PROMETHEUS

* Configuración de la aplicación para pruebas con prometheus y Grafana

<https://github.com/osmergit/metricas>

pueden crear la imagen y subirla a Docker-hub

* Archivos de Configuración Aplicación “Manifiestos”

Ingrese a la carpeta deployments

<https://github.com/osmergit/kubernetes.git>

kubectl apply –f

deployprometheus.yml

lbprot.yml

* **Instalación:**

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

helm repo update

* Archivos de Configuración Prometheus “Manifiestos”

Ingrese a la carpeta prometheus

https://github.com/osmergit/kubernetes.git

kubectl apply –f

final.yml

servicio.yml

prom-svc.yml

* Arranque Prometheus

kubectl port-forward svc/prometheus-service 9090:9090

Grafana

Instalación

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

helm repo update

Arranque Grafana

kubectl port-forward svc/prometheus-grafana -n monitoring 3000:80

usuario: admin

password: prom-operator

Buscamos el puerto por el cual corre prometheus internamente

kubectl get svc prometheus-service

Aca buscamos la ip del nodo

kubectl get nodes -o wide

Para ver las métricas deben ponerle tráfico a la aplicación ingresen a un contenedor de un pod

kubectl exec -it hello-v2-6dc57d6dfd-qfggd -c alpine -- /bin/sh

# apk add --update curl

#while sleep 1; do curl -o /dev/null -s -w %{http\_code} <http://svc-metricas:3002/>metrics ; done

QUERY PROTHEUS Y GRAFANA

rate(http\_requests\_total[1m])

Peticiones http por método

sum by (method) (rate(http\_requests\_total[1m]))

process\_cpu\_user\_seconds\_total

sum(process\_resident\_memory\_bytes)

Peticiones por código de estado:

sum by (status\_code) (rate(http\_requests\_total[1m]))

Tasa de peticiones HTTP por método y código de estado

rate(http\_requests\_total[1m])