Clase 3

**Ejercicio 1**

- Vamos a configurar un monitoreo en prometheus y grabana.

- Vamos a editar el archivo :

/etc/docker/daemon.json

y le vamos a agregar los siguiente parametros:

"metrics-addr" : "0.0.0.0:9323",

"experimental" : true

- Vamos a recargar la configuración

sudo systemctl daemon-reload ; sudo systemctl restart docker.service

- Vamos a iniciar docker en modo swarm

docker swarm init

- Vamos a obtener la ip de nuestra maquina.

Ifconfig

\*Importante: No puede cambiar la ip de nuestra maquina mientras hacemos estos ejercicios sino fallaran.

- Vamos a crear la carpeta para el config de prometheus

mkdir -p /data/prometheus

- Vamos a copiar el archivo prometheus.yml que se encuentra dentro de la carpeta del ejercicio a la ruta creada anteriormente.

- Vamos a crear el container de prometheus

docker service create --replicas 1 --name my-prometheus --mount type=bind,source=/data/prometheus/config/prometheus.yml,destination=/etc/prometheus/prometheus.yml --publish published=9090,target=9090,protocol=tcp prom/prometheus

- Vamos a verificar que haya levantado el container

docker ps

- Vamos a acceder por web para ver si se conecto prometehus

<http://localhost:9090/targets/>.

Tiene que aparecer algo así:



- Vamos a crear el container para grafana

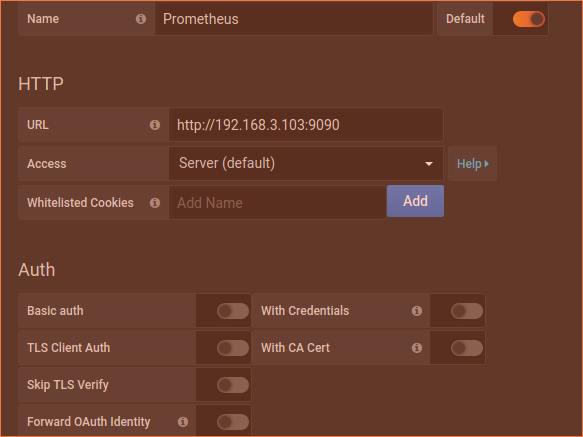
docker run -dit --name=grafana -p 3000:3000 grafana/grafana

- Vamos a aceder via web por

[http://localhost:3000](http://localhost:3000/)

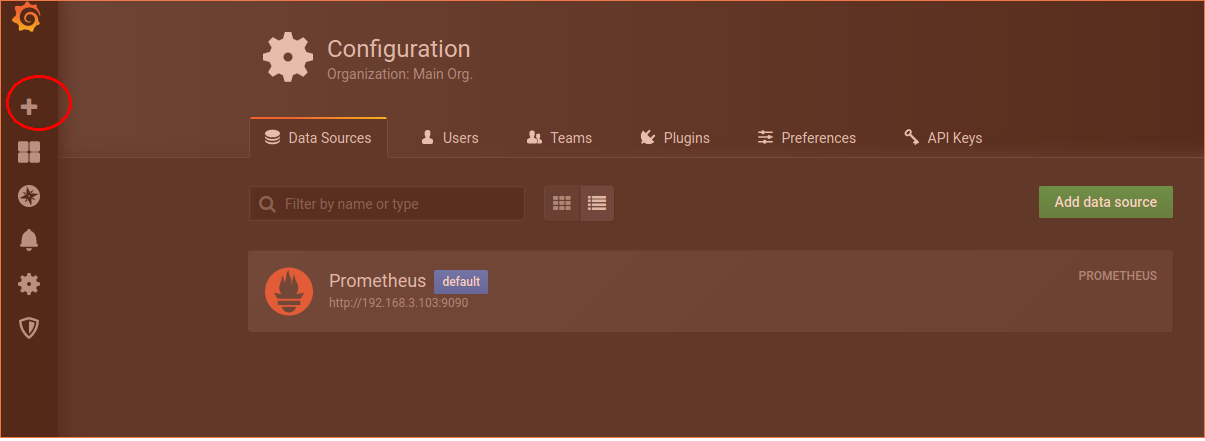
user: admin – pass: admin

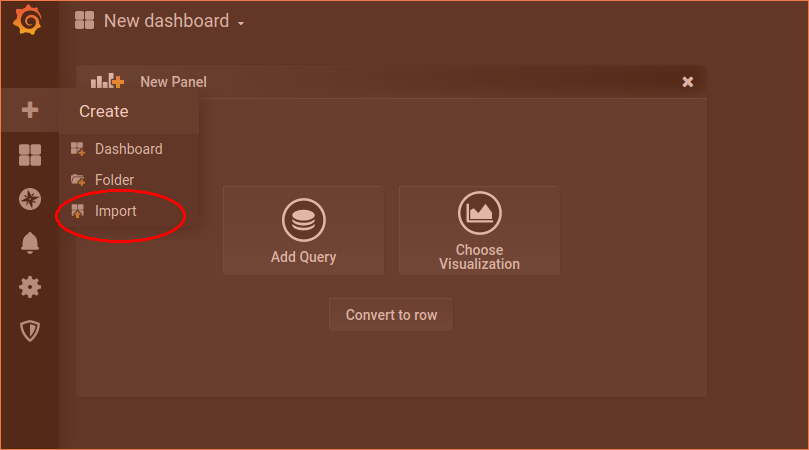
- Vamos a configurar el datasource de la siguiente manera, reemplazando por la ip de nuestra maquina.

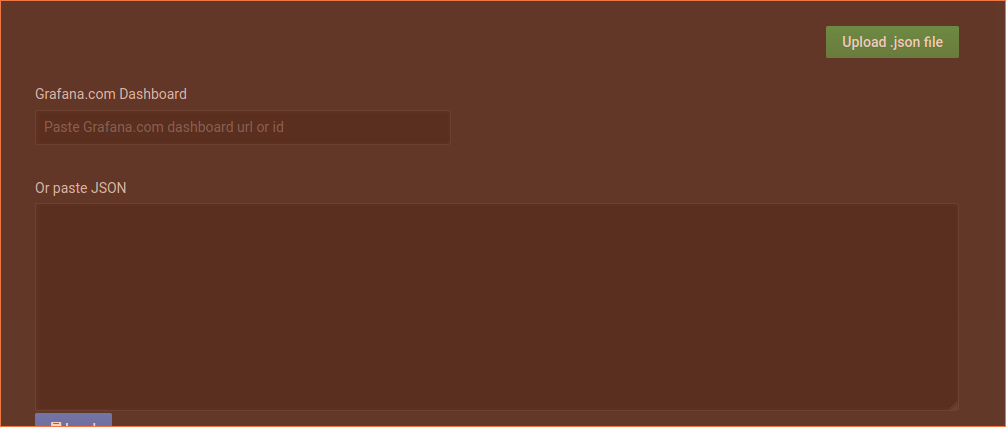


- Vamos a importar el gráfico para docker siguiendo los pasos.

Ir a la solapa izquierda y apretar el boton import, como se muestra en las imágenes:



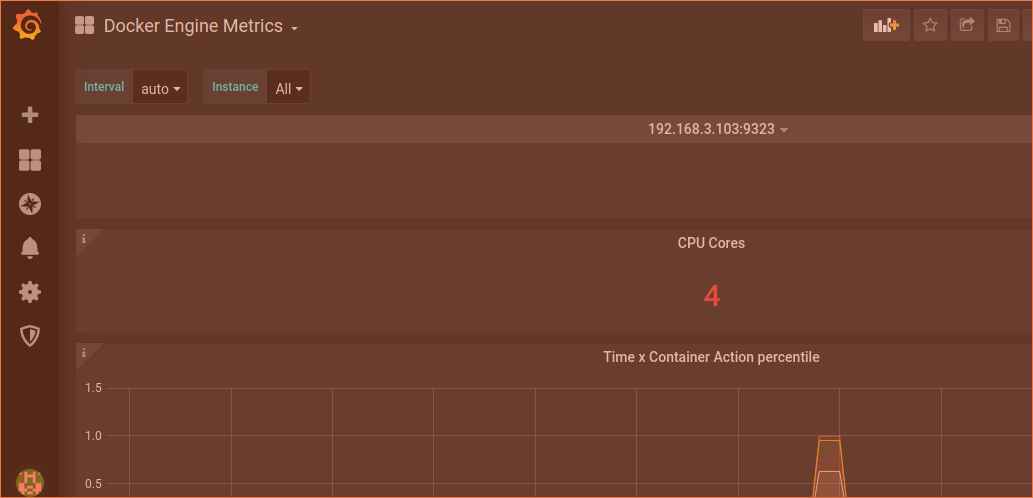




Importar el archivo suministrado llamado :

docker-engine-metrics\_rev3.json

Luego de apretar cargar, les aparecerá el gráfico :



**Ejercicio 2**

- Vamos a crear un cron job

- Vamos a levantar minikube

minikube start

- Vamos copiar el archivo cronjob.yaml a la consola y luego vamos a ejecutar:

kubectl apply -f cronjob.yaml

- Vamos a ver si se creo el cronjob :

kubectl get cronjob –all-namespaces

- Vamos a ver la descripción del cronjob :

kubectl describe cronjob hello

- Vamos a borrar el cronjob :

kubectl delete cronjob hello

**Ejercicio 3**

- Vamos a crear una ReplicaSet

- Vamos a levantar minikube

minikube start

- Vamos a copiar el archivo replicaset.yaml y ejecutar el siguiente comando

kubectl apply -f replicaset.yaml

- Vamos a buscar la ejecución

kubectl get rs

- Vamos a buscar los pods creados

kubectl get pods

- Vamos a borrar uno de los pods y ver que pasa

kubectl delete <pod nombre>

- Vamos a buscar el nombre del rs para luego borrarlo

kubectl get rs

kubectl delete rs <nombre>

**Ejercicio 4**

- Vamos a crear una DaemonSet

- Vamos a levantar minikube

minikube start

- Vamos a copiar el archivo daemonset.yaml y ejecutar el siguiente comando

kubectl apply -f daemonset.yaml

- Vamos a buscar la ejecución

kubectl get ds --all-namespaces

- Vamos a buscar los pods creados

kubectl get pods --all-namespaces

- Vamos a borrar uno de los pods y ver que pasa

kubectl delete <pod nombre> -n kube-system

- Vamos a buscar el nombre del ds para luego borrarlo

kubectl get ds –all-namespaces

kubectl delete ds fluentd-elasticsearch -n kube-system

**Ejercicio 5**

- Vamos a crear un AutoScaling

- Vamos a levantar minikube

minikube start

- Vamos a copiar el archivo pod.yaml en la carpeta autoscaling y ejecutar el siguiente comando

kubectl apply -f pod.yaml

- Vamos a ver si se creo el pod

kubectl get pods

- Vamos a crear el crecimiento por cpu

kubectl autoscale deployment php-apache --cpu-percent=50 --min=1 --max=10

- Vamos a ver el actual estado

kubectl get hpa

- Ahora vamos a crear un nuevo pod que le genere carga al anterior para probar el crecimiento

kubectl run --generator=run-pod/v1 -it --rm load-generator --image=busybox /bin/sh

#Hit enter for command prompt

while true; do wget -q -O- http://php-apache.default.svc.cluster.local; done

- Abrimos otra terminal y vemos el consumo

kubectl get hpa --watch

- Al no tener mas nodos, no veremos el crecimiento.

- Vamos a borrar el scaling y el pod

kubectl delete hpa php-apache

kubectl delete pods --all

**Ejercicio 6**

- Limpieza de kubernetes.

- Segun el proveedor que usemos, hay diferentes opciones de limpieza. Para nuestro ejercicio vamos a crear nuestro propio recolector de basura

- Vamos a crear nuestra imagen desde el dockerfile que esta en la carpeta cleanup

\* Revisar ejercicios anteriores

docker build -t "cleanup:01" .

- Luego vamos a enviarla a docker hub.

\* Revisar ejercicios anteriores

- Vamos a crear el secret como hicimos en ejercicios anteriores para poder traer la imagen de docker hub.

\* Revisar ejercicios anteriores

kubectl create secret docker-registry regcred --docker-server=<your-registry-server> --docker-username=<your-name> --docker-password=<your-pword> --docker-email=<your-email>

- Vamos a crear la tarea. Reemplazar por el nombre de la imagen que pusimos nosotros en el archivo yaml

kubectl apply -f cronjob.yaml

- Ver el cronjob creado

kubectl get cronjob

- Ver los pods creadores

kubectl get pods

- Ver los logs de los pods creadores

kubectl logs <nombre pods>

- Vamos a borrar la tarea

kubectl delete cronjob <nombre cronjob>