PIN DEVOPS 2401

Grupo 01
Daniel Bertoni
Emiliano Palacio

Repositorio Solución 1 - https://github.com/palacioea/Pin-2024
Repositorio Solución 2 - https://github.com/dnlbertoni/devops2401-pin

Contenido

Enfoque de la Solución Nro: 1 2

Resumen 2

Creación de la Instancia EC2 en AWS 2

Acceso SSH a la Instancia 4

Instalación de Aplicaciones 5

Creación del Cluster EKS 6

Configuración de kubectl 8

Instalación del EBS Driver 9

Despliegue de Nginx 10

Configuración de Helm y Despliegue de Prometheus 11

Instalación de Grafana 11

Monitoreo 12

Limpieza de Recursos 14

Enfoque de la Solución Nro: 2 16

Resumen 16

Creación de la Instancia EC2 en AWS 16

Acceso SSH a la Instancia 18

Aprovisionamiento de scripts 19

Instalación de las aplicaciones 19

Instalación del EKS 20

Configuración de kubectl 20

Instalación del EBS Driver 21

Despliegue de Nginx 22

Despliegue de Prometheus 24

Instalación de Grafana 25

Monitoreo 26

Limpieza de Recursos 29

Anexo I - Reaseguro de Limpieza 31

Enfoque de la Solución Nro: 1

Resumen

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.

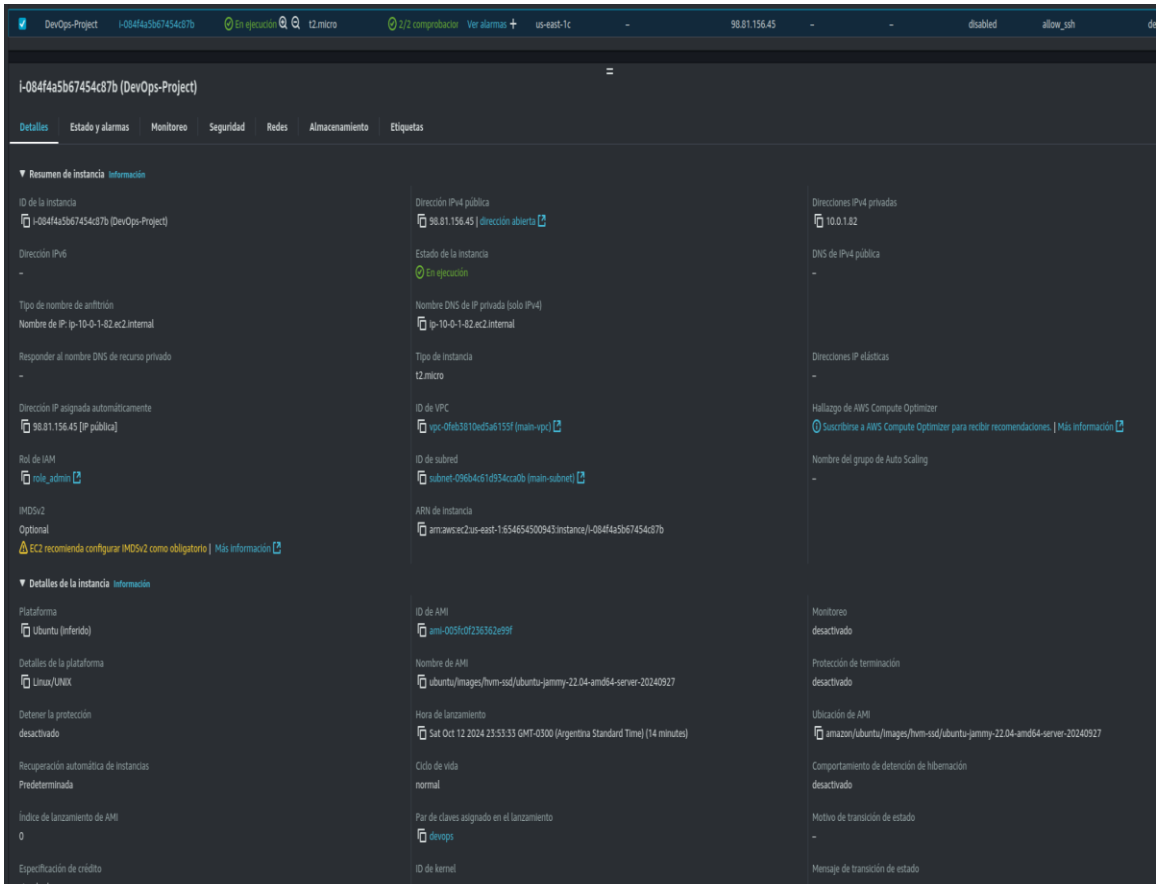
Además, utilizaremos una herramienta Cloud-Nuke para poder eliminar también todos los demás recursos que se hayan creado si fuese necesario por otras vías.

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.

```
      + tenancy                              = (known after apply)
      + timeouts                             = null
      + user_data                            = (known after apply)
      + user_data_base64                     = (known after apply)
      + user_data_replace_on_change          = false
      + volume_tags                          = null
      + vpc_security_group_ids               = (known after apply)
    }
tls_private_key.devops_key: Creating...
aws_iam_policy.my_policy: Creating...
aws_vpc.main: Creating...
aws_iam_role.my_role: Creating...
tls_private_key.devops_key: Creation complete after 2s [id=82cdcd39434b8e3871752978a6751adee99d5001]
aws_key_pair.devops: Creating...
local_file.private_key: Creating...
local_file.private_key: Creation complete after 0s [id=c6d3abd8dc6da41fe6bfc753e07948e707f4df12]
null_resource.set_permissions: Creating...
null_resource.set_permissions: Provisioning with 'local-exec'...
null_resource.set_permissions (local-exec): Executing: ["/bin/sh" "-c" "chmod 400 ./devops.pem"]
null_resource.set_permissions: Creation complete after 0s [id=635613157551006064]
aws_iam_policy.my_policy: Creation complete after 1s [id=arn:aws:iam::654654500943:policy/MyEC2Policy]
aws_iam_role.my_role: Creation complete after 1s [id=role_admin]
aws_iam_role_policy_attachment.ebs_csi_driver_attachment: Creating...
aws_iam_role_policy_attachment.my_attachment: Creating...
aws_iam_instance_profile.ec2_profile: Creating...
aws_key_pair.devops: Creation complete after 0s [id=devops]
aws_iam_role_policy_attachment.ebs_csi_driver_attachment: Creation complete after 1s [id=role_admin-20241013025325456800000001]
aws_iam_role_policy_attachment.my_attachment: Creation complete after 1s [id=role_admin-20241013025325477400000002]
aws_iam_instance_profile.ec2_profile: Creation complete after 1s [id=ec2_instance_profile]
aws_vpc.main: Creation complete after 3s [id=vpc-0feb3810ed5a6155f]
aws_internet_gateway.main: Creating...
aws_subnet.main: Creating...
aws_security_group.allow_ssh: Creating...
aws_subnet.main: Creation complete after 1s [id=subnet-096b4c61d934cca0b]
aws_internet_gateway.main: Creation complete after 1s [id=igw-08977330d7267be88]
aws_route_table.main: Creating...
aws_security_group.allow_ssh: Creation complete after 3s [id=sg-043da7f3a8768a1e4]
aws_route_table.main: Creation complete after 2s [id=rtb-0d67579d3b66d4d64]
aws_instance.mi_instancia: Creating...
aws_route_table_association.a: Creating...
aws_route_table_association.a: Creation complete after 1s [id=rtbassoc-0404f8deb9b595d01]
aws_instance.mi_instancia: Still creating... [10s elapsed]
aws_instance.mi_instancia: Creation complete after 17s [id=i-084f4a5b67454c87b]

Apply complete! Resources: 16 added, 0 changed, 0 destroyed.
```

```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.

```
ubuntu@ip-10-0-1-82: ~
mhoes@local ..ia/Hdd/Containers/proyectos/Definitivo % ssh -i "devops.pem" ubuntu@98.81.156.45
The authenticity of host '98.81.156.45 (98.81.156.45)' can't be established.
ED25519 key fingerprint is SHA256:5dRz3AY4STaqrDalmpZ0VZpHTU3ctBxKe7/OhKFz3yI.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '98.81.156.45' (ED25519) to the list of known hosts.
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 6.8.0-1015-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Sun Oct 13 03:09:59 UTC 2024

  System load:  0.0               Processes:             102
  Usage of /:   21.1% of 7.57GB   Users logged in:       0
  Memory usage: 22%               IPv4 address for eth0: 10.0.1.82
  Swap usage:   0%

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


The list of available updates is more than a week old.
To check for new updates run: sudo apt update


The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-10-0-1-82:~$
```

Instalación de Aplicaciones

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

main.tf > resource "aws_iam_instance_profile" "ec2_profile" > role

```hcl
# Crear el perfil de instancia IAM
resource "aws_iam_instance_profile" "ec2_profile" {
  name = "ec2_instance_profile"
  role = aws_iam_role.my_role.name
}

# Crear la instancia EC2
resource "aws_instance" "mi_instancia" {
  ami                    = "ami-005fc0f236362e99f"
  instance_type          = var.ec2_instance_type
  iam_instance_profile   = aws_iam_instance_profile.ec2_profile.name
  subnet_id              = aws_subnet.main.id
  key_name               = aws_key_pair.devops.key_name
  vpc_security_group_ids = [aws_security_group.allow_ssh.id]
  associate_public_ip_address = true
  user_data = file("script/apps_install.sh")

  tags = {
    Name = var.ec2_name
  }
}
```

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
```

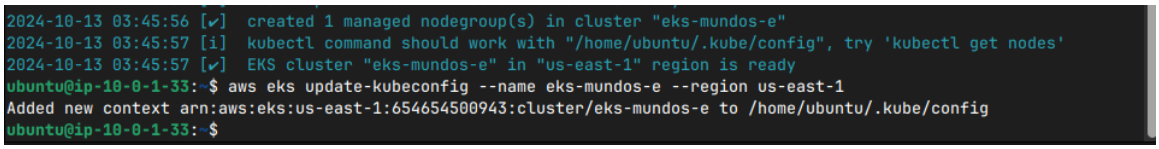
```
ubuntu@ip-10-0-1-33:~$ eksctl create cluster \
  --name eks-mundos-e \
  --region us-east-1 \
  --with-oidc \
  --nodegroup-name testv3 \
  --node-type t2.large \
  --nodes 1 \
  --nodes-min 1 \
  --nodes-max 3 \
  --node-volume-size 20 \
  --ssh-access=false \
  --managed \
  --asg-access \
  --external-dns-access \
  --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:~$
```

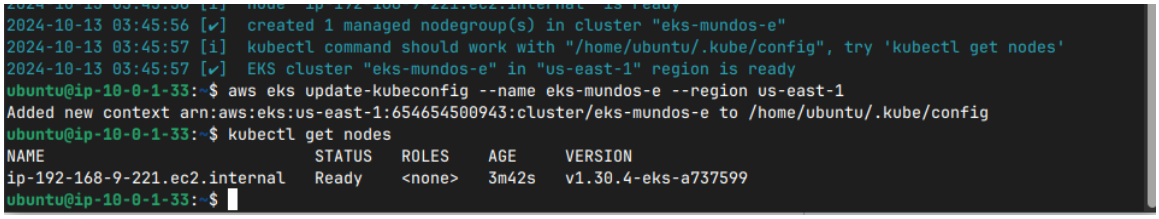
Configuración de kubectl

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:~$
```

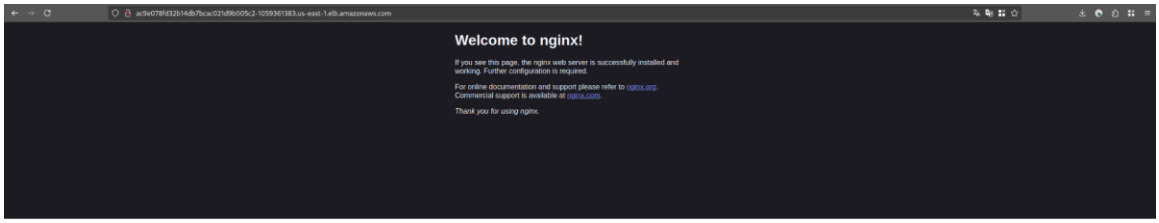
Despliegue de Nginx

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
```

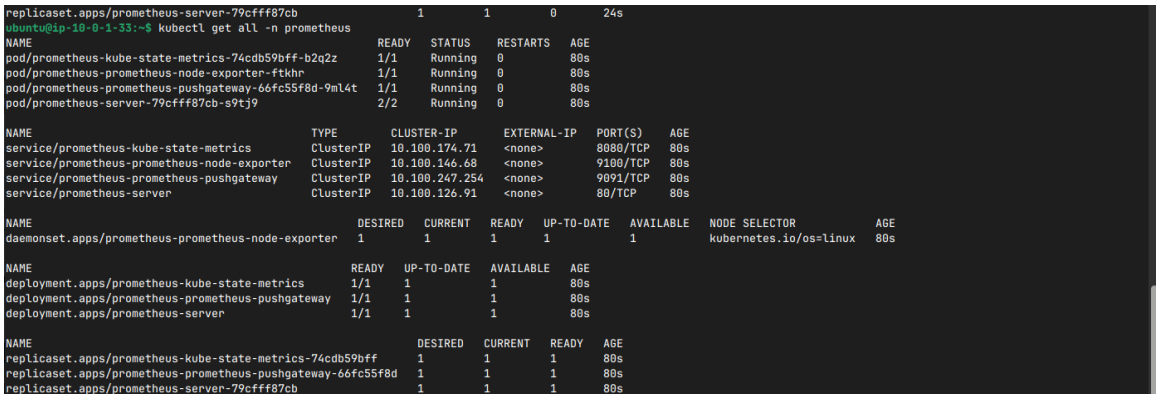
Luego, verificamos el acceso a Nginx desde el navegador.



Configuración de Helm y Despliegue de Prometheus

Agregamos los repositorios de Prometheus utilizando Helm:

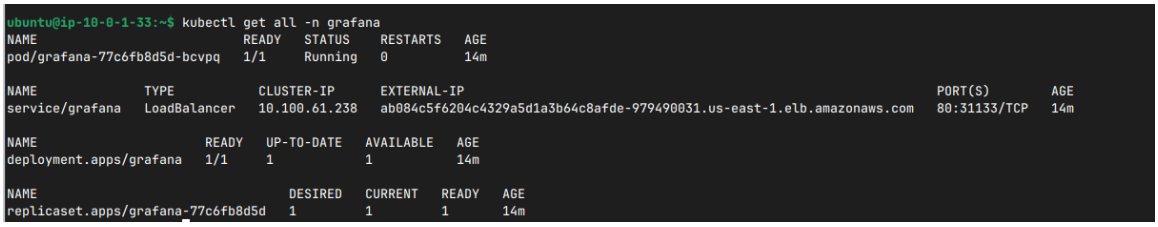
```
helm repo add prometheus-community https://prometheus-community.github.io/helm-charts
helm repo update
kubectl create namespace prometheus
helm install prometheus prometheus-community/prometheus \
        --namespace prometheus \
        --set alertmanager.persistentVolume.storageClass="gp2" \
 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



Instalación de Grafana

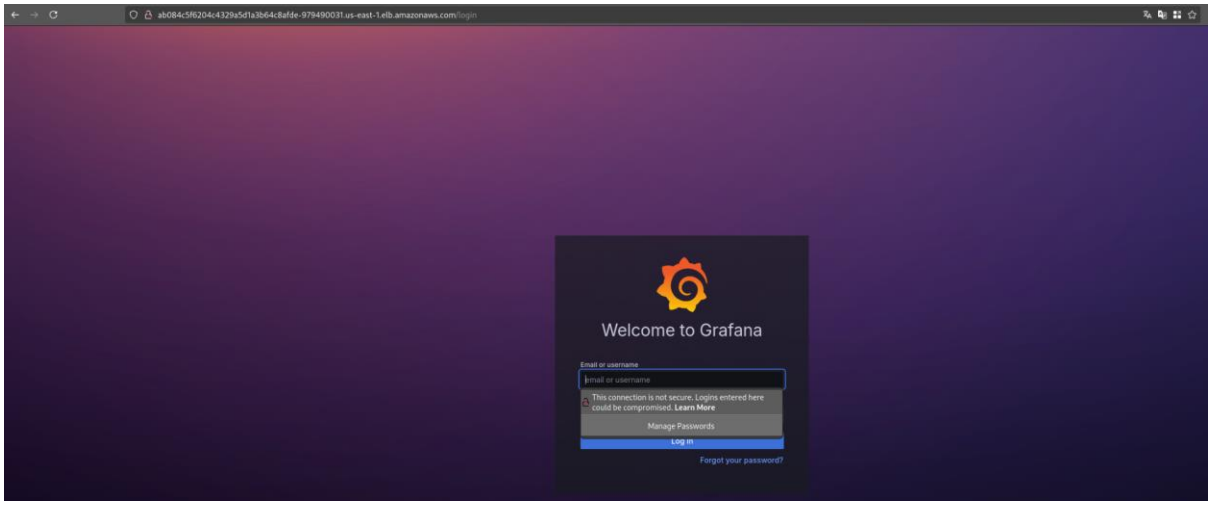
Instalamos Grafana utilizando Helm, creando previamente el namespace:

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

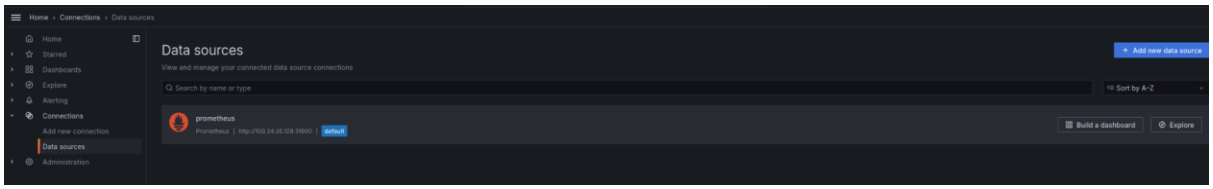


Monitoreo

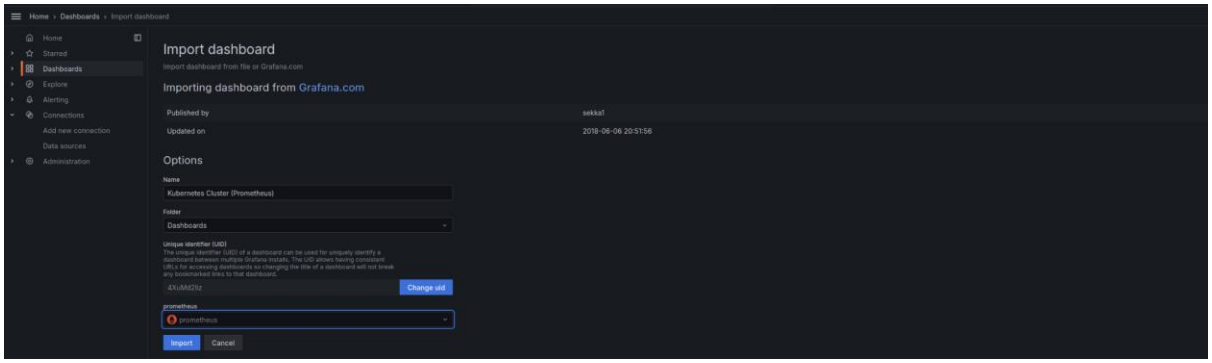
Accedemos al external-ip

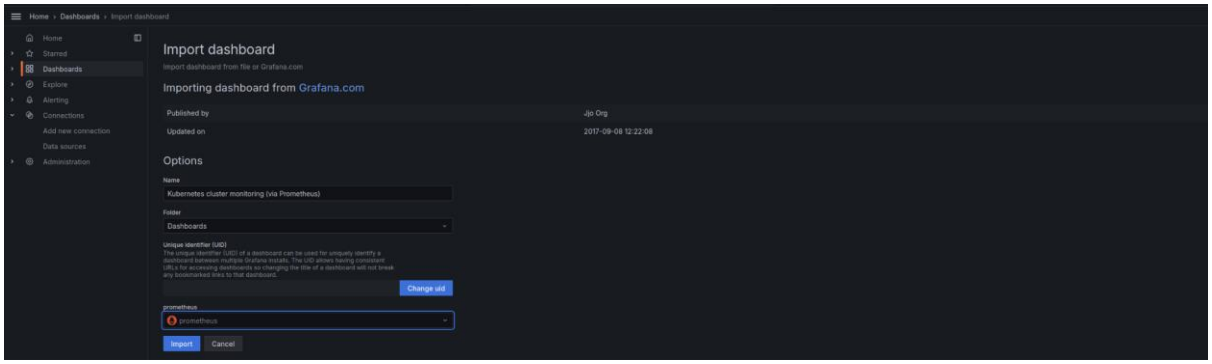


Agregamos Prometheus como datasource:

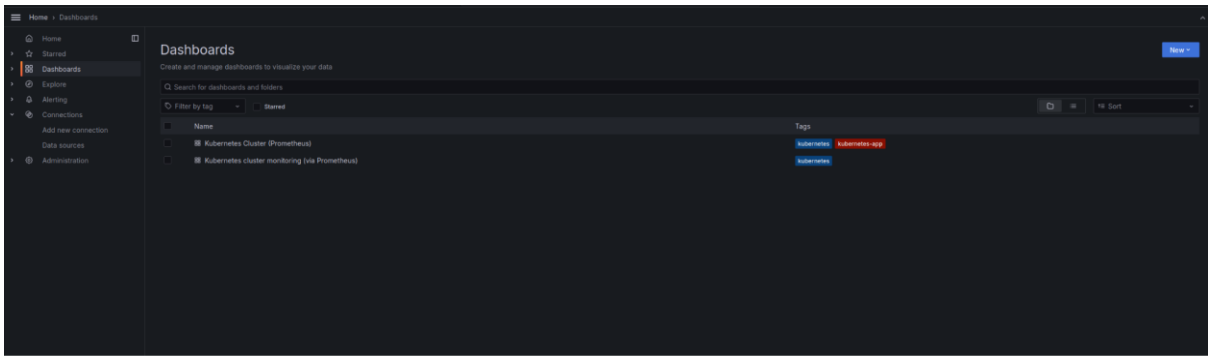


importamos los dashboards 6417 y 3119 en la carpeta Dashboard y confiugramos el prometheus que creamos en el datasource.

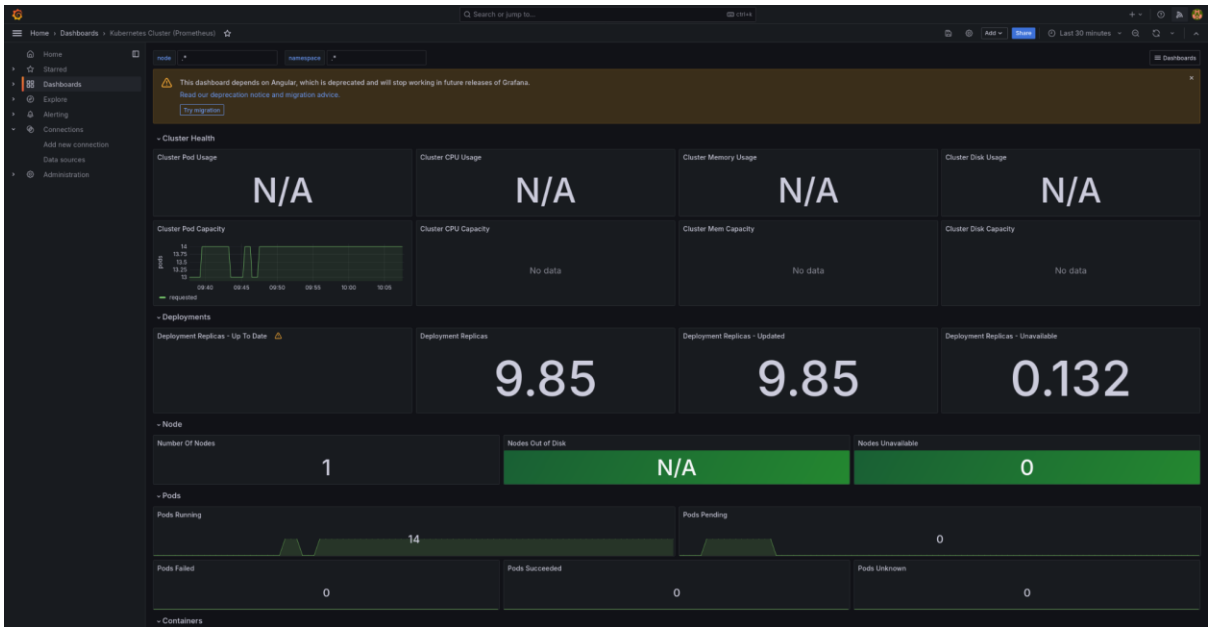




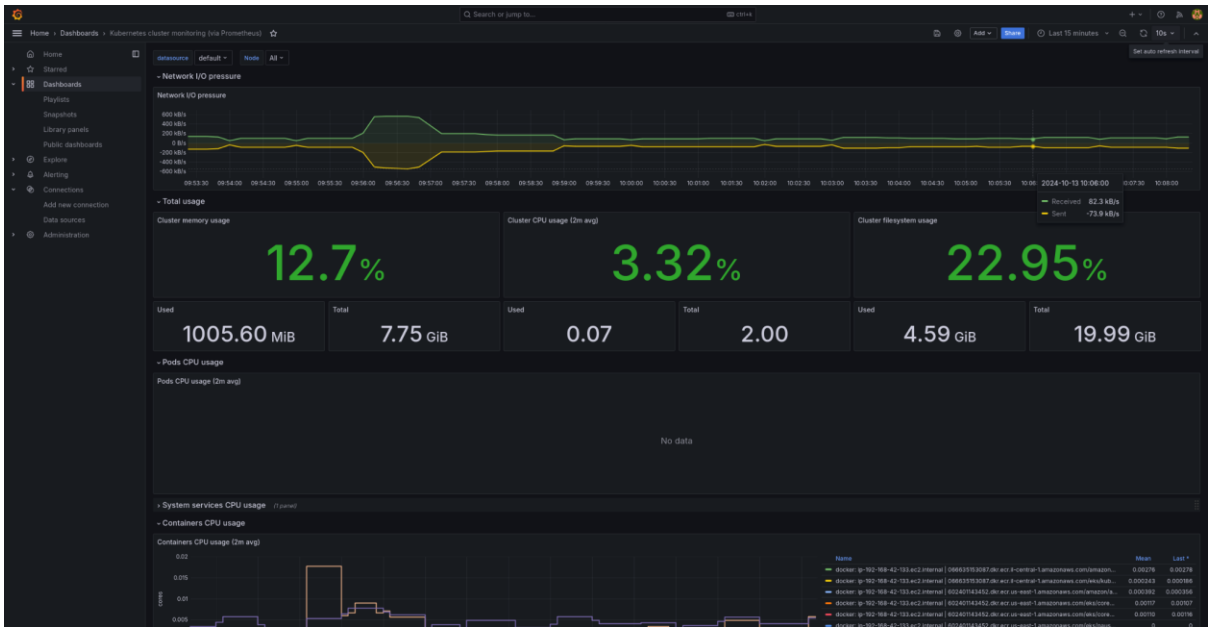
Verificamos importación de ambos



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.
```

Enfoque de la Solución Nro: 2

Resumen

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

Una vez implementado el EC2, procederemos a levantar el EKS, el Nginx y las aplicaciones de monitoreo con un script en bash aprovisionado también al EC2 desde Terraform y que solo reste aplicarlo desde el bastión.

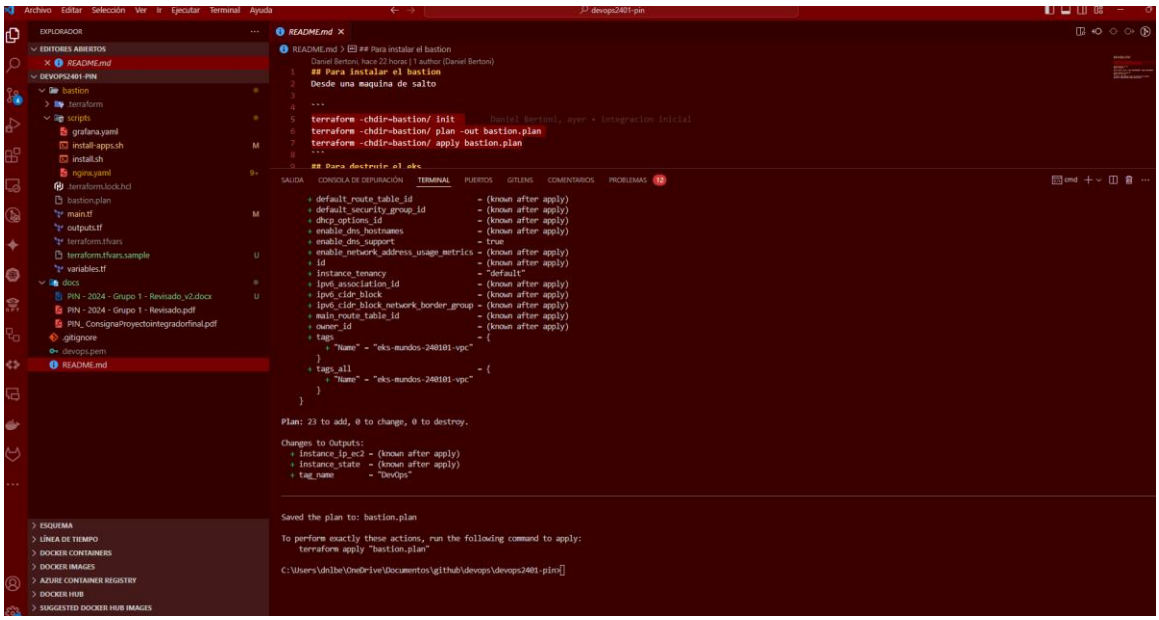
Cade destacar que cambia es la manera de aprovisionar, ya que en el método anterior usamos **user_data** y con este enfoque estamos usando **file** y **remote-exec** como provisioner desde Terraform

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

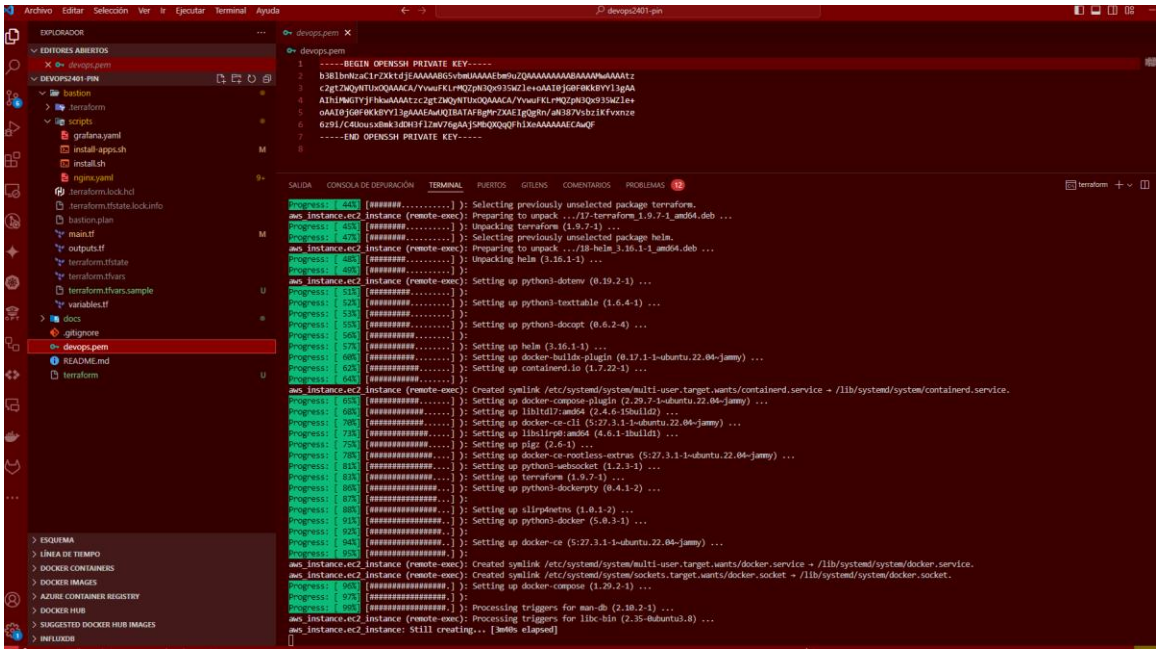
Creación de la Instancia EC2 en AWS

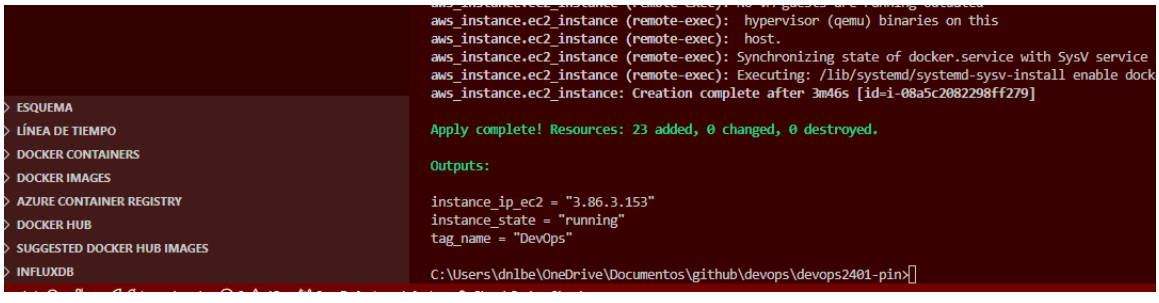
Utilizamos Terraform para aprovisionar una instancia EC2 en AWS. En el proceso, se generaron 16 recursos, incluyendo la creación del par de claves (KEY_PAIR) necesario para establecer la conexión SSH.

```
terraform -chdir=bastion/ init
terraform -chdir=bastion/ plan -out bastion.plan
terraform -chdir=bastion/ apply bastion.plan
```


Archivo Editar Selección Ver Ir Ejecutar Terminal Ayuda
devops2401-pin
EXPLORADOR
EDITORES ABIERTOS
README.md
DEVOPS2401-PIN
bastion
.terraform
scripts
grafana.yaml
install-apps.sh
install.sh
nginx.yaml
.terraform.lock.hcl
bastion.plan
main.tf
outputs.tf
terraform.tfvars
terraform.tfvars.sample
variables.tf
docs
PIN - 2024 - Grupo 1 - Revisado_v2.docx
PIN - 2024 - Grupo 1 - Revisado.pdf
PIN_ConsignaProyectointegradorfinal.pdf
.gitignore
devops.pem
README.md
README.md > ## Para instalar el bastion
Daniel Bertoni, hace 22 horas | 1 author (Daniel Bertoni)
Para instalar el bastion
Desde una maquina de salto
---
terraform -chdir=bastion/ init
terraform -chdir=bastion/ plan -out bastion.plan
terraform -chdir=bastion/ apply bastion.plan

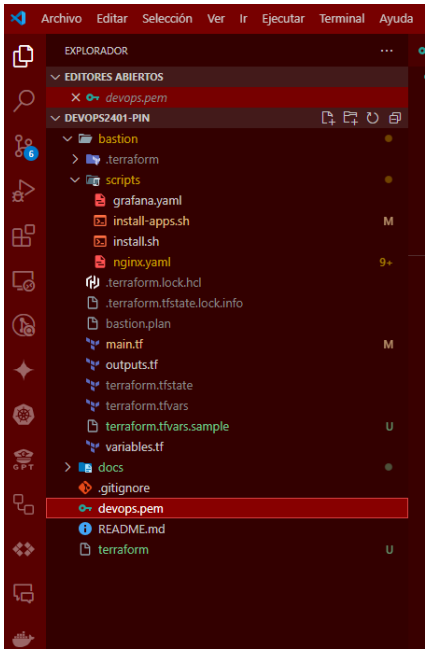
Para destruir el eks
Daniel Bertoni, ayer • Integracion Inicial
SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS GITLENS COMENTARIOS PROBLEMAS 12
cmd
+ default_route_table_id = (known after apply)
+ default_security_group_id = (known after apply)
+ dhcp_options_id = (known after apply)
+ enable_dns_hostnames = (known after apply)
+ enable_dns_support = true
+ enable_network_address_usage_metrics = (known after apply)
+ id = (known after apply)
+ instance_tenancy = "default"
+ ipv6_association_id = (known after apply)
+ ipv6_cidr_block = (known after apply)
+ ipv6_cidr_block_network_border_group = (known after apply)
+ main_route_table_id = (known after apply)
+ owner_id = (known after apply)
+ tags = {
+ "Name" = "eks-mundos-240101-vpc"
}
+ tags_all = {
+ "Name" = "eks-mundos-240101-vpc"
}
}
Plan: 23 to add, 0 to change, 0 to destroy.
Changes to Outputs:
+ instance_ip_ec2 = (known after apply)
+ instance_state = (known after apply)
+ tag_name = "DevOps"
Saved the plan to: bastion.plan
To perform exactly these actions, run the following command to apply:
terraform apply "bastion.plan"
C:\Users\dnlbe\OneDrive\Documentos\github\devops\devops2401-pin>
ESQUEMA
LÍNEA DE TIEMPO
DOCKER CONTAINERS
DOCKER IMAGES
AZURE CONTAINER REGISTRY
DOCKER HUB
SUGGESTED DOCKER HUB IMAGES

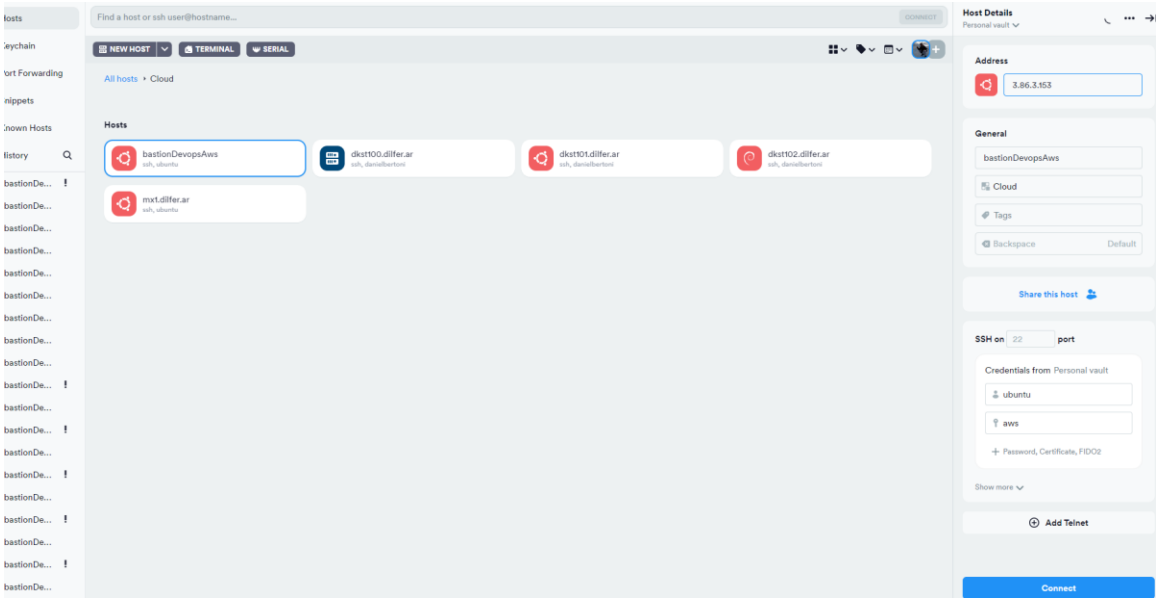

Archivo Editar Selección Ver Ir Ejecutar Terminal Ayuda
devops2401-pin
EXPLORADOR
EDITORES ABIERTOS
devops.pem
DEVOPS2401-PIN
bastion
.terraform
scripts
grafana.yaml
install-apps.sh
install.sh
nginx.yaml
.terraform.lock.hcl
.terraform.tfstate.lock.info
bastion.plan
main.tf
outputs.tf
terraform.tfstate
terraform.tfvars
terraform.tfvars.sample
variables.tf
docs
.gitignore
devops.pem
README.md
terraform
devops.pem
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAAAMwAAAAtz
c2gtZWQyNTUxOQAAACA/YvwuFKLrMQZpN3Qx935WZle+oAAI0jGBF0KkBYYl3gAA
AIhiPMGTYjFhkwAAAAtzc2gtZWQyNTUxOQAAACA/YvwuFKLrMQZpN3Qx935WZle+
oAAI0jGBF0KkBYYl3gAAAEAwUQIBATAFBgMrZXAEIgQgRn/aN387VsbziKfvxnze
6z9l/C4UoosxBmk3dDH3Fl1mV7bgAAjSMbQXQqQFhiXeAAAAAECAwQF
-----END OPENSSH PRIVATE KEY-----
SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS GITLENS COMENTARIOS PROBLEMAS 12
terraform
Progress: [44%] [#######.....]): Selecting previously unselected package terraform.
aws_instance.ec2_instance (remote-exec): Preparing to unpack .../17-terraform_1.9.7-1_amd64.deb ...
Progress: [45%] [#######.....]): Unpacking terraform (1.9.7-1) ...
Progress: [47%] [#######.....]): Selecting previously unselected package helm.
aws_instance.ec2_instance (remote-exec): Preparing to unpack .../18-helm_3.16.1-1_amd64.deb ...
Progress: [48%] [#######.....]): Unpacking helm (3.16.1-1) ...
Progress: [49%] [########....]):
aws_instance.ec2_instance (remote-exec): Setting up python3-dotenv (0.19.2-1) ...
Progress: [51%] [########....]):
Progress: [52%] [########....]): Setting up python3-texttable (1.6.4-1) ...
Progress: [53%] [########....]):
Progress: [55%] [#########...]): Setting up python3-docopt (0.6.2-4) ...
Progress: [56%] [#########...]):
Progress: [57%] [#########...]): Setting up helm (3.16.1-1) ...
Progress: [60%] [#########...]): Setting up docker-buildx-plugin (0.17.1-1~ubuntu.22.04~jammy) ...
Progress: [62%] [##########..]): Setting up containerd.io (1.7.22-1) ...
Progress: [64%] [##########..]):
aws_instance.ec2_instance (remote-exec): Created symlink /etc/systemd/system/multi-user.target.wants/containerd.service → /lib/systemd/system/containerd.service.
Progress: [65%] [##########..]): Setting up docker-compose-plugin (2.29.7-1~ubuntu.22.04~jammy) ...
Progress: [68%] [###########.]): Setting up libltdl7:amd64 (2.4.6-15build2) ...
Progress: [70%] [###########.]): Setting up docker-ce-cli (5:27.3.1-1~ubuntu.22.04~jammy) ...
Progress: [73%] [############]): Setting up libslirp0:amd64 (4.6.1-1build1) ...
Progress: [75%] [############]): Setting up pigz (2.6-1) ...
Progress: [78%] [############]): Setting up docker-ce-rootless-extras (5:27.3.1-1~ubuntu.22.04~jammy) ...
Progress: [81%] [#############]): Setting up python3-websocket (1.2.3-1) ...
Progress: [83%] [#############]): Setting up terraform (1.9.7-1) ...
Progress: [86%] [#############]): Setting up python3-dockerpty (0.4.1-2) ...
Progress: [87%] [#############]):
Progress: [88%] [##############]): Setting up slirp4netns (1.0.1-2) ...
Progress: [91%] [##############]): Setting up python3-docker (5.0.3-1) ...
Progress: [92%] [##############]):
Progress: [94%] [###############]): Setting up docker-ce (5:27.3.1-1~ubuntu.22.04~jammy) ...
Progress: [95%] [###############]):
aws_instance.ec2_instance (remote-exec): Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /lib/systemd/system/docker.service.
aws_instance.ec2_instance (remote-exec): Created symlink /etc/systemd/system/sockets.target.wants/docker.socket → /lib/systemd/system/docker.socket.
Progress: [95%] [###############.]): Setting up docker-compose (1.29.2-1) ...
Progress: [97%] [###############.]):
Progress: [99%] [################.]): Processing triggers for man-db (2.10.2-1) ...
aws_instance.ec2_instance (remote-exec): Processing triggers for libc-bin (2.35-0ubuntu3.8) ...
aws_instance.ec2_instance: Still creating... [3m40s elapsed]
ESQUEMA
LÍNEA DE TIEMPO
DOCKER CONTAINERS
DOCKER IMAGES
AZURE CONTAINER REGISTRY
DOCKER HUB
SUGGESTED DOCKER HUB IMAGES
INFLUXDB


aws_instance.ec2_instance (remote-exec): hypervisor (qemu) binaries on this
aws_instance.ec2_instance (remote-exec): host.
aws_instance.ec2_instance (remote-exec): Synchronizing state of docker.service with SysV service
aws_instance.ec2_instance (remote-exec): Executing: /lib/systemd/systemd-sysv-install enable dock
aws_instance.ec2_instance: Creation complete after 3m46s [id=i-08a5c2082298ff279]
Apply complete! Resources: 23 added, 0 changed, 0 destroyed.
Outputs:
instance_ip_ec2 = "3.86.3.153"
instance_state = "running"
tag_name = "DevOps"
C:\Users\dnlbe\OneDrive\Documentos\github\devops\devops2401-pin>

Acceso SSH a la Instancia

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

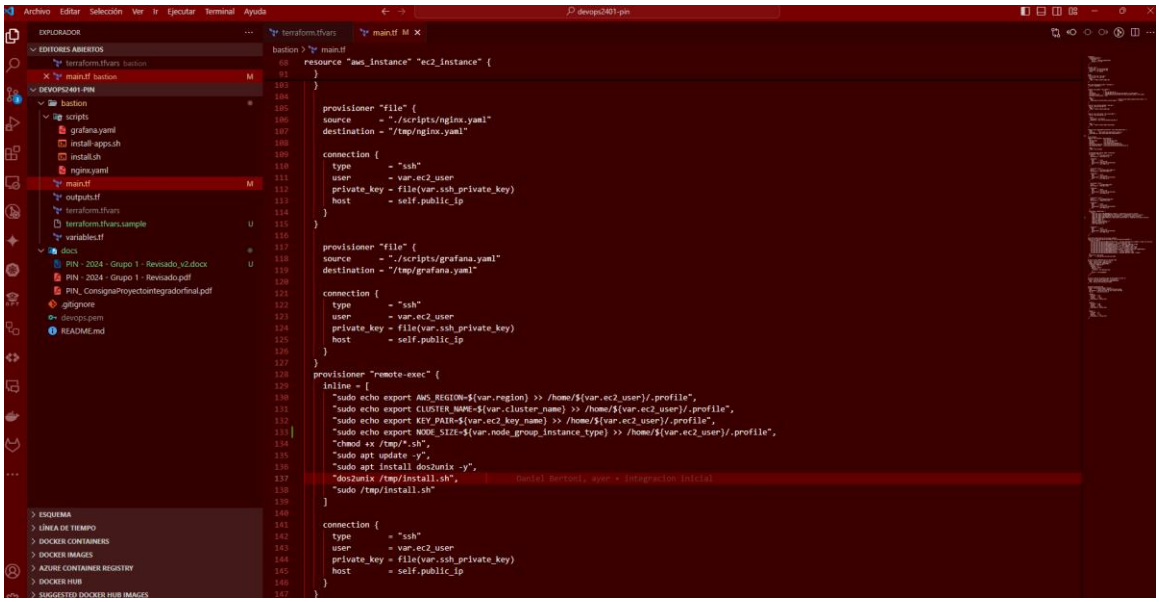




Aprovisionamiento de scripts

Aquí usamos la misma llave que vamos a usar para conectarnos al bastión , para transferirle los archivo de script al directorio /tmp del bastión y así poder controlar de manera versionada los cambios que vayamos haciendo en los script de despliegue.

Ademas es requerido pasarle las variable de Terraform al Sistema Opertivo mediante variables de entorno.



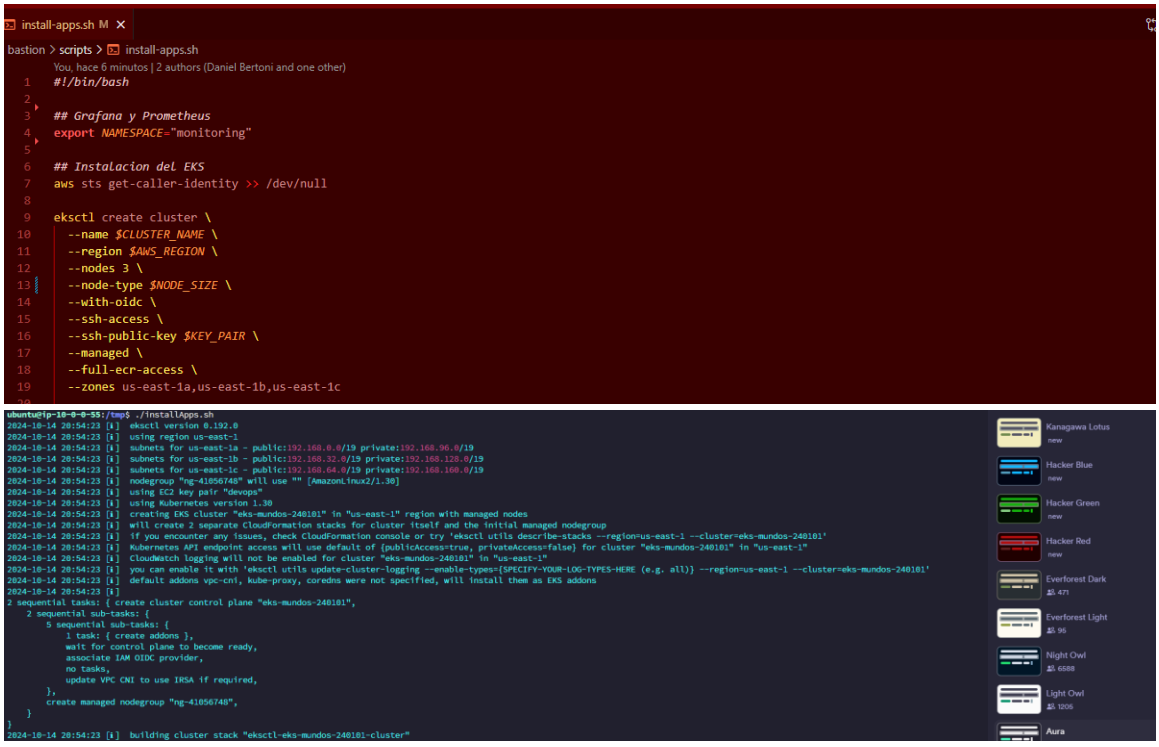
Instalación de las aplicaciones

Iniciamos el proceso de instalación utilizando el siguiente comando:

```
cd /tmp

./installApps.sh
```

Instalación del EKS

Enel script se instalación se ejecutara esta parte



Configuración de kubectl

En el script de instalacion se ejecutara en esta parte

```
install-apps.sh M
bastion > scripts > install-apps.sh
  eksctl create cluster \
    --managed \
    --full-ecr-access \
    --zones us-east-1a,us-east-1b,us-east-1c

  ## Actualizar kubeconfig
  aws eks update-kubeconfig --name $CLUSTER_NAME --region $AWS_REGION
```

```
024-10-14 21:10:08 [✓]  created 1 managed nodegroup(s) in cluster "eks-mundos-240101"
024-10-14 21:10:09 [ℹ]  kubectl command should work with "/home/ubuntu/.kube/config", try 'kubectl get nodes'
024-10-14 21:10:09 [✓]  EKS cluster "eks-mundos-240101" in "us-east-1" region is ready
dded new context arn:aws:eks:us-east-1:714317586335:cluster/eks-mundos-240101 to /home/ubuntu/.kube/config
```

Instalación del EBS Driver

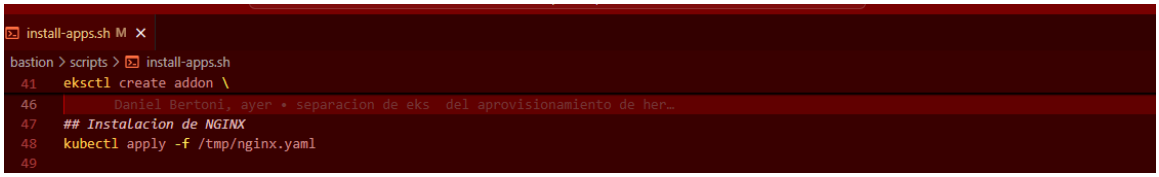
En el script de instalacion se ejecutara en esta parte

```
devops2401-pin
install-apps.sh M
bastion > scripts > install-apps.sh
  ##Intalacion el EBS CSI Driver
  kubectl apply -k "github.com/kubernetes-sigs/aws-ebs-csi-driver/deploy/kubernetes/overlays/stable/?ref=release-1.30"

  eksctl create iamserviceaccount \
   --name ebs-csi-controller-sa \
   --region $AWS_REGION \
   --namespace kube-system \
   --cluster $CLUSTER_NAME \
   --attach-policy-arn arn:aws:iam::aws:policy/service-role/AmazonEBSCSIDriverPolicy \
   --approve \
   --role-only \
   --role-name AmazonEKS_EBS_CSI_DriverRole

   # Obtener el ID de la cuenta y almacenarlo en una variable
  ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)

  # Ejecutar el comando eksctl con la variable ACCOUNT_ID
  eksctl create addon \
    --name aws-ebs-csi-driver \
    --cluster $CLUSTER_NAME \
    --service-account-role-arn arn:aws:iam::$ACCOUNT_ID:role/AmazonEKS_EBS_CSI_DriverRole \
    --force
        Daniel Bertoni, ayer • separacion de eks  del aprovisionamiento de her…
```

```
Added new context arn:aws:eks:us-east-1:714317586335:cluster/eks-mundos-240101 to /home/ubuntu/.kube/config
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
2024-10-14 21:10:13 [ℹ]  1 iamserviceaccount (kube-system/ebs-csi-controller-sa) was included (based on the include/exclude rules)
2024-10-14 21:10:13 [!]  serviceaccounts in Kubernetes will not be created or modified, since the option --role-only is used
2024-10-14 21:10:13 [ℹ]  1 task: { create IAM role for serviceaccount "kube-system/ebs-csi-controller-sa" }
2024-10-14 21:10:13 [ℹ]  building iamserviceaccount stack "eksctl-eks-mundos-240101-addon-iamserviceaccount-kube-system-ebs-csi-controller-sa"
2024-10-14 21:10:13 [ℹ]  deploying stack "eksctl-eks-mundos-240101-addon-iamserviceaccount-kube-system-ebs-csi-controller-sa"
2024-10-14 21:10:13 [ℹ]  waiting for CloudFormation stack "eksctl-eks-mundos-240101-addon-iamserviceaccount-kube-system-ebs-csi-controller-sa"
2024-10-14 21:10:43 [ℹ]  waiting for CloudFormation stack "eksctl-eks-mundos-240101-addon-iamserviceaccount-kube-system-ebs-csi-controller-sa"
2024-10-14 21:10:44 [ℹ]  Kubernetes version "1.30" in use by cluster "eks-mundos-240101"
2024-10-14 21:10:45 [ℹ]  IRSA is set for "aws-ebs-csi-driver" addon; will use this to configure IAM permissions
2024-10-14 21:10:45 [!]  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`
2024-10-14 21:10:45 [ℹ]  using provided ServiceAccountRoleARN "arn:aws:iam::714317586335:role/AmazonEKS_EBS_CSI_DriverRole"
2024-10-14 21:10:45 [ℹ]  creating addon
```

Despliegue de Nginx

En el script de instalacion se ejecutara en esta parte



El deployment es el siguiguiente:

```
apiVersion: v1
kind: Namespace
metadata:
  name: devops2401
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  namespace: devops2401
  labels:
    app: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.21.6
        ports:
        - containerPort: 80
---
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
  namespace: devops2401
  labels:
    app: nginx
spec:
  type: LoadBalancer
  ports:
  - port: 80
    targetPort: 80
  selector:
    app: nginx
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: nginx-ingress
  namespace: devops2401
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  rules:
  - host: nginx.local
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: nginx-service
            port:
              number: 80
```

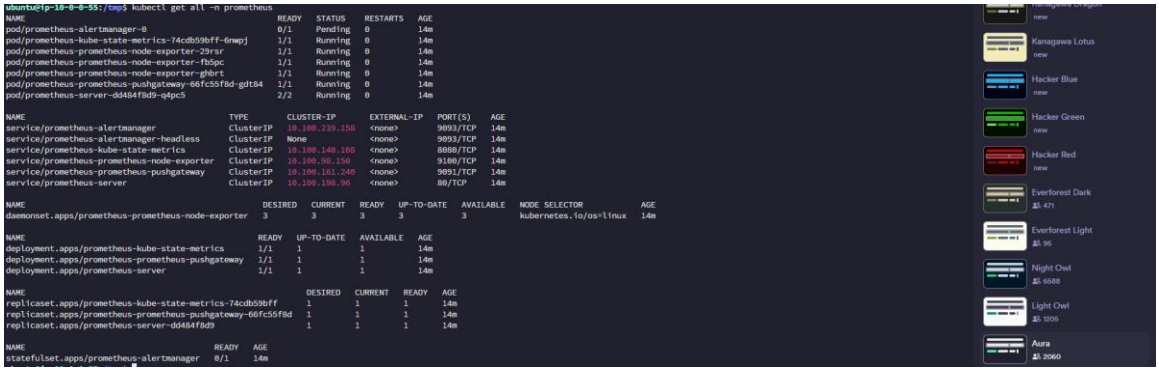
```
ubuntu@ip-10-0-0-55:/tmp$ kubectl get all -n devops2401
NAME                                    READY   STATUS    RESTARTS   AGE
pod/nginx-deployment-6f894d6c97-24b6v   1/1     Running   0          13m
pod/nginx-deployment-6f894d6c97-2hsxr   1/1     Running   0          13m
pod/nginx-deployment-6f894d6c97-t26s6   1/1     Running   0          13m

NAME                    TYPE           CLUSTER-IP       EXTERNAL-IP                                                                PORT(S)        AGE
service/nginx-service   LoadBalancer   10.100.250.158   a11ad52303b1f4316a99ea31ef475578-2017000061.us-east-1.elb.amazonaws.com   80:32267/TCP   13m

NAME                               READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/nginx-deployment   3/3     3            3           13m

NAME                                          DESIRED   CURRENT   READY   AGE
replicaset.apps/nginx-deployment-6f894d6c97   3         3         3       13m
ubuntu@ip-10-0-0-55:/tmp$
```

Luego, verificamos el acceso a Nginx desde el navegador.

Despliegue de Prometheus

En el script de instalacion se ejecutara en esta parte

```
# Agregar repo de prometheus
helm repo add prometheus-community https://prometheus-community.github.io/helm-charts

# Agregar repo de grafana
helm repo add grafana https://grafana.github.io/helm-charts

helm repo update
# Crear el namespace prometheus
kubectl create namespace prometheus

# Desplegar prometheus en EKS

helm install prometheus prometheus-community/prometheus \
--namespace prometheus \
--set alertmanager.persistentVolume.storageClass="gp2" \
--set server.persistentVolume.storageClass="gp2"

# Verificar la instalación
kubectl get all -n prometheus

# Exponer prometheus en la instancia de EC2 en el puerto 8080
kubectl port-forward -n prometheus deploy/prometheus-server 8080:9090 --address 0.0.0.0
```



Instalación de Grafana

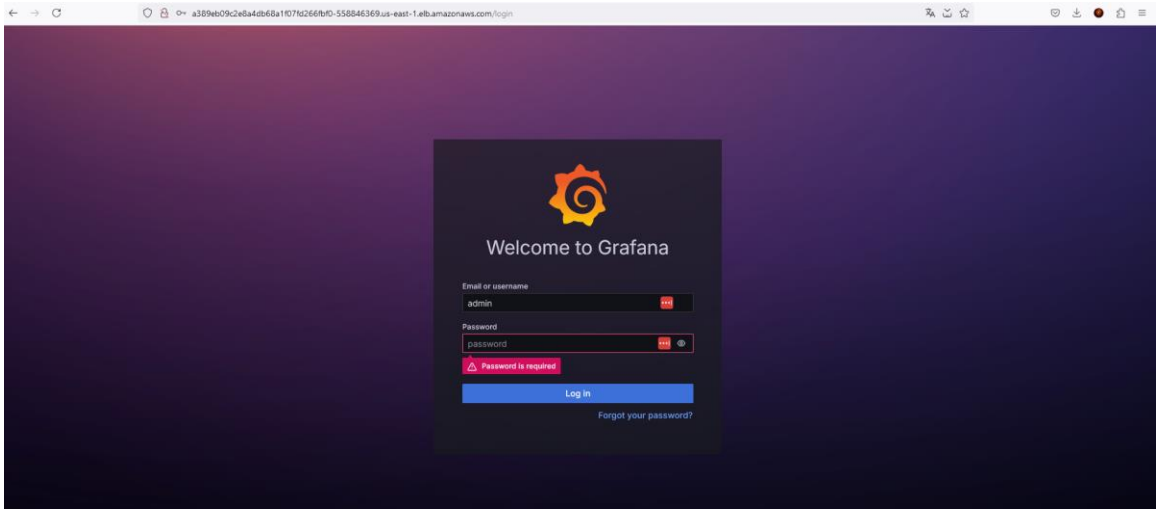
En el proceso de instalacion de grafana se utiliza un archivo grafana.yaml que sera donde se configura el datasource de prometheus que se despliega en el paso anterior.

En el script de instalacion se ejecutara en esta parte

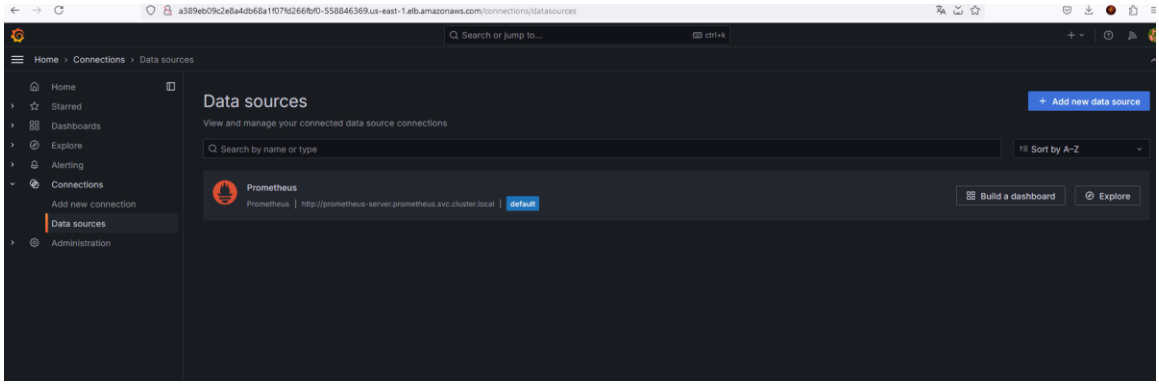
```
kubectl create namespace grafana

helm install grafana grafana/grafana \
    --namespace grafana \
    --set persistence.storageClassName="gp2" \
    --set persistence.enabled=true \
    --set adminPassword='EKS!sAWSome' \
    --values /tmp/grafana.yaml \
    --set service.type=LoadBalancer
```

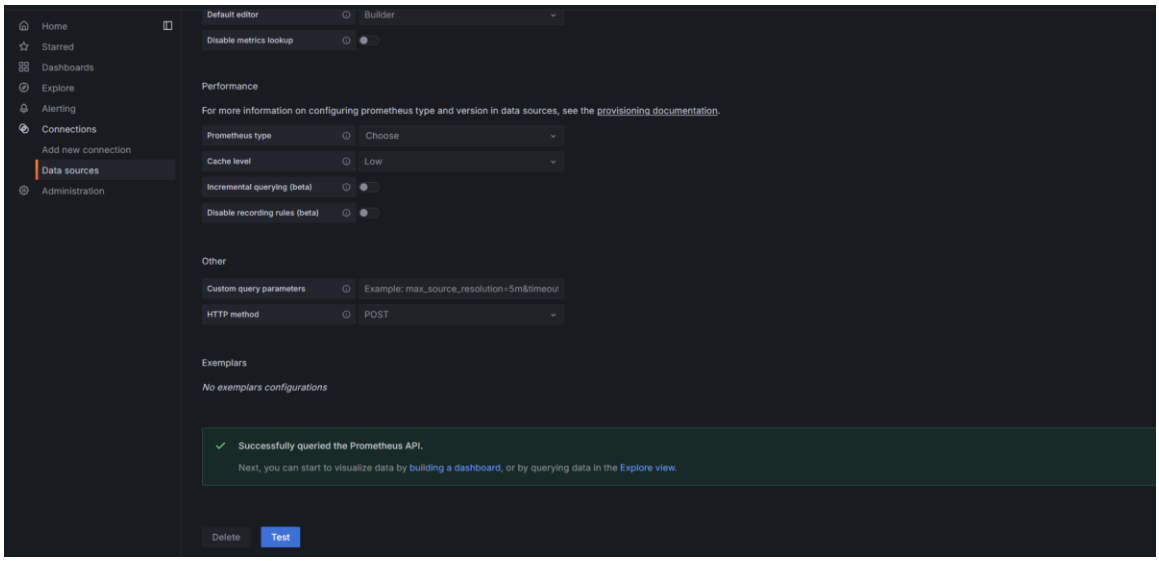
Monitoreo

Accedemos al external-ip

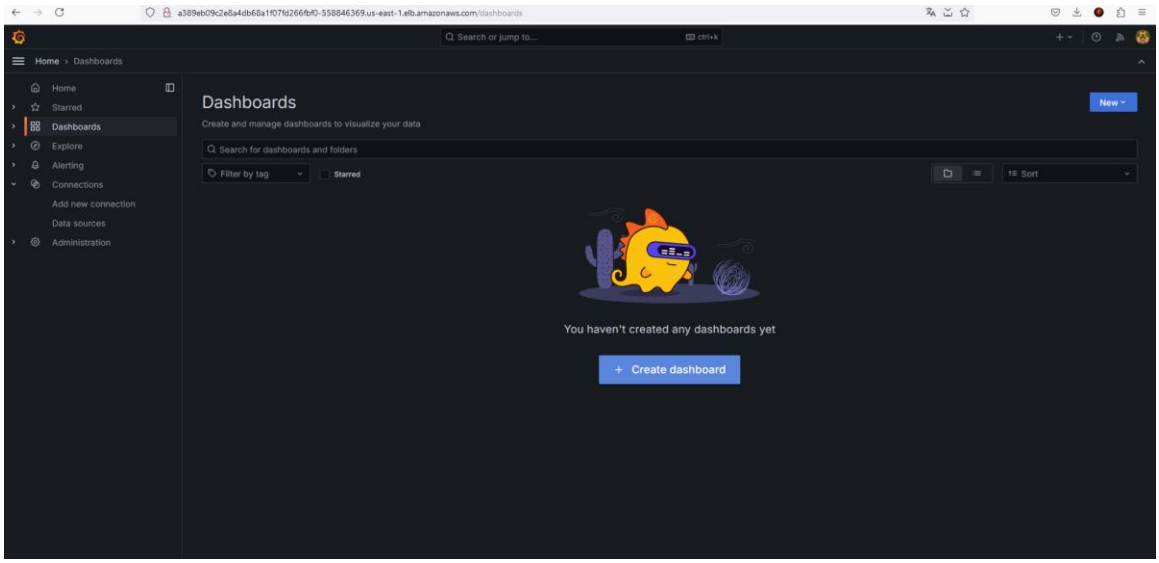


Verificamos que exista el datasource

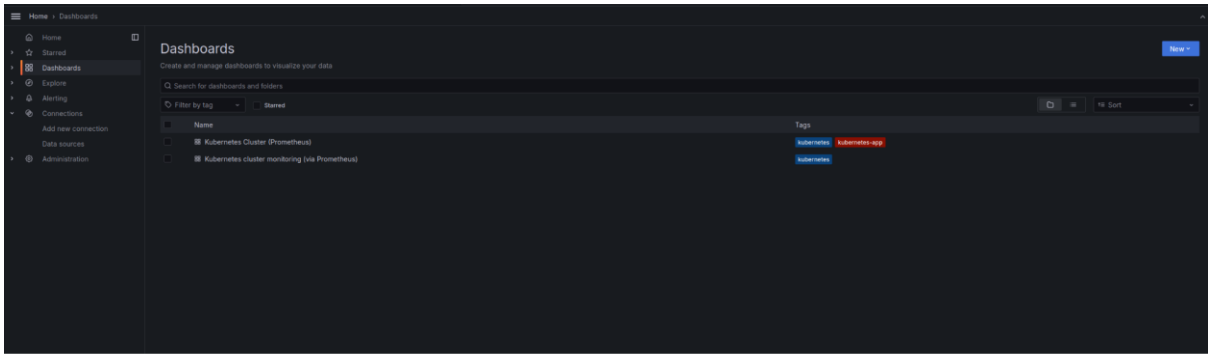




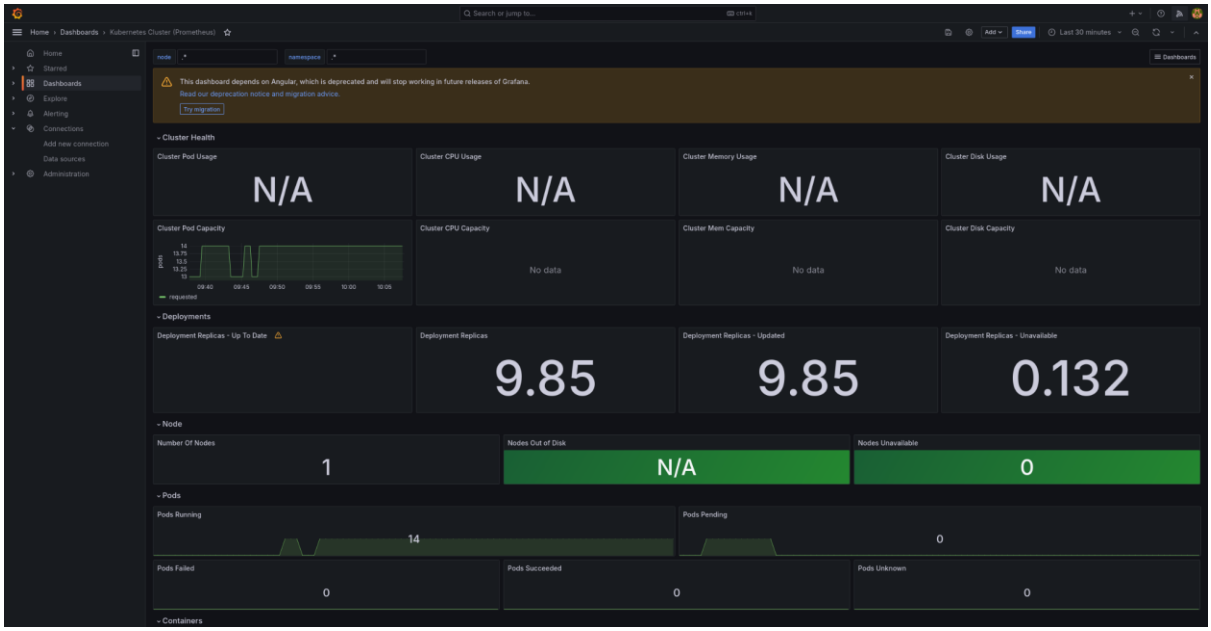
Importamos los dashboards 6417 y 3119 en la carpeta Dashboard



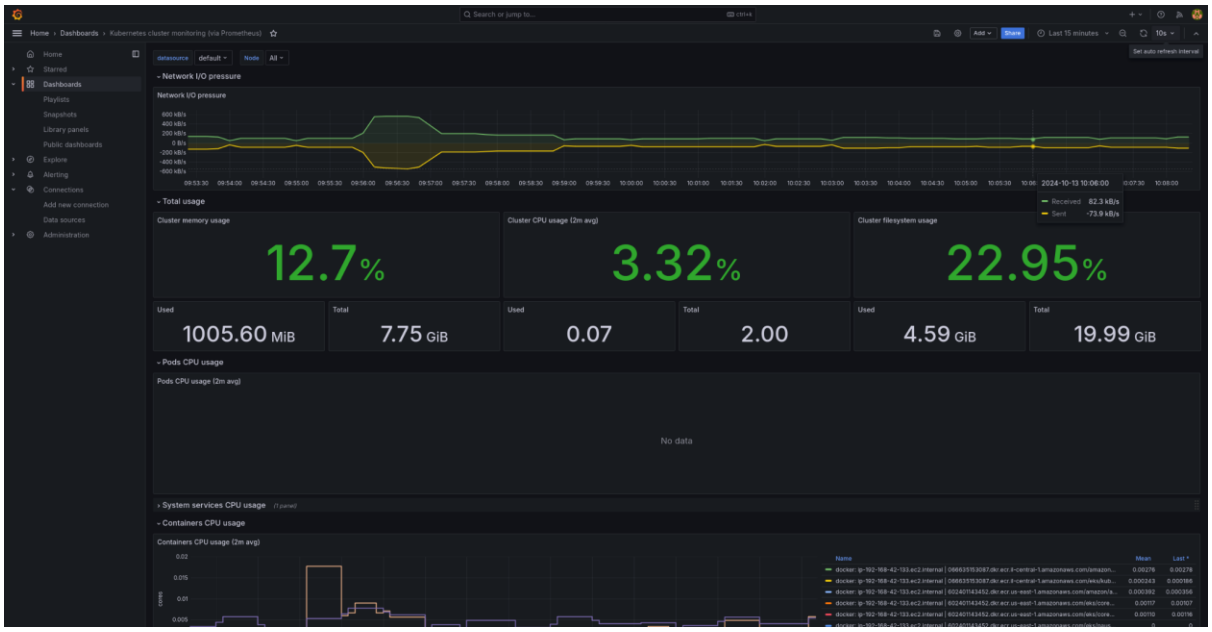
Verificamos importación de ambos



ID 6417:


Home › Dashboards › Kubernetes Cluster (Prometheus)
Search or jump to...
Add
Share
Last 30 minutes
Home
Starred
Dashboards
Explore
Alerting
Connections
Add new connection
Data sources
Administration
node
.*
namespace
.*
Dashboards
This dashboard depends on Angular, which is deprecated and will stop working in future releases of Grafana.
Read our deprecation notice and migration advice.
Try migration
Cluster Health
Cluster Pod Usage
N/A
Cluster CPU Usage
N/A
Cluster Memory Usage
N/A
Cluster Disk Usage
N/A
Cluster Pod Capacity
pods
16
13.75
13.5
13.25
13
09:40
09:45
09:50
09:55
10:00
10:05
requested
Cluster CPU Capacity
No data
Cluster Mem Capacity
No data
Cluster Disk Capacity
No data
Deployments
Deployment Replicas - Up To Date
Deployment Replicas
9.85
Deployment Replicas - Updated
9.85
Deployment Replicas - Unavailable
0.132
Node
Number Of Nodes
1
Nodes Out of Disk
N/A
Nodes Unavailable
0
Pods
Pods Running
14
Pods Pending
0
Pods Failed
0
Pods Succeeded
0
Pods Unknown
0
Containers

ID 3119:

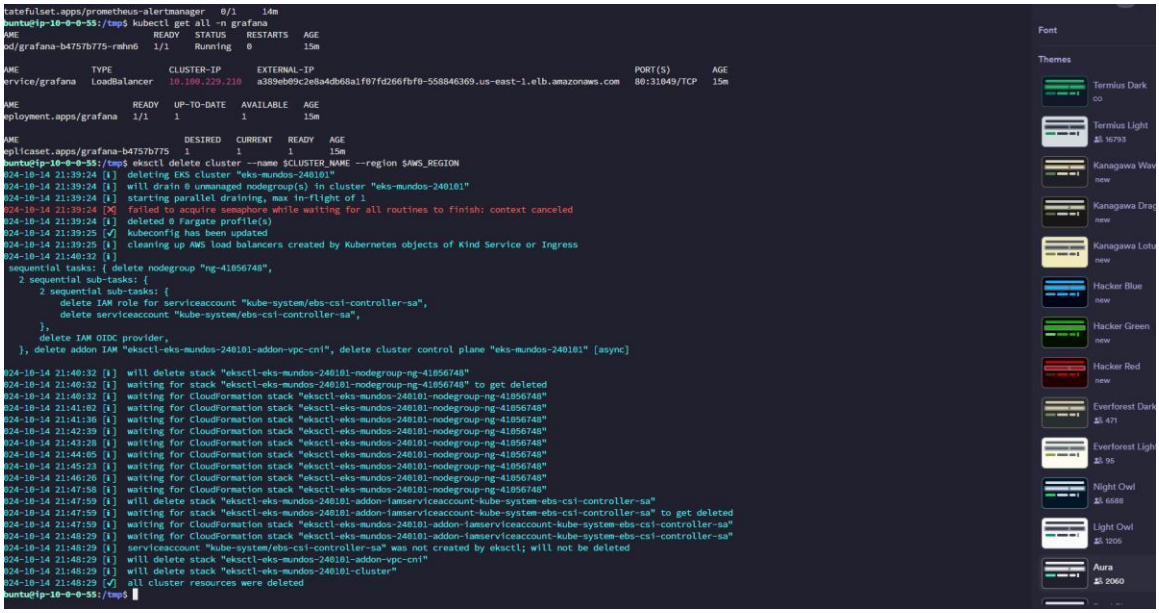


Limpieza de Recursos

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

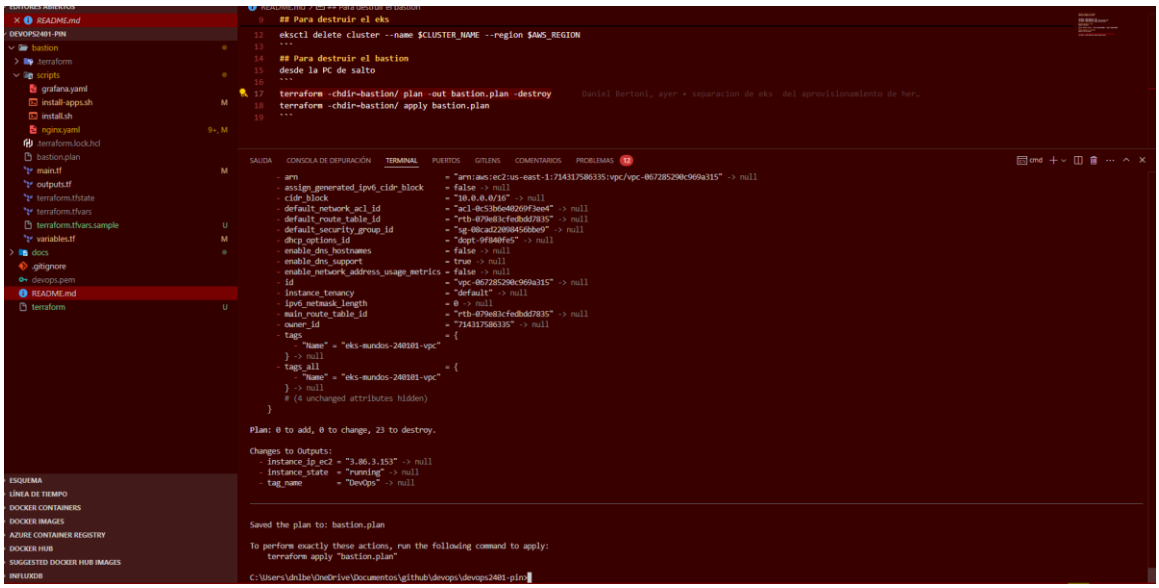
Desde el bastion (EC2)

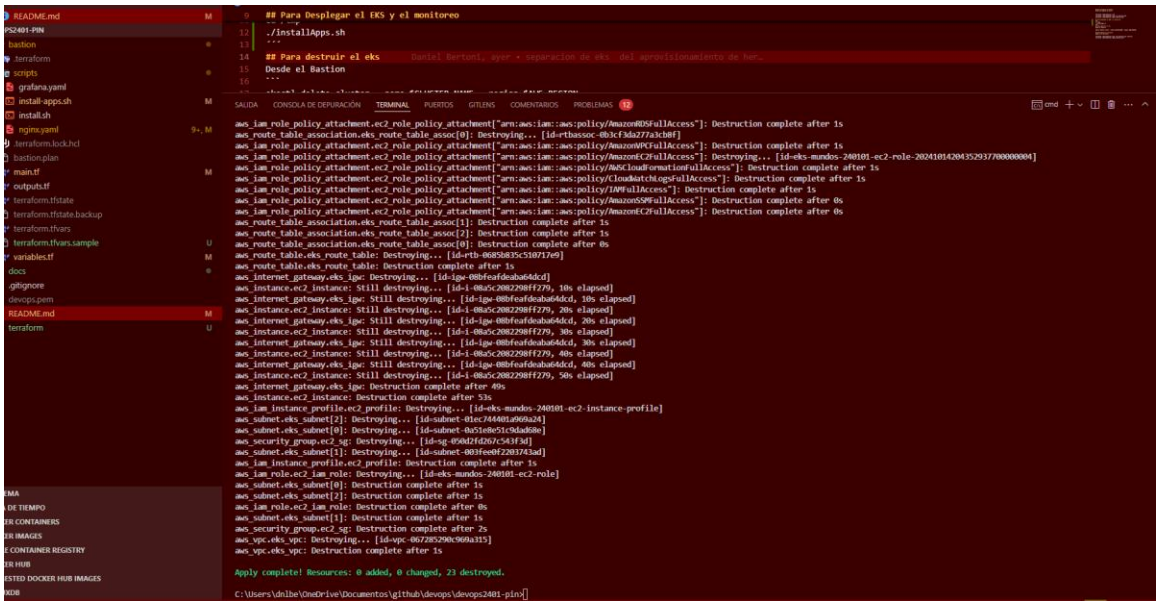
```
eksctl delete cluster --name $CLUSTER_NAME --region $AWS_REGION
```



Desde el pc de salto:

```
terraform -chdir=bastion/ plan -out bastion.plan -destroy
terraform -chdir=bastion/ apply bastion.plan
```

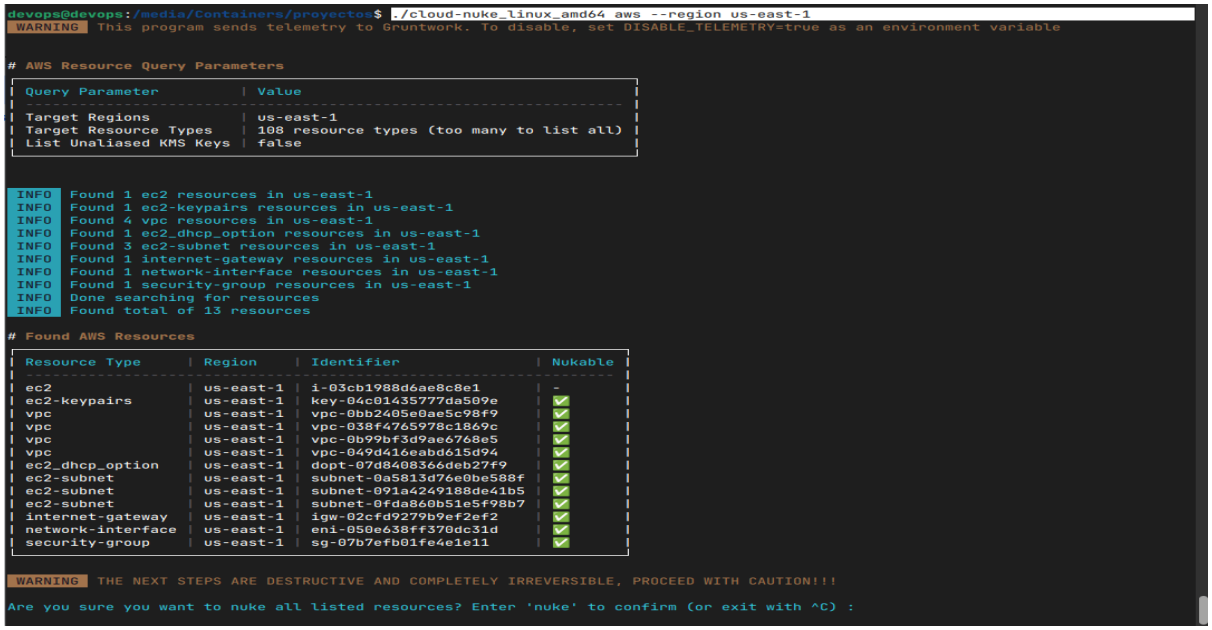




Anexo I - Reaseguro de Limpieza

CLOUD-NUKE - Eliminar otros servios creado a mano.

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



Confirmamos