PIN DEVOPS 2401

Grupo 01
Daniel Bertoni
Emiliano Palacio

Repositorio - https://github.com/palacioea/Pin-2024

Contenido

Enfoque de la Solución

Vamos a desplegar el EC2 desde un pc de salto vía Terraform, además desplegaremos todas las herramientas que se solicitan para poder montar el EKS con un script en Shell para automatizar el despliegue.

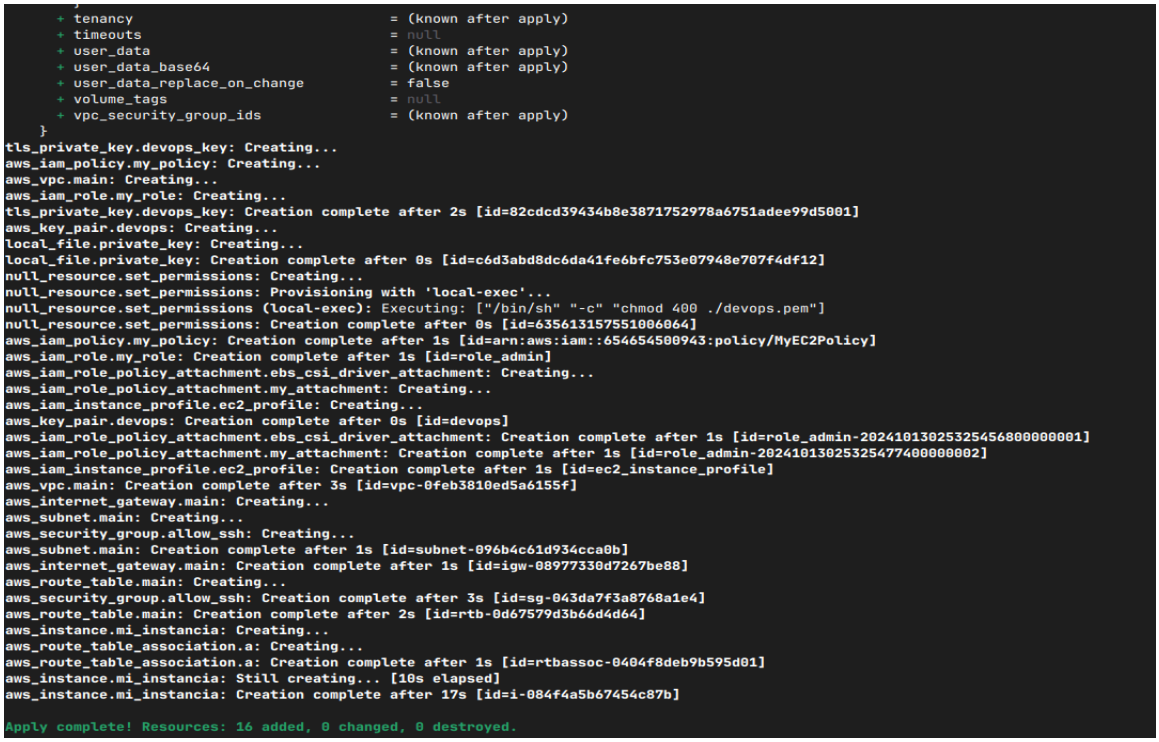
Una vez implementado el EC2, procederemos a levantar el EKS desde línea de comandos desde el bastión previamente desplegado.

Terminada todas las capturas de pantalla, se procederá a borrar el EKS desde el bastion y luego con Terraform se destruirá el bastión construido.

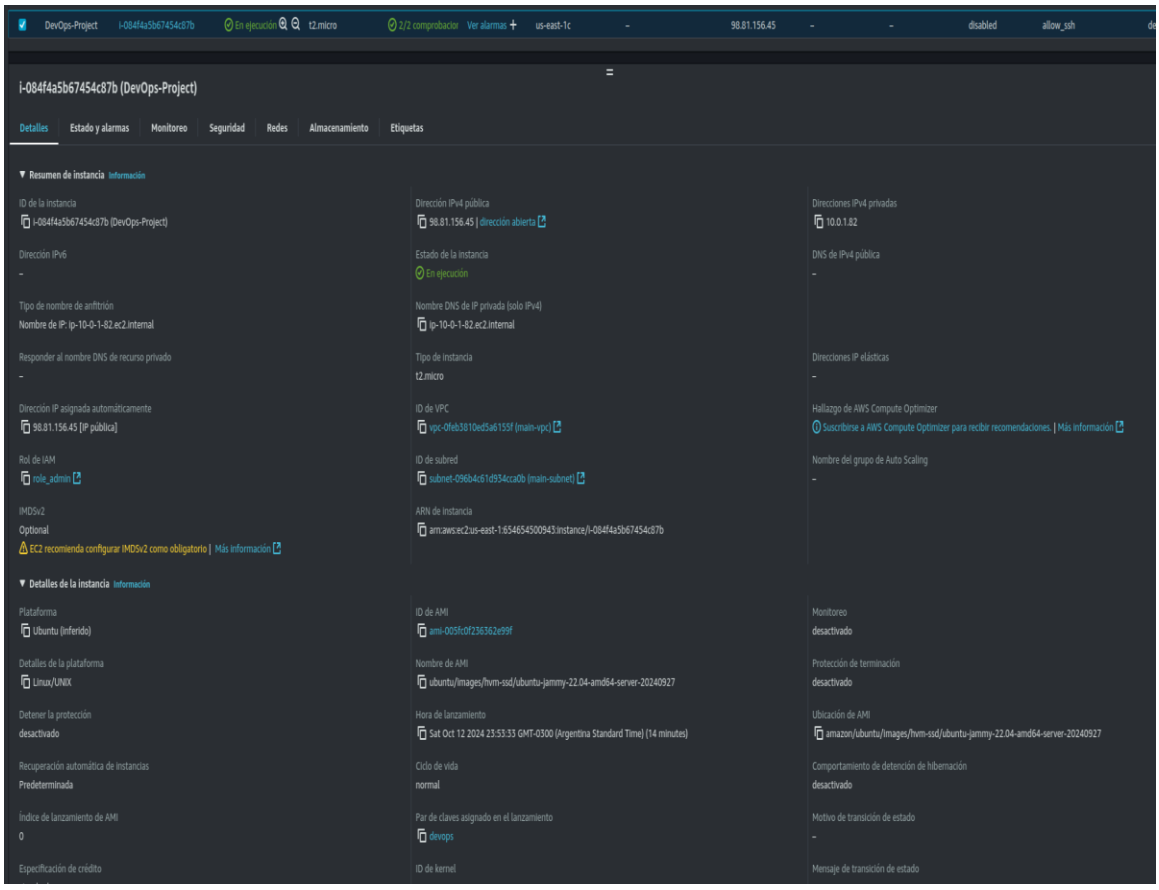
Además, utilizaremos una herramienta Cloud-Nuke para poder eliminar también todos los demás recursos que se hayan creado si fuese necesario por otras vías.

Creación de la Instancia EC2 en AWS

Utilizamos Terraform para aprovisionar una instancia EC2 en AWS. En el proceso, se generaron 16 recursos, incluyendo la creación del par de claves (KEY_PAIR) necesario para establecer la conexión SSH. Verificamos exitosamente la creación de la instancia a través de la consola de AWS.

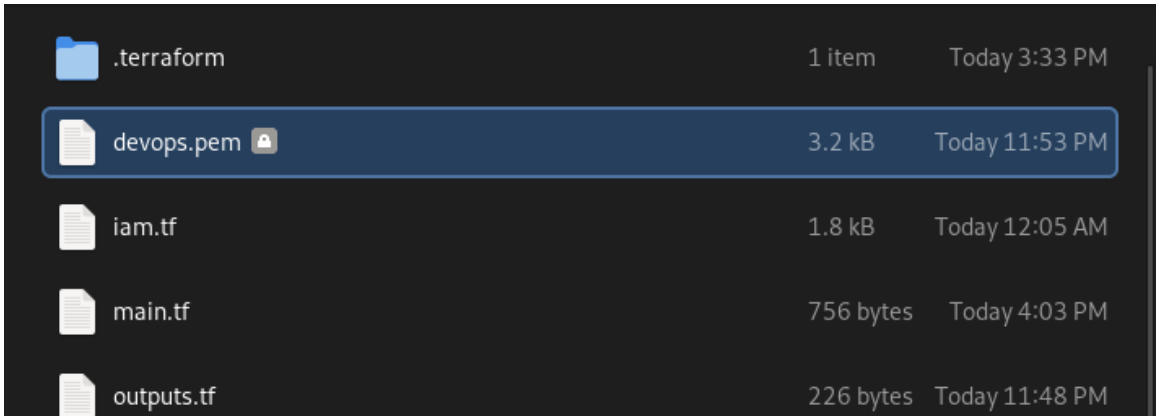


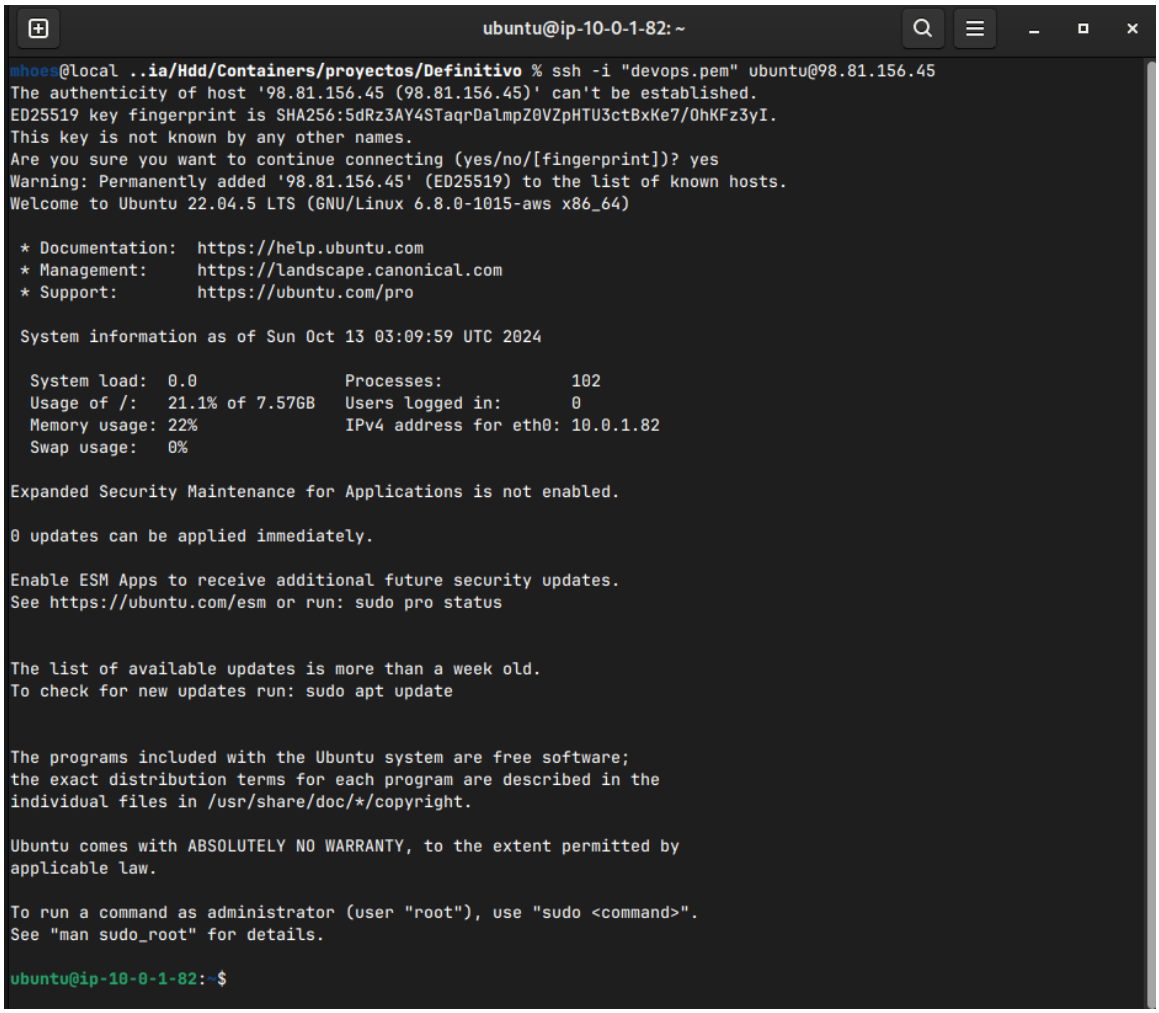
```hcl
# Generar una nueva clave privada
resource "tls_private_key" "devops_key" {
  algorithm = "RSA"
  rsa_bits  = 4096
}

# Crear un Key Pair en AWS usando la clave generada
resource "aws_key_pair" "devops" {
  key_name   = "devops"
  public_key = tls_private_key.devops_key.public_key_openssh
}

# Opcional: Guardar la clave privada en un archivo local
resource "local_file" "private_key" {
  content  = tls_private_key.devops_key.private_key_pem
  filename = "${path.module}/devops.pem"
}

resource "null_resource" "set_permissions" {
  depends_on = [local_file.private_key]

  provisioner "local-exec" {
    command = "chmod 400 ${local_file.private_key.filename}"
  }
}
```



Acceso SSH a la Instancia

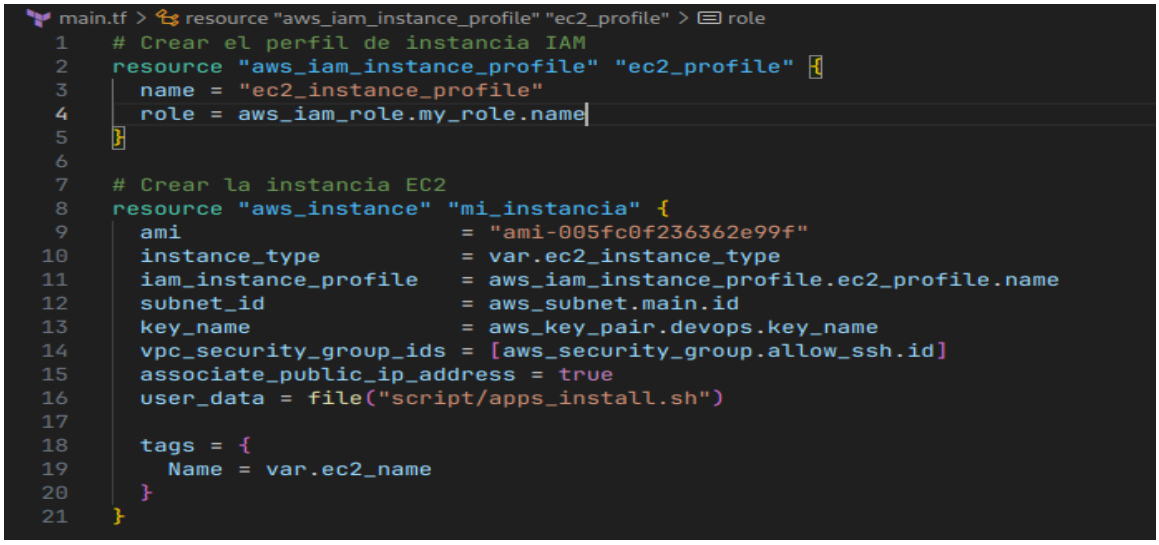
La clave PEM se descargó oportunamente, se logró la conexión SSH a la instancia de forma exitosa.





Instalación de Aplicaciones

Se creó un script que permite la instalación de todas las aplicaciones requeridas que se aprovisiono directamente desde Terraform.



Creación del Cluster EKS

Iniciamos el proceso de creación del cluster EKS utilizando el siguiente comando:

```
eksctl create cluster \
  --name eks-mundos-e \
  --region us-east-1 \
  --with-oidc \
  --nodegroup-name testv3 \
  --node-type t2.large \
  --nodes 1 \
  --nodes-min 1 \
  --nodes-max 3 \
  --node-volume-size 20 \
  --ssh-access=false \
  --managed \
  --asg-access \
  --external-dns-access \
  --full-ecr-access \
  --appmesh-access \
  --alb-ingress-access
```

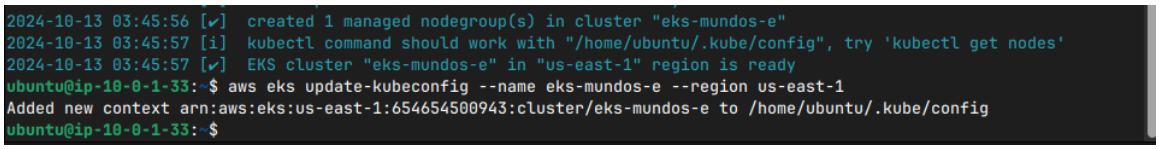
```
ubuntu@ip-10-0-1-33:~$ eksctl create cluster \
  --name eks-mundos-e \
  --region us-east-1 \
  --with-oidc \
  --nodegroup-name testv3 \
  --node-type t2.large \
  --nodes 1 \
  --nodes-min 1 \
  --nodes-max 3 \
  --node-volume-size 20 \
  --ssh-access=false \
  --managed \
  --asg-access \
  --external-dns-access \
  --full-ecr-access \
  --appmesh-access \
  --alb-ingress-access
2024-10-13 03:33:27 [i]  eksctl version 0.192.0
2024-10-13 03:33:27 [i]  using region us-east-1
2024-10-13 03:33:27 [i]  setting availability zones to [us-east-1c us-east-1b]
2024-10-13 03:33:27 [i]  subnets for us-east-1c - public:192.168.0.0/19 private:192.168.64.0/19
2024-10-13 03:33:27 [i]  subnets for us-east-1b - public:192.168.32.0/19 private:192.168.96.0/19
2024-10-13 03:33:27 [i]  nodegroup "testv3" will use "" [AmazonLinux2/1.30]
2024-10-13 03:33:27 [i]  using Kubernetes version 1.30
2024-10-13 03:33:27 [i]  creating EKS cluster "eks-mundos-e" in "us-east-1" region with managed nodes
2024-10-13 03:33:27 [i]  will create 2 separate CloudFormation stacks for cluster itself and the initial managed nodeg
roup
2024-10-13 03:33:27 [i]  if you encounter any issues, check CloudFormation console or try 'eksctl utils describe-stack
s --region=us-east-1 --cluster=eks-mundos-e'
2024-10-13 03:33:27 [i]  Kubernetes API endpoint access will use default of {publicAccess=true, privateAccess=false} f
or cluster "eks-mundos-e" in "us-east-1"
2024-10-13 03:33:27 [i]  CloudWatch logging will not be enabled for cluster "eks-mundos-e" in "us-east-1"
2024-10-13 03:33:27 [i]  you can enable it with 'eksctl utils update-cluster-logging --enable-types={SPECIFY-YOUR-LOG-
TYPES-HERE (e.g. all)} --region=us-east-1 --cluster=eks-mundos-e'
2024-10-13 03:33:27 [i]  default addons vpc-cni, kube-proxy, coredns were not specified, will install them as EKS addo
ns
2024-10-13 03:33:27 [i]
2 sequential tasks: { create cluster control plane "eks-mundos-e",
    2 sequential sub-tasks: {
        5 sequential sub-tasks: {
            1 task: { create addons },
            wait for control plane to become ready,
            associate IAM OIDC provider,
            no tasks,
            update VPC CNI to use IRSA if required,
        },
        create managed nodegroup "testv3",
    }
}
2024-10-13 03:33:27 [i]  building cluster stack "eksctl-eks-mundos-e-cluster"
2024-10-13 03:33:28 [i]  deploying stack "eksctl-eks-mundos-e-cluster"
```

```
2024-10-13 03:40:29 [i]  creating addon
2024-10-13 03:40:29 [i]  successfully created addon
2024-10-13 03:40:30 [i]  creating addon
2024-10-13 03:40:30 [i]  successfully created addon
2024-10-13 03:40:30 [i]  creating addon
2024-10-13 03:40:31 [i]  successfully created addon
2024-10-13 03:42:32 [i]  deploying stack "eksctl-eks-mundos-e-addon-vpc-cni"
2024-10-13 03:42:32 [i]  waiting for CloudFormation stack "eksctl-eks-mundos-e-addon-vpc-cni"
2024-10-13 03:43:02 [i]  waiting for CloudFormation stack "eksctl-eks-mundos-e-addon-vpc-cni"
2024-10-13 03:43:02 [i]  updating addon
2024-10-13 03:43:12 [i]  addon "vpc-cni" active
2024-10-13 03:43:13 [i]  building managed nodegroup stack "eksctl-eks-mundos-e-nodegroup-testv3"
2024-10-13 03:43:13 [i]  deploying stack "eksctl-eks-mundos-e-nodegroup-testv3"
2024-10-13 03:43:13 [i]  waiting for CloudFormation stack "eksctl-eks-mundos-e-nodegroup-testv3"
2024-10-13 03:43:43 [i]  waiting for CloudFormation stack "eksctl-eks-mundos-e-nodegroup-testv3"
2024-10-13 03:44:25 [i]  waiting for CloudFormation stack "eksctl-eks-mundos-e-nodegroup-testv3"
2024-10-13 03:45:55 [i]  waiting for CloudFormation stack "eksctl-eks-mundos-e-nodegroup-testv3"
2024-10-13 03:45:55 [i]  waiting for the control plane to become ready
2024-10-13 03:45:56 [✔]  saved kubeconfig as "/home/ubuntu/.kube/config"
2024-10-13 03:45:56 [i]  no tasks
2024-10-13 03:45:56 [✔]  all EKS cluster resources for "eks-mundos-e" have been created
2024-10-13 03:45:56 [✔]  created 0 nodegroup(s) in cluster "eks-mundos-e"
2024-10-13 03:45:56 [i]  nodegroup "testv3" has 1 node(s)
2024-10-13 03:45:56 [i]  node "ip-192-168-9-221.ec2.internal" is ready
2024-10-13 03:45:56 [i]  waiting for at least 1 node(s) to become ready in "testv3"
2024-10-13 03:45:56 [i]  nodegroup "testv3" has 1 node(s)
2024-10-13 03:45:56 [i]  node "ip-192-168-9-221.ec2.internal" is ready
2024-10-13 03:45:56 [✔]  created 1 managed nodegroup(s) in cluster "eks-mundos-e"
2024-10-13 03:45:57 [i]  kubectl command should work with "/home/ubuntu/.kube/config", try 'kubectl get nodes'
2024-10-13 03:45:57 [✔]  EKS cluster "eks-mundos-e" in "us-east-1" region is ready
ubuntu@ip-10-0-1-33:~$
```

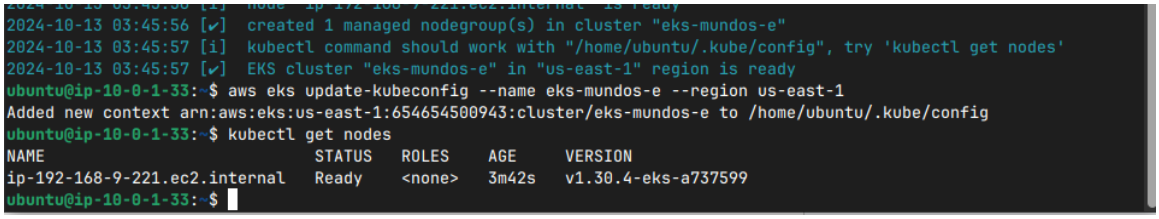
Configuración de kubectl

Configuramos `kubectl` para interactuar con el cluster EKS recién creado utilizando el siguiente comando:

```
aws eks update-kubeconfig --name eks-mundos-e --region us-east-1
```



Verificación de la conexión con el cluster



Instalación del EBS Driver

Para habilitar el soporte de volúmenes EBS, aplicamos el controlador EBS driver con los siguientes comandos:

```
kubectl apply -k "github.com/kubernetes-sigs/aws-ebs-csi-driver/deploy/kubernetes/overlays/stable/?ref=release-1.35
eksctl create iamserviceaccount \
        --name ebs-csi-controller-sa \
        --region us-east-1 \
        --namespace kube-system \
        --cluster eks-mundos-e \
        --attach-policy-arn arn:aws:iam::aws:policy/service-role/AmazonEBSCSIDriverPolicy \
        --approve \
        --role-only \
         --role-name AmazonEKS_EBS_CSI_DriverRole
eksctl create addon \
        --name aws-ebs-csi-driver
        --cluster eks-mundos-e
        --service-account-role-arn arn:aws:iam::xxxxxxxxxx:role/AmazonEKS_EBS_CSI_DriverRole
        --force
```

```
ubuntu@ip-10-0-1-33:~$ kubectl apply -k "github.com/kubernetes-sigs/aws-ebs-csi-driver/deploy/kubernetes/overlays/stab
le/?ref=release-1.35"
serviceaccount/ebs-csi-controller-sa created
serviceaccount/ebs-csi-node-sa created
role.rbac.authorization.k8s.io/ebs-csi-leases-role created
clusterrole.rbac.authorization.k8s.io/ebs-csi-node-role created
clusterrole.rbac.authorization.k8s.io/ebs-external-attacher-role created
clusterrole.rbac.authorization.k8s.io/ebs-external-provisioner-role created
clusterrole.rbac.authorization.k8s.io/ebs-external-resizer-role created
clusterrole.rbac.authorization.k8s.io/ebs-external-snapshotter-role created
rolebinding.rbac.authorization.k8s.io/ebs-csi-leases-rolebinding created
clusterrolebinding.rbac.authorization.k8s.io/ebs-csi-attacher-binding created
clusterrolebinding.rbac.authorization.k8s.io/ebs-csi-node-getter-binding created
clusterrolebinding.rbac.authorization.k8s.io/ebs-csi-provisioner-binding created
clusterrolebinding.rbac.authorization.k8s.io/ebs-csi-resizer-binding created
clusterrolebinding.rbac.authorization.k8s.io/ebs-csi-snapshotter-binding created
deployment.apps/ebs-csi-controller created
poddisruptionbudget.policy/ebs-csi-controller created
daemonset.apps/ebs-csi-node created
csidriver.storage.k8s.io/ebs.csi.aws.com created
ubuntu@ip-10-0-1-33:~$
```

```
^C
ubuntu@ip-10-0-1-33:~$ eksctl create iamserviceaccount \
 --name ebs-csi-controller-sa \
 --region us-east-1 \
 --namespace kube-system \
 --cluster eks-mundos-e \
 --attach-policy-arn arn:aws:iam::aws:policy/service-role/AmazonEBSCSIDriverPolicy \
 --approve \
 --role-only \
 --role-name AmazonEKS_EBS_CSI_DriverRole
2024-10-13 03:59:26 [i]  1 iamserviceaccount (kube-system/ebs-csi-controller-sa) was included (based on the include/ex
clude rules)
2024-10-13 03:59:26 [!]  serviceaccounts in Kubernetes will not be created or modified, since the option --role-only i
s used
2024-10-13 03:59:26 [i]  1 task: { create IAM role for serviceaccount "kube-system/ebs-csi-controller-sa" }
2024-10-13 03:59:26 [i]  building iamserviceaccount stack "eksctl-eks-mundos-e-addon-iamserviceaccount-kube-system-ebs
-csi-controller-sa"
2024-10-13 03:59:26 [i]  deploying stack "eksctl-eks-mundos-e-addon-iamserviceaccount-kube-system-ebs-csi-controller-s
a"
2024-10-13 03:59:26 [i]  waiting for CloudFormation stack "eksctl-eks-mundos-e-addon-iamserviceaccount-kube-system-ebs
-csi-controller-sa"
2024-10-13 03:59:57 [i]  waiting for CloudFormation stack "eksctl-eks-mundos-e-addon-iamserviceaccount-kube-system-ebs
-csi-controller-sa"
ubuntu@ip-10-0-1-33:~$
```

```
ubuntu@ip-10-0-1-33:~$  eksctl create addon --name aws-ebs-csi-driver --region us-east-1 --cluster eks-mundos-e --service-account-role-arn arn:aws:iam::654654500943:role/AmazonEKS_EBS_CSI_DriverRole --force
2024-10-13 04:05:15 [i]  Kubernetes version "1.30" in use by cluster "eks-mundos-e"
2024-10-13 04:05:15 [i]  IRSA is set for "aws-ebs-csi-driver" addon; will use this to configure IAM permissions
2024-10-13 04:05:15 [!]  IRSA has been deprecated; the recommended way to provide IAM permissions for "aws-ebs-csi-driver" addon is via pod identity associations; after addon creation is completed, run `eksctl utils migrate-to-pod-identity`
2024-10-13 04:05:15 [i]  using provided ServiceAccountRoleARN "arn:aws:iam::654654500943:role/AmazonEKS_EBS_CSI_DriverRole"
2024-10-13 04:05:15 [i]  creating addon
ubuntu@ip-10-0-1-33:~$
```

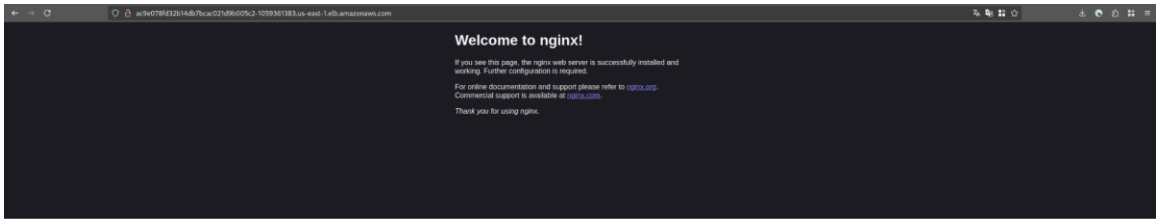
Despliegue de Nginx

Para desplegar Nginx, aplicamos el manifiesto `nginx.yaml` usando el comando:

```
kubectl apply -f nginx.yaml
```

```
ubuntu@ip-10-0-1-33:~$ kubectl get all
NAME                                    READY   STATUS    RESTARTS   AGE
pod/nginx-deployment-576c6b7b6-6w57l    1/1     Running   0          2m9s
pod/nginx-deployment-576c6b7b6-tlcmb    1/1     Running   0          2m9s

NAME                    TYPE           CLUSTER-IP      EXTERNAL-IP                                                                PORT(S)        AGE
service/kubernetes      ClusterIP      10.100.0.1      <none>                                                                     443/TCP        22m
service/nginx-service   LoadBalancer   10.100.43.237   ac9e078fd32b14db7bcac021d9b005c2-1059361383.us-east-1.elb.amazonaws.com   80:31797/TCP   2m9s

NAME                               READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/nginx-deployment   2/2     2            2           2m9s

NAME                                          DESIRED   CURRENT   READY   AGE
replicaset.apps/nginx-deployment-576c6b7b6    2         2         2       2m9s
```

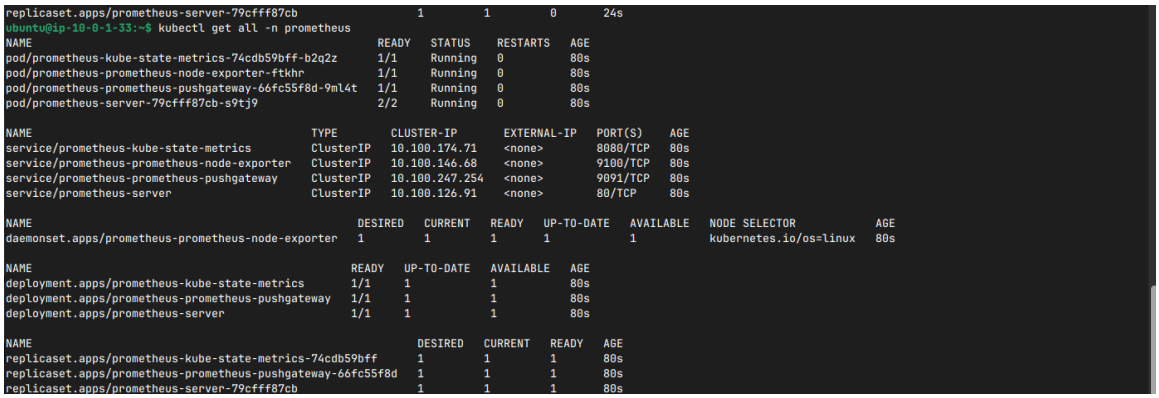
Luego, verificamos el acceso a Nginx desde el navegador.



Configuración de Helm y Despliegue de Prometheus

Agregamos los repositorios de Prometheus utilizando Helm:

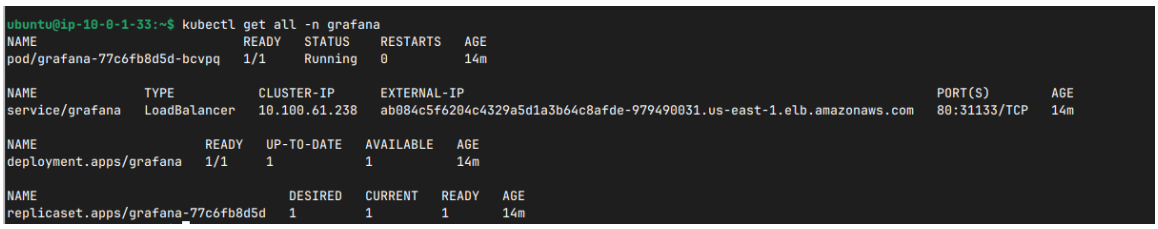
```
helm repo add prometheus-community https://prometheus-community.github.io/helm-charts
helm repo update
kubectl create namespace prometheus
helm install prometheus prometheus-community/prometheus \
        --namespace prometheus \
        --set alertmanager.persistentVolume.storageClass="gp2" \
        --set server.persistentVolume.storageClass="gp2"
kubectl patch svc prometheus-server -n prometheus -p '{"spec": {"type": "NodePort"}}'
```



Instalación de Grafana

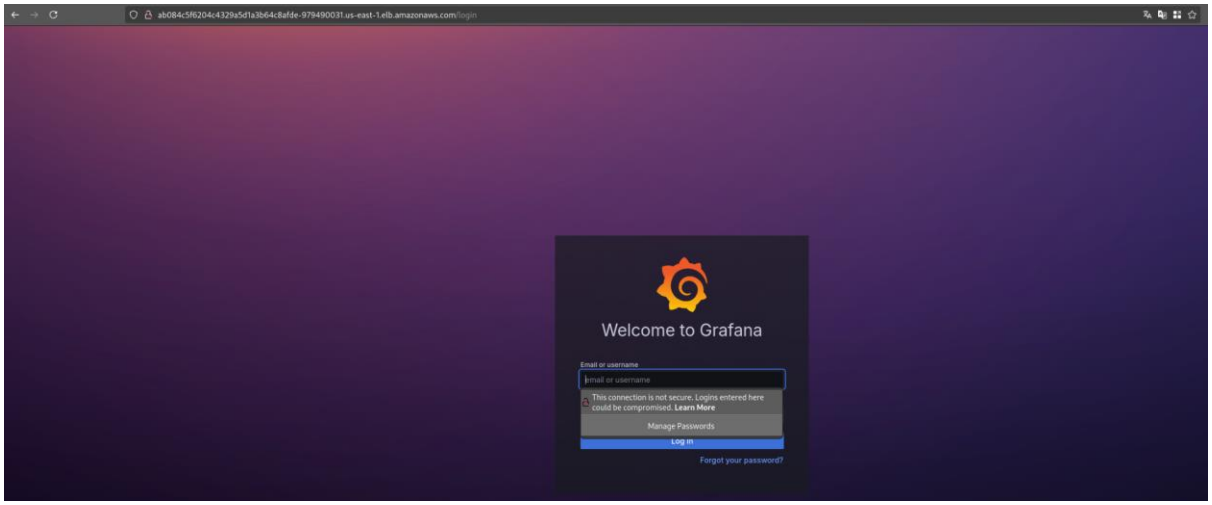
Instalamos Grafana utilizando Helm, creando previamente el namespace:

```
kubectl create namespace grafana
helm install grafana grafana/grafana \
        --namespace grafana \
        --set adminPassword='EKS!sAWSome' \
        --values grafana.yaml \
        --set service.type=LoadBalancer
```

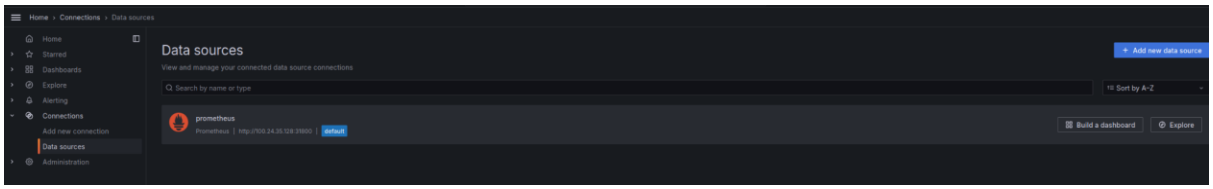


Monitoreo

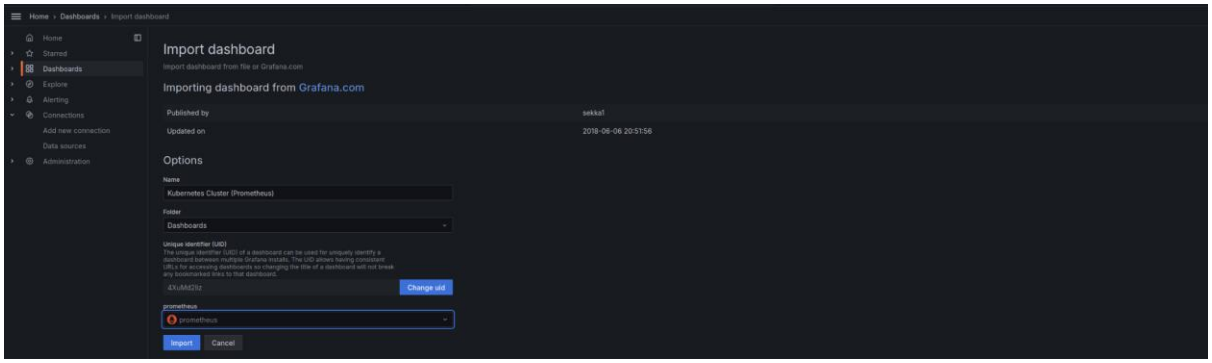
Accedemos al external-ip

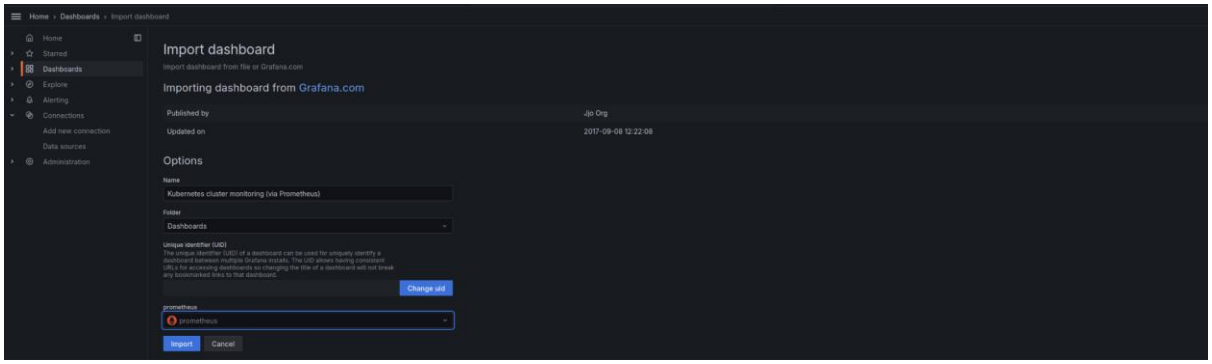


Agregamos Prometheus como datasource:

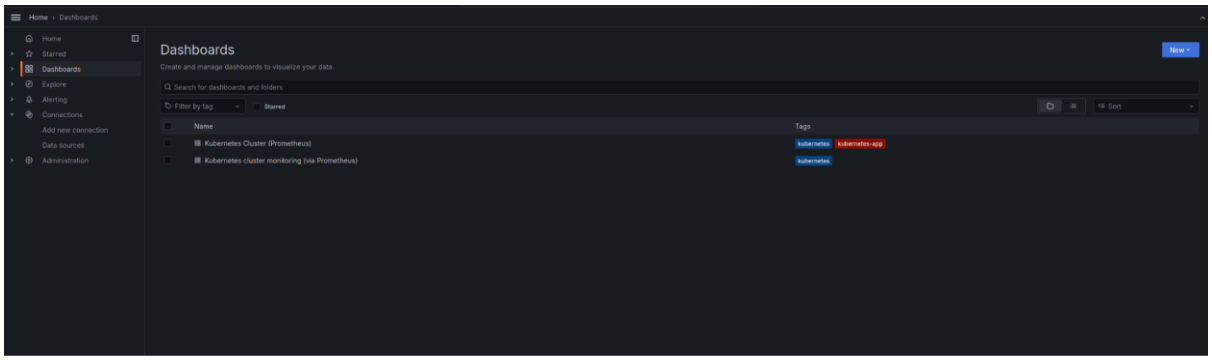


importamos los dashboards 6417 y 3119 en la carpeta Dashboard y confiugramos el prometheus que creamos en el datasource.

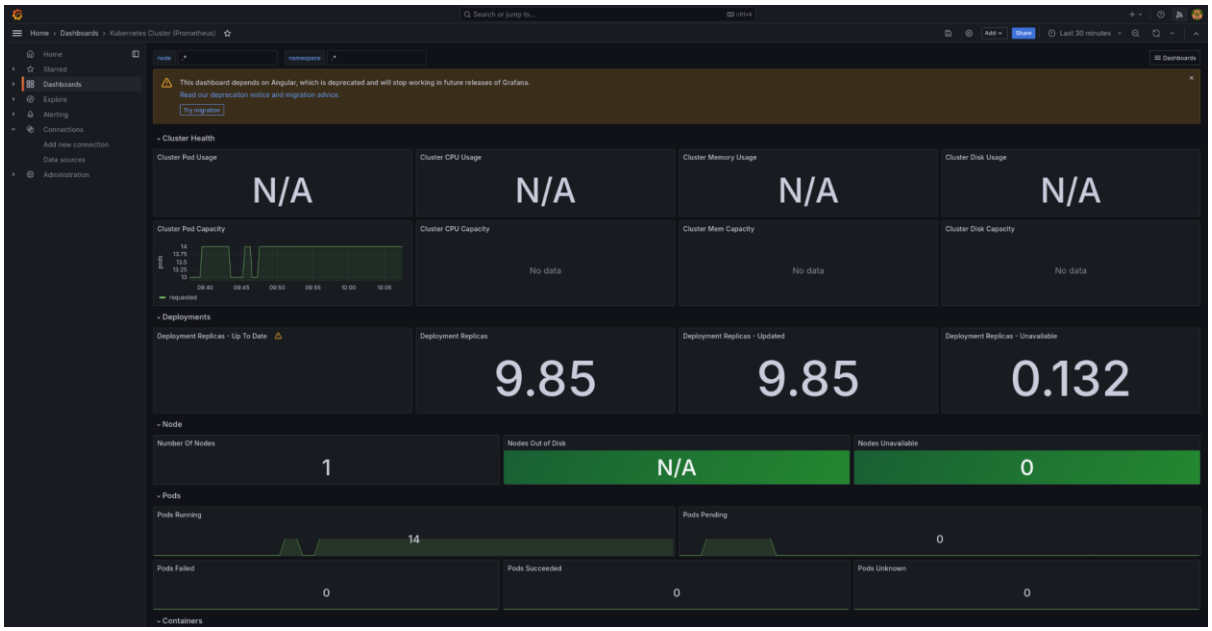




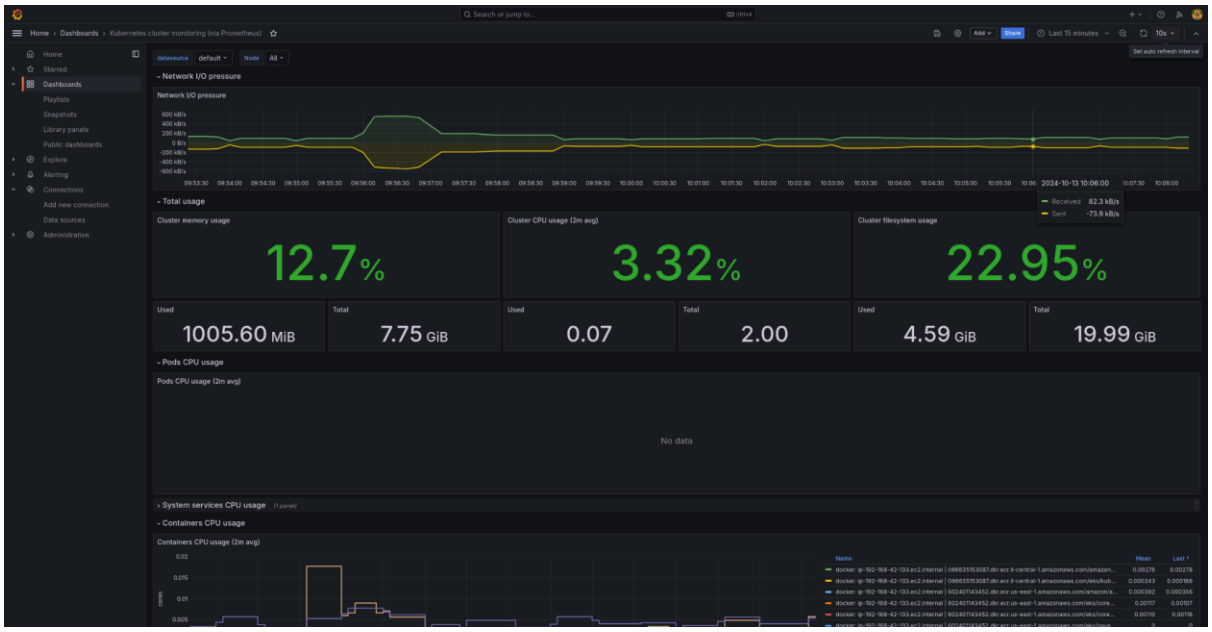
Verificamos importación de ambos



ID 6417:



ID 3119:



Limpieza de Recursos

Procedimos a limpiar todos los recursos desplegados para evitar costos innecesarios.

Desde el bastion (EC2)

```
helm uninstall prometheus --namespace prometheus
kubectl delete ns prometheus
helm uninstall grafana --namespace grafana
eksctl delete cluster --name mundos-e --region us-east-1
```

```
4 sequential tasks: { delete nodegroup "mundose",
    2 sequential sub-tasks: {
        2 sequential sub-tasks: {
            delete IAM role for serviceaccount "kube-system/ebs-csi-controller-sa",
            delete serviceaccount "kube-system/ebs-csi-controller-sa",
        },
        delete IAM OIDC provider,
    }, delete addon IAM "eksctl-mundos-e-addon-vpc-cni", delete cluster control plane "mundos-e" [async]
}
2024-10-13 13:14:29 [i]  will delete stack "eksctl-mundos-e-nodegroup-mundose"
2024-10-13 13:14:29 [i]  waiting for stack "eksctl-mundos-e-nodegroup-mundose" to get deleted
2024-10-13 13:14:29 [i]  waiting for CloudFormation stack "eksctl-mundos-e-nodegroup-mundose"
2024-10-13 13:14:59 [i]  waiting for CloudFormation stack "eksctl-mundos-e-nodegroup-mundose"
2024-10-13 13:15:33 [i]  waiting for CloudFormation stack "eksctl-mundos-e-nodegroup-mundose"
2024-10-13 13:16:37 [i]  waiting for CloudFormation stack "eksctl-mundos-e-nodegroup-mundose"
2024-10-13 13:17:12 [i]  waiting for CloudFormation stack "eksctl-mundos-e-nodegroup-mundose"
2024-10-13 13:18:23 [i]  waiting for CloudFormation stack "eksctl-mundos-e-nodegroup-mundose"
2024-10-13 13:20:12 [i]  waiting for CloudFormation stack "eksctl-mundos-e-nodegroup-mundose"
2024-10-13 13:21:03 [i]  waiting for CloudFormation stack "eksctl-mundos-e-nodegroup-mundose"
2024-10-13 13:21:50 [i]  waiting for CloudFormation stack "eksctl-mundos-e-nodegroup-mundose"
2024-10-13 13:21:51 [i]  will delete stack "eksctl-mundos-e-addon-iamserviceaccount-kube-system-ebs-csi-controller-sa"
2024-10-13 13:21:51 [i]  waiting for stack "eksctl-mundos-e-addon-iamserviceaccount-kube-system-ebs-csi-controller-sa" to get deleted
2024-10-13 13:21:51 [i]  waiting for CloudFormation stack "eksctl-mundos-e-addon-iamserviceaccount-kube-system-ebs-csi-controller-sa"
2024-10-13 13:22:21 [i]  waiting for CloudFormation stack "eksctl-mundos-e-addon-iamserviceaccount-kube-system-ebs-csi-controller-sa"
2024-10-13 13:22:21 [i]  serviceaccount "kube-system/ebs-csi-controller-sa" was not created by eksctl; will not be deleted
2024-10-13 13:22:21 [i]  will delete stack "eksctl-mundos-e-addon-vpc-cni"
2024-10-13 13:22:21 [i]  will delete stack "eksctl-mundos-e-cluster"
2024-10-13 13:22:21 [✔]  all cluster resources were deleted
ubuntu@ip-10-0-1-33:~$
```

Desde el pc de salto:

terraform destroy -auto-approve

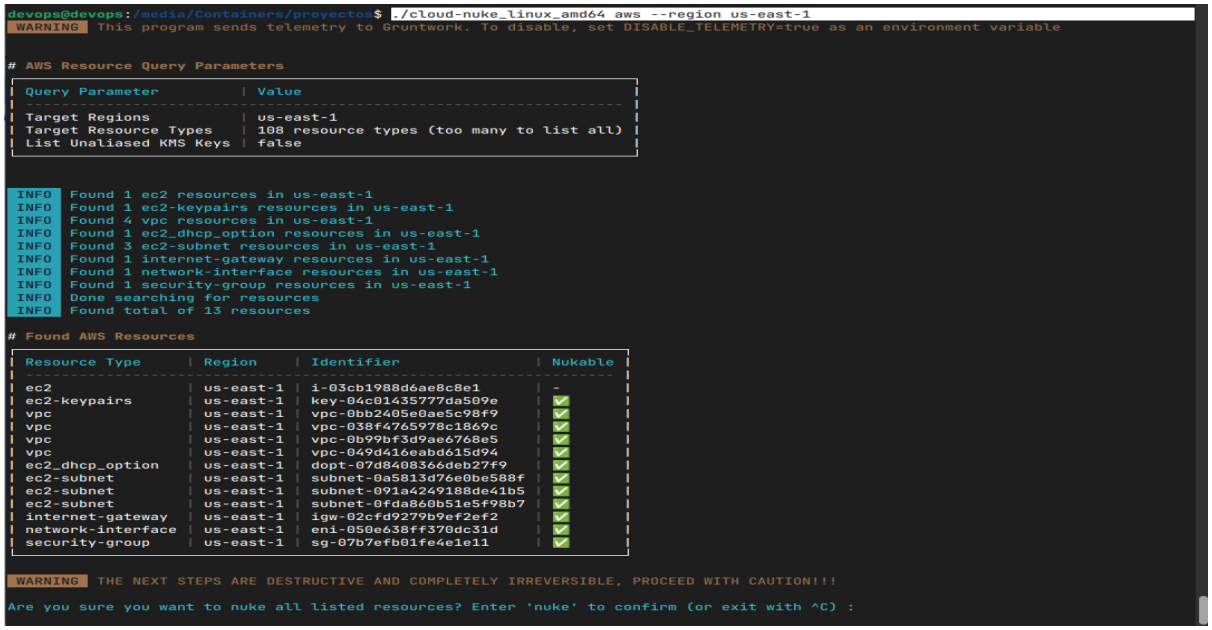
```
}
devops@devops:/media/Containers/proyectos/Definitivo$ terraform destroy -auto-approve
tls_private_key.devops_key: Refreshing state... [id=095c942864d7376c16a0104c31ae62c70dc3caa2]
local_file.private_key: Refreshing state... [id=4f3d84b57fea148435a3cd64adfeae2200d13afb]
aws_key_pair.devops: Refreshing state... [id=devops]
aws_iam_policy.my_policy: Refreshing state... [id=arn:aws:iam::654654500943:policy/MyEC2Policy]
aws_vpc.main: Refreshing state... [id=vpc-038f4765978c1869c]
aws_iam_role.my_role: Refreshing state... [id=role_admin]
aws_security_group.allow_ssh: Refreshing state... [id=sg-07b7efb01fe4e1e11]
aws_internet_gateway.main: Refreshing state... [id=igw-02cfd9279b9ef2ef2]
aws_subnet.main: Refreshing state... [id=subnet-091a4249188de41b5]
aws_iam_role_policy_attachment.ebs_csi_driver_attachment: Refreshing state... [id=role_admin-20241013065551885100000001]
aws_iam_role_policy_attachment.my_attachment: Refreshing state... [id=role_admin-20241013065551890400000002]
aws_iam_instance_profile.ec2_profile: Refreshing state... [id=ec2_instance_profile]
aws_route_table.main: Refreshing state... [id=rtb-0a6e8b71f9f310cc4]
aws_instance.mi_instancia: Refreshing state... [id=i-03cb1988d6ae8c8e1]
aws_route_table_association.a: Refreshing state... [id=rtbassoc-099674736b2048040]
```

```
local_file.private_key: Destroying... [id=4f3d84b57fea148435a3cd64adfeae2200d13afb]
local_file.private_key: Destruction complete after 0s
tls_private_key.devops_key: Destroying... [id=095c942864d7376c16a0104c31ae62c70dc3caa2]
tls_private_key.devops_key: Destruction complete after 0s
aws_iam_role_policy_attachment.my_attachment: Destroying... [id=role_admin-20241013065551890400000002]
aws_iam_role_policy_attachment.ebs_csi_driver_attachment: Destroying... [id=role_admin-20241013065551885100000001]
aws_iam_instance_profile.ec2_profile: Destroying... [id=ec2_instance_profile]
aws_iam_role_policy_attachment.my_attachment: Destruction complete after 0s
aws_iam_role_policy_attachment.ebs_csi_driver_attachment: Destruction complete after 0s
aws_iam_policy.my_policy: Destroying... [id=arn:aws:iam::654654500943:policy/MyEC2Policy]
aws_iam_instance_profile.ec2_profile: Destruction complete after 0s
aws_iam_role.my_role: Destroying... [id=role_admin]
aws_iam_policy.my_policy: Destruction complete after 1s
aws_iam_role.my_role: Destruction complete after 1s

Destroy complete! Resources: 7 destroyed.
```

Anexo

CLOUD-NUKE - Eliminar otros servios creado a mano.

cloud-nuke_linux_amd64 aws --region us-east-1



Confirmamos