

Summary

- 1) Prepare the infrastructure to install Jenkins

Create an EC2 machine (AMI=ubuntu) with 4 GB RAM and 15 HDD

Create an IAM role with AdministratorAccess Policy

Attach this role to EC2 machine

- Install AWS client

sudo apt-get install awscli on master and node1

- Install Docker

sudo apt-get update

```
---sudo apt-get install -y lsb-release software-properties-common
```

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

```
---- 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
```

```
--- sudo apt-get update -y
```

```
---- sudo apt-get install -y docker-ce docker-ce-cli containerd.io
```

Changing Docker Cgroup Driver

```
cat <<EOF | sudo tee /etc/docker/daemon.json
{ "exec-opts": ["native.cgroupdriver=systemd"],
"log-driver": "json-file",
"log-opts": {
    "max-size": "100m"
},
"storage-driver": "overlay2"
}
EOF
```

```
sudo systemctl enable docker
sudo systemctl daemon-reload
sudo systemctl restart docker
```

```
-----sudo apt-get update
```

- Install Kubectl

```
sudo apt-get install -y apt-transport-https ca-certificates curl
```

```
sudo curl -fsSLo /usr/share/keyrings/kubernetes-archive-keyring.gpg
https://packages.cloud.google.com/apt/doc/apt-key.gpg
```

```
echo "deb [signed-by=/usr/share/keyrings/kubernetes-archive-keyring.gpg]
https://apt.kubernetes.io/ kubernetes-xenial main" | sudo tee /etc/apt/sources.list.d/kubernetes.list
```

```
sudo apt-get update
```

```
sudo apt-get install -y kubectl
```

```
sudo rm /etc/containerd/config.toml
```

```
sudo systemctl restart containerd
```

```
sudo swapoff -a
```

Install Jenkins

```
sudo apt install default-jdk
```

```
sudo mkdir -p /usr/share/keyrings
```

```
curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo tee  
/usr/share/keyrings/jenkins-keyring.asc > /dev/null
```

Enable Jenkins

```
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
```

```
systemctl status jenkins --no-pager -l
```

```
sudo systemctl enable --now jenkins
```

```
sudo ufw allow 8080
```

```
sudo cat /var/lib/jenkins/secrets/initialAdminPassword
```

Copy password and paste to Jenkins console after launched the Jenkins app via a browser

- Add user to Docker group

```
$ sudo groupadd docker
```

```
$ sudo usermod -a -G docker $USER
```

change righth on file docker

```
sudo chmod 777 /var/run/docker.sock
```

- Binary Terraform

binary

```
wget https://releases.hashicorp.com/terraform/1.2.5/terraform_1.2.5_linux_amd64.zip
```

```
unzip
```

```
mv terraform /usr/bin/
```

```
sudo mv terraform /usr/bin
```

which terraform

2) Install Plugin and configure them on Jenkins

Plugins

Docker

Docker Pipeline

Terraform

Kubernetes CLI

Configuration of Terraform on Jenkins

Go to Global Tool Configuration

And click on Terraform to add this path : “/usr/bin/” and click on save

Create Pipeline to automate eks cluster and deploy application on it

In my Demo I create 2 pipelines

One to automate build image and push image to ECR

The second pipeline to create EKS cluster with one node and after to deploy image on ECR to EKS with YAML file for kubernetes

1) Pipeline 1

pipeline {

agent any

environment {

registry = "143527018359.dkr.ecr.us-east-1.amazonaws.com/my-app"

}

stages {

stage('Checkout') {

```
    steps {
        checkout([$class: 'GitSCM', branches: [[name: '*/main']], extensions: [],
userRemoteConfigs: [[url: 'https://github.com/asejour/genspark-sre-training/']]])
    }
}
```

```
stage('build image') {
    steps{
        script{
            docker.build registry
        }
    }
}
```

```
stage('test') {

    steps{

        echo "Ok continue---Emppty"
    }

}
```

```
stage('Login Docker image to ecr') {
    steps{
        sh "aws ecr get-login-password --region us-east-1 | docker login --username AWS --
password-stdin 143527018359.dkr.ecr.us-east-1.amazonaws.com"
```

```

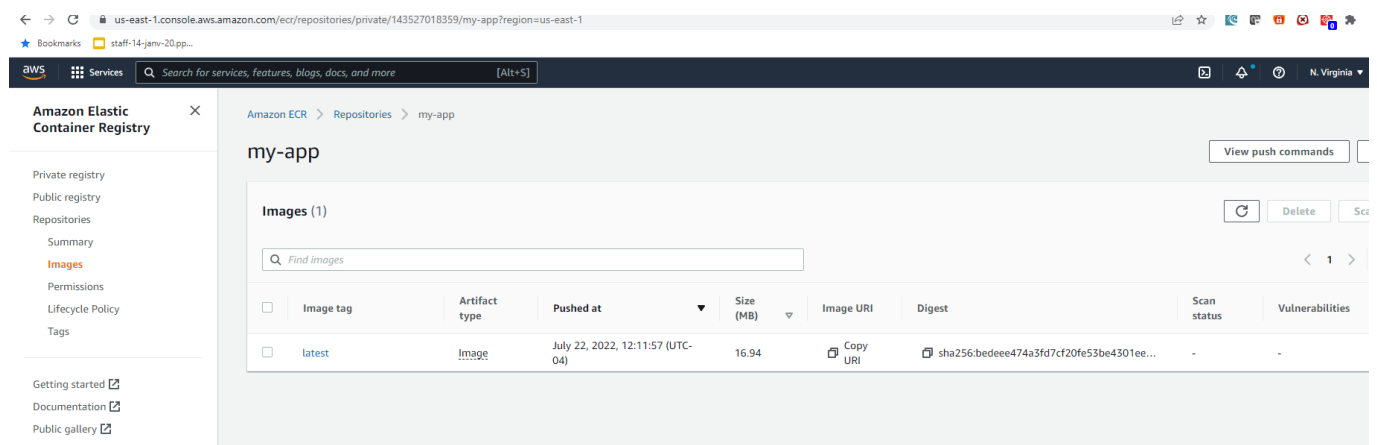
    }
}

stage('Push Docker image to ecr') {
    steps{
        sh "docker push 143527018359.dkr.ecr.us-east-1.amazonaws.com/my-app:latest"

    }
}

}
}

```



2) Pipeline 2

```
pipeline {
    agent any

    tools {
        terraform 'Terraform-11'
```

```
}  
stages{  
  stage('Checkout') {  
    steps {  
      checkout([$class: 'GitSCM', branches: [[name: '*/main']], extensions: [],  
userRemoteConfigs: [[url: 'https://github.com/asejour/adomaa-amos']]])  
    }  
  }  
  
  stage('Launch init terraform'){  
  
    steps{  
  
      sh "terraform init"  
  
    }  
  }  
  
  stage('Launch plan terraform'){  
  
    steps{  
  
      sh "terraform plan"  
  
    }  
  }  
  
  stage('Launch apply terraform'){
```



```
sh "terraform apply --auto-approve"
```

```
}  
}  
  
}
```

```
}
```

The screenshot shows the AWS Management Console for the 'hr-prod-eks-ng-public' EKS cluster. The top navigation bar includes the AWS logo, 'Services' link, a search bar, and the current region 'N. Virginia'. The left sidebar shows the 'Amazon Elastic Kubernetes Service' header and a 'Clusters' section with a 'New' button. The main content area has a breadcrumb trail: 'EKS > Clusters > hr-prod-eks-prod > Node group: hr-prod-eks-ng-public'. A warning box at the top states: 'Your current user or role does not have access to Kubernetes objects on this EKS nodegroup. This may be due to the current user or role not having Kubernetes RBAC permissions to describe cluster resources or not having an entry in the cluster's auth config map. Learn more'. Below this, the 'Node group configuration' section shows details for the 'hr-prod-eks-ng-public' nodegroup, including Kubernetes version (1.22), AMI type (AL2_x86_64), AMI release version (1.22.9-20220629), instance types (t2.micro), and status (Create failed). The 'Details' section at the bottom provides further information: Node group ARN, Autoscaling group name, Capacity type (On-Demand), Desired size (1 node), Minimum size (1 node), Maximum size, Subnets, and SSH key pair (srevm-key).

us-east-1.console.aws.amazon.com/eks/home?region=us-east-1#/clusters/hr-prod-eks-ng-public

Bookmarks staff-14-jan-20pm...

aws Services Search for services, features, blogs, docs, and more [Alt+S]

Amazon Elastic Kubernetes Service

Clusters New

Related services

Amazon ECR Container storage for EKS

Documentation Submit feedback

EKS > Clusters > hr-prod-eks-prod > Node group: hr-prod-eks-ng-public

hr-prod-eks-ng-public

⌂ Edit Delete

Warning: Your current user or role does not have access to Kubernetes objects on this EKS nodegroup. This may be due to the current user or role not having Kubernetes RBAC permissions to describe cluster resources or not having an entry in the cluster's auth config map. [Learn more](#)

Node group configuration info

Kubernetes version 1.22	AMI type Info AL2_x86_64	Status ⛔ Create failed
AMI release version Info 1.22.9-20220629	Instance types t2.micro	Disk size 20 GiB

Details Nodes Health issues 2 Kubernetes labels Update config Kubernetes taints Update history Tags

Details

Node group ARN arn:aws:eks:us-east-1:143527018359:nodegroup/hr-prod-eks-prod/hr-prod-eks-ng-public/a6c11402-93d6-cf8a-d85b-8150f6ee2216 Created 7 hours ago	Autoscaling group name eks-hr-prod-eks-ng-public-a6c11402-93d6-cf8a-d85b-8150f6ee2216 Node IAM role ARN arn:aws:iam:143527018359:role/hr-prod-eks-nodegroup-role	Capacity type On-Demand Desired size 1 node Minimum size 1 node Maximum size	Subnets subnet-021122990508f84cb subnet-07dc71069b91ca8c5 Configure SSH access to nodes Enabled SSH key pair srevm-key
---	---	--	---

History on Ubuntu

```
ubuntu@ip-172-31-5-130:~$ history
1  ps
2  ps
3  sudo systemctl enable docker
4  sudo systemctl daemon-reload
5  sudo systemctl restart docker
6  docker network create jenkins
7  wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo gpg --dearmor -o /usr/share/keyrings/jenkins.gpg
8  sudo sh -c 'echo deb [signed-by=/usr/share/keyrings/jenkins.gpg] http://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'
9  sudo apt-get update
10 sudo apt-get install jenkins
11 sudo systemctl start jenkins.service
12 clear
13 sudo apt-get install default-jdk
14 java -version
15 sudo apt-get update
16 java -version
17 sudo apt-get install maven
18 sudo mkdir -p /usr/share/keyrings
19 curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo tee /usr/share/keyrings/jenkins-keyring.asc > /dev/null
20 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
21 sudo apt-get update
22 sudo apt-get install jenkins
23 systemctl status jenkins --no-pager -l
24 sudo systemctl enable --now jenkins
25 mv -v
26 mv -v
27 mv -v
28 java -version
29 sudo systemctl status jenkins
30 sudo ufw allow 8080
31 sudo cat /var/lib/jenkins/secrets/initialAdminPassword
32 pwd
33 aws ec2 describe-addresses
34 wget https://releases.hashicorp.com/terraform/1.2.5/terraform_1.2.5_linux_amd64.zip
35 ls
36 unzip terraform_1.2.5_linux_amd64.zip
37 ls
38 sudo apt-get update
39 sudo apt-get install zip
40 sudo apt-get install unzip
41 unzip terraform_1.2.5_linux_amd64.zip
42 ls
43 ls -l
44 ls -ltr
45 sudo mv terraform /usr/bin/
46 which terraform
47 terraform -v
48 terraform --version
49 docker images
50 docker create
51 sudo groupadd docker
52 sudo systemctl enable docker
53 sudo systemctl daemon-reload
54 sudo systemctl restart docker
55 grep docker /etc/group
56 docker ps
57 docker run hello-world
58 docker ps
```

Quick connect...

home/ubuntu/

Name

..

.cache

.ssh

.bash_logout

.bashrc

.profile

.Xauthority

Remote monitoring

☐ Follow terminal folder

8 ec2-3-234-251-180.compute-1.amazonaws.com

```
15 sudo apt-get update
16 java -version
17 sudo apt-get install maven
18 sudo mkdir -p /usr/share/keyrings
19 curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo tee /usr/share/keyrings/jenkins-keyring.asc > /dev/null
20 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
21 sudo apt-get update
22 sudo apt-get install jenkins
23 systemctl status jenkins --no-pager -l
24 sudo systemctl enable --now jenkins
25 mv -v
26 maven -v
27 maven -version
28 java -version
29 sudo systemctl status jenkins
30 sudo ufw allow 8080
31 sudo cat /var/lib/jenkins/secrets/initialAdminPassword
32 pwd
33 aws ec2 describe-addresses
34 wget https://releases.hashicorp.com/terraform/1.2.5/terraform_1.2.5_linux_amd64.zip
35 ls
36 unzip terraform_1.2.5_linux_amd64.zip
37 ls
38 sudo apt-get update
39 sudo apt-get install zip
40 sudo apt-get install unzip
41 unzip terraform_1.2.5_linux_amd64.zip
42 ls
43 ls -l
44 ls -ltr
45 sudo mv terraform /usr/bin/
46 which terraform
47 terraform -v
48 terraform --version
49 docker images
50 docker create
51 sudo groupadd docker
52 sudo systemctl enable docker
53 sudo systemctl daemon-reload
54 sudo systemctl restart docker
55 grep docker /etc/group
56 docker ps
57 docker run hello-world
58 docker ps
59 docker-compose up
60 sudo service jenkins restart
61 grep docker /etc/group
62 sudo ls -la /var/run/docker.sock
63 chmod 777 /var/run/docker.sock
64 sudo chmod 777 /var/run/docker.sock
65 sudo systemctl restart docker
66 docker images
67 history
ubuntu@ip-172-31-5-130:~$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
143527018359.dkr.ecr.us-east-1.amazonaws.com/my-app        latest             539d27fd536c       8 hours ago        54MB
hello-world          latest             feb5d9fea6a5       10 months ago      13.3kB
nginx                1.10.1-alpine      2cd900f340dd       5 years ago        54MB
```