

## First make a public EKS repository: I called mine jenkins

Second we have to create a dockerfile. This is what mine looks like:

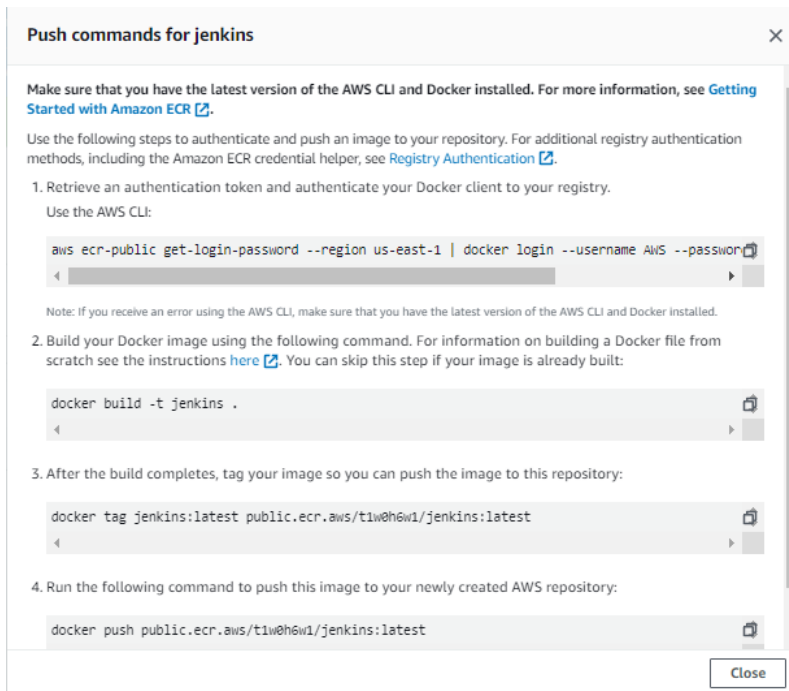
### This is what my dockerfile looks like

```
FROM jenkins/jenkins:2.319.1-jdk11
USER root
RUN apt-get update && apt-get install -y lsb-release
RUN curl -fsSLo /usr/share/keyrings/docker-archive-keyring.asc \
  https://download.docker.com/linux/debian/gpg
RUN echo "deb [arch=$(dpkg --print-architecture) \
  signed-by=/usr/share/keyrings/docker-archive-keyring.asc] \
  https://download.docker.com/linux/debian \
  $(lsb_release -cs) stable" > /etc/apt/sources.list.d/docker.list
RUN apt-get update && apt-get install -y docker-ce-cli
USER jenkins
RUN jenkins-plugin-cli --plugins "blueocean:1.25.2 docker-workflow:1.26"
```

### CD into the folder with the dockerfile

cd C:\Users\Bishajit\Deployment7

After changing into the folder with the dockerfile, run these commands from your eks repository's "view push commands".



The screenshot shows a dialog box titled "Push commands for jenkins" with a close button (X) in the top right corner. The dialog contains instructions for pushing a Docker image to an AWS ECR repository. It includes a note about having the latest version of AWS CLI and Docker installed, and a link to "Getting Started with Amazon ECR". The instructions are numbered 1 through 4, each followed by a code block for the required command. The commands are: 1. Retrieve an authentication token and authenticate your Docker client to your registry. Use the AWS CLI: `aws ecr-public get-login-password --region us-east-1 | docker login --username AWS --password-` 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 jenkins .` 3. After the build completes, tag your image so you can push the image to this repository: `docker tag jenkins:latest public.ecr.aws/t1w0h6w1/jenkins:latest` 4. Run the following command to push this image to your newly created AWS repository: `docker push public.ecr.aws/t1w0h6w1/jenkins:latest`. A "Close" button is located at the bottom right of the dialog.

**Push commands for jenkins**

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-public get-login-password --region us-east-1 | docker login --username AWS --password-
```
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 jenkins .
```
3. After the build completes, tag your image so you can push the image to this repository:  

```
docker tag jenkins:latest public.ecr.aws/t1w0h6w1/jenkins:latest
```
4. Run the following command to push this image to your newly created AWS repository:  

```
docker push public.ecr.aws/t1w0h6w1/jenkins:latest
```

Close

## Create the cluster

Configure cluster

Cluster name\*  ⓘ

Networking

Create a new VPC for your cluster to use. A VPC is an isolated portion of the AWS Cloud populated by AWS objects, such as Fargate tasks.

Create VPC ☐ Create a new VPC for this cluster

Tags

Key	Value
<input type="text" value="Add key"/>	<input type="text" value="Add value"/>

CloudWatch Container Insights

CloudWatch Container Insights is a monitoring and troubleshooting solution for containerized applications and microservices. It collects, aggregates, and summarizes compute utilization such as CPU, memory, disk, and network; and diagnostic information such as container restart failures to help you isolate issues with your clusters and resolve them quickly. [Learn more](#)

CloudWatch Container Insights ☐ Enable Container Insights

Configure task definitions


Create new Task Definition

- Step 1: Select launch type compatibility
- Step 2: Configure task and container definitions

Select launch type compatibility


Select which launch type you want your task definition to be compatible with based on where you want to launch your task.

**FARGATE**



Price based on task size  
Requires network mode awsipc  
AWS-managed infrastructure, no Amazon EC2 instances to manage

**EC2**



Price based on resource usage  
Multiple network modes available  
Self-managed infrastructure using Amazon EC2 instances

# Configure task and container definitions

A task definition specifies which containers are included in your task and how they interact with each other. You can also specify data volumes for your containers to use. [Learn more](#)

Task definition name\*

jenkins-task-defintions

Requires compatibilities\* FARGATE

Task role

ecsTaskExecutionRole

Optional IAM role that tasks can use to make API requests to authorized AWS services. Create an Amazon Elastic Container Service Task Role in the [IAM Console](#)

Network mode

awsvpc

If you choose <default>, ECS will start your container using Docker's default networking mode, which is Bridge on Linux and NAT on Windows. Windows tasks support the <default> and awsvpc network modes.

**Configure port mapping for the container. This allows us to access application in port 80 and port 8080**

Add container

▼ Standard

Container name\*

jenkins-container

Image\*

public.ecr.aws/t1w0h6w1/jenkins:latest

Private repository authentication\*

☐

Memory Limits (MiB)

Soft limit

128

+

 Add Hard limit

Define hard and/or soft memory limits in MiB for your container. Hard and soft limits correspond to the 'memory' and 'memoryReservation' parameters, respectively, in task definitions.  
ECS recommends 300-500 MiB as a starting point for web applications.

Port mappings

Container port	Protocol	
80	tcp	✕
8080	tcp	✕

+

 Add port mapping

Host port mappings are not valid when the network mode for a task definition is host or awsvpc. To specify different host and container port mappings, choose the Bridge network mode.

## Task size



The task size allows you to specify a fixed size for your task. Task size is required for tasks using the Fargate launch type and is optional for the EC2 or External launch type. Container level memory settings are optional when task size is set. Task size is not supported for Windows containers.

Task memory (GB)

The valid memory range for 0.25 vCPU is: 0.5GB - 2GB.

Task CPU (vCPU)

The valid CPU for 0.5 GB memory is: 0.25 vCPU

### Task memory maximum allocation for container memory reservation



0 512 shared of 512 MiB

### Task CPU maximum allocation for containers



0 256 shared of 256 CPU units

## Container definitions



Add container

Container Na...	Image	Hard/Soft m...	CPU Units ...	GPU	Inference Ac...	Essential ...	
No results							

## Service integration

AWS App Mesh is a service mesh based on the Envoy proxy that makes it easy to monitor and control microservices. App Mesh standardizes how your microservices communicate, giving you end-to-end visibility and helping to ensure high-availability for your

## Configure and add the container use the uri you used for the tag

Add container ✕

▼ Standard

Container name\*

Jenkins-container

?

Image\*

public.ecr.aws/t1w0h6w1/jenkins:latest

G

?

Private repository authentication\*

☐

?

Memory Limits (MiB)

Soft limit

128

?

➤ Add Hard limit

Define hard and/or soft memory limits in MiB for your container. Hard and soft limits correspond to the "memory" and "memoryReservation" parameters, respectively, in task definitions. ECS recommends 300-500 MiB as a starting point for web applications.

Port mappings

Container port

Protocol

tcp

?

➤ Add port mapping

Host port mappings are not valid when the network mode for a task definition is host or awsvpc. To specify different host and container port mappings, choose the Bridge network mode.

Advanced container configuration

Create

Once created this will pop up

Launch Status

Task definition status - 2 of 2 completed

Create Task Definition: jenkins-task-definition

jenkins-task-definition succeeded

Create CloudWatch Log Group

CloudWatch Log Group created  
CloudWatch Log Group /ecs/jenkins-task-definition

Back View task definition

Now go to your cluster and click on the task. And then click run new task

Cluster : JenkinsCluster

Get a detailed view of the resources on your cluster.

Cluster ARN   arn:aws:ecs:us-east-2:323867645900:cluster/JenkinsCluster

Status   ACTIVE

Registered container instances   0

Pending tasks count   0 Fargate, 0 EC2, 0 External

Running tasks count   0 Fargate, 0 EC2, 0 External

Active service count   0 Fargate, 0 EC2, 0 External

Draining service count   0 Fargate, 0 EC2, 0 External

Services Tasks ECS Instances Metrics Scheduled Tasks Tags Capacity Providers

Run new Task Stop Stop All Actions

Desired task status: Running Stopped

Filter in this page Launch type ALL

<input type="checkbox"/>	Task	Task definition	Container instance	Last status	Desire
--------------------------	------	-----------------	--------------------	-------------	--------

# Run Task

Select the cluster to run your task definition on and the number of copies of that task to run. To apply container c

Launch type

☒ FARGATE ☐ EC2 ☐ EXTERNAL

Switch to capacity provider strategy

Operating system family

Linux

Task Definition

Family

jenkins-task-definition

Enter a value

Revision

2 (latest)

Platform version

LATEST

Cluster

Jenkins-cluster

Number of tasks

1

Task Group

...

...

...

## Configure security groups



A security group is a set of firewall rules that control the traffic for your task. On this page, you can add rules to allow specific traffic to reach your task, or you can choose to use an existing security group.  
[Learn more.](#)

Assigned security groups ☒ Create new security group  
☐ Select existing security group

Security group name\*  ⓘ

Description  ⓘ

### Inbound rules for security group

Type	Protocol	Port range	Source	
HTTP ▾	TCP ▾	80 ▾	Anywhere ▾	0.0.0.0/0, ::/0 ⓘ
Custom TCP ▾	TCP ▾	8080 ▾	Anywhere ▾	0.0.0.0/0, ::/0 ⓘ

[Add rule](#)

After configuring click run task

task group ▾

VPC and security groups

VPC and security groups are configurable when your task definition uses the awsvpc network mode.

Cluster VPC\*  ⓘ

Subnets\*

subnet-0ea2783e9ec09e4b1  
(172.31.32.0/20) - us-east-2c  
assign ipv6 on creation: Disabled

 ⓘ

Security groups\*   ⓘ

Auto-assign public IP  ⓘ

Task Placement



## Cluster : jenkins-cluster

Get a detailed view of the resources on your cluster.

Cluster ARN    `arn:aws:ecs:us-east-2:323867645900:cluster/jenkins-cluster`

Status    **ACTIVE**

Registered container instances    0

Pending tasks count    0 Fargate, 0 EC2, 0 External

Running tasks count    1 Fargate, 0 EC2, 0 External

Active service count    0 Fargate, 0 EC2, 0 External

Draining service count    0 Fargate, 0 EC2, 0 External

---

Services   **Tasks**   ECS Instances   Metrics   Scheduled Tasks   Tags   Capacity Providers

[Run new Task](#)   [Stop](#)   [Stop All](#)   [Actions](#) ▾

Desired task status: **Running**   [Stopped](#)

Filter in this page    Launch type    ALL ▾

<input type="checkbox"/>	Task	Task definition	Container instance	Last status	Desired status
<input type="checkbox"/>	<b>8cd656cdffd4405ab56</b>	Jenkins-task-definition:1	--	<b>RUNNING</b>	RUNNING

Click on it

## Task : 8cd656cdffd4405ab5688d4c43cf7b1c

**Details**   **Tags**   **Logs**

---

Filter logs    ✕

Timestamp (UTC+00:00) ▾	Message
▶ 2021-12-13 21:36:33	Please use the following password to proceed to installation:
▶ 2021-12-13 21:36:33	<b>d6deabf8b16940f393ba8ce2a0aa596e</b>
▶ 2021-12-13 21:36:33	This may also be found at: <code>/var/jenkins_home/secrets/initialAdminPassword</code>
▶ 2021-12-13 21:36:33	*****
▶ 2021-12-13 21:36:33	*****
▶ 2021-12-13 21:36:33	*****

Go on the public ip on port 8080

http://3.138.194.153:8080/

## Task : 8cd656cdffd4405ab5688d4c43cf7b1c

Details

Tags

Logs

Cluster

jenkins-cluster

Launch type

FARGATE

Platform version

1.4.0

Task definition

Jenkins-task-definition:1

Group

family:Jenkins-task-definition

Task role

ecsTaskExecutionRole

Last status

RUNNING

Desired status

RUNNING

Created at

2021-12-13 21:35:28 -0500

Started at

2021-12-13 21:36:00 -0500

Network

Network mode

awsvpc

ENI Id

eni-066e830aa92687763

Subnet Id

subnet-0ea2783e9ec09e4b1

Private IP

172.31.43.10

Public IP

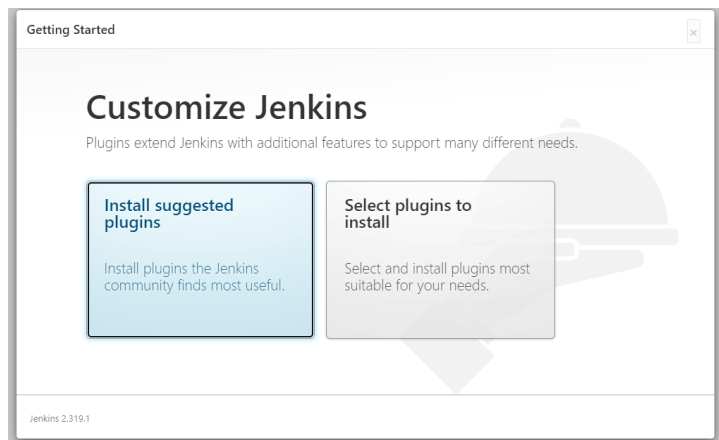
3.138.194.153

Mac address

0a:2f:44:53:6d:fe

Containers

Install suggested plugins, and you can configure an account if you want or just login as admin



## Manage Jenkins → Manage Plugins → available → install docker pipeline and Amazon EC2

<input checked="" type="checkbox"/>	<b>Docker Pipeline</b> Build and use Docker containers from pipelines.	1.26
Enabled	Name ↓	Version
<input checked="" type="checkbox"/>	<b>Amazon EC2 plugin</b> This plugin integrates Jenkins with <a href="#">Amazon EC2</a> or anything implementing the EC2 API's such as an Ubuntu.	1.66

### Create a agent EC2 where your pipeline will build

- That is a ubuntu ami
- And make sure it is in subnet 2c so it is on the same subnet as the task from your ECS cluster
- Also install docker and java using these command

```
1. sudo apt-get update
2. sudo apt-get install default-jre -y git -y nodejs -y npm -y
3. sudo apt install maven
```

```
4. sudo apt-get install \
    ca-certificates \
    curl \
    gnupg \
    lsb-release
```

```
5. curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg
    --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg
```

```
6. echo \
```

```
"deb [arch=$(dpkg --print-architecture)
signed-by=/usr/share/keyrings/docker-archive-keyring.gpg]
https://download.docker.com/linux/ubuntu \
```

```
$(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list >
/dev/null
```

```
7. sudo apt-get update
8. sudo apt-get install docker-ce docker-ce-cli containerd.io
```

After making the EC2 and doing all the installations set it up as a agent

## For credentials -> jenkins

By the way: The host is private ip of agent1

Name	agent1
Description	agent for deploy7
Number of executors	1
Remote root directory	/home/ubuntu
Labels	jenkins-agent
Usage	Use this node as much as possible
Launch method	Launch agents via SSH
Host	172.31.43.10

SSH Username with private key	
Scope	Global (Jenkins, nodes, items, all child items, etc)
ID	agen1
Description	dep7 agen1
Username	ubuntu
<input type="checkbox"/> Treat username as secret	
Private Key	<input checked="" type="radio"/> Enter directly
Key	<div>ukhQoQkegQonHnXxmqDzsaGzrHw9eY3UzZA9KKk8zgsQWAcnyJyY1g1gze8 Wofm3SD6M4C8PmxkhzP1yvZDgCthwXnGdX4QG+s7u5z7rHF03T8nDLX4Mp1ToTGx kB1Wqh1bWZdbbTMAaDXQ88x1Q+XUMKQJG19Xq8uu51G8DJEzOE2OQg== -----END RSA PRIVATE KEY-----</div>
Passphrase	
Add	Cancel

Credentials

ubuntu (agent for dep7)

Add

Host Key Verification Strategy

Known hosts file Verification Strategy

Advanced...





Availability

Keep this agent online as much as possible

Node Properties

☐ Disable deferred wipeout on this node  
☐ Environment variables  
☐ Tool Locations

Save

S	Name ↓	Architecture	Clock Difference	Free Disk Space	Free Swap Space
	agent1	Linux (amd64)	In sync	4.84 GB	 0 B
	Built-In Node	Linux (amd64)	In sync	18.15 GB	 0 B
Data obtained		5.4 sec	5.4 sec	5.4 sec	5.3 sec

After configuring the agent add the docker file to your github

main

DEPLOY07\_ECS / Dockerfile

Bishajit

Add files via upload

1 contributor

5 lines (5 sloc) | 110 Bytes

1

From openjdk:17

2

WORKDIR /

3

COPY demo-0.0.1-SNAPSHOT.jar .

4

EXPOSE 8080

5

CMD java -jar demo-0.0.1-SNAPSHOT.jar

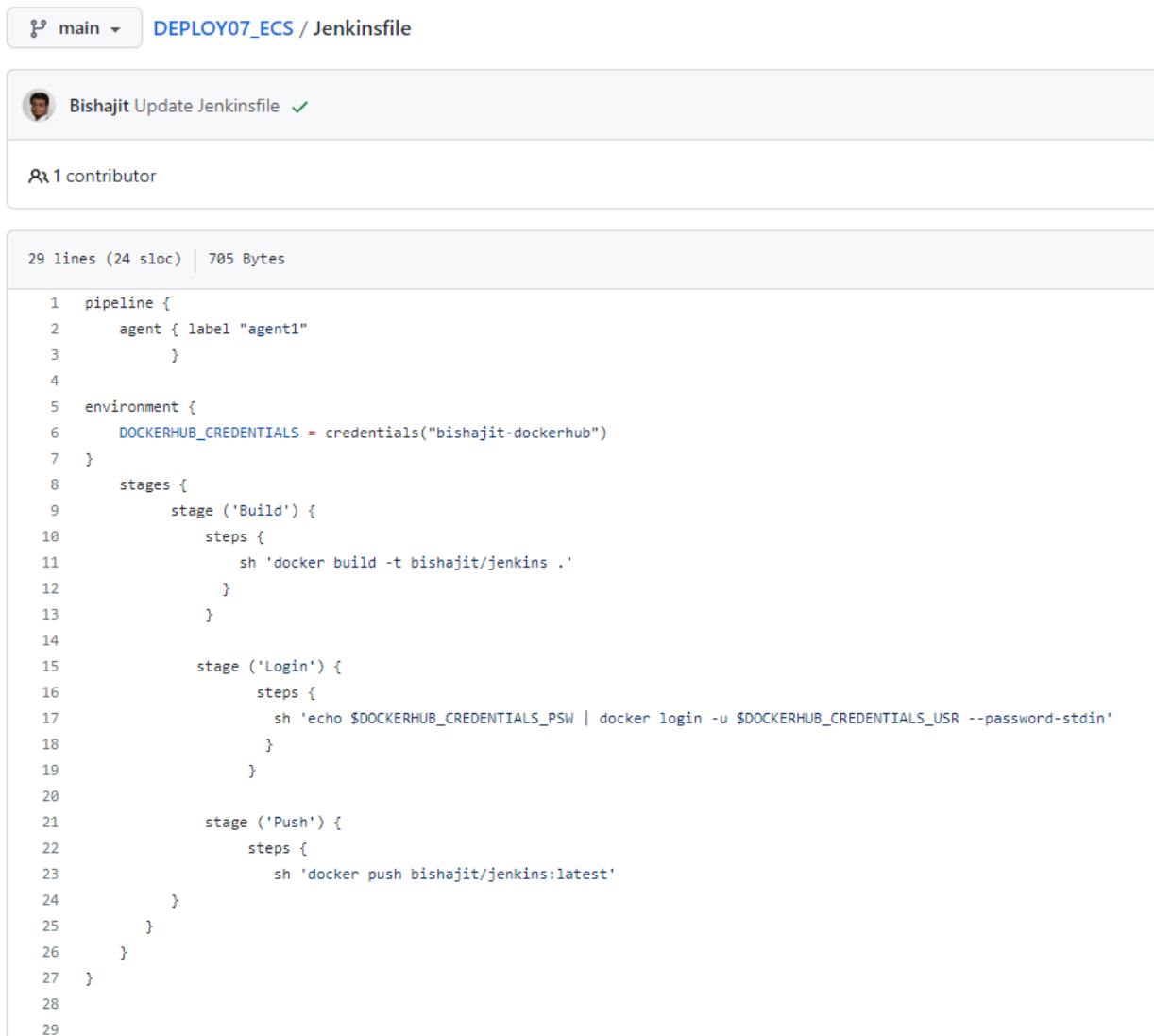
And also add the .jar file to your github as welli

demo-0.0.1-SNAPSHOT.jar

Add files via upload

Last file you need to add is the jenkins file that has the whole pipeline build.

This is what mine looks like:



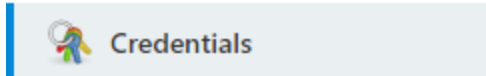
```
1 pipeline {
2   agent { label "agent1" }
3 }
4
5 environment {
6   DOCKERHUB_CREDENTIALS = credentials("bishajit-dockerhub")
7 }
8 stages {
9   stage ('Build') {
10     steps {
11       sh 'docker build -t bishajit/jenkins .'
12     }
13   }
14
15   stage ('Login') {
16     steps {
17       sh 'echo $DOCKERHUB_CREDENTIALS_PSW | docker login -u $DOCKERHUB_CREDENTIALS_USR --password-stdin'
18     }
19   }
20
21   stage ('Push') {
22     steps {
23       sh 'docker push bishajit/jenkins:latest'
24     }
25   }
26 }
27 }
```

**Configure the docker hub access token**

<https://hub.docker.com/settings/general> -> security -> new access token -> generate

**Copy that access token and keep it safe because you only get one free.**

**Now add it to your credentials global and click global**



## Credentials

T	P	Store	Domain	ID	Name
		Jenkins	(global)	agent1	ubuntu (agent for dep7)

Icon: S M L

Click global

Then click add credentials

Scope

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

Username

bishajit

☐ Treat username as secret

Password

Concealed

Change Password

ID

bishajit-dockerhub

Description

dockerhub access token

Save

Now create the pipeline by clicking a new project then click on pipeline. And then press

Enter an item name

deployment

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

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

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

**Folder**  
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

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

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

OK

ok

This is what the pipeline configurations should be.

## Pipeline

### Definition

Pipeline script from SCM

#### SCM

Git

#### Repositories

##### Repository URL

https://github.com/Bishajit/DEPLOY07\_ECS

##### Credentials

bishajit/\*\*\*\*\* (githububtoken)

Add

Advanced...

Add Repository

#### Branches to build

Branch Specifier (blank for 'any')

\*/main

Add Branch

#### Repository browser

(Auto)

#### Branches to build

Branch Specifier (blank for 'any')

\*/main

Add Branch

#### Repository browser

(Auto)

#### Additional Behaviours

Add

#### Script Path

Jenkinsfile

☒ Lightweight checkout

Pipeline Syntax



Dashboard
> deployment7

Back to Dashboard

Status

Changes

Build Now

Configure

Delete Pipeline

Full Stage View

Open Blue Ocean

Rename

Pipeline Syntax

Build History
trend

Filter builds...

#2
Dec 14, 2021, 5:05 AM

#1
Dec 14, 2021, 4:57 AM

Atom feed for all
Atom feed for failures

## Pipeline deployment7

Recent Changes

### Stage View

	Declarative: Checkout SCM	Build	Login	Push
Average stage times: (Average full run time: ~22s)	5s	8s	469ms	2s
#2 Dec 14 00:05 No Changes	557ms	14s	643ms	4s
#1 Dec 13 23:57 No Changes	11s	1s failed	295ms failed	101ms failed

### Permalinks

- Last build (#1), 6 min 8 sec ago
- Last failed build (#1), 6 min 8 sec ago
- Last unsuccessful build (#1), 6 min 8 sec ago
- Last completed build (#1), 6 min 8 sec ago

## Troubleshoot

**Problem:** With my first build I had this error first time I ran it.

Got permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Post

"http://%2Fvar%2Frun%2Fdocker.sock/v1.24/build?buildargs=%7B%7D&cachefrom=%5B%5D&cgroupparent=&cpuperiod=0&cpuquota=0&cpusetcpus=&cpusetmems=&cpushares=0&docke&rfile=Dockerfile&labels=%7B%7D&memory=0&memswap=0&networkmode=default&rm=1&shm&size=0&=bishajit%2Fjenkins&target=&ulimits=null&version=1": dial unix /var/run/docker.sock: connect: permission denied

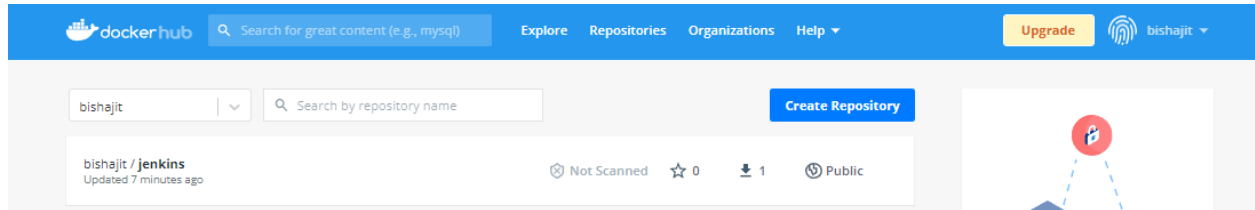
**Fix:** Second build ran smoothly because I ran this command.

**sudo chmod 666 /var/run/docker.sock**

**Helpful**

**link:** <https://www.digitalocean.com/community/questions/how-to-fix-docker-got-permission-denied-while-trying-to-connect-to-the-docker-daemon-socket>

**We are all done. My pipeline pushed therepository to docker hub.**



**`Now destroy all the resources you created.**