**Documentação**

Realizado a criação de uma instância Amazon Linux 2 na AWS e um usuário IAM chamado terraform com permissão Administrator Full Acess e instalado o terraform, Docker, cliv2, git, kubectl e helm

**Repositorio do GIT onde o projeto foi armazenado**

https://github.com/vmayrink1/testeccr.git

**Instalação terraform**:

sudo yum install -y yum-utils

sudo yum-config-manager --add-repo https://rpm.releases.hashicorp.com/AmazonLinux/hashicorp.repo

sudo yum -y install terraform

**Instalação docker**:

sudo amazon-linux-extras install Docker

sudo systemctl docker start

sudo systemctl enable docker

**Instalação kubectl**:

curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.30.0/2024-05-12/bin/linux/amd64/kubectl

chmod +x ./kubectl

sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl

kubectl –version

**Instalação helm**:

helm repo add prometheus-community https://prometheus-community.github.io/helm-charts

yum install helm

curl -fsSL -o get\_helm.sh https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3

chmod 700 get\_helm.sh

**Instalação git**:

sudo yum install git

git –version

**Instalação AWSCLIV2:**

curl "https://awscli.amazonaws.com/awscli-exe-linux-x86\_64.zip" -o "awscliv2.zip"

unzip awscliv2.zip

sudo ./aws/install --bin-dir /usr/local/bin --install-dir /usr/local/aws-cli --update

aws --version

**Instalação do Cluster EKS 1.30 e Add-ons**

Terraform utilizado para subida do cluster EKS na versão 1.30 – módulo oficial

<https://github.com/terraform-aws-modules/terraform-aws-eks.git>

**Comandos realizados**

mkdir -p testeccr/eks

cd eks

git clone <https://github.com/terraform-aws-modules/terraform-aws-eks.git>

cd terraform-aws-eks/examples/eks-managed-node-group

vim main.tf – Editei o arquivo de configuração, ajustei o provider e a região.

vim eks-al2.tf – Editei o arquivo de configuração do eks e alterei o nome do cluster para saopaulofc e defini a instancia a ser utilizada como t3.micro no instance\_types

terraform init

terraform plan

terraform apply

aws eks --region us-east-1 update-kubeconfig --name saopaulofc

kubectl get nodes

kubectl get pods -A

Após subida do Cluster estava com problemas de acesso para gerenciar os nós, pods, editei o arquivo de configuração eks-al2.tf e adicionei os seguintes parâmetros:

cluster\_endpoint\_public\_access = true

enable\_cluster\_creator\_admin\_permissions = true

Rodei novamente os seguintes comandos:

terraform plan

terraform apply

E com isso consegui gerenciar o cluster

kubectl get pods -A | grep kube-system

Texto

Descrição gerada automaticamente

**Cluster AutoScaler e Metrics-Server**

Subida do Cluster **autoscaler** e o **metrics-server** via código terraform criado por mim e mais uma pessoa ao qual nomeamos o código de **coredns** ou **core-apps** onde ficam os seguintes add-ons: **kube\_proxy, coredns, vpc\_cni, aws\_load\_balancer\_controller, autoscaler, metrics\_server, nginx\_controler, kube\_dashboard e kubecost**

**Comandos Realizados:**

cd eks/coredns/my-eks-core-apps

git clone <https://github.com/Anderson-Ferreira-Costa/my-eks-core-apps.git>

vim main.tf – Editei o arquivo de configuração e alterei o nome do cluster e a região e também o parâmetro de false para true no **autoscaler** e **metrics-server**

terraform init

terraform plan

terraform apply

**Instalação Ingress-nginx-controller**

**Comandos Realizados:**

helm repo add ingress-nginx https://kubernetes.github.io/ingress-nginx

helm install nginx-ingress ingress-nginx/ingress-nginx

kubectl apply -f <https://raw.githubusercontent.com/nginxinc/kubernetes-ingress/v3.5.2/deploy/crds.yaml>

**Instalação do KubeStack**

**Comandos Realizados:**

mkdir -p /home/ec2-user/testeccr/eks/terraform-aws-eks/examples/kube-stack

cd /home/ec2-user/testeccr/eks/terraform-aws-eks/examples/kube-stack

helm repo add prometheus-community <https://prometheus-community.github.io/helm-charts>

helm install prometheus-stack prometheus-community/kube-prometheus-stack

kubectl create ns monitoring

helm install prometheus-stack prometheus-community/kube-prometheus-stack -n monitoring -f prometheus.yaml

vim prometheus.yaml

helm upgrade prometheus-stack prometheus-community/kube-prometheus-stack -n monitoring -f prometheus.yaml

kubectl get secrets -n monitoring grafana-new --template='{{ index .data "admin-password" }}' | base64 -d

usuário: admin

senha: prom-operator

**Instalação do ArgoCD**

**Comandos Realizados:**

mkdir -p /home/ec2-user/testeccr/argocd/

cd /home/ec2-user/testeccr/argocd/

kubectl create namespace argocd  
kubectl apply -n argocd -f <https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/install.yaml>

vim argocd.yaml – Ajustar as configurações.

vim argocd-ingress.yaml – Criação do Ingress do Argocd

kubectl apply -f argocd-ingress.yaml

kubectl get secret argocd-initial-admin-secret -n argocd -o jsonpath="{.data.password}" | base64 -d

Usuario: admin

Senha: ffmlDAMnqvl-FR-M

**FLASK**

Foi realizado a criação de um repositório privado na aws chamado de flask

Foi realizado a criação de três arquivos para o build o app.py, dockerfile e o requirements.txt

Ao rodar o docker build tomei erro no requirements.txt e pesquisando no stackoverflow encontrei que a solução era adicionar o parâmetro de uma dependencia

# Werkzeug

Werkzeug==2.2.2

Após este ajuste rodei o comando docker build -t flask .

**Comandos Realizados para subir a imagem no repositório**

aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin 608420491255.dkr.ecr.us-east-1.amazonaws.com

docker tag 4c652e04d083 608420491255.dkr.ecr.us-east-1.amazonaws.com/flask:v1

docker image ls

docker push 608420491255.dkr.ecr.us-east-1.amazonaws.com/flask:v1

**Comandos Realizados para criação do ingress e deploy do Flask**

kubectl create deploy flask --image 608420491255.dkr.ecr.us-east-1.amazonaws.com/flask:v1

kubectl expose --name svc-flask deploy flask --port 80

kubectl describe svc svc-flask

cd /home/ec2-user/flask/

vim flask-ingress.yaml

kubectl apply -f flask-ingress.yaml

**Criação dos arquivos de manifesto Deployment e Service do Flask**

**Deployment**

kubectl get deploy -o yaml flask

**Service**

kubectl get svc -o yaml flask

**Realizado a criação de um domínio privado manualmente chamado de Mayrink.local**

E criado as seguintes hosted zones:

argocd.mayrink.local CNAME ad8ee76817ec74ea7b4cadcea7cc52ad-2100129322.us-east-1.elb.amazonaws.com

flask.mayrink.local CNAME a827d83c5869d4b27ae3d4f6e6502159-948530912.us-east-1.elb.amazonaws.com

grafana.mayrink.local CNAME a827d83c5869d4b27ae3d4f6e6502159-948530912.us-east-1.elb.amazonaws.com

**Foi realizado a configuração do ARGOCD para atualizar automaticamente o deploy.**