

## Exercice 1

**I. Configuration mariadb**

- ConfigMap : Définit les paramètres de configuration de MariaDB
- StatefulSet : Gère le déploiement et la mise à jour des pods MariaDB, garantissant un stockage stable
- Service : Expose MariaDB pour permettre la communication entre les pods et les autres services

**II. Configuration wordpress**

- Deployment : Gère le déploiement des pods WordPress et assure leur scalabilité
- Service : Expose WordPress pour permettre la communication avec les utilisateurs et les autres services
- Ingress : Gère les règles d'accès HTTP et HTTPS au service WordPress

**III. Etape d'application des configurations**

1. Création de namespace pour organiser et isoler les ressources de maria db et WordPress :

```
(*)minikube:default)josias@Josiass-MacBook-Pro exercice 1 % kubectl create namespace mariadb
namespace/mariadb created
(*)minikube:default)josias@Josiass-MacBook-Pro exercice 1 % kubectl create namespace wordpress
namespace/wordpress created
```

2. Application des cofiguration dans les namespace :

**a. Mariadb**

- Application de la configuration de MariaDB dans le namespace mariadb

```
(*)minikube:default)josias@Josiass-MacBook-Pro exercice 1 % kubectl apply -f mariadb-configmap.yaml -n mariadb
configmap/mariadb-config created
```

- Déploiement des pods MariaDB avec stockage stable dans le namespace mariadb

```
(*)minikube:default)josias@Josiass-MacBook-Pro exercice 1 % kubectl apply -f mariadb-statefulset.yaml -n mariadb
statefulset.apps/mariadb created
```

- Exposition du service MariaDB dans le namespace mariadb

```
(*)minikube:default)josias@Josiass-MacBook-Pro exercice 1 % kubectl apply -f mariadb-service.yaml -n mariadb
service/mariadb created
```

**b. Wordpress**

- Déploiement des pods WordPress

```
(*)minikube:default)josias@Josiass-MacBook-Pro exercice 1 % kubectl apply -f wordpress-deployment.yaml -n wordpress
deployment.apps/wordpress created
```

- Exposition du service WordPress dans le namespace wordpress

```
(*)minikube:default)josias@Josiass-MacBook-Pro exercice 1 % kubectl apply -f wordpress-service.yaml -n wordpress
service/wordpress created
```

- Ingress : Configuration desrègles d'accès HTTP/HTTPS pour WordPress dans le namespace wordpress

```
(*)minikube:default)josias@Josiass-MacBook-Pro exercise 1 % kubectl apply -f wordpress-ingress.yaml -n wordpress
ingress.networking.k8s.io/wordpress-ingress created
```

### 3. Verification

- Pods :

```
(*)minikube:default)josias@Josiass-MacBook-Pro exercise 1 % kubectl get pods -n mariadb
kubectl get pods -n wordpress
```

NAME	READY	STATUS	RESTARTS	AGE
mariadb-0	1/1	Running	0	13m

NAME	READY	STATUS	RESTARTS	AGE
wordpress-c49454cbf-9bsft	0/1	CreateContainerConfigError	0	6m41s

- Services :

```
(*)minikube:default)josias@Josiass-MacBook-Pro exercise 1 % kubectl get svc -n mariadb
kubectl get svc -n wordpress
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
mariadb	ClusterIP	10.110.249.88	<none>	3306/TCP	22m

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
wordpress	LoadBalancer	10.103.111.18	<pending>	80:30644/TCP	16m

- Ingress

```
(*)minikube:default)josias@Josiass-MacBook-Pro exercise 1 % kubectl get ingress -n wordpress
```

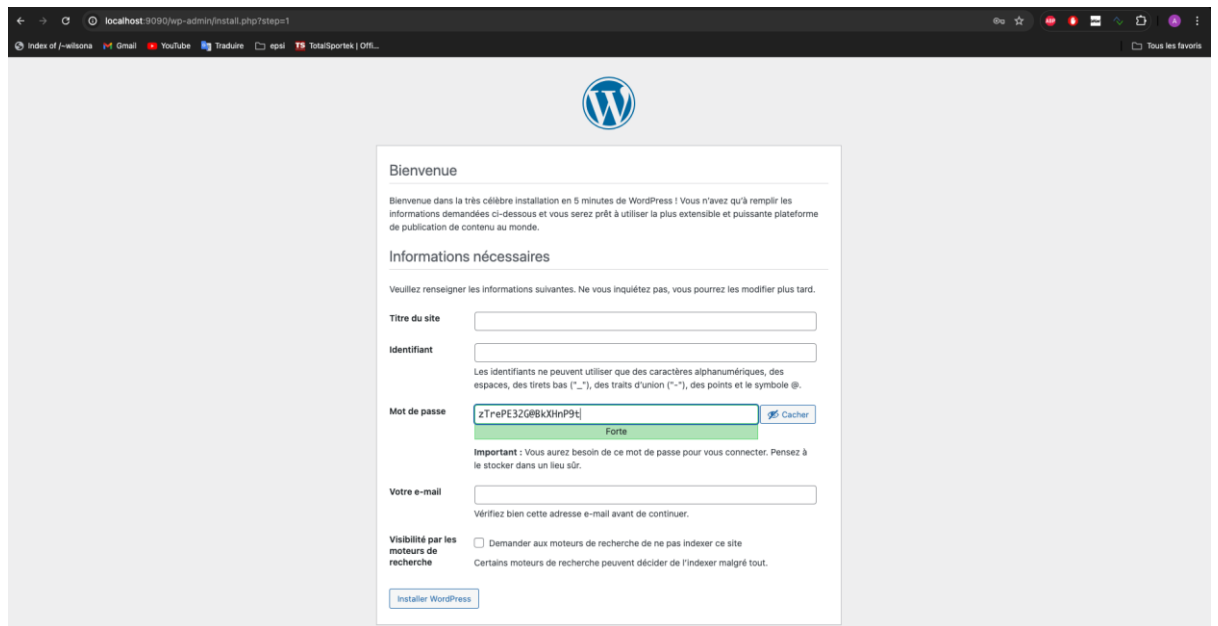
NAME	CLASS	HOSTS	ADDRESS	PORTS	AGE
wordpress-ingress	<none>	wordpress.local		80	15m

- Port forwarding pour accéder au WordPress

```
(*)minikube:default)josias@Josiass-MacBook-Pro exercise 1 % kubectl port-forward svc/wordpress 9090:80 -n wordpress
```

```
Forwarding from 127.0.0.1:9090 -> 80
Forwarding from [::1]:9090 -> 80
Handling connection for 9090
Handling connection for 9090
Handling connection for 9090
Handling connection for 9090
Handling connection for 9090
Handling connection for 9090
```

- Resultat :



The screenshot shows the WordPress installation interface on a local host. The browser's address bar indicates the URL is `localhost:3030/wp-admin/install.php?step=1`. The page features the WordPress logo at the top center. Below the logo, the heading "Bienvenue" (Welcome) is followed by a paragraph: "Bienvenue dans la très célèbre installation en 5 minutes de WordPress ! Vous n'avez qu'à remplir les informations demandées ci-dessous et vous serez prêt à utiliser la plus extensible et puissante plateforme de publication de contenu au monde." (Welcome to the very famous 5-minute WordPress installation! You just have to fill in the information requested below and you will be ready to use the most extensible and powerful content publishing platform in the world.)

The next section is titled "Informations nécessaires" (Required information). It instructs the user: "Veuillez renseigner les informations suivantes. Ne vous inquiétez pas, vous pourrez les modifier plus tard." (Please provide the following information. Don't worry, you can modify them later.)

The form contains the following fields and options:

- Titre du site** (Site title): A text input field.
- Identifiant** (Username): A text input field. Below it, a note states: "Les identifiants ne peuvent utiliser que des caractères alphanumériques, des espaces, des tirets bas ("\_-"), des traits d'union ("~"), des points et le symbole @." (Usernames can only use alphanumeric characters, spaces, underscores, tildes, hyphens, and the @ symbol.)
- Mot de passe** (Password): A text input field containing the text "zTrePE3ZGR8kXhNP9t". To the right of the field is a "Cacher" (Hide) button. Below the input field, a green bar indicates the password strength as "Forte" (Strong).
- Important**: A note below the password field states: "Important : Vous aurez besoin de ce mot de passe pour vous connecter. Pensez à le stocker dans un lieu sûr." (Important: You will need this password to connect. Think about storing it in a safe place.)
- Votre e-mail** (Your email): A text input field. Below it, a note says: "Vérifiez bien cette adresse e-mail avant de continuer." (Check this email address carefully before continuing.)
- Visibilité par les moteurs de recherche** (Search engine visibility): A checkbox labeled "Demander aux moteurs de recherche de ne pas indexer ce site" (Ask search engines not to index this site). Below the checkbox, a note states: "Certains moteurs de recherche peuvent décider de l'indexer malgré tout." (Some search engines may decide to index it anyway.)

At the bottom of the form is a button labeled "Installer WordPress" (Install WordPress).

**Exercice 2 :**

- 1- **Démarrage de Minikube** : Démarrage de minikube pour éviter les conflits avec d'autres configurations possibles sur la machine

```
PS C:\Users\Yassamine\Desktop\Kubernetes> minikube start --profile=metallb
```

- 2- **Activation de l'addon de configuration de MetalLB** : L'addon MetalLB simplifie le processus d'installation.

```
[(*|minikube:default)josias@Josias-MacBook-Pro Kube_project % minikube start ]
🐳 minikube v1.33.1 sur Darwin 14.4.1
🌟 Utilisation du pilote docker basé sur le profil existant
👍 Démarrage du nœud "minikube" primary control-plane dans le cluster "minikube"
"
🏃 Extraction de l'image de base v0.0.44...
🏃 Mise à jour du container docker en marche "minikube" ...
🌐 Préparation de Kubernetes v1.30.0 sur Docker 26.1.1...
🔍 Vérification des composants Kubernetes...
  ▪ Utilisation de l'image quay.io/metallb/controller:v0.9.6
  ▪ Utilisation de l'image quay.io/metallb/speaker:v0.9.6
  ▪ Utilisation de l'image docker.io/kubernetes/dashboard:v2.7.0
  ▪ Utilisation de l'image gcr.io/k8s-minikube/storage-provisioner:v5
  ▪ Utilisation de l'image docker.io/kubernetes/metrics-scrapers:v1.0.8
💡 Certaines fonctionnalités du tableau de bord nécessitent le module complémentaire metrics-server. Pour activer toutes les fonctionnalités, veuillez exécuter :
    minikube addons enable metrics-server
```

- 3- **Configuration de MetalLB** : le but est de savoir quelle adresse IP utilise pour les services de type LoadBalancer. Le fichier de config s'appelle "metallb-config.yaml" et on l'applique au cluster

```

! metallb-config.yaml > {} data > ⌘ config
1  apiVersion: v1
2  kind: ConfigMap
3  metadata:
4    namespace: metallb-system
5    name: config
6  data:
7    config: |
8      address-pools:
9      - name: default
10        protocol: layer2
11        addresses:
12        - 192.168.49.240-192.168.49.250

```

Application de la config avec la commande suivante :

```

(*|minikube:default)josiass@Josiass-MacBook-Pro exercice 2 % kubectl apply -f metallb-config.yaml
configmap/config configured

```

#### 4- Vérification de l'installation de MetalLB :

```

(*|minikube:default)josiass@Josiass-MacBook-Pro exercice 2 % kubectl get pods -n metallb-system
NAME                                READY   STATUS    RESTARTS   AGE
controller-74f6c6d9b4-qdbqz        1/1     Running   0           75m
speaker-xzj58                       1/1     Running   0           75m
NAME      TYPE          CLUSTER-IP   EXTERNAL-IP   PORT(S)          AGE
kubernetes ClusterIP    10.96.0.1     <none>        443/TCP          46h
mariadb   ClusterIP    None         <none>        3306/TCP         46h
nginx     LoadBalancer 10.97.250.170 192.168.49.241 80:32340/TCP     28h
nginx-service LoadBalancer 10.110.159.80 192.168.49.242 80:30090/TCP     28h
wordpress ClusterIP    10.102.104.205 <none>        80/TCP          46h

```

Ces deux commandes nous permettent de vérifier l'état des pods MetalLB dans le namespace "metallb-system" et affichent les services existants pour s'assurer que tout fonctionne correctement.

#### 5- Utilisation de MetalLB avec un service de type LoadBalancer

Pour tester MetalLB, nous allons créer un service de type LoadBalancer et un déploiement Nginx, pour ça nous créons le fichier "nginx-service.yaml" :

```

! nginx-service.yaml > {} spec > ⌘ type
1  apiVersion: v1
2  kind: Service
3  metadata:
4    name: nginx
5  spec:
6    selector:
7      app: nginx
8    ports:
9      - protocol: TCP
10        port: 80
11        targetPort: 80
12    type: LoadBalancer

```

Et un fichier "nginx-deployment.yaml" pour déployer :

```
! nginx-deployment.yaml > {} spec > {} template > {} spec > [ ] containers > {} 0 > [ ] ports > {} 0 > # containerPort
5 spec:
7   selector:
8     matchLabels:
9       app: nginx
10  template:
11    metadata:
12      labels:
13        app: nginx
14    spec:
15      containers:
16        - name: nginx
17          image: nginx
18          ports:
19            - containerPort: 80
```

Et pour appliquer les fichiers on lance les commandes suivantes :

```
• (*|minikube:default)josias@Josiass-MacBook-Pro exercice 2 % kubectl apply -f nginx-deployment.yaml
kubectl apply -f nginx-service.yaml
deployment.apps/nginx-deployment created
service/nginx unchanged
• (*|minikube:default)josias@Josiass-MacBook-Pro exercice 2 %
```

Résultat

## Exercice 3 :


- 1- Configuration des requêtes CPU et mémoire : les requêtes CPU et mémoire garantissent qu'un pod dispose d'une quantité minimale de ressources pour fonctionner correctement, on crée le fichier "nginx-requests.yaml" :

```
! nginx-requests.yaml > {} spec > [ ] containers > {} 0 > {} resources > {} requests > [ ] cpu
1  apiVersion: v1
2  kind: Pod
3  metadata:
4    name: nginx-requests
5  spec:
6    containers:
7      - name: nginx
8        image: nginx
9        resources:
10         requests:
11           memory: "64Mi"
12           cpu: "250m"
```

- 2- Application du fichier de configuration :

```
PS C:\Users\Yassamine\Desktop\Kubernetes> kubectl apply -f nginx-requests.yaml
>>
pod/nginx-requests created
```

- 3- Configuration des limites CPU et mémoire :

```
! nginx-limits.yaml > {} spec > [ ] containers > {} 0 > {} resources > {} limits >  cpu
1   apiVersion: v1
2   kind: Pod
3   metadata:
4     name: nginx-limits
5   spec:
6     containers:
7     - name: nginx
8       image: nginx
9       resources:
10        limits:
11          memory: "128Mi"
12          cpu: "500m"
```

- 4- Application du fichier de configuration :

```
PS C:\Users\Yassamine\Desktop\Kubernetes> kubectl apply -f nginx-limits.yaml
>>
pod/nginx-limits created
```

- 5- Vérification que les requêtes et les limites ont été correctement appliquées :

```
PS C:\Users\Yassamine\Desktop\Kubernetes> kubectl get pod nginx-limits -o jsonpath='{.spec.containers[*].resources}'
{"limits":{"cpu":"500m","memory":"128Mi"},"requests":{"cpu":"500m","memory":"128Mi"}}
```

- 6- Les commandes “kubectl delete pod nginx-requests” & “kubectl delete pod nginx-limits” permettent de supprimer les pods créés.



## Exercice 4

### I. Creation du volume persistant (PV) et une requete de volume persistant (PVC)

- pv.yaml : définit un Volume Persistant qui est une ressource de stockage dans Kubernetes
- Pvc.yaml : définit une Requête de Volume Persistant qui est une demande d'espace de stockage par un user

### II. Application des fichiers pv et pvc

```
(*)minikube:default)josias@Josias-MacBook-Pro exercice 4 % kubectl apply -f pv.yaml
kubectl apply -f pvc.yaml

persistentvolume/example-pv created
persistentvolumeclaim/example-pvc created
```

### III. Configuration du job

- Example-Job.yaml : définit un Job qui exécute des tâches jusqu'à leur achèvement.
- Application

```

• (*|minikube:default)jossias@Josiass-MacBook-Pro exercice 4 % kubectl apply -f example-job.yaml
job.batch/example-job created
○ (*|minikube:default)jossias@Josiass-MacBook-Pro exercice 4 % █

```

#### IV. Configuration cronjob

- Example-cronjob.yaml : définit un CronJob qui planifie des tâches répétées à des intervalles spécifiques

```

job.batch/example-job created
• (*|minikube:default)jossias@Josiass-MacBook-Pro exercice 4 % kubectl apply -f example-cronjob.yaml
cronjob.batch/example-cronjob created
○ (*|minikube:default)jossias@Josiass-MacBook-Pro exercice 4 % █

```

#### V. Vérification

- PV et PVC

```

cronjob.batch/example-cronjob created
• (*|minikube:default)jossias@Josiass-MacBook-Pro exercice 4 % kubectl get pv,pvc -n exercice4

```

NAME	CAPACITY	ACCESS MODES	RECLAIM POLICY	STATUS	CLAIM	STORAGECLASS	VOLUMEATTRIBUTESCLASS	REASON	AGE
persistentvolume/example-pv	1Gi	RWO	Retain	Available					6m48s
persistentvolume/pvc-21256aed-aa84-429a-9020-c2f79aeb396f	1Gi	RWO	Delete	Bound	default/mariadb-persistent-storage-mariadb-0	standard	<unset>		264h
persistentvolume/pvc-6f12037b-90b4-40b3-9c2f-b55dc1346467	1Gi	RWO	Delete	Bound	mariadb/mariadb-persistent-storage-mariadb-0	standard	<unset>		79m
persistentvolume/pvc-b5be8487-4fc0-40ea-95db-b217a606e792	1Gi	RWO	Delete	Bound	default/example-pvc	standard	<unset>		6m48s

```

• (*|minikube:default)jossias@Josiass-MacBook-Pro exercice 4 % █

```

- Job

```

• (*|minikube:default)jossias@Josiass-MacBook-Pro exercice 4 % kubectl get jobs example-job
NAME          STATUS    COMPLETIONS  DURATION  AGE
example-job   Complete  1/1           12s       10m
○ (*|minikube:default)jossias@Josiass-MacBook-Pro exercice 4 % █

```

- Cronjob

```

Error from server (NotFound): jobs.batch "example-cronjob" not found
• (*|minikube:default)jossias@Josiass-MacBook-Pro exercice 4 % kubectl get cronjob example-cronjob
NAME          SCHEDULE    TIMEZONE    SUSPEND    ACTIVE    LAST SCHEDULE    AGE
example-cronjob */1 * * * * <none>    False      0            46s       7m55s
○ (*|minikube:default)jossias@Josiass-MacBook-Pro exercice 4 % █

```

## Exercise 5 :

### I. Installation helm

```
(*minikube:default)josiass@Josiass-MacBook-Pro kustomize % brew install helm
=> Auto-updating Homebrew...
Adjust how often this is run with HOMEBREW_AUTO_UPDATE_SECS or disable with
HOMEBREW_NO_AUTO_UPDATE. Hide these hints with HOMEBREW_NO_ENV_HINTS (see 'man brew').
=> Auto-updated Homebrew!
Updated 3 taps (homebrew/services, homebrew/core and homebrew/cask).
=> New Formulae
codecov-cli                               geni                                     kubelogin                               poutine
=> New Casks
iterm2                                     xnapper
You have 59 outdated formulae and 2 outdated casks installed.

=> Downloading https://ghcr.io/v2/homebrew/core/helm/manifests/3.15.2
##### 100.0%
=> Fetching helm
=> Downloading https://ghcr.io/v2/homebrew/core/helm/blobs/sha256:aad9245f76cf88b3872c04b2e33d63c4a7c860974bc97a0928901aa084da8a1c
##### 100.0%
=> Pouring helm--3.15.2.sonoma.bottle.tar.gz
=> Caveats
zsh completions have been installed to:
  /usr/local/share/zsh/site-functions
=> Summary
📦 /usr/local/Cellar/helm/3.15.2: 66 files, 51.4MB
=> Running 'brew cleanup helm'...
Disable this behaviour by setting HOMEBREW_NO_INSTALL_CLEANUP.
Hide these hints with HOMEBREW_NO_ENV_HINTS (see 'man brew').
(*minikube:default)josiass@Josiass-MacBook-Pro kustomize %
```

### II. Ajout de prometheus

```
(*minikube:default)josiass@Josiass-MacBook-Pro kustomize % helm repo add prometheus-community https://prometheus-community.github.io/helm-charts
helm repo add grafana https://grafana.github.io/helm-charts
helm repo update

"prometheus-community" has been added to your repositories
"grafana" has been added to your repositories
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "grafana" chart repository
...Successfully got an update from the "prometheus-community" chart repository
Update Complete. ✨Happy Helming!✨
(*minikube:default)josiass@Josiass-MacBook-Pro kustomize %
```

### III. Installation de prometheus dans le namespace monitoring

```
(*minikube:default)josiass@Josiass-MacBook-Pro kustomize % kubectl create namespace monitoring
namespace/monitoring created
(*minikube:default)josiass@Josiass-MacBook-Pro kustomize % helm install prometheus-stack prometheus-community/kube-prometheus-stack --namespace monitoring
NAME: prometheus-stack
LAST DEPLOYED: Fri Jun 14 16:38:08 2024
NAMESPACE: monitoring
STATUS: deployed
REVISION: 1
NOTES:
kube-prometheus-stack has been installed. Check its status by running:
  kubectl --namespace monitoring get pods -l "release=prometheus-stack"

Visit https://github.com/prometheus-operator/kube-prometheus for instructions on how to create & configure Alertmanager and Prometheus instances using the Operator.
```

#### IV. Vérification des déploiements dans le namespace monitoring

```
(*)|minikube:default)josias@Josiass-MacBook-Pro kustomize % kubectl get pods -n monitoring
NAME                                READY   STATUS              RESTARTS   AGE
prometheus-stack-grafana-56f4b98db-64lv9    0/3     ContainerCreating    0           2m5s
prometheus-stack-kube-prom-operator-74fb5fc57-xjcfp    1/1     Running              0           2m5s
prometheus-stack-kube-state-metrics-86bf69bfc6-s7dp4    1/1     Running              0           2m5s
prometheus-stack-prometheus-node-exporter-lvflp        1/1     Running              0           2m5s
(*)|minikube:default)josias@Josiass-MacBook-Pro kustomize %
```

Les pods prometheus sont bien lancés

#### V. Exposition de grafana en utilisant ingress

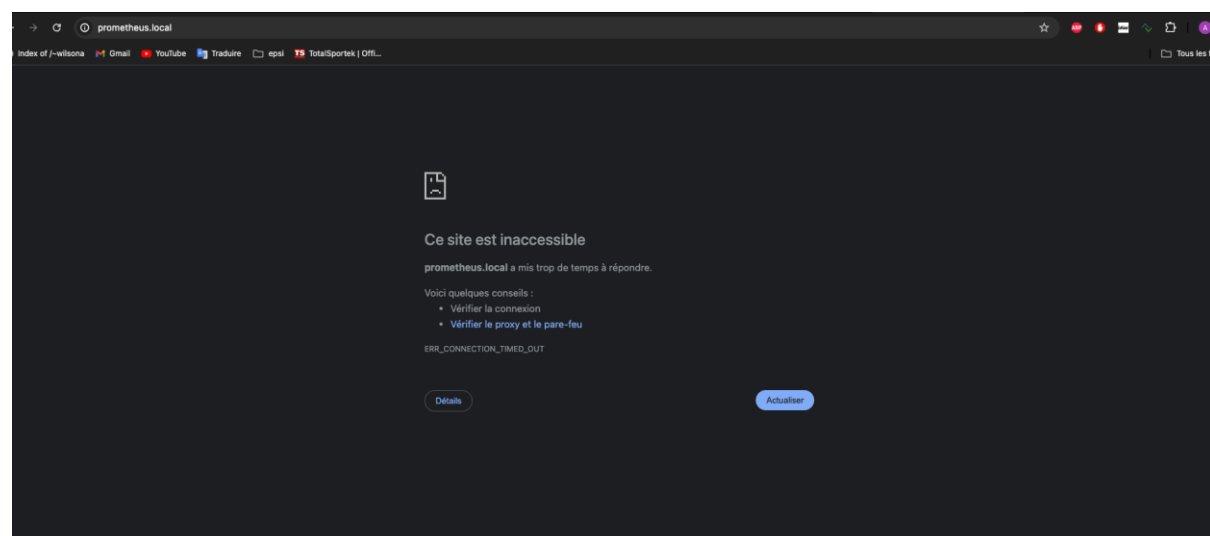
```
(*)|minikube:default)josias@Josiass-MacBook-Pro exercice 5 % kubectl apply -f prometheus-ingress.yaml
ingress.networking.k8s.io/prometheus-ingress created
(*)|minikube:default)josias@Josiass-MacBook-Pro exercice 5 % kubectl apply -f grafana-ingress.yaml
ingress.networking.k8s.io/grafana-ingress created
(*)|minikube:default)josias@Josiass-MacBook-Pro exercice 5 %
```

#### VI. Mise a jour des hosts

```
(*)|minikube:default)josias@Josiass-MacBook-Pro exercice 5 % cat /etc/hosts
##
# Host Database
#
# localhost is used to configure the loopback interface
# when the system is booting. Do not change this entry.
##
127.0.0.1        localhost
255.255.255.255 broadcasthost
::1              localhost
192.168.49.2     prometheus.local
192.168.49.2     grafana.local
(*)|minikube:default)josias@J
```

CodeTime: Head to the dashboard for statistics

#### VII. Resultat



## Exercice Kustomize

### I. Vérification d'installation de kustomize

```
(*)minikube:default)josias@Josiass-MacBook-Pro exercice kustomize % kubectl kustomize --help
Build a set of KRM resources using a 'kustomization.yaml' file. The DIR argument must be a path to a directory
containing 'kustomization.yaml', or a git repository URL with a path suffix specifying same with respect to the
repository root. If DIR is omitted, '.' is assumed.

Examples:
# Build the current working directory
kubectl kustomize

# Build some shared configuration directory
kubectl kustomize /home/config/production

# Build from github
kubectl kustomize https://github.com/kubernetes-sigs/kustomize.git/examples/helloWorld?ref=v1.0.6

Options:
--as-current-user=false:
    use the uid and gid of the command executor to run the function in the container
--enable-alpha-plugins=false:
    enable kustomize plugins
```

### II. Création des répertoires

```
(*)minikube:default)josias@Josiass-MacBook-Pro exercice kustomize % mkdir -p kustomize/base kustomize/overlays/dev kustomize/overlays/prod
```

### III. Application des Configurations

```
(*)minikube:default)josias@Josiass-MacBook-Pro kustomize % kubectl apply -k overlays/dev
service/dev-nginx created
deployment.apps/dev-nginx created
(*)minikube:default)josias@Josiass-MacBook-Pro kustomize % kubectl apply -k overlays/prod
service/prod-nginx created
deployment.apps/prod-nginx created
(*)minikube:default)josias@Josiass-MacBook-Pro kustomize %
```

### IV. Verifications

- Services

```
(*)minikube:default)josias@Josiass-MacBook-Pro kustomize % kubectl get services
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
dev-nginx	ClusterIP	10.97.83.155	<none>	80/TCP	2m5s
kubernetes	ClusterIP	10.96.0.1	<none>	443/TCP	47h
mariadb	ClusterIP	None	<none>	3306/TCP	47h
nginx	LoadBalancer	10.97.250.170	192.168.49.241	80:32340/TCP	28h
nginx-service	LoadBalancer	10.110.159.80	192.168.49.242	80:30090/TCP	28h
prod-nginx	ClusterIP	10.97.103.150	<none>	80/TCP	112s
wordpress	ClusterIP	10.102.104.205	<none>	80/TCP	47h

On a le service dev-nginx et prod-nginx créés

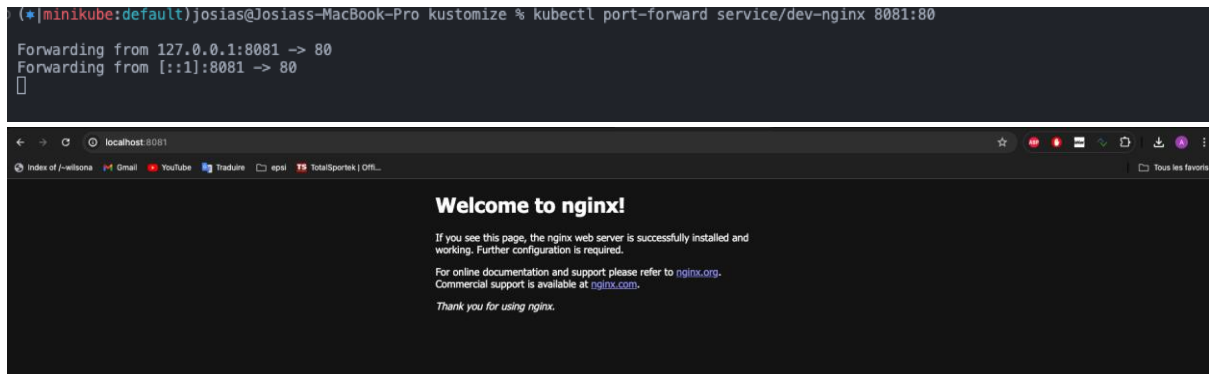
- Déploiements

```
(*)minikube:default)josias@Josiass-MacBook-Pro kustomize % kubectl get deployments
```

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
dev-nginx	1/1	1	1	105s
nginx-deployment	1/1	1	1	37m
prod-nginx	1/1	1	1	92s

On voit le déploiement dev-nginx et prod-nginx créés

- Dev



- Prod

