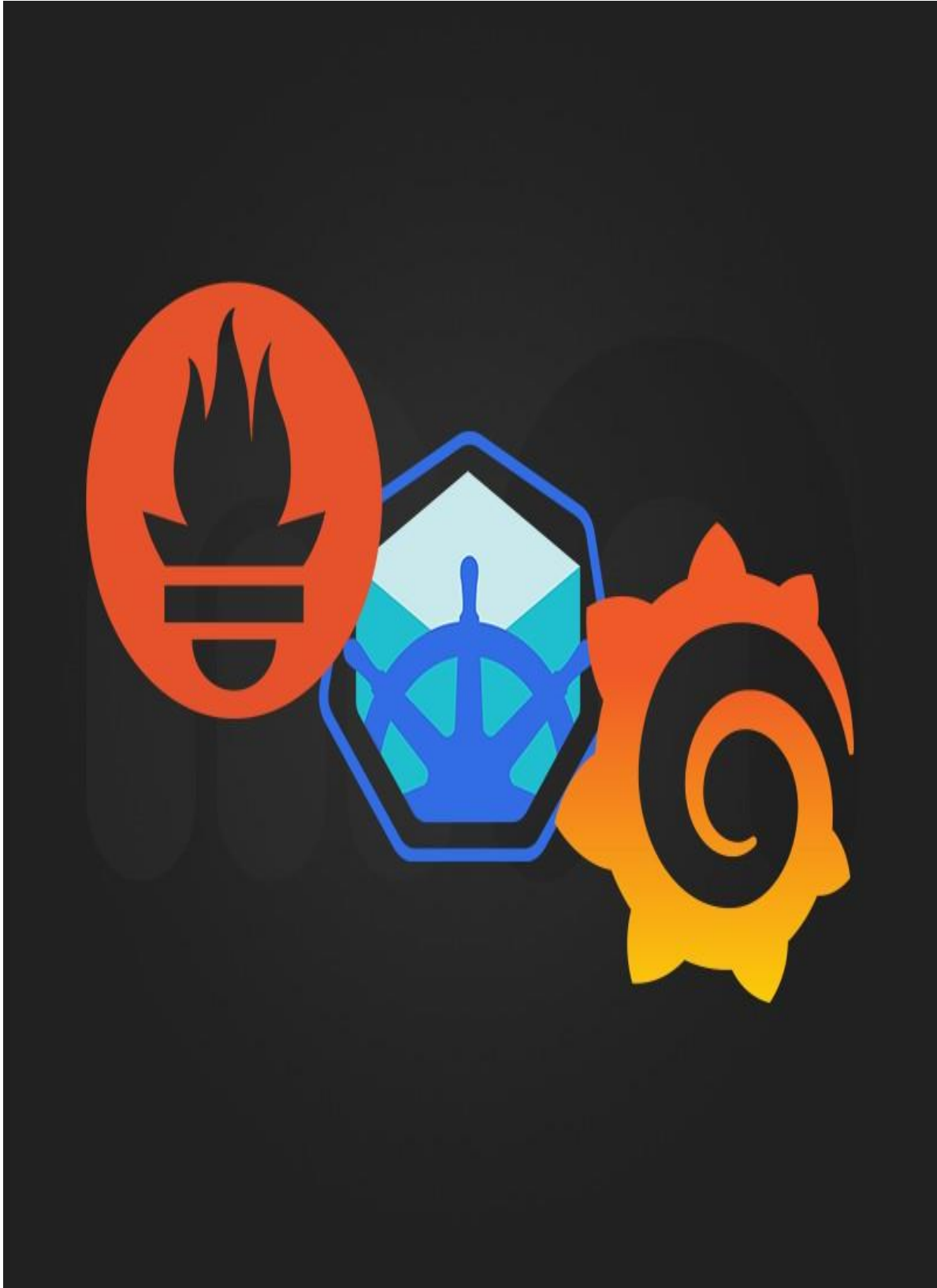


**Belhadj Walid**  
**Sicom 2101413**

**En deux documents**

**Superviser Kubernetes avec Prometheus et Grafana**

**Superviser Kubernetes avec Prometheus et Grafana**



Problématique de la supervision :

## **1 Installation VM Minikube**

Création VM

Installer sur cette VM Centos 8 comme dans le TP5. Installer Docker sur minikube-host

Installer Kubectl sur minikube-host Installation contrack

Installation minikube

Dashboard

Kubernetes

Namespace (Espace de nom) ConfigMap

## **2 Prometheus dans un pod**

Exposer le POD

Service

Kubernetes

Kubectl port-forward

## **3 Grafana avec helm**

Installation de Helm :

Ajouter le repository Grafana

Paramétrage du déploiement

Grafana Ajout de la source de données Importation d'un dashboard

## Problématique de la supervision :

---

- Auto-scaling : élasticité du SI. Par exemple si un container s'arrête un autre redémarre aussitôt (ils sont « jetables »)
- Infrastructure plus complexe (plus de composants interconnectés).
- Les métriques traditionnelles ne suffisent pas à mesurer le SI et sa performance.

L'objectif de ce TP est de vous faire pratiquer les outils de supervision adaptés à ces nouvelles problématiques de supervision, dont font partie [Prometheus](#) et [Grafana](#) pour la partie visualisation.

Vous allez dans un premier temps installer sur une VM un cluster Minikube, puis installer Prometheus et Grafana (sous forme de container).

## 1 Installation VMMinikube

---

### Création VM

---

Créer une vm avec les spécifications suivantes :

- nom : minikube-host
- 4 CPUS
- 3 Go RAM
- 30 Go HDD
- carte réseau en NAT

Installer sur cette VM [Centos 8](#) comme dans le TP5.

---

Reprendre la même configuration pour les users, le type d'installation (minimale), le partitionnement, etc...

Pour gagner du temps et surtout se concentrer sur l'intérêt du sujet vous pouvez pré charger une image virtualbox :

<https://www.linuxvmimages.com/images/centos-8/>

## Installer Docker sur minikube-host

### Installation de yum-utils

```
sudo yum install -y yum-utils
```

```
centos@centos8:~  
File Edit View Search Terminal Help  
[centos@centos8 ~]$ sudo yum install -y yum-utils  
Last metadata expiration check: 0:12:25 ago on Sat 16 Jan 2021 02:32:13 PM IST.  
Dependencies resolved.  
=====
```

Package	Architecture	Version	Repository	Size
Installing: <b>yum-utils</b>	noarch	4.0.17-5.el8	baseos	68 k

```
=====
```

Transaction Summary

Install 1 Package

Total download size: 68 k  
Installed size: 20 k  
Downloading Packages:  
yum-utils-4.0.17-5.el8.noarch.rpm 392 kB/s | 68 kB 00:00

### Ajout du référentiel docker au système centos :

```
sudo yum-config-manager --add-repo  
https://download.docker.com/linux/centos/docker-ce.repo
```

```
[root@centos8 centos]# sudo yum-config-manager --add-repo http://download.docker.com/linux/centos/docker-ce.repo  
Adding repo from: http://download.docker.com/linux/centos/docker-ce.repo  
[root@centos8 centos]#
```

### Activer les référentiel nightly et test

```
sudo yum-config-manager --enable docker-ce-nightly
```

```
Adding repo from: http://download.docker.com/linux/centos/docker-ce.repo  
[root@centos8 centos]# sudo yum-config-manager --enable docker-ce-nightly  
[root@centos8 centos]#
```

### Installer le docker Engine et containerd

```
sudo yum install docker-ce docker-ce-cli containerd.io
```

```
[centos@centos8 ~]$ sudo yum install docker-ce docker-ce-cli containerd.io --nobest --allowdowngrade  
Last metadata expiration check: 0:29:39 ago on Sat 16 Jan 2021 03:33:40 PM IST.  
Dependencies resolved.  
=====
```

Package	Arch	Version	Repository	Size
Installing: <b>containerd.io</b>	x86_64	1.4.3-3.1.el8	docker-ce-stable	33 M
replacing <b>runc</b>	x86_64	1.0.0-68.rc92.module_el8.3.0+475+c50ce30b		
<b>docker-ce</b>	x86_64	3:20.10.2-3.el8	docker-ce-stable	27 M
<b>docker-ce-cli</b>	x86_64	1:20.10.2-3.el8	docker-ce-stable	33 M

```
=====
```

Installing dependencies:

<b>docker-ce-rootless-extras</b>	x86_64	20.10.2-3.el8	docker-ce-stable	9.1 M
<b>libcgroup</b>	x86_64	0.41-19.el8	baseos	70 k

```
=====
```

Removing dependent packages:

## Activation du service docker

```
sudo systemctl enable docker.service
```

```
Complete!
[centos@centos8 ~]$ sudo systemctl enable docker.service
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /usr/lib/systemd/system/docker.service.
[centos@centos8 ~]$
```

## Démarrage de docker

```
sudo systemctl start docker
```

```
[centos@centos8 ~]$ sudo systemctl start docker
[centos@centos8 ~]$
```

**Q1. Vérifier que le docker engine tourne, détailler dans votre compte rendu ce que vous avez fait pour cette vérification.**

```
root@centos8:/home/centos
File Edit View Search Terminal Help
[centos@centos8 ~]$ sudo systemctl start docker
[centos@centos8 ~]$ sudo su
[root@centos8 centos]# systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; vendor preset: disabled)
   Active: active (running) since Sat 2021-01-16 16:07:27 IST; 2min 13s ago
     Docs: https://docs.docker.com
   Main PID: 5681 (dockerd)
    Tasks: 12
   Memory: 60.3M
   CGroup: /system.slice/docker.service
           └─5681 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

Jan 16 16:07:23 centos8.linuxvmimages.local dockerd[5681]: time="2021-01-16T16:07:23.344647762+05:30"
Jan 16 16:07:23 centos8.linuxvmimages.local dockerd[5681]: time="2021-01-16T16:07:23.344676249+05:30"
Jan 16 16:07:23 centos8.linuxvmimages.local dockerd[5681]: time="2021-01-16T16:07:23.344808760+05:30"
Jan 16 16:07:26 centos8.linuxvmimages.local dockerd[5681]: time="2021-01-16T16:07:26.679648009+05:30"
Jan 16 16:07:26 centos8.linuxvmimages.local dockerd[5681]: time="2021-01-16T16:07:26.877659719+05:30"
Jan 16 16:07:27 centos8.linuxvmimages.local dockerd[5681]: time="2021-01-16T16:07:27.282799951+05:30"
Jan 16 16:07:27 centos8.linuxvmimages.local dockerd[5681]: time="2021-01-16T16:07:27.682269979+05:30"
Jan 16 16:07:27 centos8.linuxvmimages.local dockerd[5681]: time="2021-01-16T16:07:27.682639626+05:30"
Jan 16 16:07:27 centos8.linuxvmimages.local systemd[1]: Started Docker Application Container Engine.
Jan 16 16:07:27 centos8.linuxvmimages.local dockerd[5681]: time="2021-01-16T16:07:27.829747610+05:30"
lines 1-20/20 (END)
```

La commande **systemctl status docker** fournit des informations indiquant si le service / processus est en cours d'exécution (**active**), et s'il est **enabled** doit démarrer au démarrage de la machine.

Cela en soi aurait nécessité plusieurs programmes ( chkconfigetc.) pour interroger. Affiche le fichier d'unité par défaut chargé dans la ligne suivante:

**Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; vendor preset: disabled)**

**Status :** active depuis la date d'activation

**Lien pour la documentation :** docs.docker.com

**PID principal numéro 5681 de son daemon (dockerd)**

**Tasks :** nombres des tâches qu'il run

Et informations, des messages utiles sur le PID, la mémoire consommée et occupée, les arguments ( DOCKER\_OPTS) et les 10 dernières lignes (aka tail) sont aussi

affichés.

## Ajouter votre user au groupe docker

```
sudo usermod -aG docker m1xxx && newgrp docker
```

```
vboxadd:x:974:1::/var/run/vboxadd:/bin/false
m1wbe:x:1001:1001::/home/m1wbe:/bin/bash
[root@centos8 home]#
```

## Installer Kubectl sur minikube-host

Télécharger la dernière version de kubectl :

```
sudo curl -LO https://storage.googleapis.com/kubernetes-release/release/$(curl -s https://storage.googleapis.com/kubernetes-release/release/stable.txt)/bin/linux/amd64/kubectl
```

```
[m1wbe@centos8 ~]$ curl -LO https://storage.googleapis.com/kubernetes-release/release/$(curl -s https://storage.googleapis.com/kubernetes-release/release/stable.txt)/bin/linux/amd64/kubectl
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           %             %             Dload  Upload  Total   Spent    Left     Speed
100 38.3M  100 38.3M    0     0 3911k      0  0:00:10  0:00:10 --:--:-- 3825k
[m1wbe@centos8 ~]$
```

Attribuez les droits d'exécution et déplacez le programme kubectl dans /usr/local/bin (cf TP4)

```
[root@centos8 home]# sudo chmod +x ./kubectl
[root@centos8 home]# mv ./kubectl /usr/local/bin/kubectl
[root@centos8 home]#
```

## Installation conntrack

sudo yum install conntrack

```
[root@centos8 home]# sudo yum install conntrack
Last metadata expiration check: 2:10:35 ago on Sat 16 Jan 2021 03:33:40 PM IST.
Dependencies resolved.
=====
Package                Arch      Version      Repository    Size
=====
Installing:
conntrack-tools         x86_64    1.4.4-10.el8 baseos        204 k
Installing dependencies:
libnetfilter_cthelper   x86_64    1.0.0-15.el8 baseos         24 k
libnetfilter_cttimeout  x86_64    1.0.0-11.el8 baseos         24 k
libnetfilter_queue      x86_64    1.0.4-3.el8  baseos         31 k
Transaction Summary
=====
Install 4 Packages
```

**Q2. Faites une recherche sur l'outil conntrack et notez dans votre rapport à quoi sert ce composant.**

**Conntrack** est une fonctionnalité essentielle de la pile réseau du noyau Linux. Il permet au noyau de garder une trace de toutes les connexions ou flux réseau logiques, et ainsi d'identifier tous les paquets qui composent chaque flux afin qu'ils puissent être traités ensemble de manière cohérente.

De plus, conntrack améliore normalement les performances (processeur

réduit et latences de paquets réduites) car seul le premier paquet d'un flux doit passer par le traitement complet de la pile réseau pour déterminer ce qu'il faut en faire. Voir le blog « Comparaison des modes kube-proxy » pour un exemple de cela en action.

Le suivi des connexions est la base de nombreux services et applications réseau telque Kubernetes Service,

Conntrack permet également d'interroger, de modifier ou de détruire des entrées du suivi de connexions, et il permet aussi de garder en mémoire l'état de connexions réseau (réaliser un firewall statefull(avec état)).Il peut être utilisé pour activer la haute disponibilité des clusters basés sur des firewalls avec état (statefull) ainsi de collecter des statistiques sur l'utilisation des firewalls.

## **Installation minikube**

---

[Minikube](#) est un outil qui va faire tourner un cluster Kubernetes à un noeud unique dans une machine virtuelle.

Ça permet d'émuler un environnement Kubernetes sans trop de ressources.

C'est très utile pour les développer pour tester des déploiements et application sur un environnement isolé.

Récupérer la dernière version de minikube

```
sudo curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64
```

Installer minikube

```
[root@centos8 home]# sudo curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload   Total       Spent    Left     Speed
100 55.2M  100 55.2M    0     0  558k      0  0:01:41  0:01:41 --:--:-- 1047k
[root@centos8 home]# sudo install minikube-linux-amd64 /usr/local/bin/minikube
[root@centos8 home]#
```

Démarrez minikube (attention vous ne devez pas être root)

```
minikube start --driver=docker
```

```
[centos@centos8 ~]$ su mlwbe
Password:
[mlwbe@centos8 centos]$ minikube start --driver=docker
😄 minikube v1.16.0 on Centos 8.3.2011 (vbox/amd64)
👉 Using the docker driver based on user configuration

🔧 The requested memory allocation of 2200MiB does not leave room for system overhead (total system memory: 2824MiB). You may face stability issues.
💡 Suggestion: Start minikube with less memory allocated: 'minikube start --memory=2200mb'

👍 Starting control plane node minikube in cluster minikube
📦 Pulling base image ...
📦 Downloading Kubernetes v1.20.0 preload ...
   > preloaded-images-k8s-v8-v1....: 28.23 MiB / 491.00 MiB  5.75% 2.26 MiB p/
```

La commande peut prendre un certain temps à s'exécuter. Attendez qu'elle soit finie avant de continuer.

Vérifier l'état de votre mini cluster avec la commande

```
minikube status
```

**Q3. Quel est le résultat ? Commentez.**

```
[mlwbe@centos8 centos]$ minikube status
minikube
type: Control Plane
host: Running
kubelet: Running
apiserver: Running
kubeconfig: Configured
timeToStop: Nonexistent
[mlwbe@centos8 centos]$
```

**Commentaire :**

Cette commande permet de nous montrer l'état du cluster minikube donc il nous montre que minikube est bien configuré et en état de marche et sans temps d'arrêt précis,

Le host et sur quel api serveur que minikube est exécuté sont aussi en état run aussi

Donc il est opérationnel,



le paramètre pour limité la mémoire :c'est le paramètre memory '--memory'.•Les nodes et les pods :Pour les pods'kubectlgetpods--all-namespaces'et pour lesnodes'kubectlgetnodes'.

**Répondez aux questions suivantes, en recherchant les réponses sur le site documentaire de minikube.**

**- A quoi correspond le paramètre driver de la commande minikube ?**

A la base minikube a un driver propre à lui mais qui est virtual machine, dans notre cas, minikube va être déployer avec le pilote docker container.

le paramètre **driver** permet de spécifier le nom de l'hyperviseur qu'on a installé, comme dans notre cas c'est le docker .

**- Quels paramètres de la commande minikube seraient utilisés pour limiter la mémoire allouée au cluster.**

**Limits --memory**

**Dans le fichier** memory-constraints.yaml

```
resources:
  limits:
    memory: 800Mi
  requests:
    memory: 600Mi
```

**- Quelle commande vous permet de voir les nodes et les pods de votre mini-cluster ?**

kubectl get pods

Kubectl get nodes mini-cluster

```
no resources found in default namespace.
[m1wbe@centos8 centos]$ kubectl get nodes
NAME        STATUS    ROLES                  AGE    VERSION
minikube    Ready     control-plane,master   123m   v1.20.0
[m1wbe@centos8 centos]$
```

## Dashboard Kubernetes

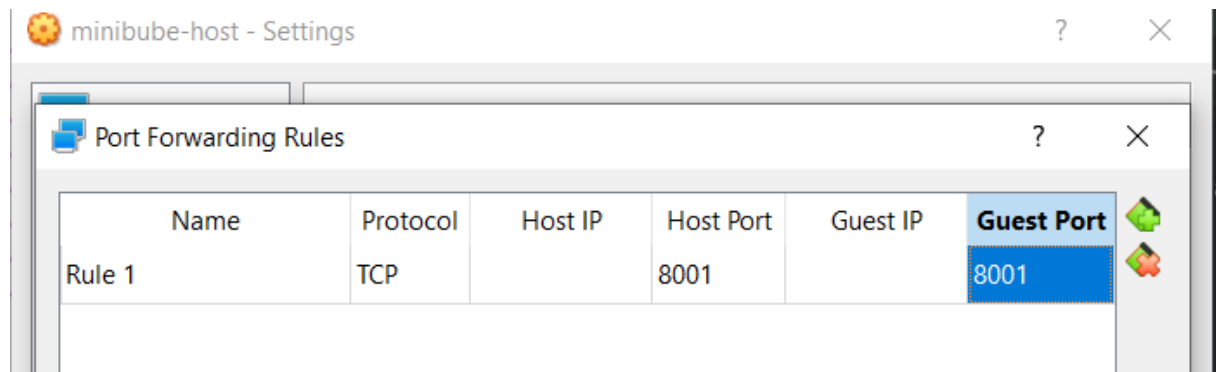
Pour accéder au dashboard Kubernetes, il faut utiliser le service mandataire de kubectl. Exécuter la commande suivante :

```
kubectl proxy --address 0.0.0.0 &
```

```
[mlwbe@centos8 centos]$ kubectl proxy --address 0.0.0.0 &
[1] 41280
[mlwbe@centos8 centos]$ Starting to serve on [::]:8001

[mlwbe@centos8 centos]$ kubectl config view
apiVersion: v1
clusters:
- cluster:
    certificate-authority: /home/mlwbe/.minikube/ca.crt
    server: https://192.168.49.2:8443
    name: minikube
contexts:
- context:
    cluster: minikube
    namespace: default
    user: minikube
    name: minikube
current-context: minikube
kind: Config
preferences: {}
users:
- name: minikube
  user:
    client-certificate: /home/mlwbe/.minikube/profiles/minikube/client.crt
    client-key: /home/mlwbe/.minikube/profiles/minikube/client.key
[mlwbe@centos8 centos]$
```

Ajuster la propagation des ports de votre VM en fonction du port qui a été ouvert par le mandataire kubectl.



```
client-key: /home/mlwbe/.minikube/profiles/minikube/client.key
[mlwbe@centos8 centos]$ kubectl apply -f https://raw.githubusercontent.com/kubernetes/dashboard
rd/v2.0.0/aio/deploy/recommended.yaml
namespace/kubernetes-dashboard created
serviceaccount/kubernetes-dashboard created
service/kubernetes-dashboard created
secret/kubernetes-dashboard-certs created
secret/kubernetes-dashboard-csrf created
secret/kubernetes-dashboard-key-holder created
configmap/kubernetes-dashboard-settings created
role.rbac.authorization.k8s.io/kubernetes-dashboard created
clusterrole.rbac.authorization.k8s.io/kubernetes-dashboard created
rolebinding.rbac.authorization.k8s.io/kubernetes-dashboard created
clusterrolebinding.rbac.authorization.k8s.io/kubernetes-dashboard created
deployment.apps/kubernetes-dashboard created
service/dashboard-metrics-scraper created
deployment.apps/dashboard-metrics-scraper created
[mlwbe@centos8 centos]$
```

## Ajout d'un admin-user

```
[mlwbe@centos8 centos]$ cat <<EOF | kubectl apply -f -
> apiVersion: v1
> kind: ServiceAccount
> metadata:
>   name: admin-user
>   namespace: kubernetes-dashboard
> EOF
serviceaccount/admin-user created
[mlwbe@centos8 centos]$
```

## Création d'un clusterRoleBinding

```
[mlwbe@centos8 centos]$ cat <<EOF | kubectl apply -f -
> apiVersion: rbac.authorization.k8s.io/v1
> kind: ClusterRoleBinding
> metadata:
>   name: admin-user
> roleRef:
>   apiGroup: rbac.authorization.k8s.io
>   kind: ClusterRole
>   name: cluster-admin
> subjects:
> - kind: ServiceAccount
>   name: admin-user
>   namespace: kubernetes-dashboard
> EOF
clusterrolebinding.rbac.authorization.k8s.io/admin-user unchanged
[mlwbe@centos8 centos]$
```

### Obtention d'un jeton :

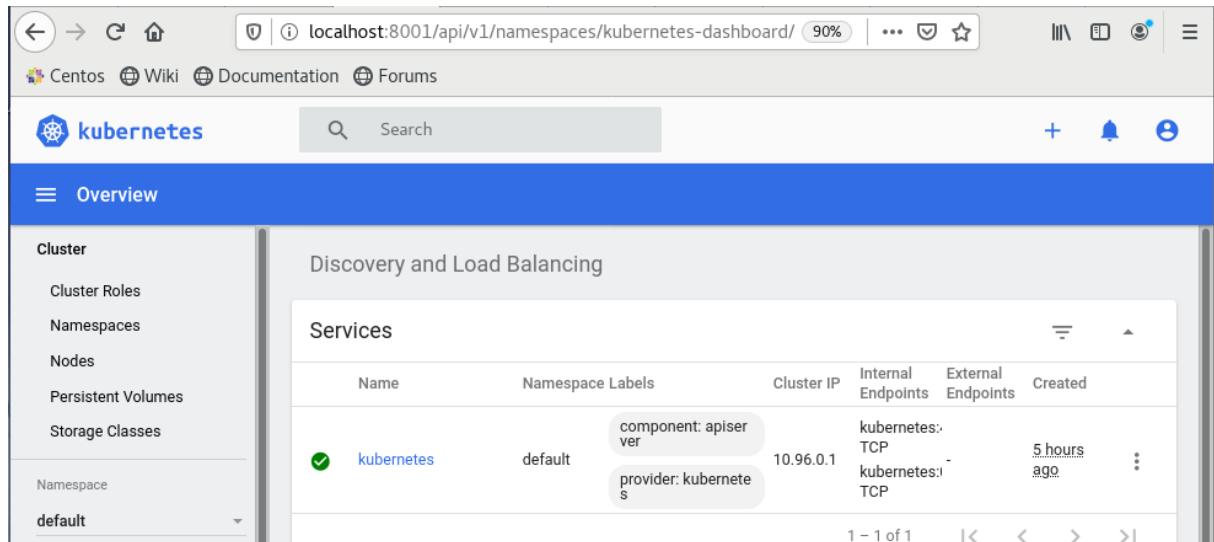
[illegible]

Accès page

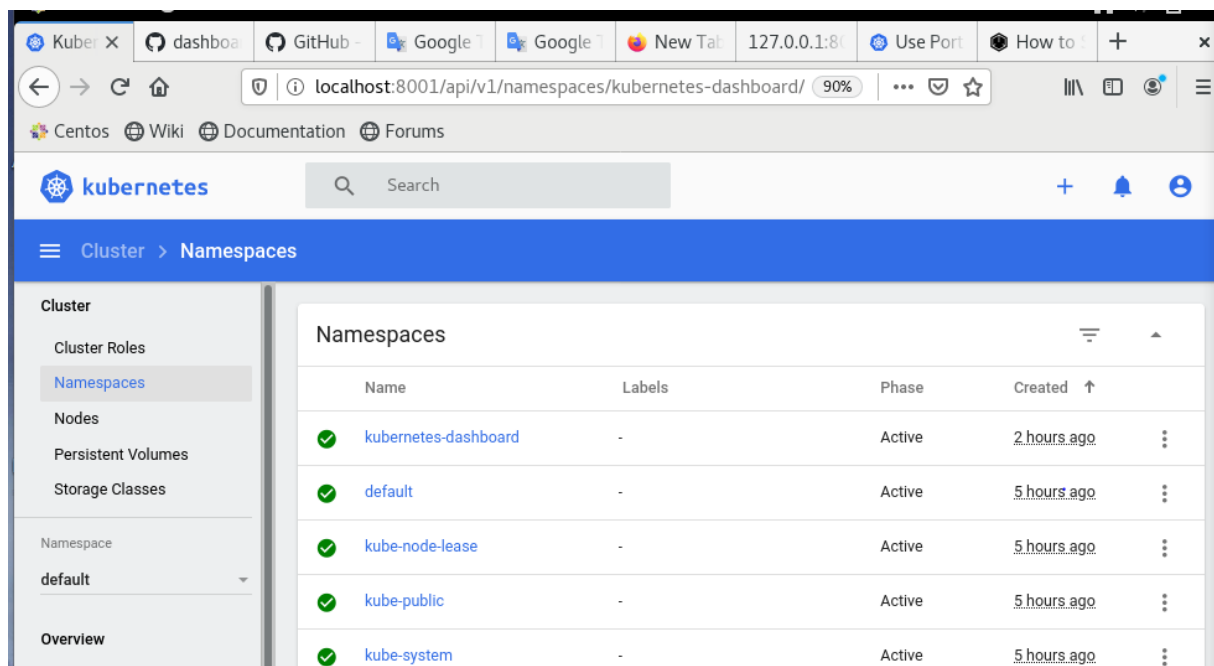
Ouvrez un navigateur sur votre host à l'URL suivante :

The screenshot shows the Kubernetes Dashboard interface. The browser address bar indicates the URL is `localhost:8001/api/v1/namespaces/kubernetes-dashboard/services/https/kubernetes-dashboard/proxy/#/namespaces`. The left sidebar contains a navigation menu with the following items: Cluster, Cluster Roles, Namespaces (highlighted), Nodes, Persistent Volumes, Storage Classes, Namespace (dropdown), default (selected), Overview, Workloads, and Cron Jobs. The main content area is titled "Namespaces" and displays a table with the following data:

Name	Labels	Phase	Created ↑
monitoring	-	Active	5 hours ago
kubernetes-dashboard	-	Active	9 hours ago
default	-	Active	12 hours ago
kube-node-lease	-	Active	12 hours ago
kube-public	-	Active	12 hours ago
kube-system	-	Active	12 hours ago



**Q4. Quels sont les Espaces de nom présents sur votre cluster ?**

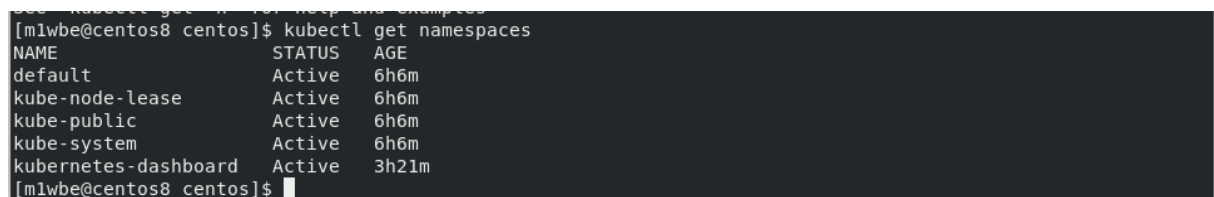


**Namespace (Espace de nom)**

La notion de namespace permet de cloisonner les différents pods qui vont être exécuté sur le cluster minikube, afin de contrôler les interactions.

**Q5. Trouver la commande kubectl qui permet d'afficher tous les namespaces de votre cluster. Quels sont les namespaces présents ?**

**Kubectl get --all-namespaces**



```
[mlwbe@centos8 centos]$ kubectl get pods --all-namespaces
NAMESPACE      NAME                                     READY   STATUS    RESTARTS   AGE
kube-system    coredns-74ff55c5b-g4zlw                1/1     Running   0          6h
kube-system    etcd-minikube                           1/1     Running   0          6h1m
kube-system    kube-apiserver-minikube                 1/1     Running   0          6h1m
kube-system    kube-controller-manager-minikube        1/1     Running   0          6h1m
kube-system    kube-proxy-tzdx2                        1/1     Running   0          6h
kube-system    kube-scheduler-minikube                 1/1     Running   0          6h1m
kube-system    storage-provisioner                     1/1     Running   4          6h1m
kubernetes-dashboard  dashboard-metrics-scraper-7b59f7d4df-pfvj5  1/1     Running   0          59m
kubernetes-dashboard  kubernetes-dashboard-74d688b6bc-dtw2v        1/1     Running   0          59m
[mlwbe@centos8 centos]$
```

**Les namespaces présents sont :**

Nous allons créer un namespace pour les outils de monitoring. Créer un répertoire **/etc/namespaces/monitoring** et créer le fichier **monitoring-namespace.yaml** dans ce répertoire, avec le contenu suivant :

```
---
apiVersion: v1
kind: Namespace
metadata:
  name: monitoring
```

**\*Note : attention à l'indentation\***, le format yaml prend en compte les espaces ! Exécuter la commande suivante :

```
kubectl apply -f /etc/namespaces/monitoring/monitoring-namespace.yaml
```

Vérifier avec la commande trouvée auparavant que votre namespace monitoring a été créé.

```
[mlwbe@centos8 monitoring]$ kubectl apply -f /etc/namespaces/monitoring/monitoring-namespace.yaml
namespace/monitoring created
[mlwbe@centos8 monitoring]$

[mlwbe@centos8 monitoring]$ kubectl get namespaces
NAME                STATUS    AGE
default             Active    6h58m
kube-node-lease     Active    6h58m
kube-public          Active    6h58m
kube-system          Active    6h58m
kubernetes-dashboard  Active    4h13m
monitoring           Active    31m
[mlwbe@centos8 monitoring]$
```

## ConfigMap

La ConfigMap est une fonctionnalité de kubernetes qui permet de définir les éléments de configuration de l'application

Le pod ou container, lui possèdera une référence à cette configmap qu'il chargera à son démarrage.

Nous allons ici utiliser une ConfigMap pour le paramétrage de Prometheus, qui

est contenue dans le fichier en pièce jointe à ce sujet, prometheus-config.yaml

Copier le fichier prometheus-config.yaml sur la vm dans le répertoire

/etc/namespaces/monitoring.

```
[mlwbe@centos8 monitoring]$ kubectl get configmap
NAME          DATA  AGE
kube-root-ca.crt 1      7h12m
[mlwbe@centos8 monitoring]$
```

Créer la ConfigMap pour Prometheus :

```
kubectl apply -f /etc/namespaces/monitoring/prometheus-config.yaml
```

```
[mlwbe@centos8 monitoring]$ kubectl apply -f /etc/namespaces/monitoring/prometheus-config.yaml
configmap/prometheus-config created
[mlwbe@centos8 monitoring]$
```

Vérifier que la ConfigMap a bien été créée :

```
kubectl get configmap --namespace=monitoring prometheus-config -o yaml
```

```
f:annotations:
  .: {}
  f:kubectl.kubernetes.io/last-applied-configuration: {}
manager: kubectl-client-side-apply
operation: Update
time: "2021-01-16T19:47:39Z"
name: prometheus-config
namespace: monitoring
resourceVersion: "17904"
uid: 9190769d-49c9-416e-9122-d35a0ba99f4b
[mlwbe@centos8 monitoring]$
```

**Q6. Quel est l'uid et la resourceVersion de la ConfigMAP ?**

**resourceVersion : 17904**

**uid : est dans la dernière ligne**

## 2 Prometheus dans un pod

[Prometheus](#) est un outil de collecte de métriques et de déclenchement d'alertes. Il utilise un modèle de données de type clé/valeur et une base de données TSDB, comme Graphite.

Vous allez exécuter Prometheus dans un pod, en utilisant le fichier de déploiement joint avec ce TP (prometheus-deployment.yaml).

**Q7. Combien de Replicas est attribué à ce déploiement ? Quel est le port d'écoute du déploiement de ce pod Prometheus ? Quelle est la durée de rétention des données dans la base TSDB ?**

Une seule réplication

```
spec:
  replicas: 1
  selector:
    matchLabels:
```

Port : 9090

```
name: prometheus
ports:
- containerPort: 9090
```

Quelle est la durée de rétention des données dans la base TSDB : 24h

```
containers:
- args:
  - --config.file=/etc/prometheus/prometheus.yml
  - --storage.tsdb.path=/prometheus
  - --storage.tsdb.retention=24h
command:
  /bin/prometheus
```

Note : dans ce fichier, il est fait référence à la ConfigMap « prometheus config » créée précédemment, il est donc important de respecter ces noms.

Editez ce fichier de déploiement pour changer  
l'image par :

**quay.io/prometheus/prometheus:v2.23.0**

Créer le déploiement :

```
kubectl apply -f /etc/namespaces/monitoring/prometheus-deployment.yaml
```

```
[mlwbe@centos8 monitoring]$ sudo vim prometheus-deployment.yaml
[mlwbe@centos8 monitoring]$ kubectl apply -f /etc/namespaces/monitoring/prometheus-deployment.yaml --validate
lse
clusterrole.rbac.authorization.k8s.io/prometheus created
serviceaccount/default unchanged
clusterrolebinding.rbac.authorization.k8s.io/prometheus configured
deployment.apps/prometheus unchanged
```

Vérifier que votre déploiement est opérationnel (attention à spécifier namespace)

```
uid: 9190769d-49c9-410e-9122-033a0ba9914b
[mlwbe@centos8 monitoring]$ kubectl get pods --namespace=monitoring
NAME                                READY   STATUS    RESTARTS   AGE
prometheus-6c4fffb969-sjxsd         1/1     Running   0           46m
[mlwbe@centos8 monitoring]$
```

```
[mlwbe@centos8 monitoring]$ kubectl get deployments -n monitoring
NAME          READY   UP-TO-DATE   AVAILABLE   AGE
prometheus    1/1     1            1           28m
[mlwbe@centos8 monitoring]$
```

Q8. Quelle est la commande que vous utilisez ?

```
uid: 9190769d-49c9-410e-9122-033a0ba9914b
[mlwbe@centos8 monitoring]$ kubectl get pods --namespace=monitoring
NAME                                READY   STATUS    RESTARTS   AGE
prometheus-6c4fffb969-sjxsd         1/1     Running   0           46m
[mlwbe@centos8 monitoring]$
```

## Exposer le POD

---

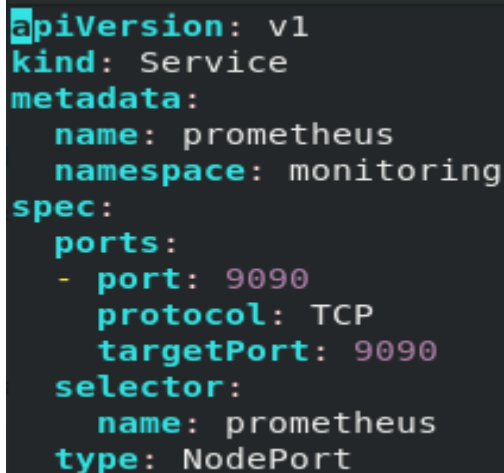
Il reste une chose à faire pour accéder à l'interface de prometheus, c'est de relier un port de la VM au pod. C'est ce que va faire le Kubernetes Service, et le port forwarding kubectl.

### Service Kubernetes

---

Créer le fichier prometheus\_service.yaml dans le répertoire /etc/namespaces/monitoring, avec le contenu suivant (attention à l'indentation!):

```
apiVersion: v1
kind: Service
metadata:
  name: prometheus
  namespace: monitoring
spec:
  ports:
    - port: 9090
      protocol: TCP
      targetPort: 9090
  selector:
    name: prometheus
  type: NodePort
```



```
apiVersion: v1
kind: Service
metadata:
  name: prometheus
  namespace: monitoring
spec:
  ports:
    - port: 9090
      protocol: TCP
      targetPort: 9090
  selector:
    name: prometheus
  type: NodePort
```

Créer le service en appliquant ce fichier de configuration au cluster Minikube.

### Q9. Quelle est la commande utilisée ?



```
monitoring_namespace.yaml prometheus_config.yaml prometheus_deployment.yaml
[m1wbe@centos8 monitoring]$ kubectl apply -f /etc/namespaces/monitoring/prometheus-service.yaml
service/prometheus created
[m1wbe@centos8 monitoring]$
```



## Kubectl port-forward

Exécuter les commandes suivantes :

```
kubectl get pods --namespace=monitoring

kubectl describe pod <nom_du_pod_prometheus>

kubectl get svc --namespace=monitoring
```

```
[mlwbe@centos8 monitoring]$ kubectl get pods --namespace=monitoring
NAME                                READY   STATUS    RESTARTS   AGE
prometheus-6c4fffb969-sjxsd         1/1     Running   0           71m

[mlwbe@centos8 monitoring]$ kubectl describe pod -n monitoring prometheus-6c4fffb969-sjxsd
No resources found in prometheus-6c4fffb969-sjxsd namespace.
[mlwbe@centos8 monitoring]$ kubectl describe pod -n monitoring prometheus-6c4fffb969-sjxsd
Name:                               prometheus-6c4fffb969-sjxsd
Namespace:                           monitoring
Priority:                             0
Node:                               minikube/192.168.49.2
Start Time:                         Sun, 17 Jan 2021 02:17:16 +0530
Labels:                              name=prometheus
                                      pod-template-hash=6c4fffb969
Annotations:                         prometheus.io/port: 9090
                                      prometheus.io/scrape: true
Status:                              Running
IP:                                  172.17.0.2
IPs:                                 IP: 172.17.0.2
Controlled By:                       ReplicaSet/prometheus-6c4fffb969
Containers:
  prometheus:
    Container ID:   docker://bd508023cd61a330f0934f95676483324ed88b80d070e544d6223a4e4fef2e01
    Image:          quay.io/prometheus/prometheus:v2.23.0
    Image ID:       docker-pullable://quay.io/prometheus/prometheus@sha256:0eac377a90d361be9da35b469def699bcd5bb26eab8a6e9068516a9910717d58
    Port:           9090/TCP
```

```
[mlwbe@centos8 monitoring]$ kubectl get svc --namespace=monitoring
NAME            TYPE        CLUSTER-IP      EXTERNAL-IP  PORT(S)          AGE
prometheus      NodePort    10.109.76.182   <none>       9090:30022/TCP   39m
[mlwbe@centos8 monitoring]$
```

### Q10. Et répondez aux questions :

Quelle est l'adresse IP du pod Prometheus ?

**172.17.0.2**

Et le port exposé par le pod ?

**9090**

Quelle est l'adresse IP du Node qui fait tourner le pod Prometheus ? (Le Node est l'équivalent du docker engine) ?

**192.168.49.2 dans le même réseau que la VM**

Quelle est l'adresse IP du Service Prometheus exposé ?

**10.109.76.182**

Quel est le port le pod et sur le node ? sur le pod **9090**

## Sur le node 30022

Pour exposer le port du service Prometheus sur la VM, il faut propager le port du node sur la VM avec la commande suivante :

```
[centos@centos8 ~]$ ifconfig
br-eec8a4c17149: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.49.1 netmask 255.255.255.0 broadcast 192.168.49.255
    inet6 fe80::42:9fff:fe82:36a1 prefixlen 64 scopeid 0x20<link>
    ether 02:42:9f:82:36:a1 txqueuelen 0 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 28 bytes 3721 (3.6 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

```
kubectl port-forward svc/prometheus --address <IP VM> --namespace=monitoring
<NODEPORT>:9090
```

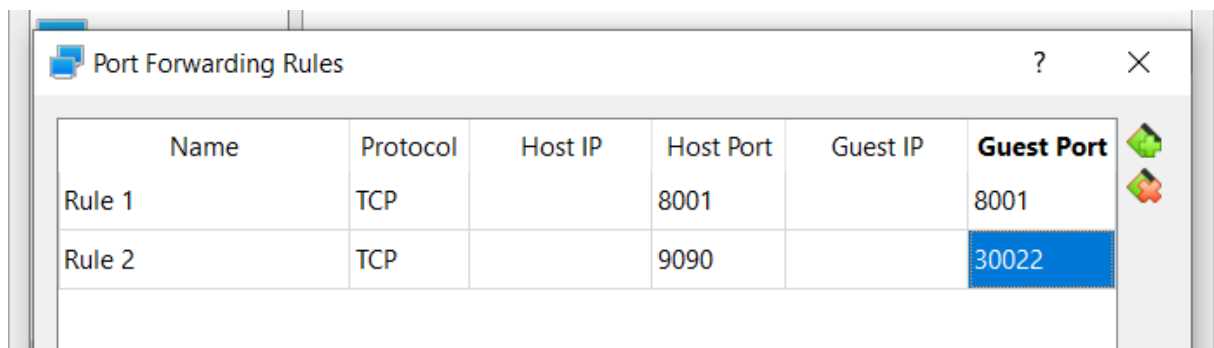
```
[mlwbe@centos8 monitoring]$ kubectl port-forward svc/prometheus --address 0.0.0.0 --namespace=monitoring 30022:9090
Forwarding from 0.0.0.0:30022 -> 9090
Handling connection for 30022
Handling connection for 30022
Handling connection for 30022
```

Vérifier que le Node Port est bien ouvert sur la VM avec la commande

```
netstat -an | grep <NodePort>
```

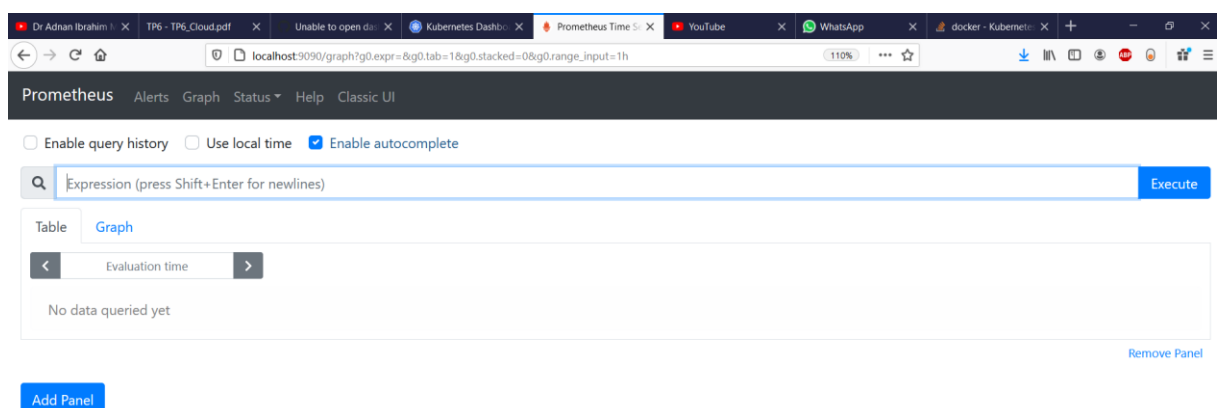
```
[mlwbe@centos8 centos]$ netstat -an | grep 30022
tcp        0      0 192.168.49.1:30022 0.0.0.0:*        LISTEN
[mlwbe@centos8 centos]$
```

**Ajuster la propagation de port de votre VM pour que le port 9090 de votre host soit redirigé sur le NodePort de votre VM.**



Name	Protocol	Host IP	Host Port	Guest IP	Guest Port
Rule 1	TCP		8001		8001
Rule 2	TCP		9090		30022

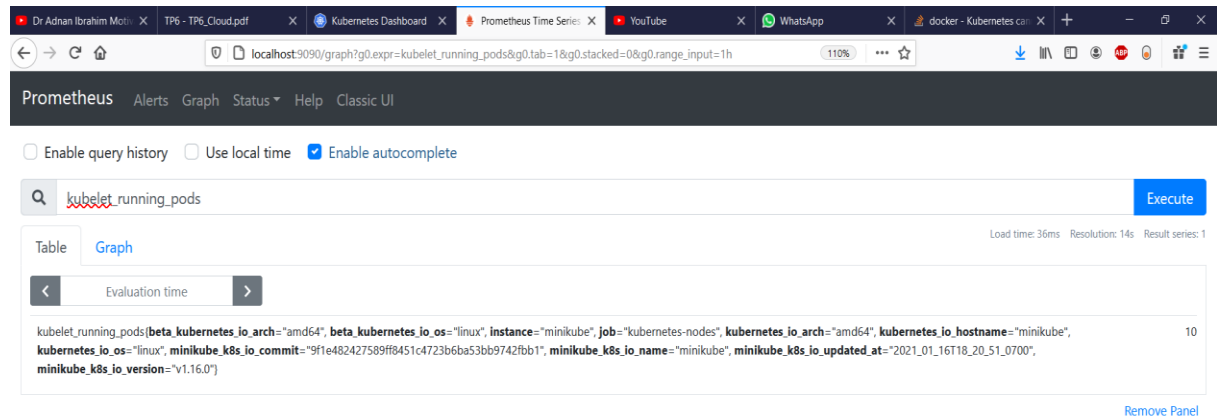
Ouvrez un navigateur à l'URL : **localhost:9090**. Vous devriez avoir la page d'accueil de Prometheus



Dans la zone Expression, saisissez « kubelet\_ » puis explorez la liste des métriques pour trouver celle qui contient le nombre de pods en cours d'exécution.

Le langage d'interrogation de Prometheus est le PromQL (<https://prometheus.io/docs/introduction/overview/>)

### Q11. Exécuter la query et commentez le résultat dans votre rapport.



The screenshot shows the Prometheus web interface. The query bar contains 'kubelet\_running\_pods'. The results are displayed in a table format, showing 10 pods. Each row represents a pod with various labels and values. The labels include 'beta.kubernetes.io.arch', 'beta.kubernetes.io.os', 'instance', 'job', 'kubernetes.io.arch', 'kubernetes.io.hostname', 'kubernetes.io.os', 'minikube.k8s.io.commit', 'minikube.k8s.io.name', 'minikube.k8s.io.updated\_at', and 'minikube.k8s.io.version'. The values for these labels are: 'amd64', 'linux', 'minikube', 'kubernetes-nodes', 'amd64', 'minikube', 'linux', '9f1e482427589ff8451c4723b6ba53bb9742fbb1', 'minikube', '2021-01-16T18:20:51.0700', and 'v1.16.0'.

cette commande nous affiche 10 pods en cours d'exécution, chacun avec un petit résumé et la version exemple : arch= « amd64 »  
on peut encore lister les pods présents à une période précise en changeant la date.

## 3 Grafana avec helm

Pour utiliser Grafana, nous allons utiliser un assistant de déploiement très pratique : helm (<http://helm.sh/>)

Cet outil va déployer des application classiques de façon très standardisé. Ça tombe bien on a juste de déployer l'app Grafana.

### Installation de Helm :

Sur la vm, exécuter les commandes suivantes :

```
[mlwbe@centos8 ~]$ sudo curl https://raw.githubusercontent.com/kubernetes/helm/master/scripts/get-helm-3 -o get_helm.sh
% Total % Received % Xferd Average Speed Time Time Time Current
Dload Upload Total Spent Left Speed
100 11213 100 11213 0 0 95837 0 --:--:-- --:--:-- --:--:-- 95837
[mlwbe@centos8 ~]$
[mlwbe@centos8 ~]$
[mlwbe@centos8 ~]$
[mlwbe@centos8 ~]$
```

```
curl https://raw.githubusercontent.com/kubernetes/helm/master/scripts/get-helm-3
> get_helm.sh

chmod 700 get_helm.sh

./get_helm.sh
```

```
[mlwbe@centos8 ~]$
[mlwbe@centos8 ~]$ chmod 700 get_helm.sh
[mlwbe@centos8 ~]$ ./get_helm.sh
Downloading https://get.helm.sh/helm-v3.5.0-linux-amd64.tar.gz
Verifying checksum... Done.
Preparing to install helm into /usr/local/bin
helm installed into /usr/local/bin/helm
[mlwbe@centos8 ~]$
```

## Ajouter le repository Grafana

Exécuter la commande suivante pour ajouter le repository bitnami/grafana, et déployer grafana dans le namespace monitoring avec un service d'exposition :

```
helm repo add bitnami https://charts.bitnami.com/bitnami
```

```
helm install grafana bitnami/grafana --namespace=monitoring --set serviceType=NodePort --set
persistence.enabled=true --set persistence.accessModes={ReadWriteOnce} --set persistence.size=8Gi
```

```
Preparing to install helm into /usr/local/bin
helm installed into /usr/local/bin/helm
[mlwbe@centos8 ~]$ helm repo add bitnami https://charts.bitnami.com/bitnami
"bitnami" has been added to your repositories
[mlwbe@centos8 ~]$
[mlwbe@centos8 ~]$
```

r

```
[mlwbe@centos8 ~]$
[mlwbe@centos8 ~]$ helm install grafana bitnami/grafana --namespace=monitoring --set serviceType=NodePort --set persistence.enabled=true --set persistence.accessModes={ReadWriteOnce} --set persistence.size=8Gi
NAME: grafana
LAST DEPLOYED: Sun Jan 17 08:00:24 2021
NAMESPACE: monitoring
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
** Please be patient while the chart is being deployed **

1. Get the application URL by running these commands:
  echo "Browse to http://127.0.0.1:8080"
  kubectl port-forward svc/grafana 8080:3000 &

2. Get the admin credentials:

  echo "User: admin"
  echo "Password: $(kubectl get secret grafana-admin --namespace monitoring -o jsonpath="{.data.GF_SECURITY_ADMIN_PASSWORD}" | base64 --decode)"
[mlwbe@centos8 ~]$
```

## Paramétrage du déploiement Grafana

Editer le service Grafana pour qu'il soit comme le service Prometheus en mode NodePort :

```
kubectl edit service grafana --namespace=monitoring
```

Changer le paramètre qui convient pour passer le service en mode NodePort.

```

name: grafana
namespace: monitoring
resourceVersion: "34697"
uid: 33c4e0bf-b825-4eda-9210-9794e4b0b5c4
spec:
  clusterIP: 10.111.106.126
  clusterIPs:
  - 10.111.106.126
  ports:
  - name: http
    port: 3000
    protocol: TCP
    targetPort: dashboard
  selector:
    app.kubernetes.io/component: grafana
    app.kubernetes.io/instance: grafana
    app.kubernetes.io/name: grafana
  sessionAffinity: None
  type: NodePort
status:
  loadBalancer: {}

```

```

Error from server (NotFound): services "namespace/monitoring" not found
[m1wbe@centos8 ~]$ kubectl edit service grafana --namespace=monitoring
service/grafana edited
[m1wbe@centos8 ~]$
[m1wbe@centos8 ~]$
[m1wbe@centos8 ~]$

```

## Q12. Exécuter la commande suivante et commentez le résultat dans votre rapport :

```

[m1wbe@centos8 ~]$ minikube service grafana --namespace=monitoring

```

NAMESPACE	NAME	TARGET PORT	URL
monitoring	grafana	http/3000	http://192.168.49.2:32670

Opening service monitoring/grafana in default browser...

```
minikube service grafana --namespace=monitoring
```

Un nouveau pod est déployé dans le namespace :monitoring avec le nom **grafana** avec l'adresse ip du pod **192.168.49.2** en mode http (tcp) en écoute ,Le port exposé par le pod **3000** côté node est **32670** à l'intérieur de la VM et l'URL d'accès depuis l'extérieur est : **http://192.168.49.2:32670**

Procéder comme pour Prometheus pour propager le port du Node

Port Forwarding Rules						
Name	Protocol	Host IP	Host Port	Guest IP	Guest Port	
Rule 1	TCP		8001		8001	
Rule 2	TCP		9090		30022	
Rule 3	TCP		3000		32670	

vers la VM. Récupérer le mot de passe de l'utilisateur admin de grafana avec la commande :

```
echo "Password: $(kubectl get secret grafana-admin --namespace monitoring -o jsonpath="{.data.GF_SECURITY_ADMIN_PASSWORD}" | base64 --decode)"
```

```

tcp 0 0 0.0.0.0:3000 0.0.0.0:* LISTEN
[m1wbe@centos8 centos]$ netstat -an | grep 32670
tcp 0 0 0.0.0.0:32670 0.0.0.0:* LISTEN
[m1wbe@centos8 centos]$

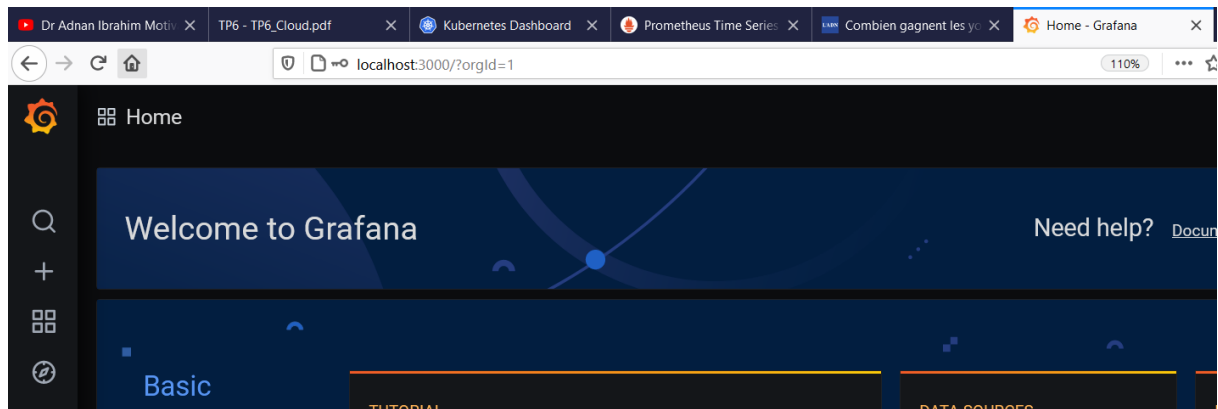
```

```

^C
[m1wbe@centos8 ~]$ echo "Password: $(kubectl get secret grafana-admin --namespace monitoring -o jsonpath="{.data.GF_SECURITY_ADMIN_PASSWORD}" | base64 --decode)"
Password: RTXDB7vPhr
[m1wbe@centos8 ~]$
[m1wbe@centos8 ~]$

```

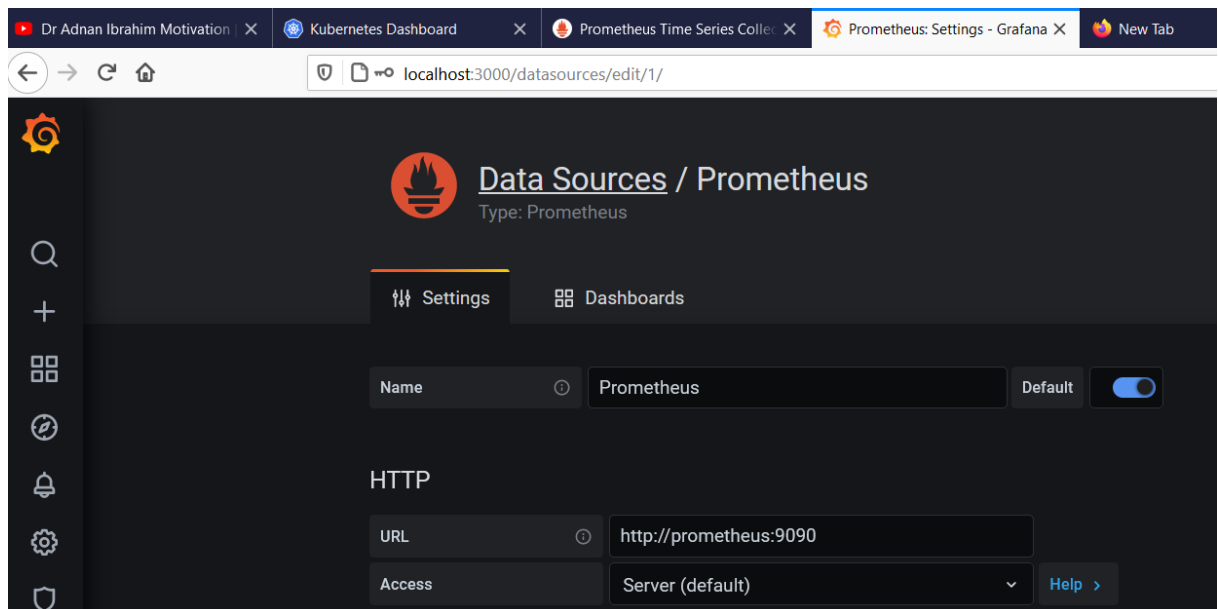
```
wbe@centos8 TP6-fichiers]$ kubectl port-forward svc/grafana --address 0.0.0.0 --namespace=monitoring 32670:3000  
warding from 0.0.0.0:32670 -> 3000
```



## Ajout de la source de données

Il nous faut associer Grafana aux métriques de Prometheus.

Menu Configuration → DataSources. Choisir le type *prometheus* et mettre dans l'url <http://prometheus:9090>.



**Q13. D'où provient cette URL ? Par quel composant de votre cluster est-elle exposée ?** l'url concerne celui d'accès à la gestion de prometheus

**Le composant est :** prometheus

## Importation d'un dashboard

L'intérêt d'une solution comme Grafana est l'accès à de nombreux dashboards créés par la communauté des utilisateurs.

Vous allez importer le Dashboard n° **7249** : Menu **Dashboard** → **Manage**. Puis cliquer sur import. Saisissez l'id du Dashboard en sélectionnant également la DataSource Grafana puis importez le Dashboard.

**Import**  
Import dashboard from file or Grafana.com

### Importing Dashboard from [Grafana.com](https://grafana.com)

Published by: buhay

Updated on: 2018-07-31 00:23:47

#### Options

Name:

Folder:

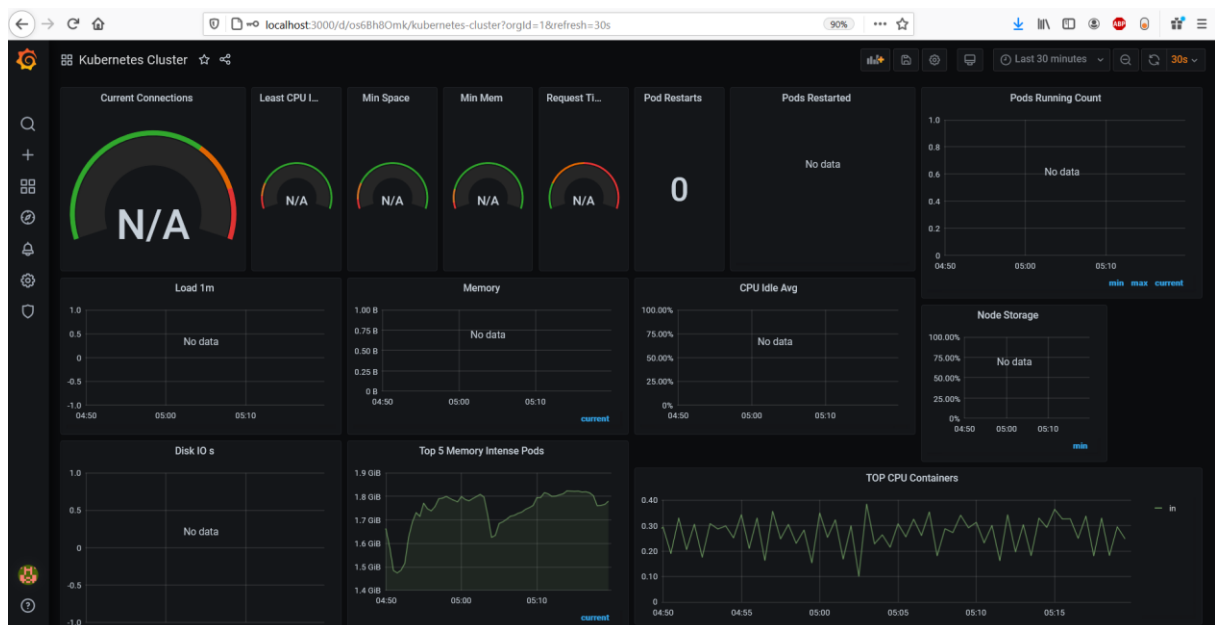
Unique identifier (uid)  
The unique identifier (uid) of a dashboard can be used to uniquely identify a dashboard between multiple Grafana installs. The uid allows having consistent URLs for accessing dashboards so changing the title of a dashboard will not break any bookmarked links to that dashboard.

os6Bh8OmK Change uid

prometheus

Import Cancel

**Q14. Faites une copie d'écran de votre dashboard et commentez le.**



#### Commentaire :

Un joli dashboard pour le monitoring des métriques prometheus  
Des statistiques qui concernent l'utilisation de mémoire, CPU top 5 memory , les ports en écoute , ports des pods les réplicats, états  
On peut même mettre un filtre afin de voir chaque éléments seuls ( mémoire par exemple)  
On peut choisir n'importe quel pod et voir son usage de ressources : et voir un graphe pour chacun de ces pods, on peut également changer la période et voir aussi par jour par semaine par mois ...etc  
On peut explorer chaque métrique à part et avoir la possibilité de l'inspecter, visualiser, partager, éditer, supprimer ... etc.

Certaines zones du Dashboard sont vides, car **Prometheus** ne remonte pas toutes les métriques.

Vous allez installer deux composants additionnels à **Prometheus** : le **node-exporter** et **kube-state-metrics** en exécutant les commandes suivantes :

```
helm install node-exporter bitnami/node-exporter -n monitoring
```

```
helm install my-release bitnami/kube-state-metrics --namespace=monitoring --set serviceType=NodePort
```

```
wbe@centos8 ~]$ helm install node-exporter bitnami/node-exporter -n monitoring
E: node-exporter
T DEPLOYED: Sun Jan 17 09:56:45 2021
NAMESPACE: monitoring
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
Please be patient while the chart is being deployed **

Watch the Node Exporter DaemonSet status using the command:

    kubectl get ds -w --namespace monitoring node-exporter

The Node Exporter can be accessed via port "9100" on the following DNS name from within your cluster:

    node-exporter.monitoring.svc.cluster.local

To access Node Exporter from outside the cluster execute the following commands:

    echo "URL: http://127.0.0.1:9100/"
    kubectl port-forward --namespace monitoring svc/node-exporter 9100:9100
wbe@centos8 ~]$ kubectl port-forward --namespace monitoring svc/node-exporter 9100:9100
[m1wbe@centos8 ~]$ helm install my-release bitnami/kube-state-metrics --namespace=monitoring --set serviceType=NodePort
NAME: my-release
LAST DEPLOYED: Sun Jan 17 09:58:19 2021
NAMESPACE: monitoring
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
** Please be patient while the chart is being deployed **

Watch the kube-state-metrics Deployment status using the command:

    kubectl get deploy -w --namespace monitoring my-release-kube-state-metrics

kube-state-metrics can be accessed via port "8080" on the following DNS name from within your cluster:

    my-release-kube-state-metrics.monitoring.svc.cluster.local

To access kube-state-metrics from outside the cluster execute the following commands:

    echo "URL: http://127.0.0.1:9100/"
    kubectl port-forward --namespace monitoring svc/my-release-kube-state-metrics 9100:8080
[m1wbe@centos8 ~]$
```

Récupérer l'IP et le port du service créé.

Ip : **10.108.28.69** , port : **9100**

**Q15. Quelle est la commande utilisée ?**

```
[m1wbe@centos8 ~]$ ^C
[m1wbe@centos8 ~]$ kubectl get svc --namespace=monitoring
NAME                                TYPE        CLUSTER-IP    EXTERNAL-IP    PORT(S)          AGE
grafana                             NodePort    10.111.106.126 <none>         3000:32670/TCP   120m
my-release-kube-state-metrics       ClusterIP   10.102.238.152 <none>         8080/TCP         2m35s
node-exporter                       ClusterIP   10.108.28.69   <none>         9100/TCP         4m12s
prometheus                          NodePort    10.109.76.182  <none>         9090:30022/TCP   6h44m
[m1wbe@centos8 ~]$
```

Ensuite ajouter un job à Prometheus pour qu'il récupère plus de métriques à partir du composant kube-state-metrics.



```

    target_label: kubernetes_pod_name
    - job_name: 'kube-state-metrics' static_configs:
      - targets: ['10.108.28.69:9100']
    - job_name: 'kube-state-metrics'
      static_configs:
        - targets: ['kube-state-metrics.kube-system.svc.cluster.10.108.20.69:9100']
kind: ConfigMap
metadata:

```

Editer le fichier de configuration de Prometheus

/etc/namespace/monitoring/prometheus-config.yaml, en ajoutant les lignes suivantes à la fin du fichier, avant la directive kind:ConfigMap comme affiché dessous.

```

[mlwbe@centos8 ~]$
[mlwbe@centos8 ~]$ sudo vim /etc/namespace/monitoring/prometheus-config.yaml
[sudo] password for mlwbe:
[mlwbe@centos8 ~]$ kubectl apply -f /etc/namespace/monitoring/prometheus-config.yaml
configmap/prometheus-config configured
[mlwbe@centos8 ~]$
[mlwbe@centos8 ~]$

```

```

    action: replace
    target_label: kubernetes_pod_name
    - job_name: 'kube-state-metrics' static_configs:
      - targets: ['10.108.28.69:9100']
kind: ConfigMap
metadata:

```

```

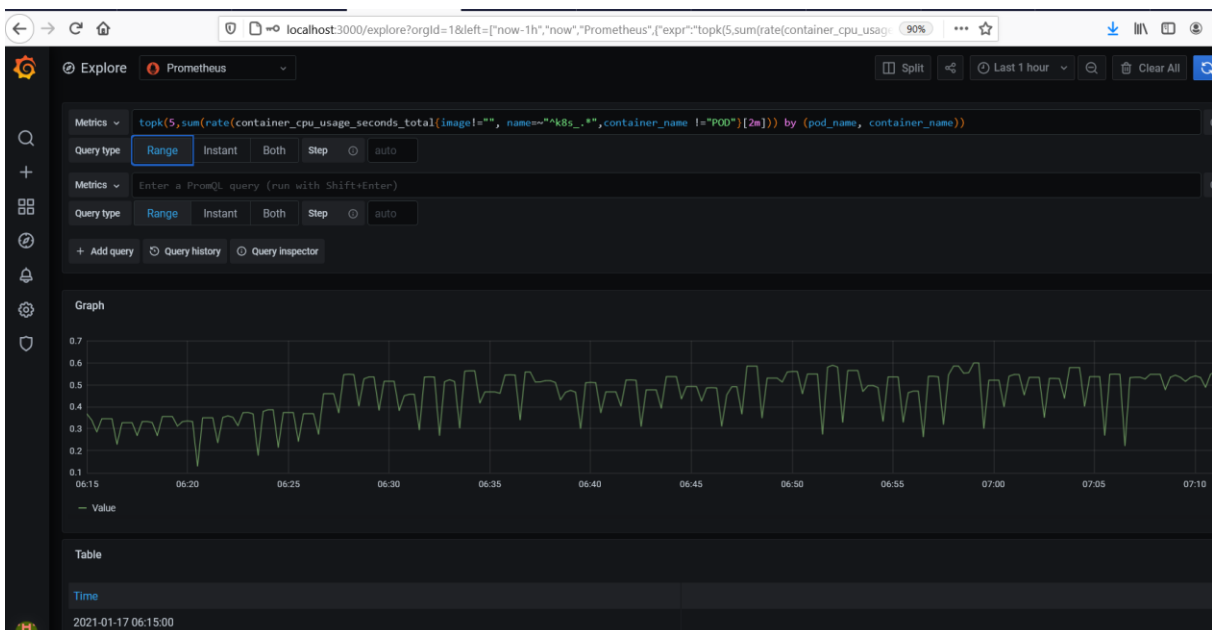
- job_name: 'kube-state-metrics'
  static_configs:
    - targets: ['<IP SVC>:<PORT SVC>']
kind: ConfigMap
metadata:
  name: prometheus-config
  namespace: monitoring

```

Edition d'un Dashboard Grafana

Pour éditer un dashboard, cliquer sur la flèche à droite du nom de chaque graphique, puis Edit. Vous allez éditer le graphique « Top CPU containers ».

**Q16. Quelle est la formule de la métrique de ce graphique ? Reportez cette formule dans Prometheus et exécuter-la. Qu'observez-vous ?**



## Commentaire :

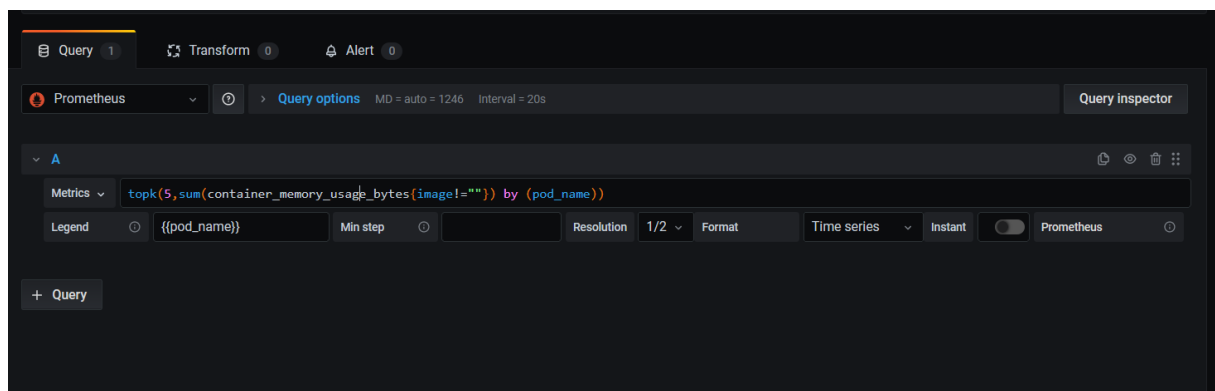
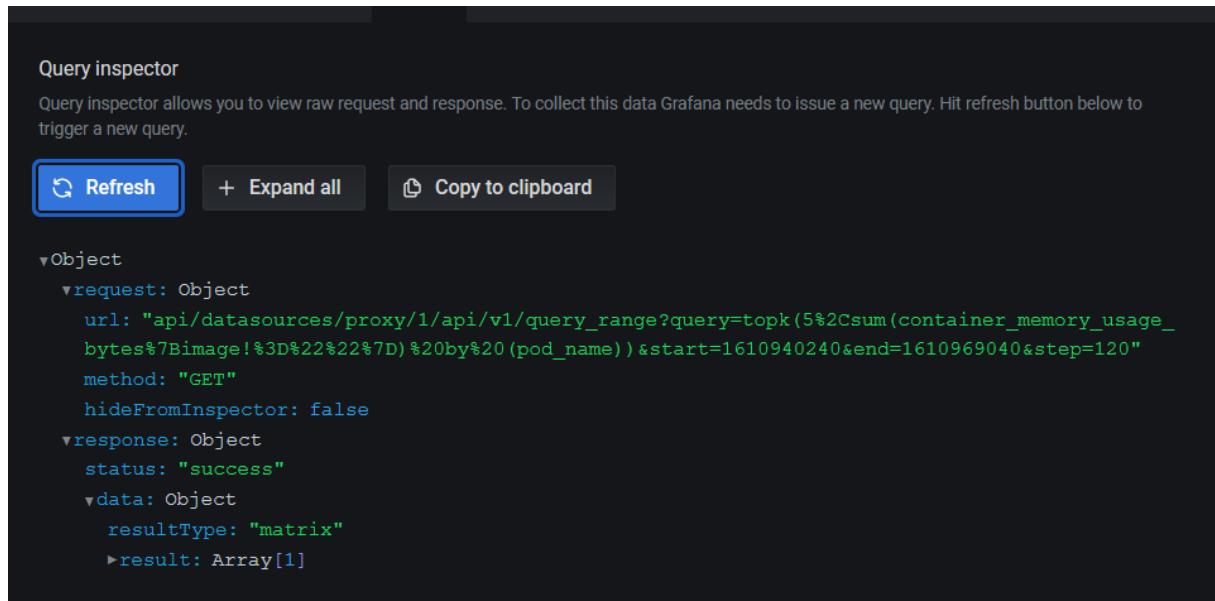
Consommation de CPU illustrer dans un graphe entre 06 :15 et 7 :15

Cette requête nous liste les top 5 des utilisations de CPU par pod avec comme préfix ^k8s\_. et qui a le nom différent de POD , le résultat est listé par les paires pod\_name / container\_name

Corriger l'erreur de formule, en vous aidant si besoin des révisions et commentaires du dashboard :

<https://grafana.com/grafana/dashboards/7249>.

Corriger également le graphique « Top 5 Memory Intense Pod ».



**Q17. Joignez une copie d'écran de votre Dashboard corrigé. Concluez votre rapport**

## Top cpu container

