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 rigth 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

1. 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{

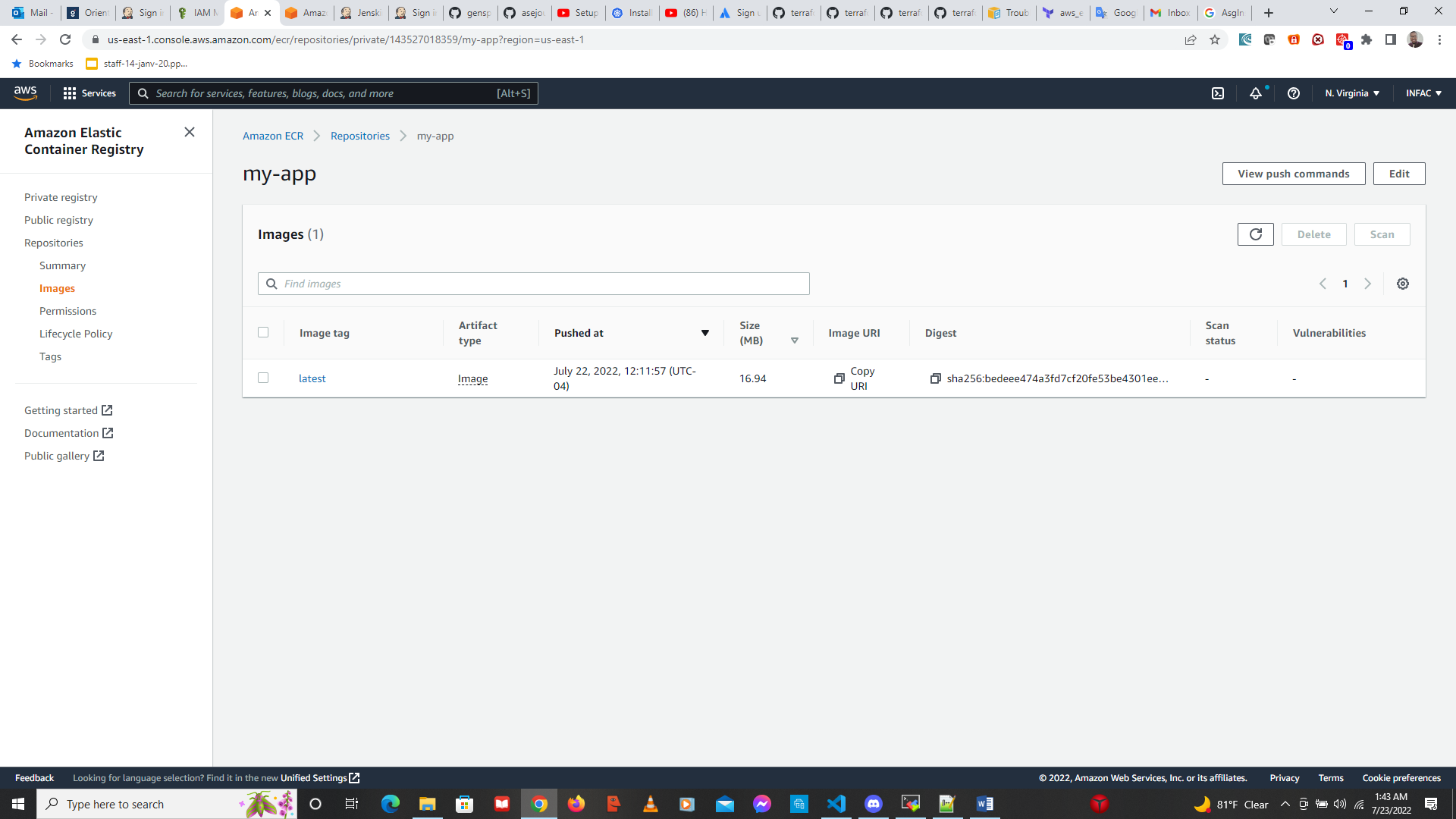
sh "docker push 143527018359.dkr.ecr.us-east-1.amazonaws.com/my-app:latest"

}

}

}

}



1. 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'){

steps{

sh "terraform apply --auto-approve"

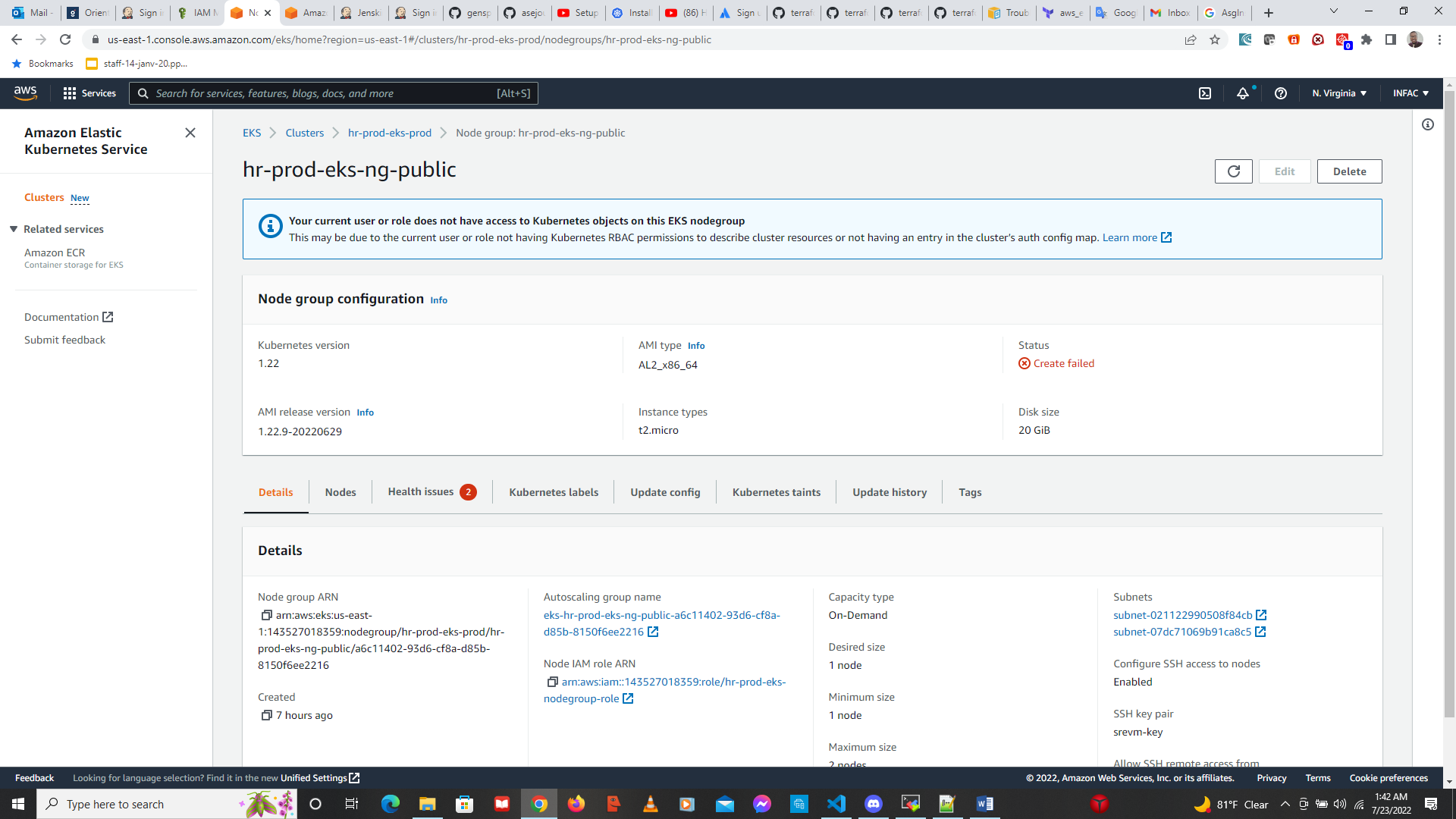
}

}

}

}

I got this error for the Node creation : AsgInstanceLaunchFailures: You've reached your quota for maximum Fleet Requests for this account. Launching EC2 instance failed.



History on Ubuntu

