docs, and make decisions together with your team. Edit description

hussainashrafi70 2:16 PM

joined #ecs-cicd-project.

Now, go to Google > Type add apps to slack > Click on Adds apps to slack link | Apps and integrations



https://slack.com › intl › en-in › apps

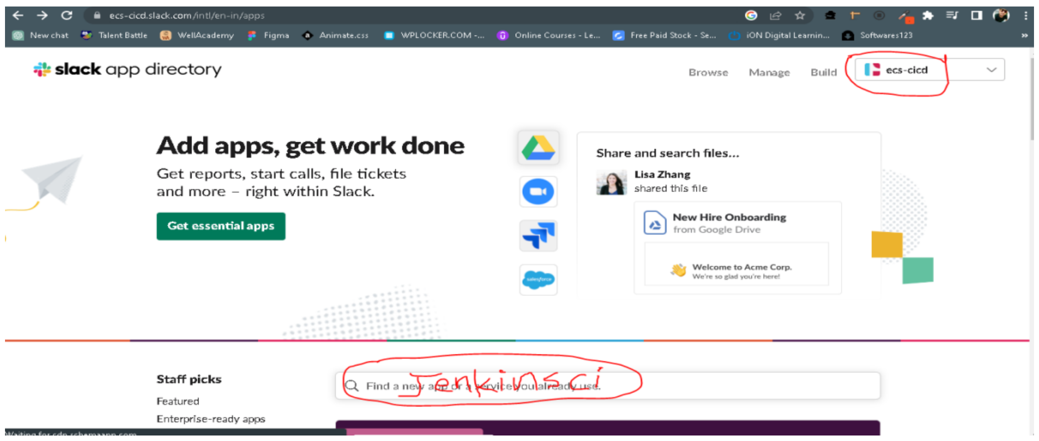
Add apps to Slack | Apps and integrations

Explore and install apps on the Slack App Directory. ... Add apps, get work done. Get reports, start calls, file tickets and more – right within Slack.

People also search for

slack apps, slack login, slack app directory, slack api, slack my apps, slack app store

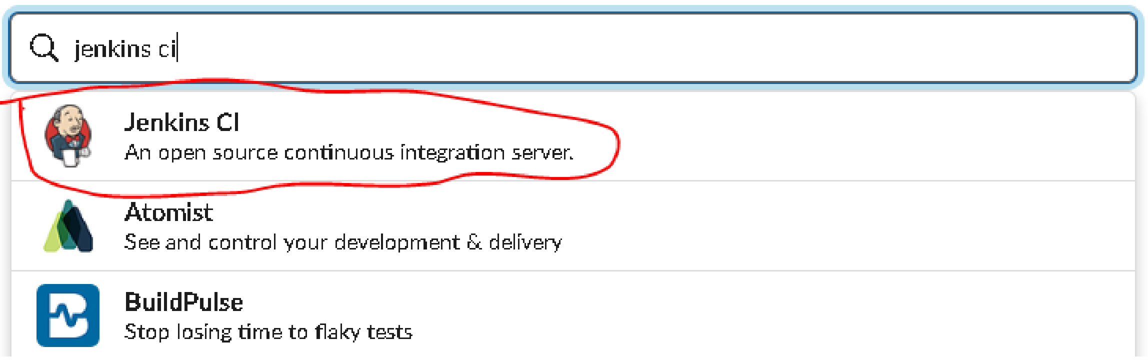
You will automatically redirected to your created workspace(ecs-cicd) and in search bar, type Jenkins ci



Add apps, get work done

Get reports, start calls, file tickets and more – right within Slack.

Click on Jenkins Ci

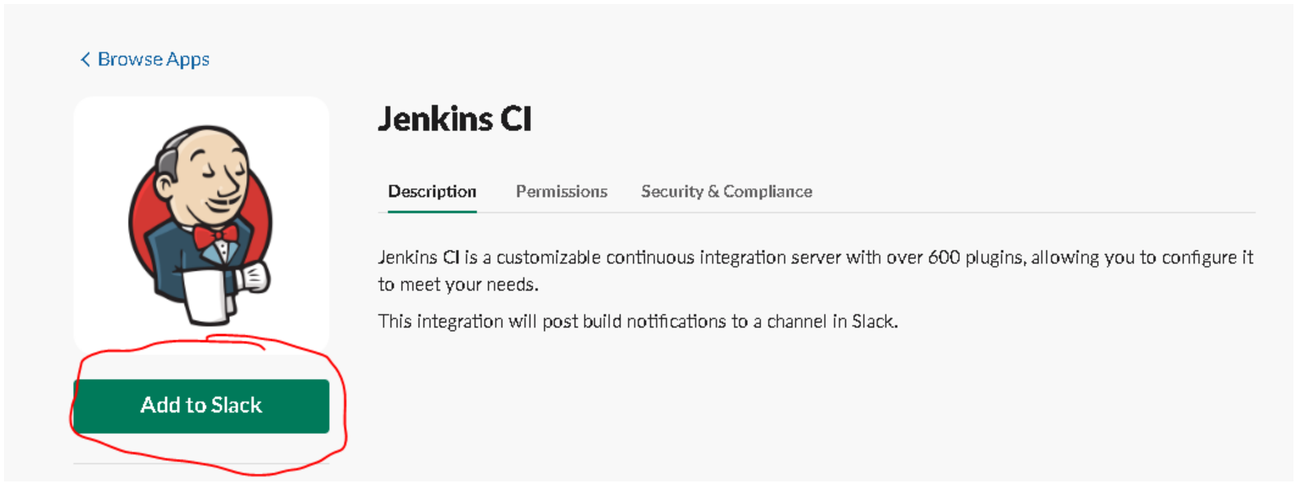


Jenkins CI
An open source continuous integration server.

Atomist
See and control your development & delivery

BuildPulse
Stop losing time to flaky tests

Click on **Add to Slack**


< Browse Apps

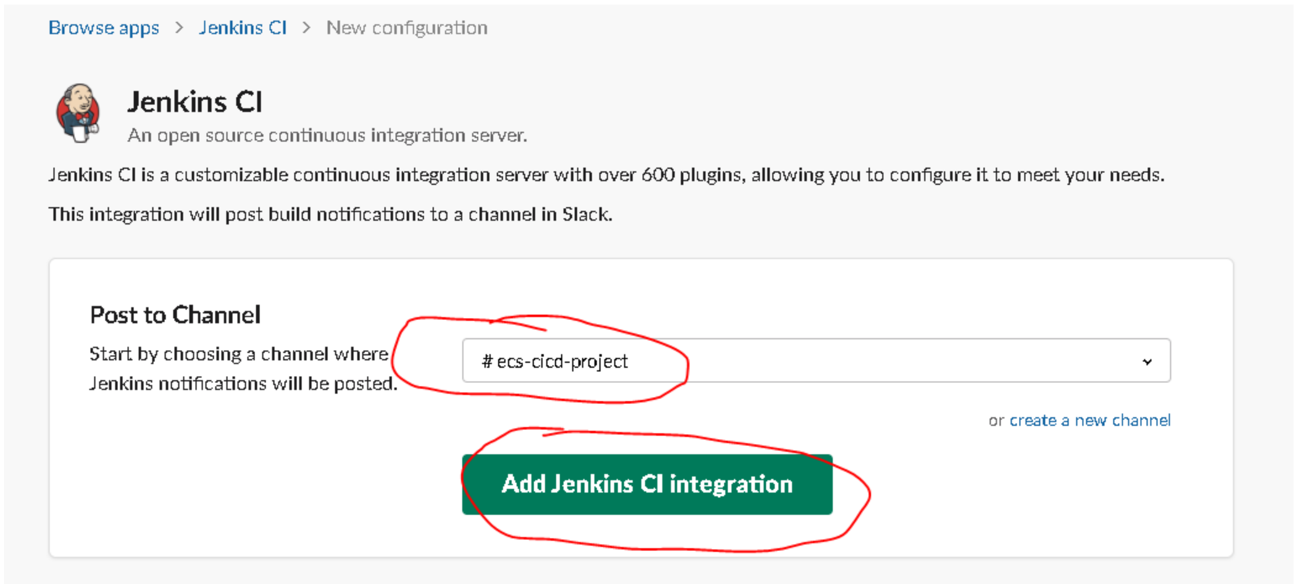
Jenkins CI

Description   Permissions   Security & Compliance

Jenkins CI is a customizable continuous integration server with over 600 plugins, allowing you to configure it to meet your needs.

This integration will post build notifications to a channel in Slack.

Now, **Choose our channel** : ecs-cicd-project


Browse apps > Jenkins CI > New configuration

Jenkins CI

An open source continuous integration server.

Jenkins CI is a customizable continuous integration server with over 600 plugins, allowing you to configure it to meet your needs.

This integration will post build notifications to a channel in Slack.

Post to Channel

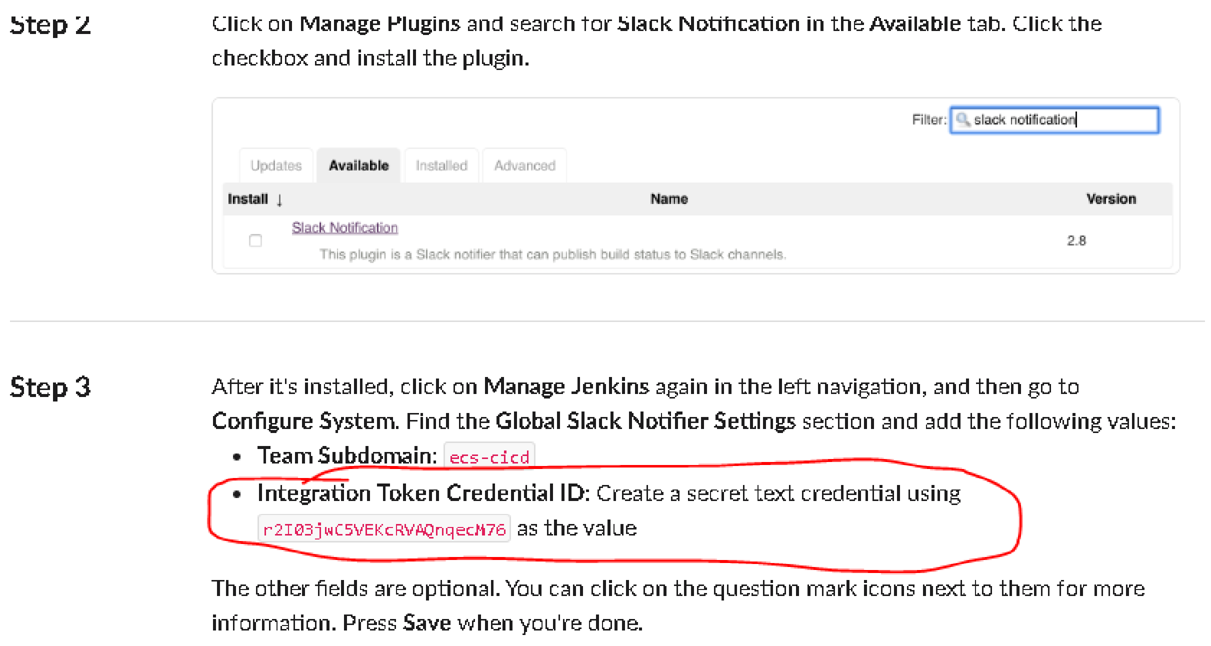
Start by choosing a channel where Jenkins notifications will be posted.

ecs-cicd-project

or create a new channel

Click **Add Jenkins CI Integration**

In Step 3, Copy the Integration Token & paste somewhere(notepad)



Step 2

Click on **Manage Plugins** and search for **Slack Notification** in the **Available** tab. Click the checkbox and install the plugin.

Install ↓	Name	Version
☐	Slack Notification This plugin is a Slack notifier that can publish build status to Slack channels.	2.8

Step 3

After it's installed, click on **Manage Jenkins** again in the left navigation, and then go to **Configure System**. Find the **Global Slack Notifier Settings** section and add the following values:

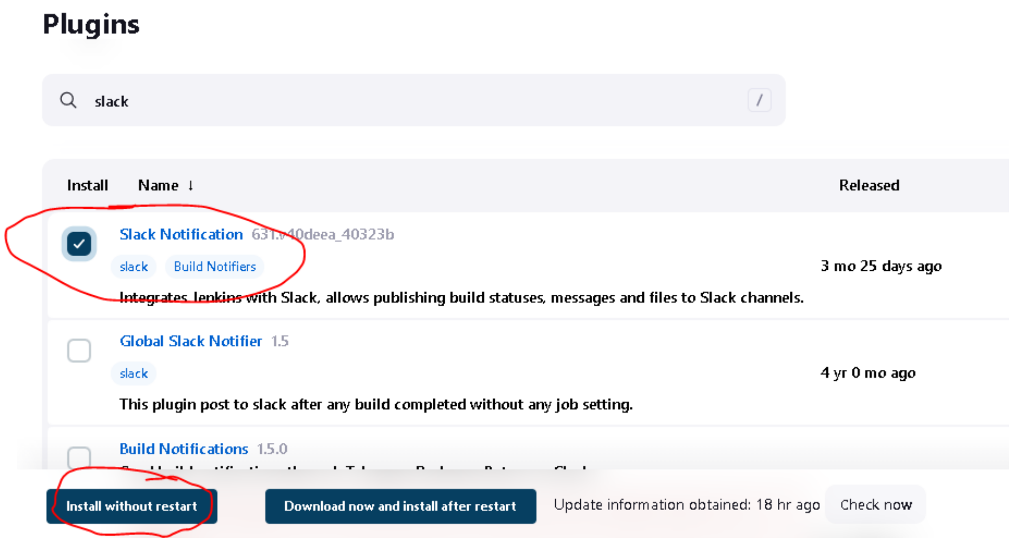
- **Team Subdomain:** `ecs-cicd`
- **Integration Token Credential ID:** Create a secret text credential using `r2I03jwC5VEKcRVAQnqecN76` as the value

The other fields are optional. You can click on the question mark icons next to them for more information. Press **Save** when you're done.

And **Scroll Down** & click **Save Settings**

Now, Come to Jenkins Dashboard

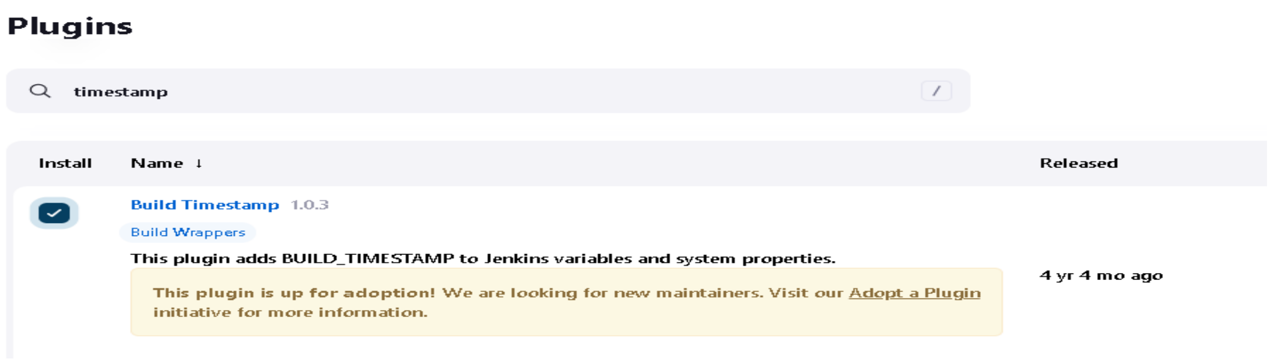
Go to Manage Jenkins > Manage Plugins > Install **Slack Notification** and **Build Timestamp**



Plugins

Install	Name ↓	Released
✓	Slack Notification 631.v40deea_40323b slack Build Notifiers Integrates Jenkins with Slack, allows publishing build statuses, messages and files to Slack channels.	3 mo 25 days ago
☐	Global Slack Notifier 1.5 slack This plugin post to slack after any build completed without any job setting.	4 yr 0 mo ago
☐	Build Notifications 1.5.0	

Install without restart
Download now and install after restart
Update information obtained: 18 hr ago
Check now


Plugins

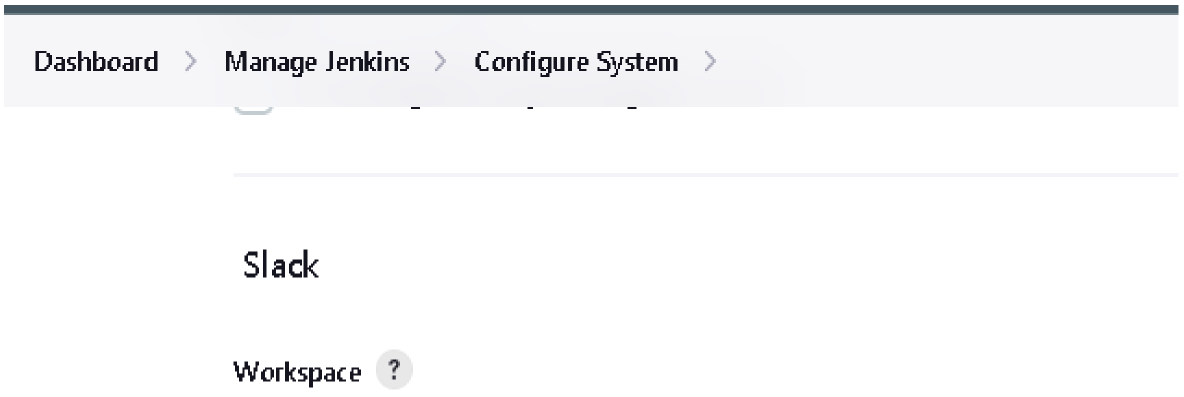
timestamp

Install	Name ↓	Released
✓	Build Timestamp 1.0.3 Build Wrappers This plugin adds BUILD_TIMESTAMP to Jenkins variables and system properties. This plugin is up for adoption! We are looking for new maintainers. Visit our Adopt a Plugin initiative for more information.	4 yr 4 mo ago

Click on **Install without restart**

Now Goto Manage Jenkins > configure System >

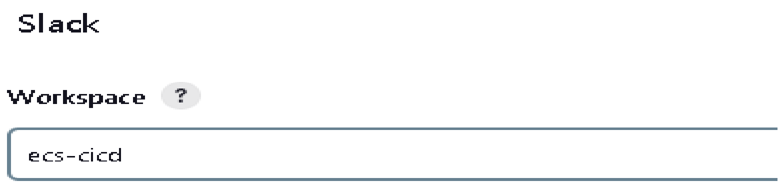
Scroll Down, you will see the Slack settings


Dashboard > Manage Jenkins > Configure System >

Slack

Workspace ?

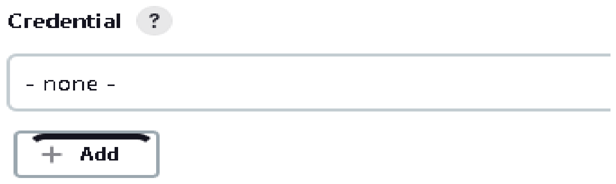
Give our **workspace name**: ecs-cicd


Slack

Workspace ?

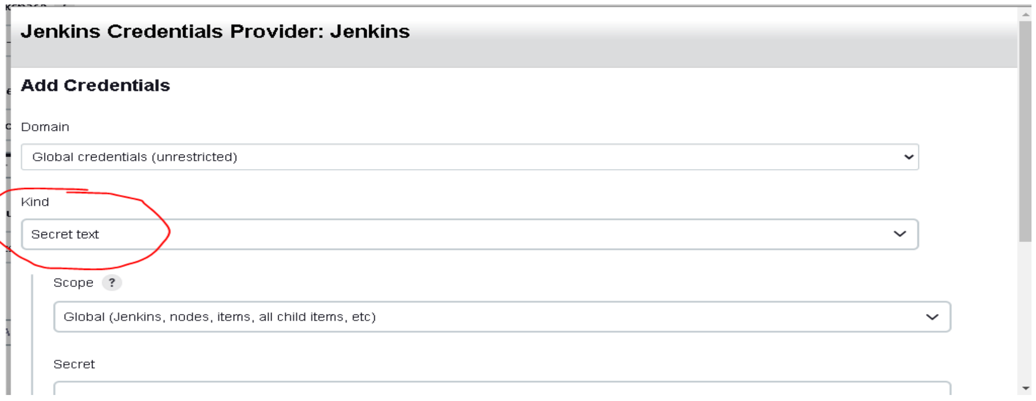
ecs-cicd

Click on Add to add Credentails


Credential ?

- none -

Select **kind** : Secret Text


Jenkins Credentials Provider: Jenkins

Add Credentials

Domain

Global credentials (unrestricted)

Kind

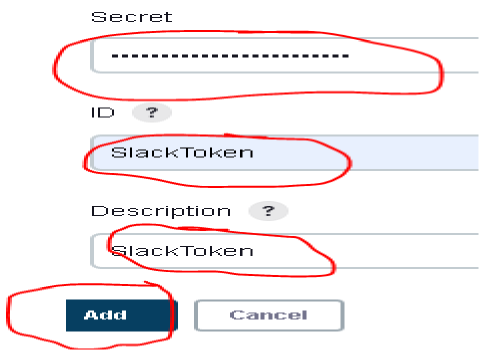
Secret text

Scope

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

Secret

Paste our Slack token from Notepad that we copied earlier & Give any ID name, description. Click on **Add**


Secret

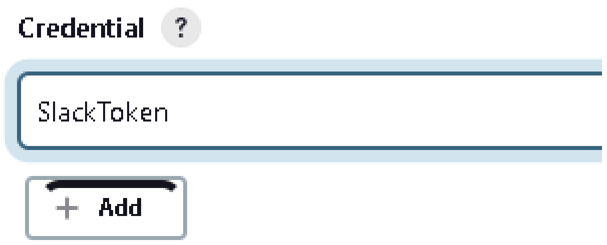
ID

SlackToken

Description

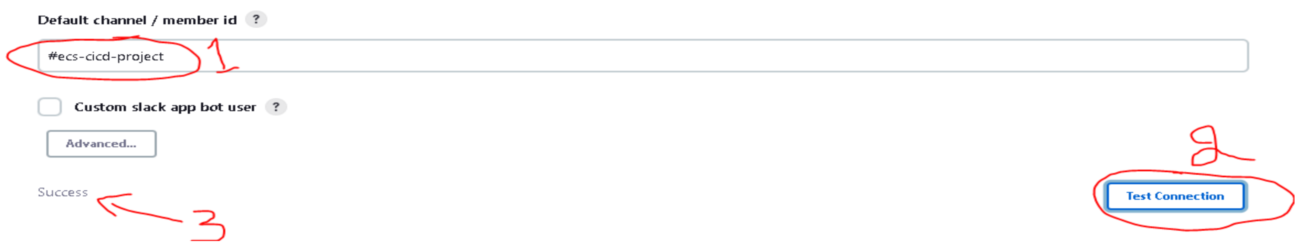
SlackToken

Now **Select** SlackToken


Credential

SlackToken

Give our channel name: #ecs-cicd-project


Default channel / member id

#ecs-cicd-project

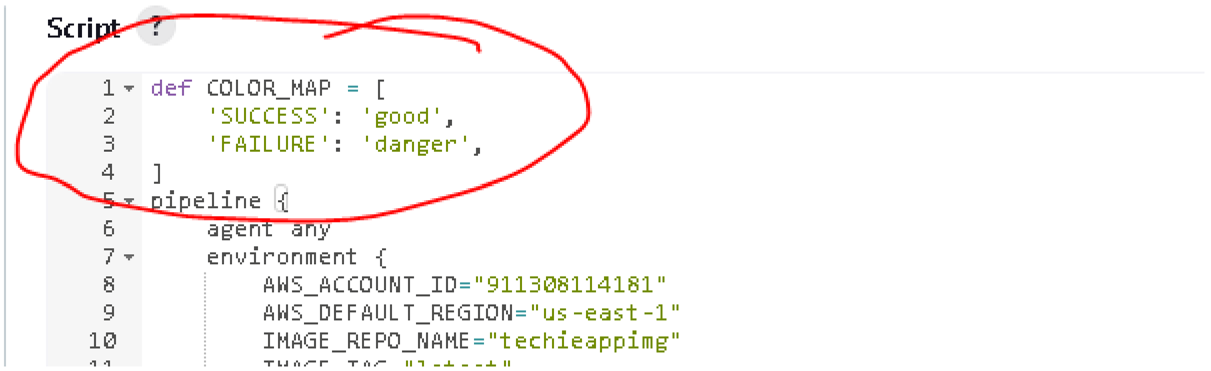
Custom slack app bot user

Click on **Test Connection**, you will get the success msg(means your successfully configured..!)

Now Click on **Save**

Copy The Below Code & paste it on the Top of the Pipeline Script:-

```
def COLOR_MAP = [
   'SUCCESS': 'good',
   'FAILURE': 'danger',
]
```



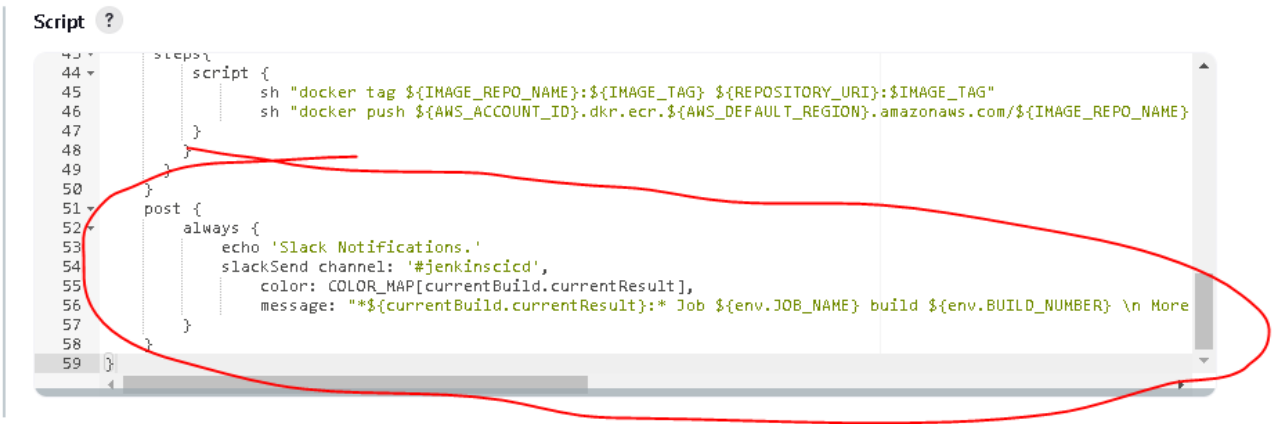
Script ?

```
def COLOR_MAP = [
    'SUCCESS': 'good',
    'FAILURE': 'danger',
]
pipeline {
    agent any
    environment {
        AWS_ACCOUNT_ID="911308114181"
        AWS_DEFAULT_REGION="us-east-1"
        IMAGE_REPO_NAME="techieappimg"
```

These below Commands(stage) should be add bottom of the pipeline script:-

```
post {
      always {
         echo 'Slack Notifications.'
         slackSend channel: '#jenkinscicd',
            color: COLOR_MAP[currentBuild.currentResult],
            message: "*${currentBuild.currentResult}:* Job ${env.JOB_NAME} build ${env.BUILD_NUMBER} \n More info at: ${env.BUILD_URL}"
```

```
    }
  }
```



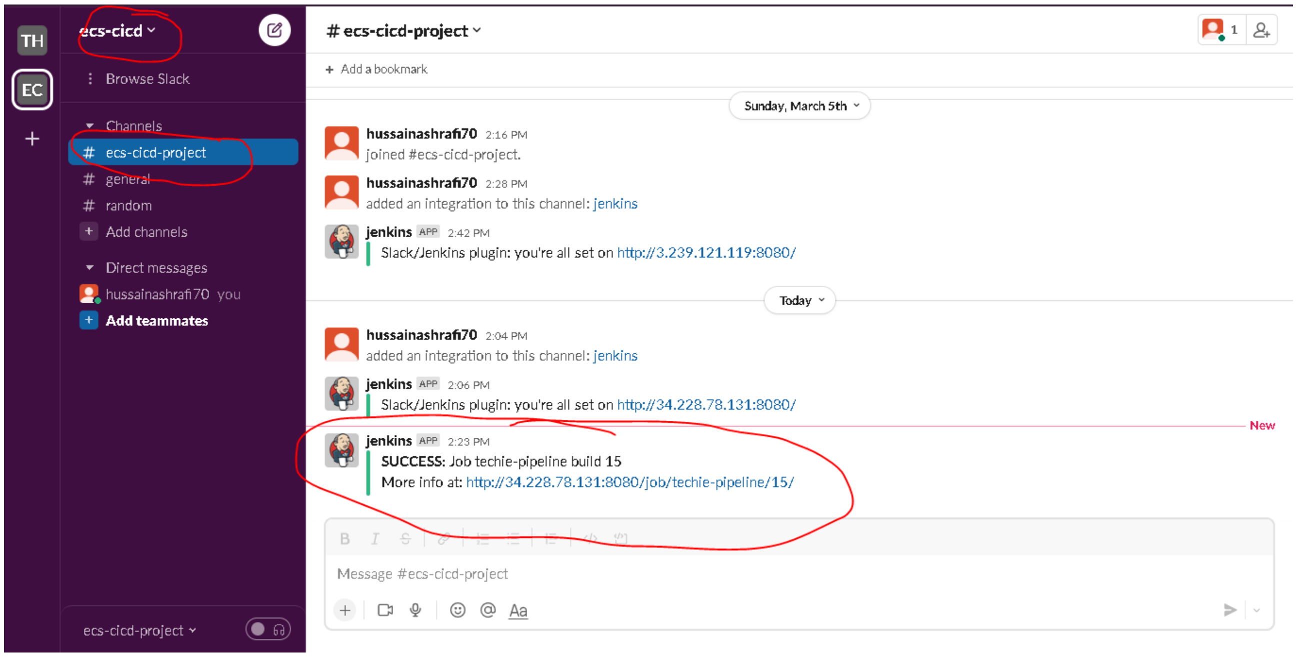
Script ?

```
        steps{
          script {
            sh "docker tag ${IMAGE_REPO_NAME}:${IMAGE_TAG} ${REPOSITORY_URI}:$IMAGE_TAG"
            sh "docker push ${AWS_ACCOUNT_ID}.dkr.ecr.${AWS_DEFAULT_REGION}.amazonaws.com/${IMAGE_REPO_NAME}
          }
        }
      }
    }
    post {
        always {
            echo 'Slack Notifications.'
            slackSend channel: '#jenkinscicd',
                color: COLOR_MAP[currentBuild.currentResult],
                message: "*${currentBuild.currentResult}:* Job ${env.JOB_NAME} build ${env.BUILD_NUMBER} \n More
        }
    }
}
```

Then **Save**

Click **Build Now,**

If the Pipeline is **Successfully Deployed the app to the ECS, then you will get the Notification on the Slack Channel (ecs-cicd-project) with Green Signal(that means success)**



hussainashrafi70 2:16 PM
joined #ecs-cicd-project.

hussainashrafi70 2:28 PM
added an integration to this channel: jenkins

jenkins APP 2:42 PM
Slack/Jenkins plugin: you're all set on http://3.239.121.119:8080/

hussainashrafi70 2:04 PM
added an integration to this channel: jenkins

jenkins APP 2:06 PM
Slack/Jenkins plugin: you're all set on http://34.228.78.131:8080/

jenkins APP 2:23 PM
SUCCESS: Job techie-pipeline build 15
More info at: http://34.228.78.131:8080/job/techie-pipeline/15/

If you click that More Info Link, Won’t Work, Because our Jenkins Ip will changes while turning off the instances. To make that link Workable. You need to configure the Jenkins in Manage Jenkins Settings and give the Private IP. Save the Changes, So next time you will get the Private Ip which does not change even we turn off & turn on the Jenkins instance. The Link Will Work…

**~~~Project END~~~**