First make a public EKS repository: I called mine jenkins

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

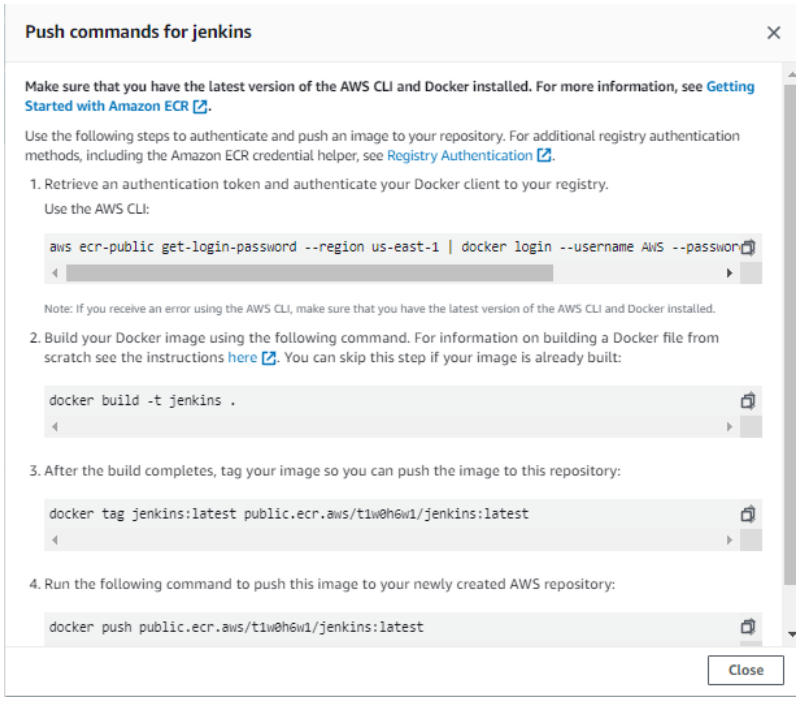
This is what my dockerfile looks like

```
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"
```

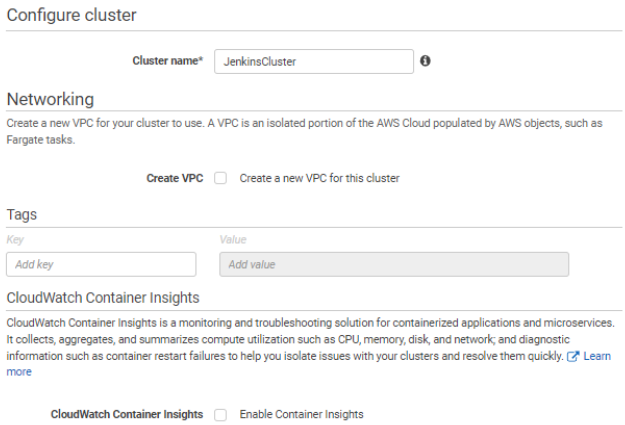
CD into the folder with the dockerfile

cd C:\Users\Bishajit\Deployment7

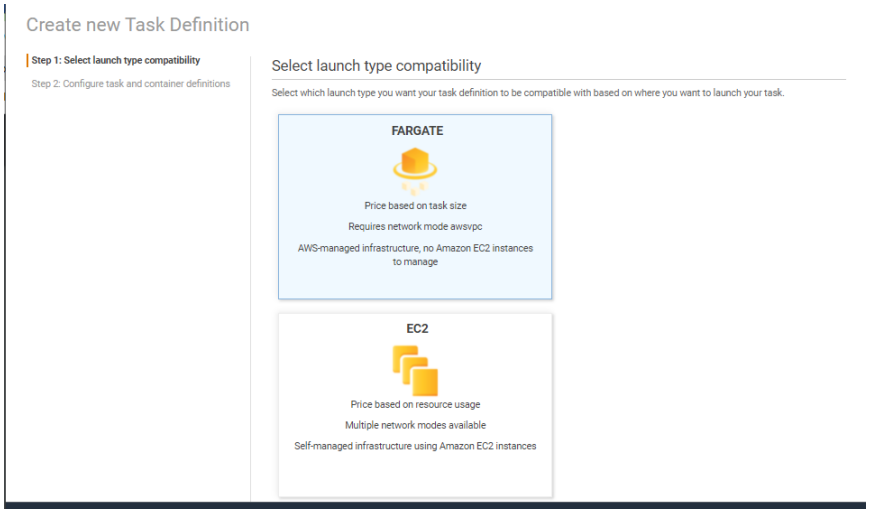
After changing into the folder with the dockerfile, run these commands from your eks repository's "view push commands".



Create the cluster



Configure task definitions



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

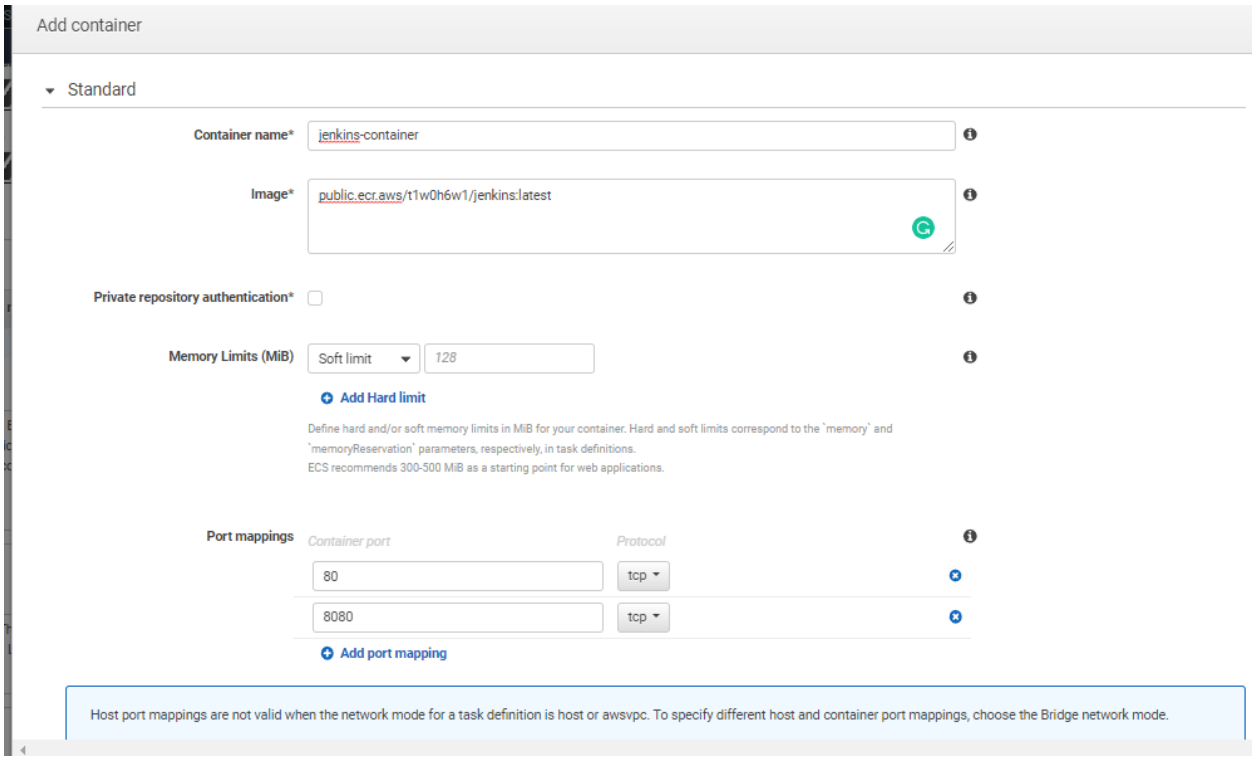
Task role ecsTaskExecutionRole

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



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) 0.5GB

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

Task CPU (vCPU) 0.25 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

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

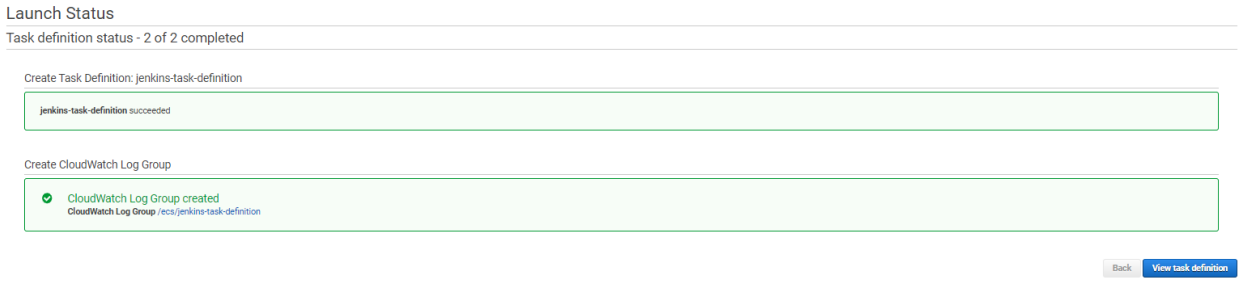
Add port mapping

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.

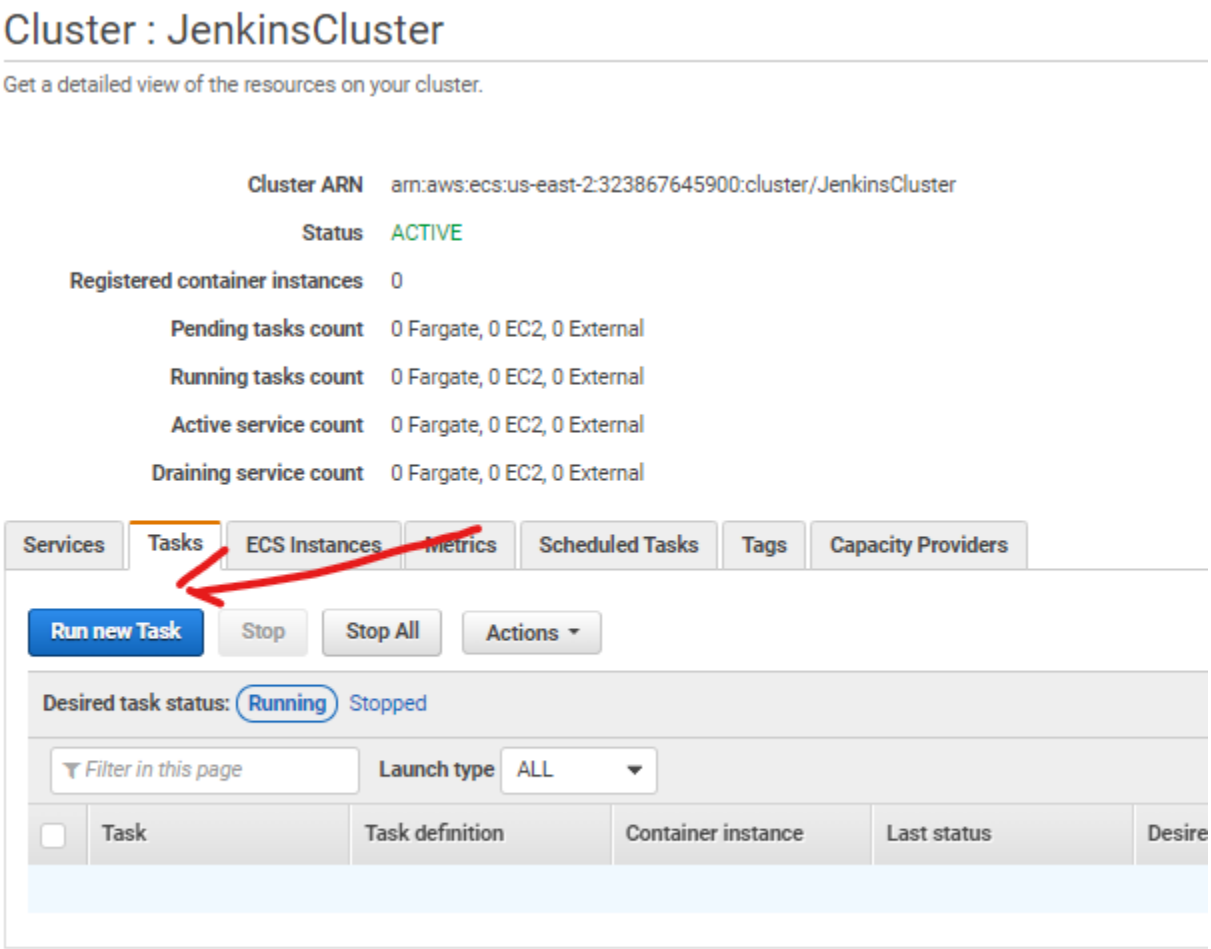
Advanced container configuration

Create

Once created this will pop up



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



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* jenkin-1583

Description Mon Dec 13 2021 19:19:08 GMT-0500 (Eastern Standard Tim

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

VPC and security groups

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

Cluster VPC* vpc-bfba24d4 (172.31.0.0/16) | Default-vpc

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

Security groups* jenkin-9764 Edit

Auto-assign public IP DISABLED

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

Task	Task definition	Container instance	Last status	Desired status
8cd656cdffd4405ab56...	Jenkins-task-definition:1	--	RUNNING	RUNNING

Click on it

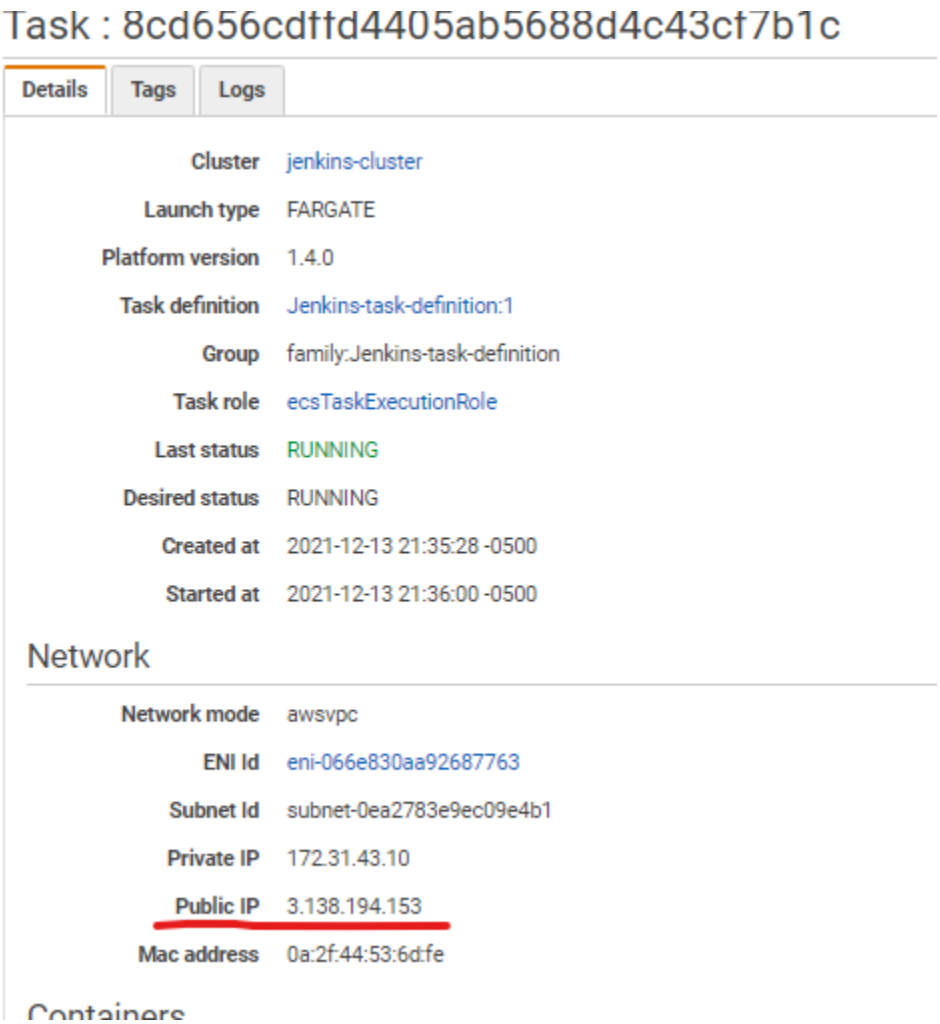
Task : 8cd656cdffd4405ab5688d4c43cf7b1c

Details | Tags | Logs

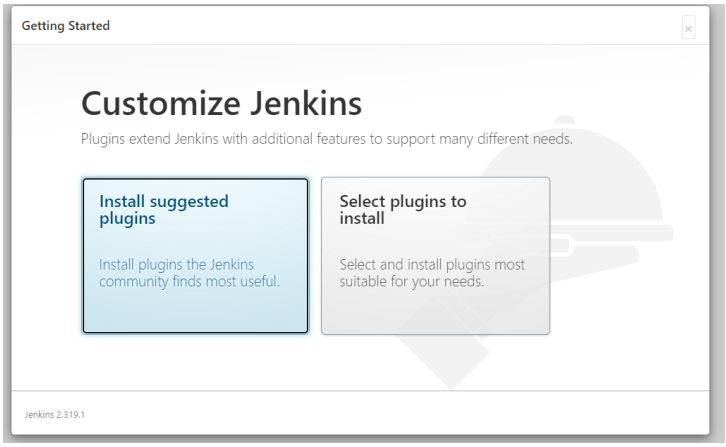
Filter logs

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	d6deabf8b16940f393ba8ce2a0aa596e
2021-12-13 21:36:33	This may also be found at: /var/jenkins_home/secrets/initialAdminPassword
2021-12-13 21:36:33	**********
2021-12-13 21:36:33	**********
2021-12-13 21:36:33	**********

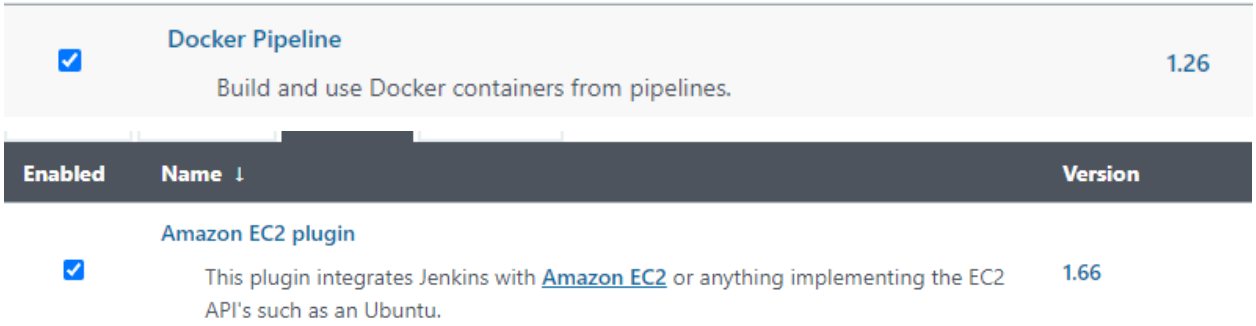
Go on the public ip on port 8080
http://3.138.194.153:8080/



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 Amazone EC2



Create a agent EC2 where your pipeline will build

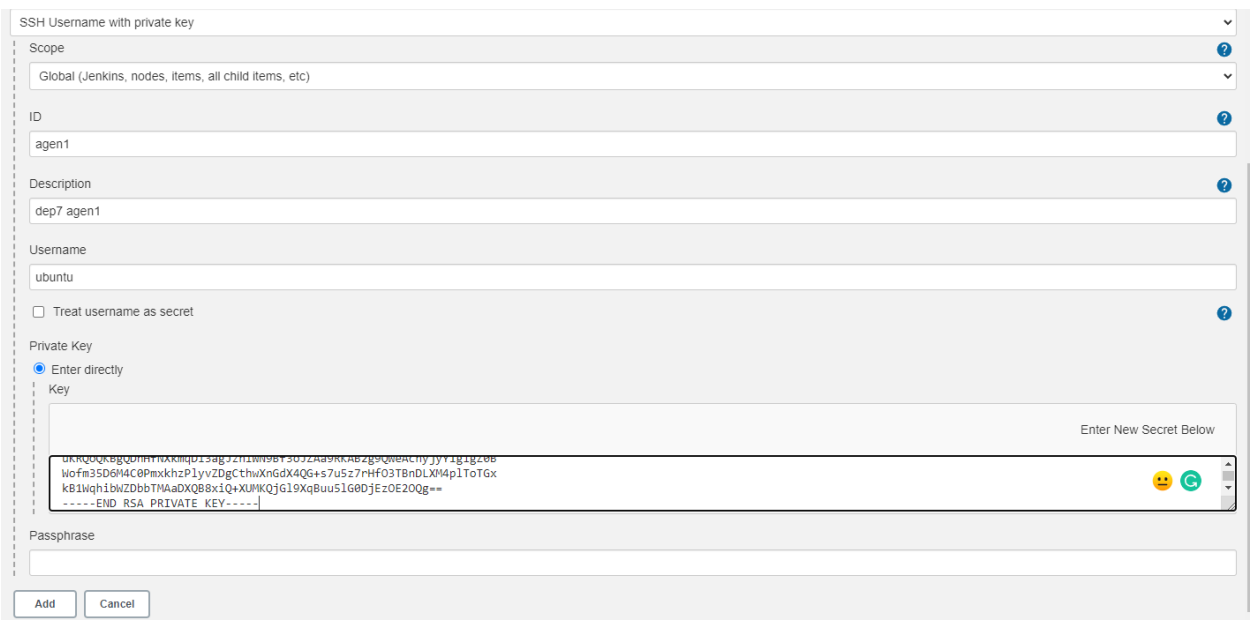
- That is a ubuntu ami
- And make sure it is in subnet 2c so it is on the same subnet as the task from your ECS cluster
- Alsoinstall docker and java using these command

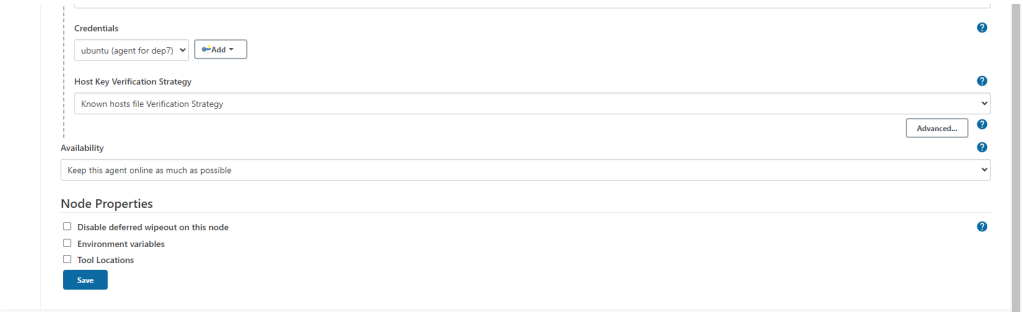
```
1. sudo apt-get update
2. sudo apt-get install default-jre -y git -y nodejs -y npm -y
3. sudo apt install maven

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

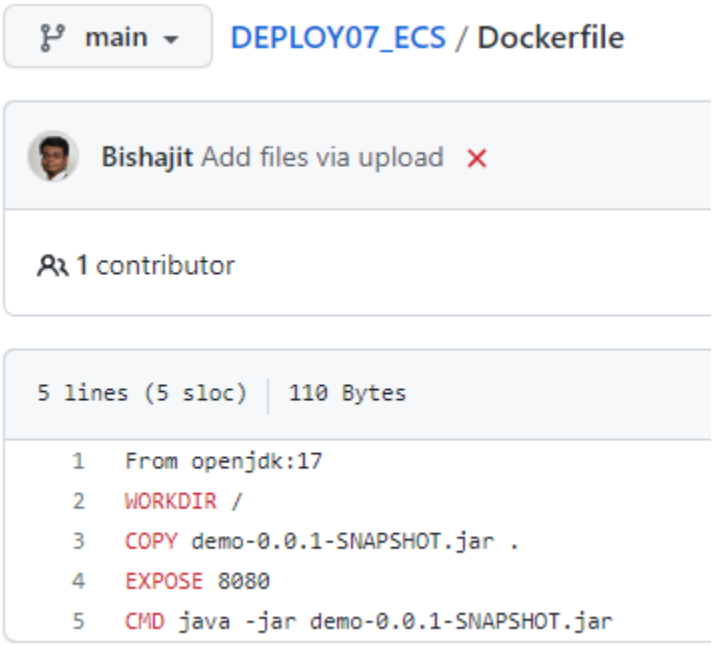
By the way: The host is private ip of agent1





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



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

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

This is what mine looks like:

main DEPLOY07_ECS / Jenkinsfile

Bishajit Update Jenkinsfile

1 contributor

29 lines (24 sloc) | 705 Bytes

```
pipeline {
    agent { label "agent1"
          }

environment {
    DOCKERHUB_CREDENTIALS = credentials("bishajit-dockerhub")
}
    stages {
          stage ('Build') {
              steps {
                  sh 'docker build -t bishajit/jenkins .'
                }
              }

             stage ('Login') {
                    steps {
                      sh 'echo $DOCKERHUB_CREDENTIALS_PSW | docker login -u $DOCKERHUB_CREDENTIALS_USR --password-stdin'
                     }
                   }

              stage ('Push') {
                   steps {
                      sh 'docker push bishajit/jenkins:latest'
          }
       }
    }
}
```

Configure the docker hub access token

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

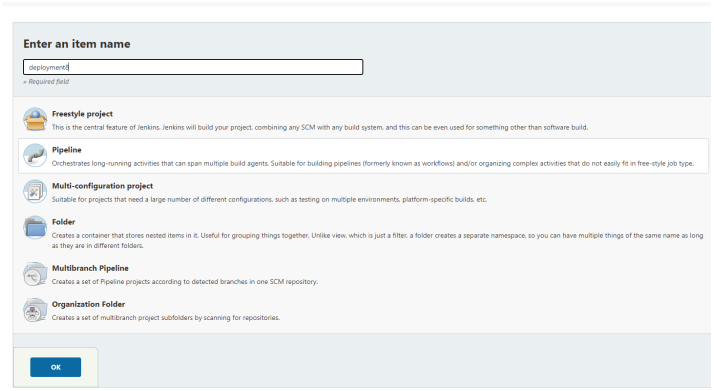
Copy that access token and keep it safe because you only get one free.

Now add it to your credentials global and click global

Click global

Then click add credentials

Now create the pipeline by clicking a new project then click on pipeline. And then press 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/****** (githuhubtoken) 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

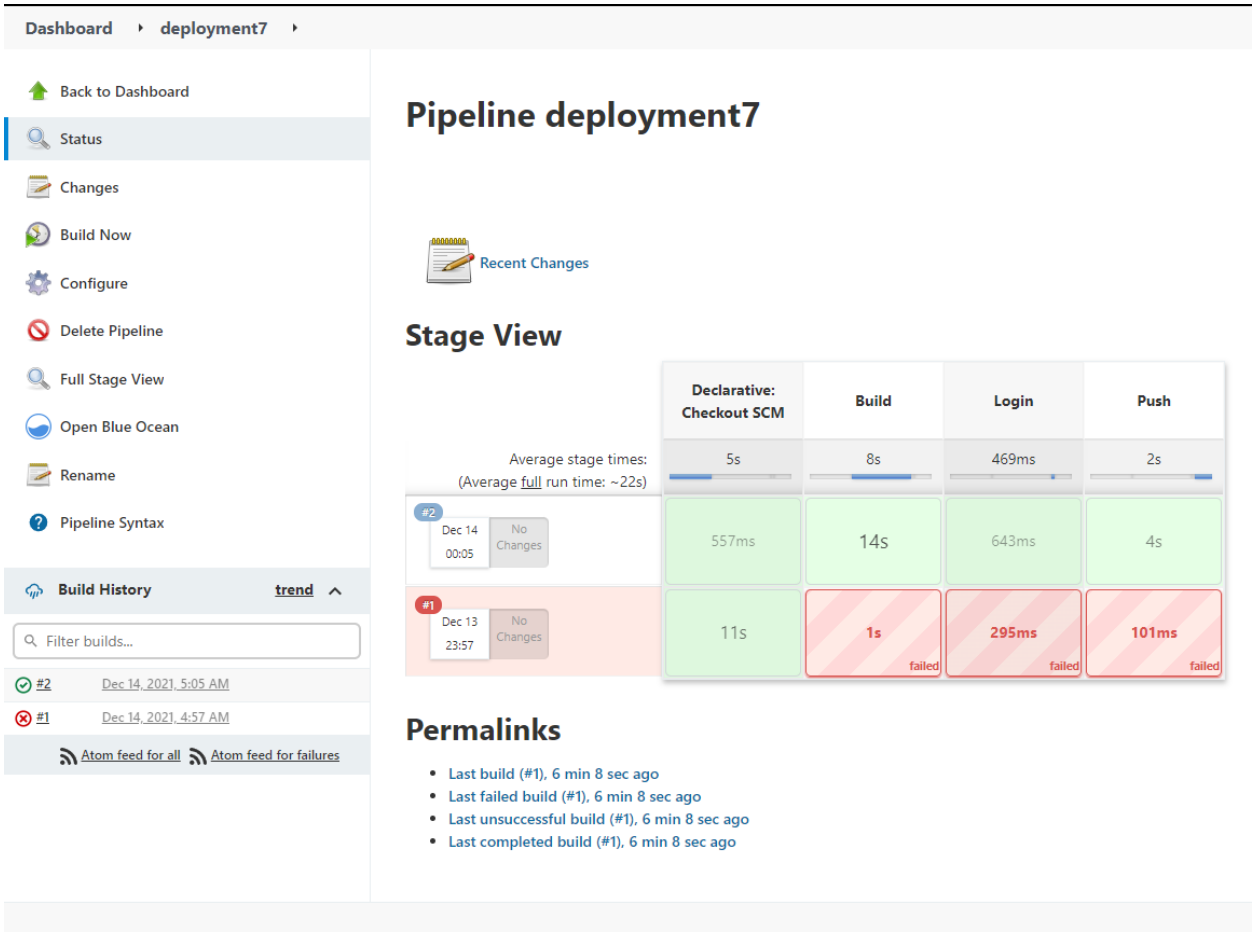
Add

Script Path

Jenkinsfile

Lightweight checkout

Pipeline Syntax



`

Troubleshoot

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

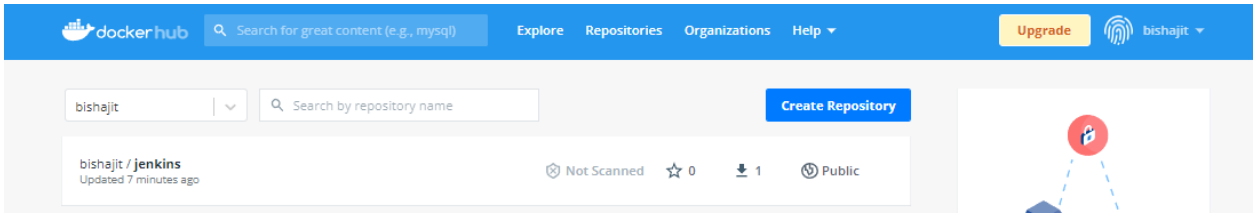
Got permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Post "http://%2Fvar%2Frun%2Fdocker.sock/v1.24/build?buildargs=%7B%7D&cachefrom=%5B%5D&cgroupparent=&cpuperiod=0&cpuquota=0&cpusetcpus=&cpusetmems=&cpushares=0&dockerfile=Dockerfile&labels=%7B%7D&memory=0&memswap=0&networkmode=default&rm=1&shmsize=0&t=bishajit%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.