Challenge Endava – Leandro Hinestroza
Documentacion:

Todo el despliegue de la infraestructura y los aplicativos fue realizada en local utilizando los siguientes recursos como prerequisitos/requeriments:

Minikube
https://minikube.sigs.k8s.io/docs/handbook/
https://minikube.sigs.k8s.io/docs/start/
https://minikube.sigs.k8s.io/docs/handbook/config/

Kubectl
https://minikube.sigs.k8s.io/docs/handbook/kubectl/

Helm Charts
https://helm.sh/docs/intro/install/
https://helm.sh/docs/intro/using_helm/
https://helm.sh/docs/intro/cheatsheet/

Terraform
https://developer.hashicorp.com/terraform/tutorials/aws-get-started/install-cli

Prometheus/Grafana setup and config
https://medium.com/globant/setup-prometheus-and-grafana-monitoring-on-kubernetes-cluster-using-helm-3484efd85891#:~:text=Setup%20Prometheus%20and%20Grafana%20monitoring%20on%20Kubernetes%20cluster,source%20into%20Grafana%20dashboard%20...%208%20Conclusion%20

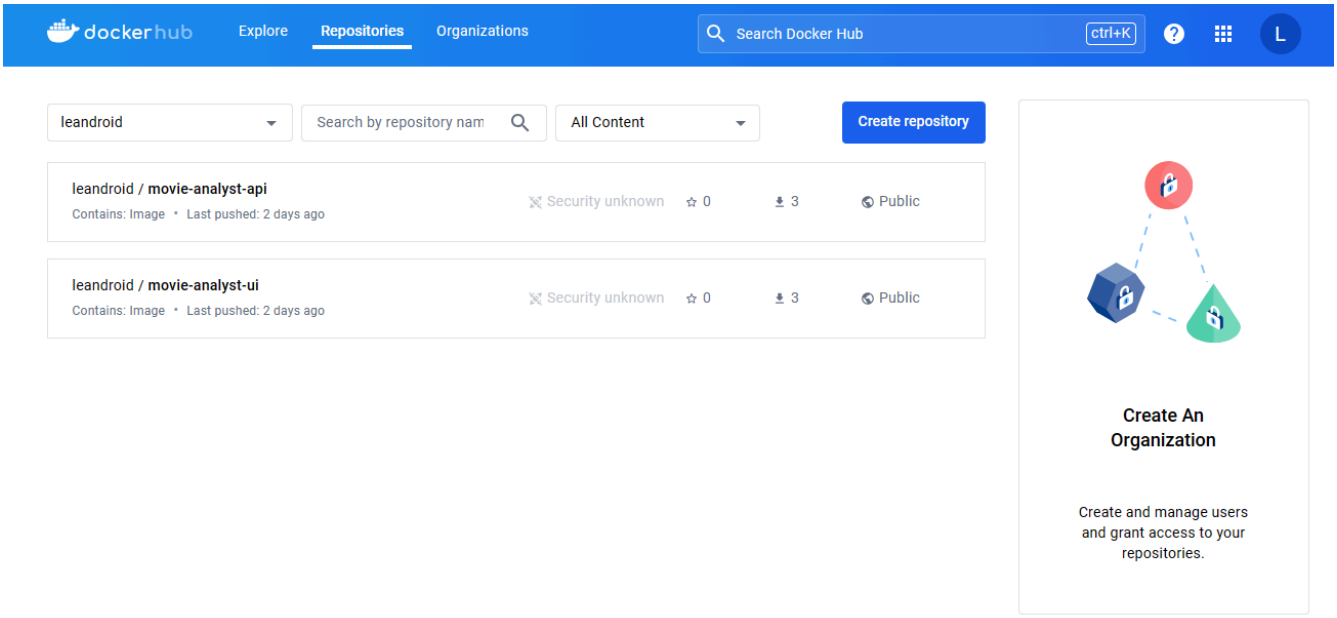
Mysql
https://dev.mysql.com/doc/mysql-operator/en/mysql-operator-installation-helm.html

Git
https://git-scm.com/download/linux

Vscode
https://code.visualstudio.com/docs/setup/linux

Luego se generaron los dockerfiles para los microservicios frontend (movie-analyst-ui) y backend (movie-analyst-api) y se pushearon las imagenes a docker hub

docker login -u leandroid -p **********
docker tag movie-analyst-api:latest leandroid/movie-analyst-api
docker tag movie-analyst-ui:latest leandroid/movie-analyst-ui
docker push leandroid/movie-analyst-api
docker push leandroid/movie-analyst-ui



Esto para testear el aplicativo local:
docker run -d -e BACKEND_URL=http://localhost:3000 -p 3030:3030 --name MOVIE-ANALYST-UI movie-analyst-ui
docker run -d -e DB_HOST=http://localhost:3000 -e DB_USER=user -e DB_PASS=P4ssw0rd -e DB_NAME=movie_db -p 3000:3000 --name MOVIE-ANALYST-API movie-analyst-api

Para ejecutar los helm charts que despliegan los distintos deployment/pods, nos situamos sobre el root folder /app y ejecutamos:

Install UI chart
helm install movie-analyst-ui ./ui
Despliega los pods en alta disponibilidad relacionados al frontend del aplicativo

Install API chart
helm install movie-analyst-api ./api
Despliega los pods en alta disponibilidad relacionados al backend del aplicativo

Install Grafana chart
helm install grafana ./grafana
Despliega los pods en alta disponibilidad relacionados a nuestro aplicativo de observabilidad, luego de desplegado se creó un datasource apuntando al los pods que ejecutan prometheus e importamos el dashboard 3662 para la visualizacion y observabilidad de los pods.
Credentials: User: admin Pass: P4ssw0rd (codificado en base 64 en los secrets pasados al helm chart.

Install Jenkins chart
helm install jenkins ./jenkins

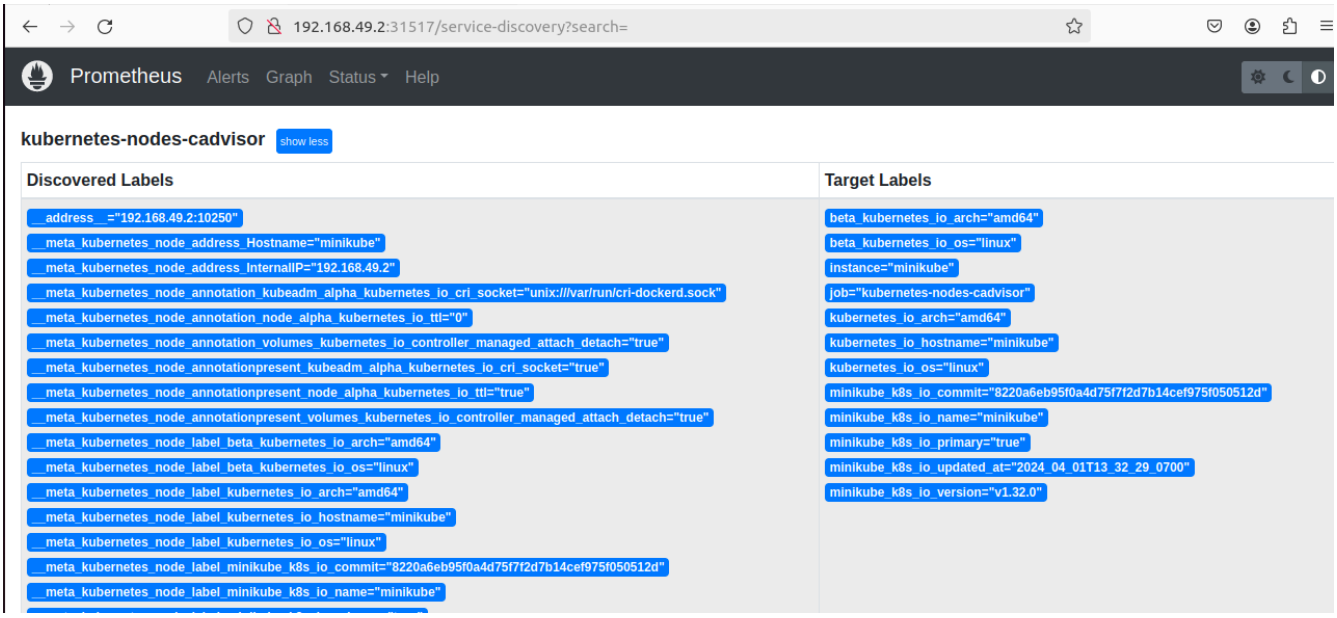
Despliega los pods en alta disponibilidad relacionados a nuestro aplicativo para ci/cd y pipelines

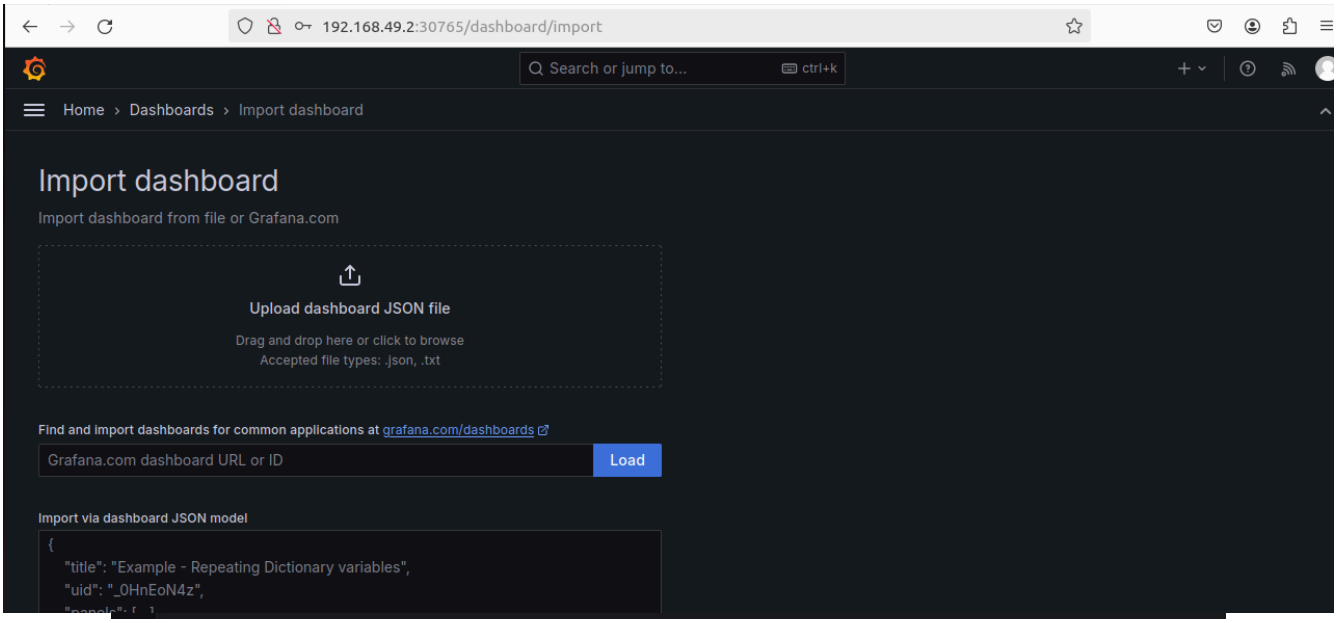
Install MySQL chart
helm install mysql ./mysql
Despliega los pods en alta disponibilidad relacionados a nuestra base de datos mysql requerida bajo el modelo entidad relacion solicitado:

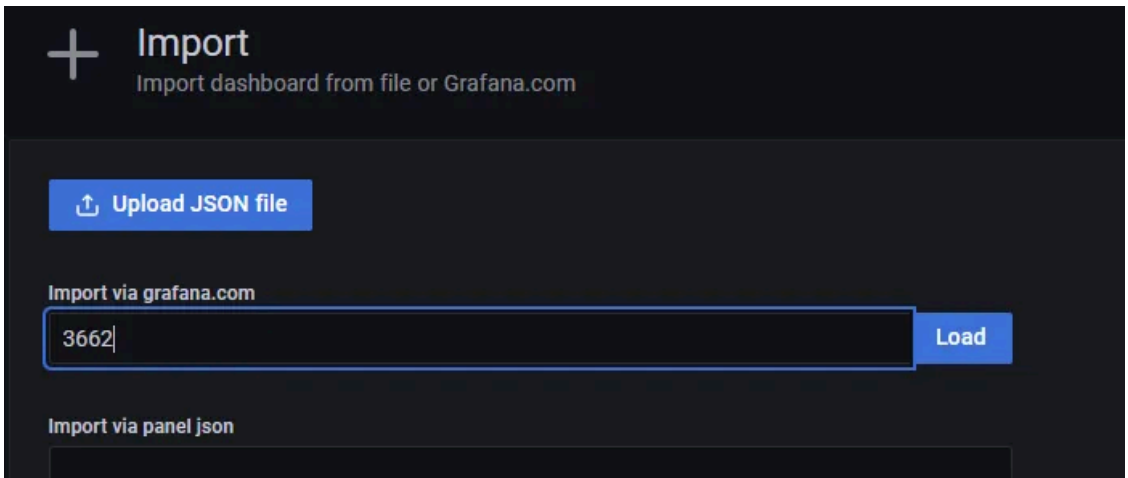
Install Prometheus chart
helm install prometheus ./prometheus
Despliega los pods en alta disponibilidad relacionados a nuestro aplicativo de correlacion de eventos, este chart tambien tiene los configmaps asociados a los jobs que hacen los scraping de las metricas de los deployment pods que corren ui, api y la base de datos mysql.
Credentials: User: admin Pass: admin (codificado en base 64 en los secrets pasados al helm chart.

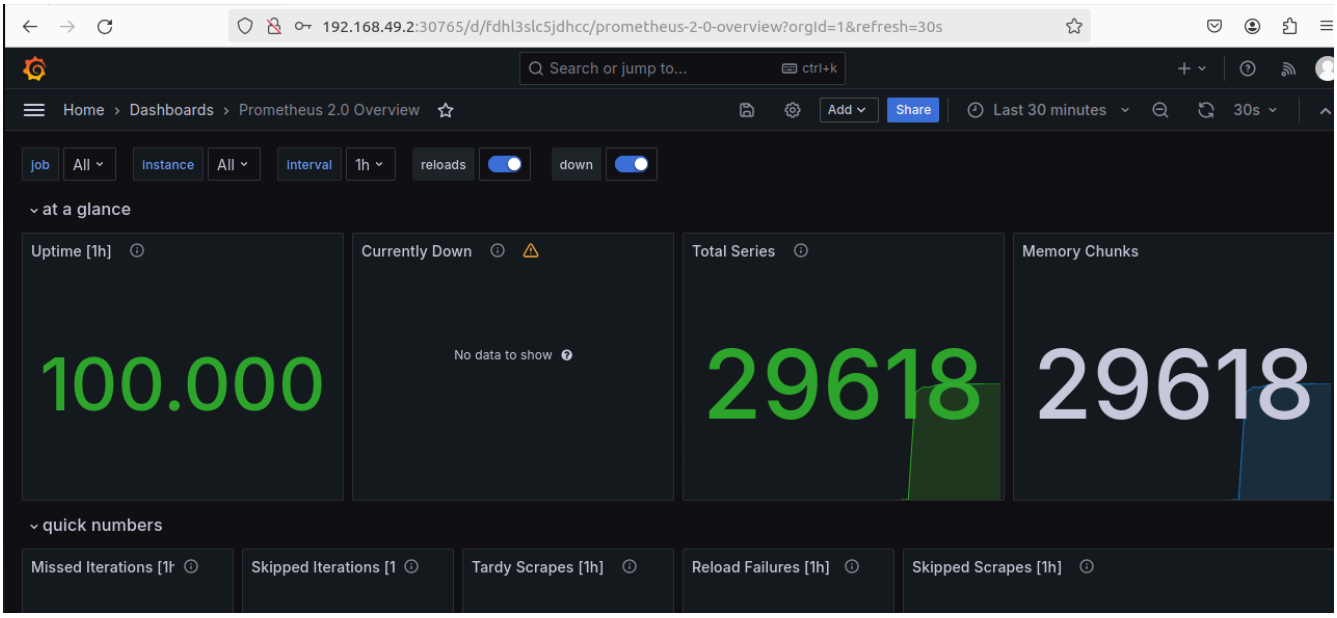
Importante, para acceder localmente a los pods via url del navegador y visualizarlos debemos usar port forward a nuestro cluster de esta manera accederemos al networking del minikube:

kubectl port-forward svc/movie-analyst-ui-service 3030:3030
kubectl port-forward svc/movie-analyst-api-service 3000:3000
kubectl port-forward svc/grafana-service 3001:3001
kubectl port-forward svc/prometheus-service 9090:9090
kubectl port-forward svc/jenkins-service 8080:8080

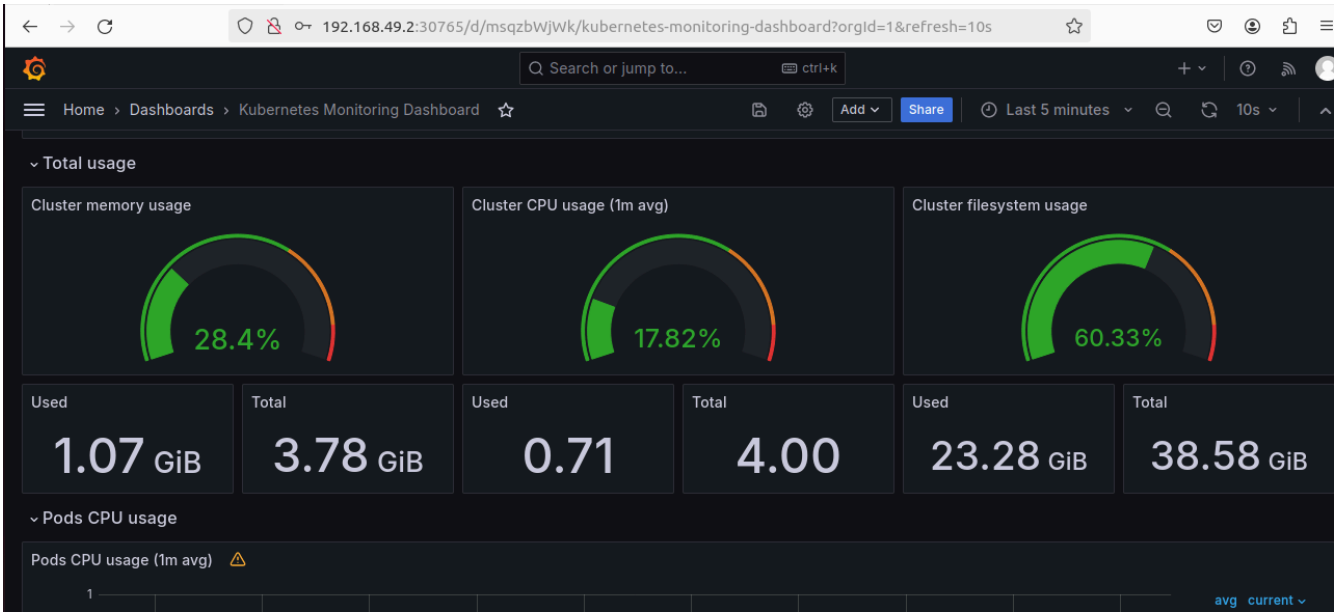

192.168.49.2:31517/service-discovery?search=
Prometheus
Alerts
Graph
Status
Help
kubernetes-nodes-cadvisor
show less
Discovered Labels
Target Labels
__address__="192.168.49.2:10250"
__meta_kubernetes_node_address_Hostname="minikube"
__meta_kubernetes_node_address_InternalIP="192.168.49.2"
__meta_kubernetes_node_annotation_kubeadm_alpha_kubernetes_io_cri_socket="unix:///var/run/cri-dockerd.sock"
__meta_kubernetes_node_annotation_node_alpha_kubernetes_io_ttl="0"
__meta_kubernetes_node_annotation_volumes_kubernetes_io_controller_managed_attach_detach="true"
__meta_kubernetes_node_annotationpresent_kubeadm_alpha_kubernetes_io_cri_socket="true"
__meta_kubernetes_node_annotationpresent_node_alpha_kubernetes_io_ttl="true"
__meta_kubernetes_node_annotationpresent_volumes_kubernetes_io_controller_managed_attach_detach="true"
__meta_kubernetes_node_label_beta_kubernetes_io_arch="amd64"
__meta_kubernetes_node_label_beta_kubernetes_io_os="linux"
__meta_kubernetes_node_label_kubernetes_io_arch="amd64"
__meta_kubernetes_node_label_kubernetes_io_hostname="minikube"
__meta_kubernetes_node_label_kubernetes_io_os="linux"
__meta_kubernetes_node_label_minikube_k8s_io_commit="8220a6eb95f0a4d75f7f2d7b14cef975f050512d"
__meta_kubernetes_node_label_minikube_k8s_io_name="minikube"
beta_kubernetes_io_arch="amd64"
beta_kubernetes_io_os="linux"
instance="minikube"
job="kubernetes-nodes-cadvisor"
kubernetes_io_arch="amd64"
kubernetes_io_hostname="minikube"
kubernetes_io_os="linux"
minikube_k8s_io_commit="8220a6eb95f0a4d75f7f2d7b14cef975f050512d"
minikube_k8s_io_name="minikube"
minikube_k8s_io_primary="true"
minikube_k8s_io_updated_at="2024_04_01T13_32_29_0700"
minikube_k8s_io_version="v1.32.0"


192.168.49.2:30765/dashboard/import
Search or jump to...
ctrl+k
Home › Dashboards › Import dashboard
Import dashboard
Import dashboard from file or Grafana.com
Upload dashboard JSON file
Drag and drop here or click to browse
Accepted file types: .json, .txt
Find and import dashboards for common applications at grafana.com/dashboards
Grafana.com dashboard URL or ID
Load
Import via dashboard JSON model
{
"title": "Example - Repeating Dictionary variables",
"uid": "_0HnEoN4z",

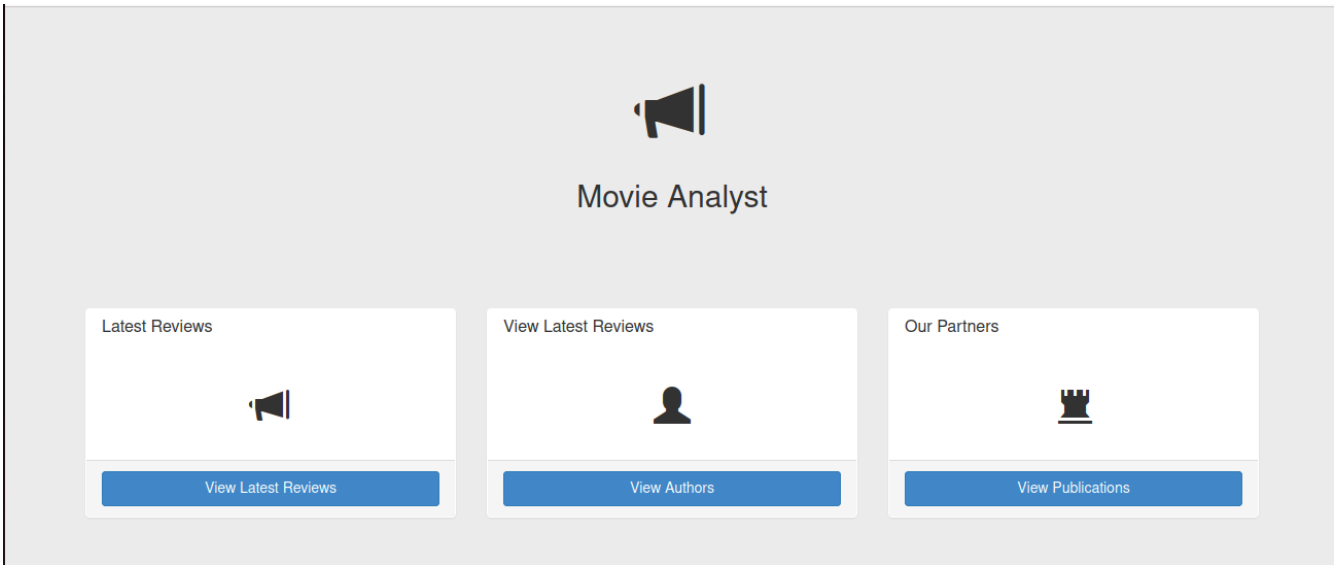

Import
Import dashboard from file or Grafana.com
Upload JSON file
Import via grafana.com
3662
Load
Import via panel json



Otro dashboard: 12740



Una vez todos los pods y deployments corriendo en el cluster de minikube nuestro aplicativo se debe ver asi, hay un servicio ingress controller dentro de la carpeta de /ui que crea por medio del helm chart que crea este servicio para ingresar por el dns name `movie-analyst-ui.info:3000`



Dentro de las carpetas movie-analyst-api y movie-analyst-ui se encuentran los archivos Jenkinsfiles para los pipelines de CI y los pipelines de CD (las credenciales para el acceso al repo se crean en el Jenkins git credentialsId: 'git' y las del acceso del pipeline a nuestro repositorio de imagenes dockerhub se crearon como credentialsId: "dockerhubaccount. Se incluye tambien un jenkinfile para el ci/cd de los microservicios teniendo en cuenta branching strat y retag, test e image remove.

En el caso de los pipelines de cd ejecutan el archivo terraform que invoca el recurso helm_release y este ejecuta los jenkins files relacionados a cd, estos dentro tienen los steps checkout, terraform init, plan y apply con los nuevos cambios hechos al codigo, es decir se completa el flujo ci/cd, cada vez que un developer hace un update sobre las ramas se corre el pipeline de ci que genera una nueva version de la imagen en el docker hub y luego el pipeline de cd corre el terraform que ejecuta el helm chart y este helm chart llama la imagen actualizada loque hace que se actualicen los pods con los nuevos cambios del aplicativo que hizo el developer.

```
provider "helm" {
  kubernetes {
    config_context_cluster = "minikube"
  }
}

resource "helm_release" "api" {
  name       = "api"
  repository = "file://./app"
  chart      = "./api"
  namespace  = "default"
}
```

PD: Steps para instalar prometheus, grafana via helm charts en minikube:

grafana:

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

helm repo update

kubectl create namespace monitoring

kubectl expose service grafana --type=NodePort --target-port=3000 --name=grafana-np --namespace monitoring

kubectl expose service grafana --type=NodePort --target-port=3000 --name=grafana-np --namespace monitoring

minikube service grafana-np -n monitoring

kubectl get secret --namespace monitoring grafana -o jsonpath="{.data.admin-password}" | base64 --decode ; echo

prometheus:

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

helm repo update

kubectl create namespace monitoring

helm install prometheus prometheus-community/prometheus --namespace monitoring

kubectl expose service prometheus-server --type=NodePort --target-port=9090 --name=prometheus-server-np --namespace monitoring

minikube service prometheus-server-np -n monitoring