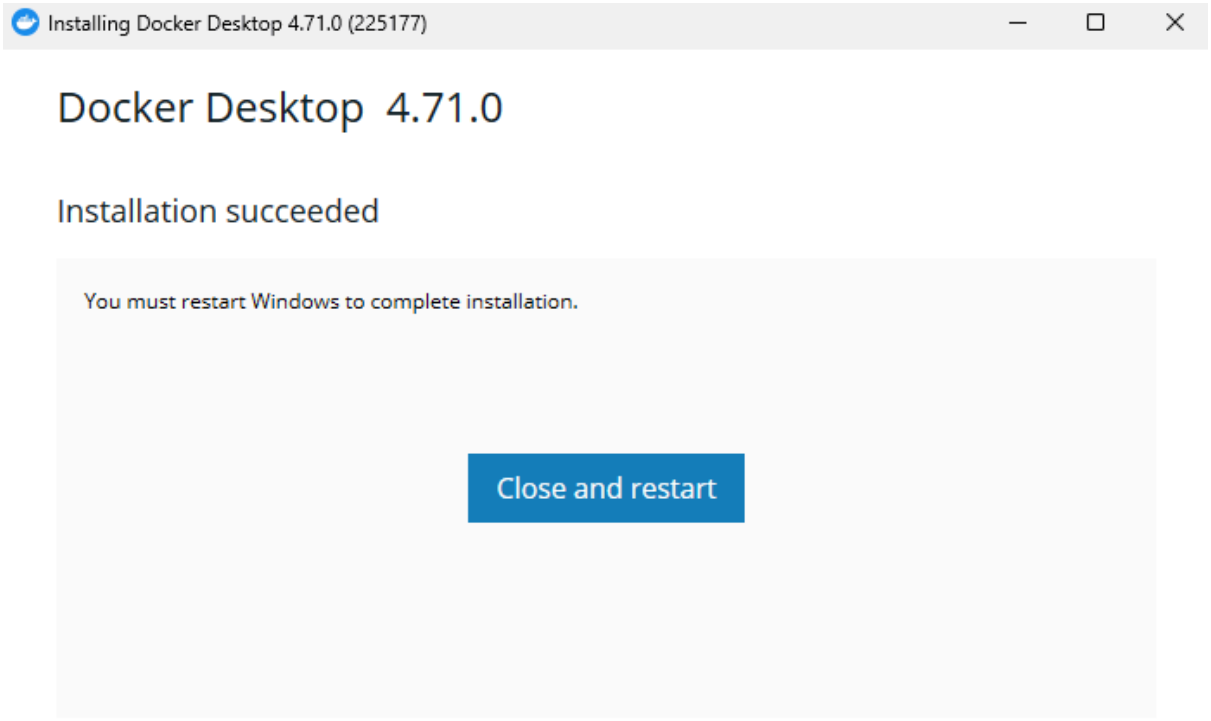
BTS 2 - Services Informatiques aux Organisations

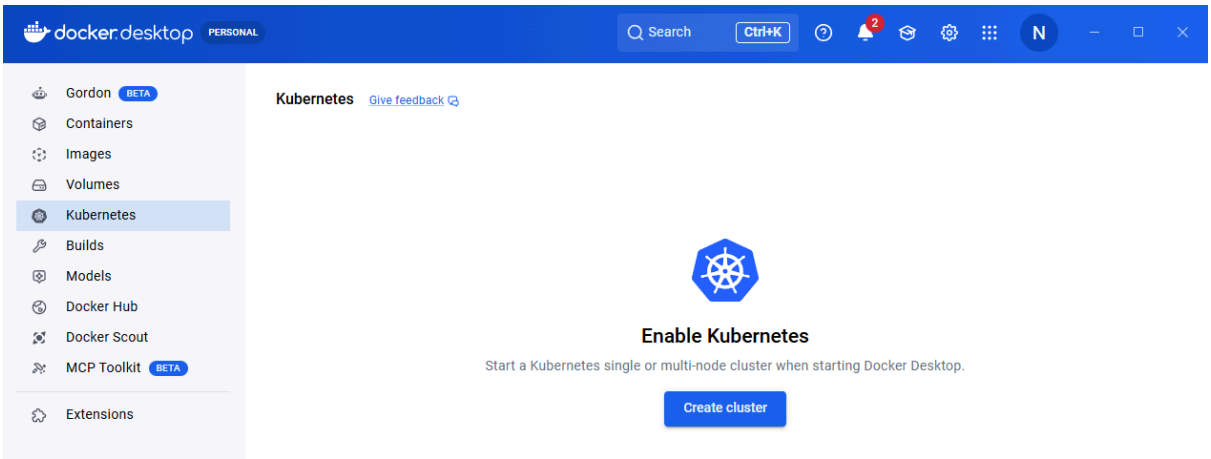
BTS SIO SISR – Chapitre 5 – Kubernetes : plateforme de déploiement des applications conteneurisées

1. Kubernetes avec Docker Desktop

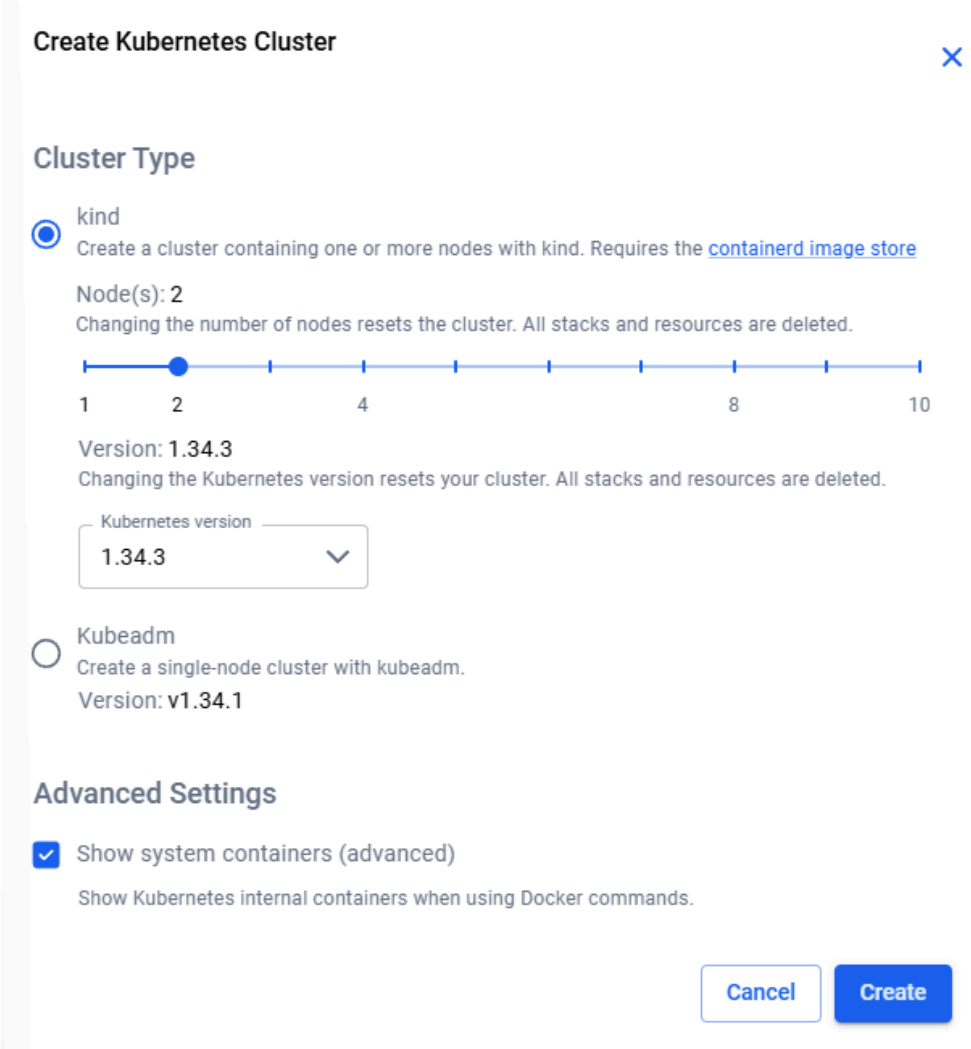
Installation de Docker Desktop :



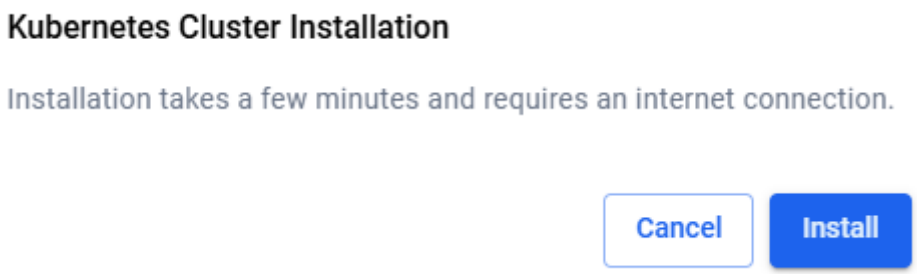
Activation de Kubernetes depuis Docker Desktop :



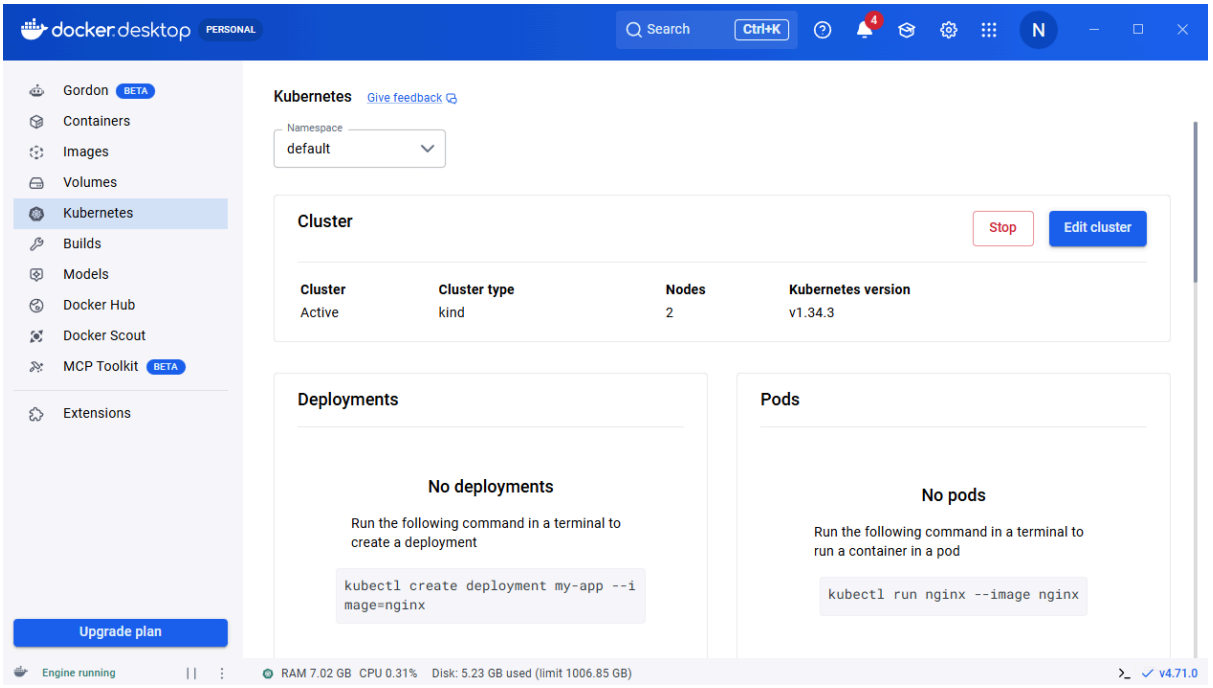
Création d'un cluster Kubernetes de type Kind avec 2 nodes :



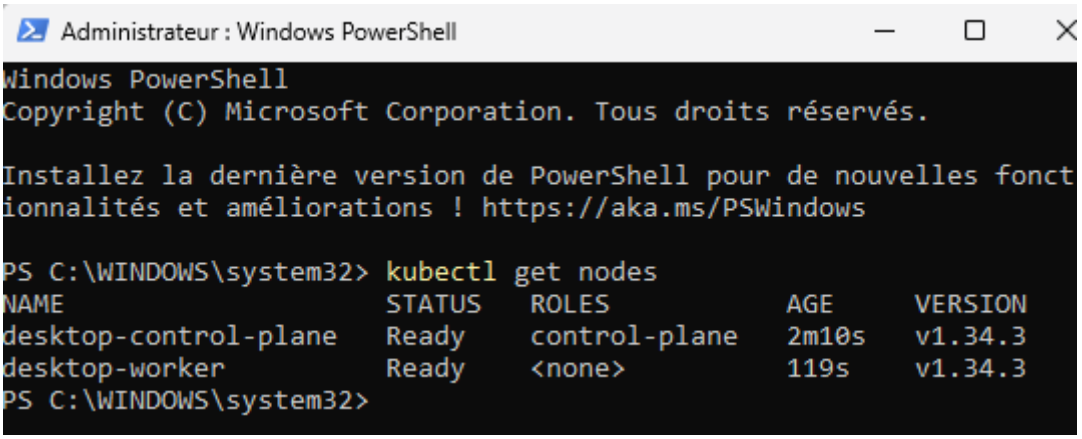
Installation du cluster Kubernetes :



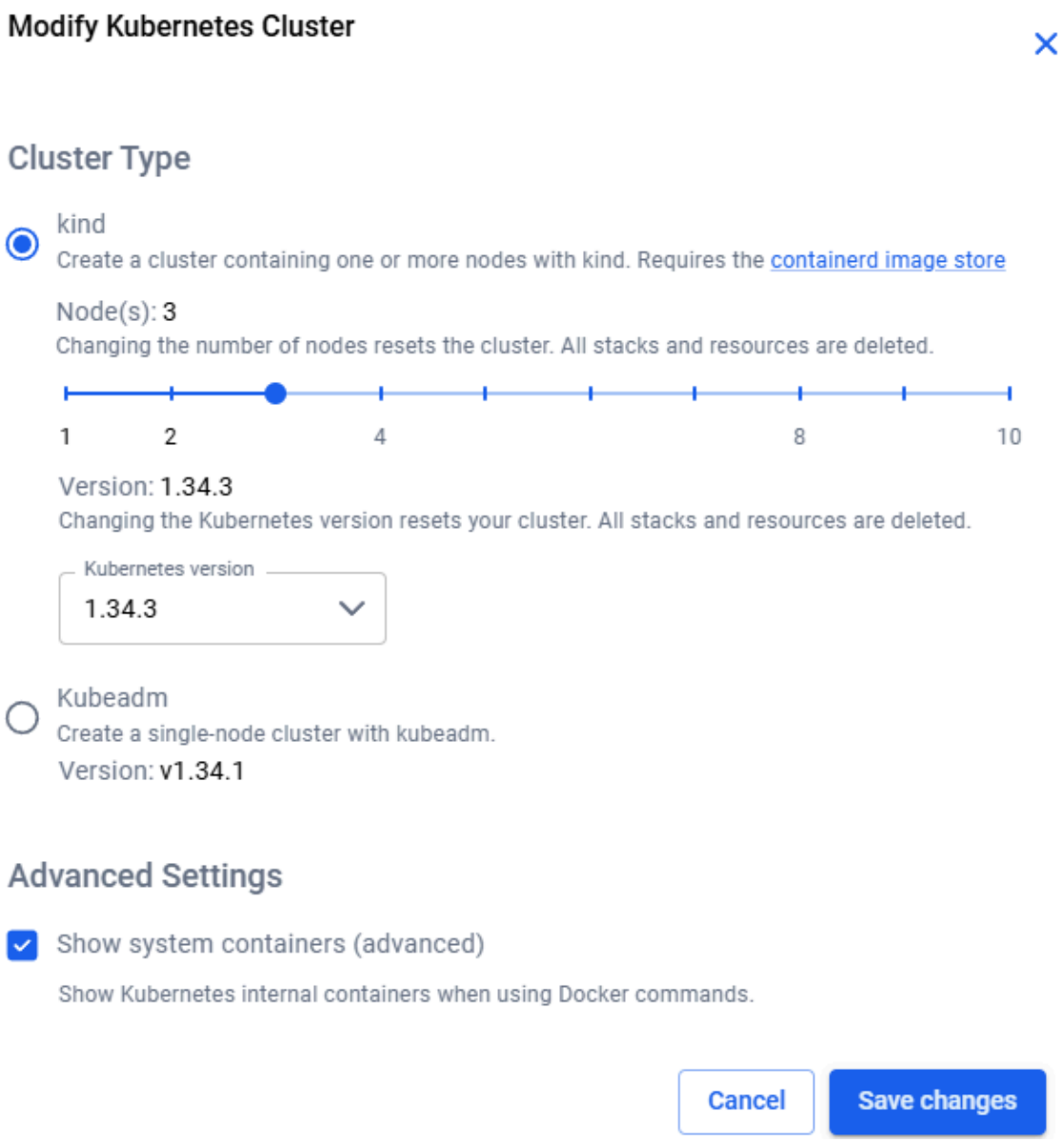
Interface de Kubernetes dans Docker Desktop :



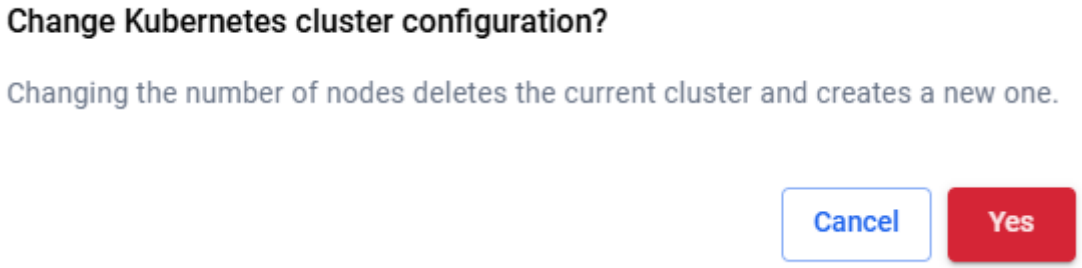
La commande **kubectl get nodes** permet de vérifier l'état des 3 nodes (noeuds) du cluster.



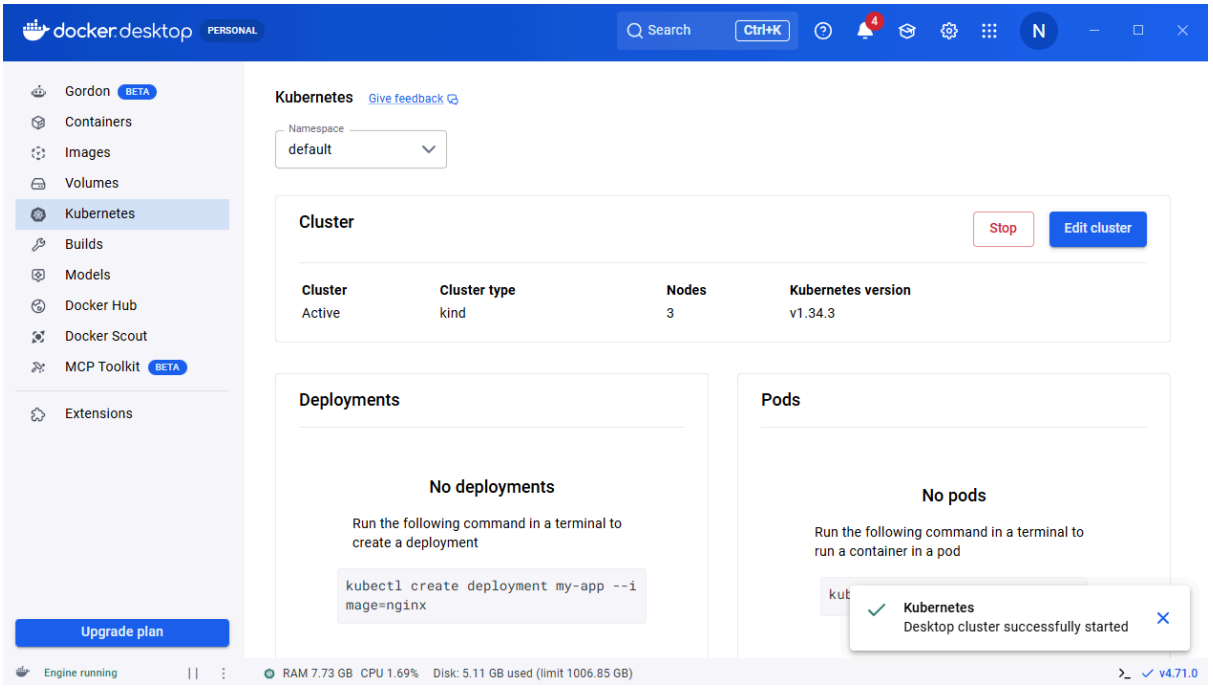
Configuration du cluster Kubernetes dans Docker Desktop (type, nœuds et version) :



Confirmation de modification du cluster Kubernetes :



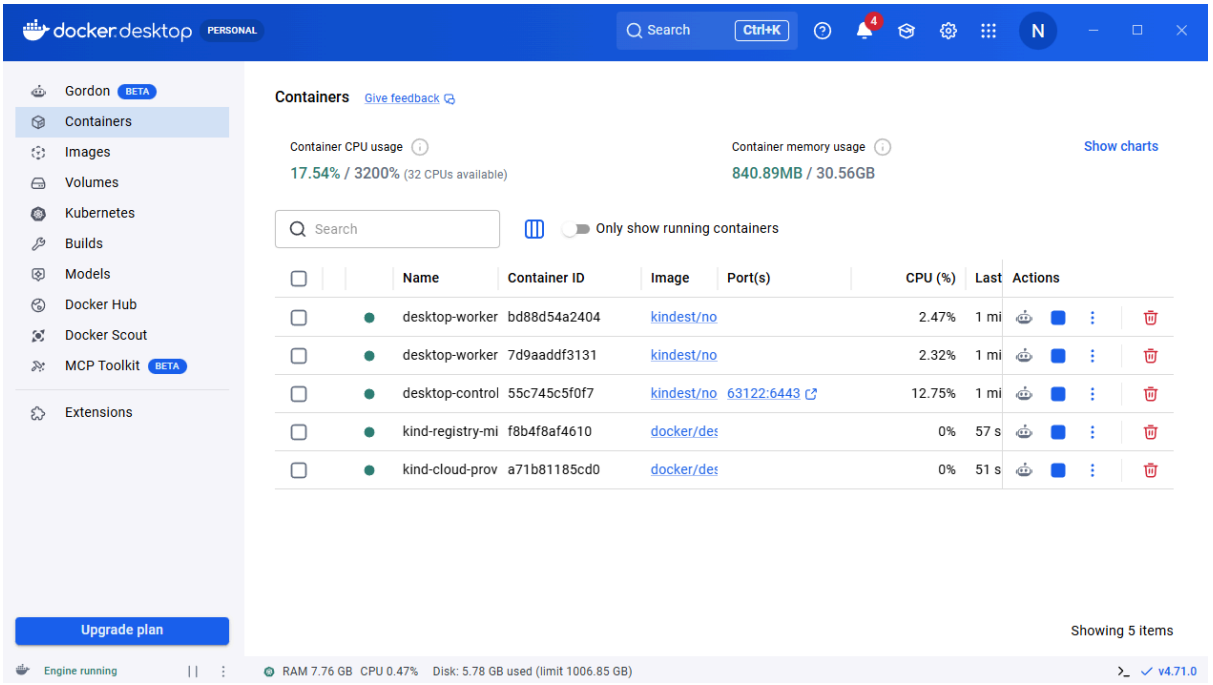
Démarrage du cluster Kubernetes dans Docker Desktop :



La commande **kubectl get nodes** permet de vérifier l'état des 3 nodes (noeuds) du cluster.

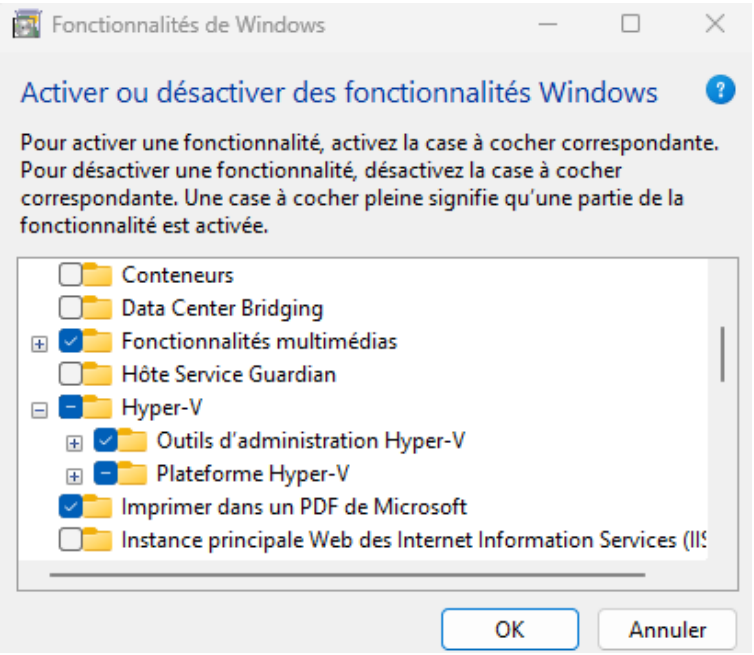
```
PS C:\WINDOWS\system32> kubectl get nodes
NAME                  STATUS   ROLES           AGE   VERSION
desktop-control-plane   Ready    control-plane   56s   v1.34.3
desktop-worker          Ready    <none>          47s   v1.34.3
desktop-worker2         Ready    <none>          47s   v1.34.3
PS C:\WINDOWS\system32>
```

Visualisation de la liste des conteneurs Kubernetes en cours d'exécution dans Docker Desktop.

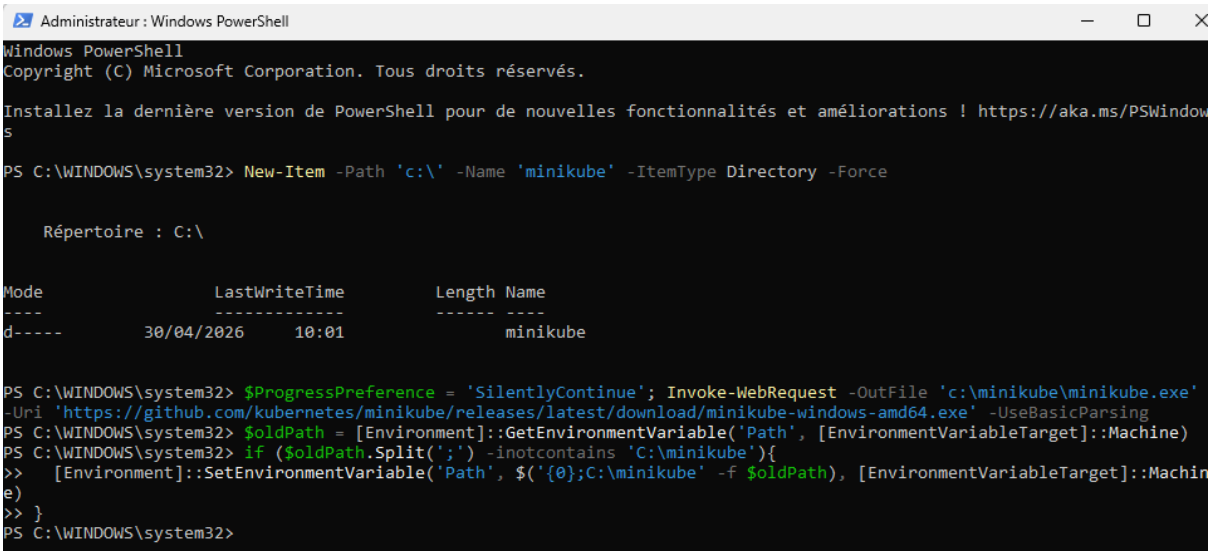


2. Installation de Minikube

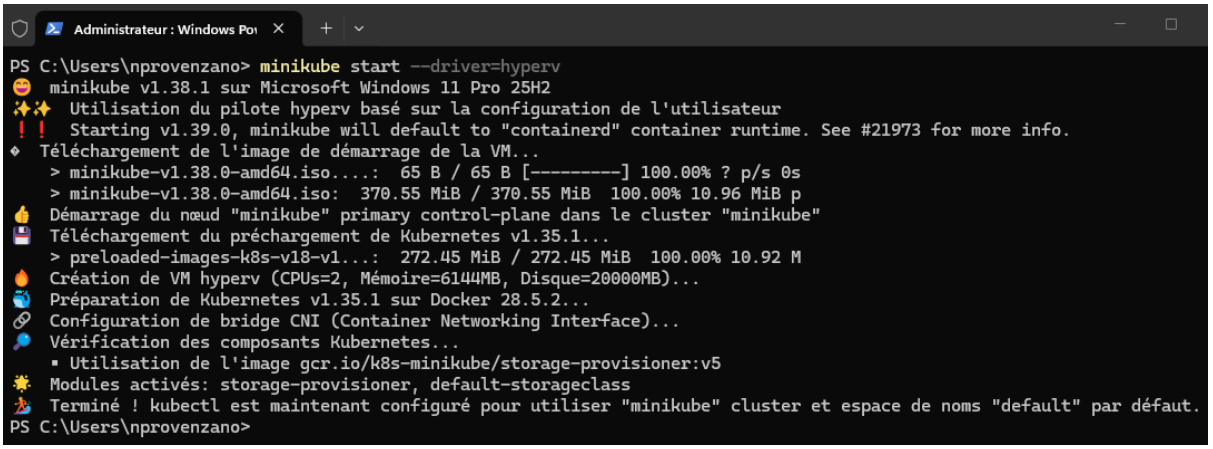
Activation de Hyper-V permet d'utiliser des machines virtuelles pour exécuter Minikube.



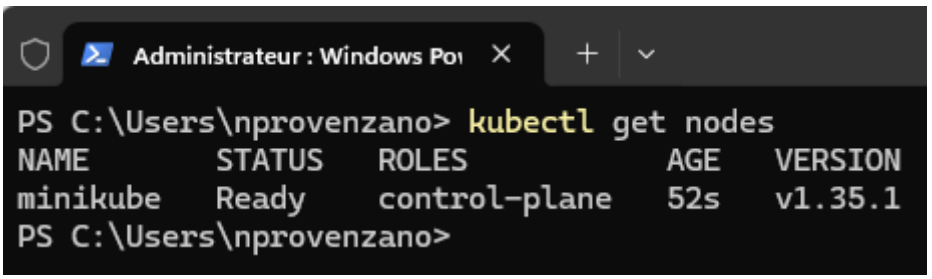
L'installation de Minikube permet de créer un cluster Kubernetes local.



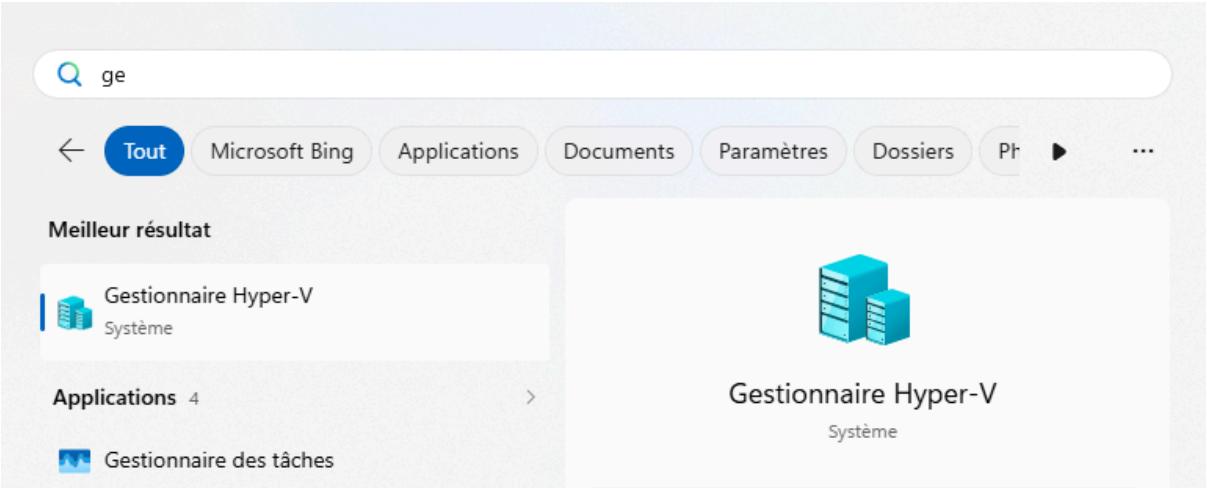
La commande **minikube start** permet de démarrer le cluster Kubernetes local.



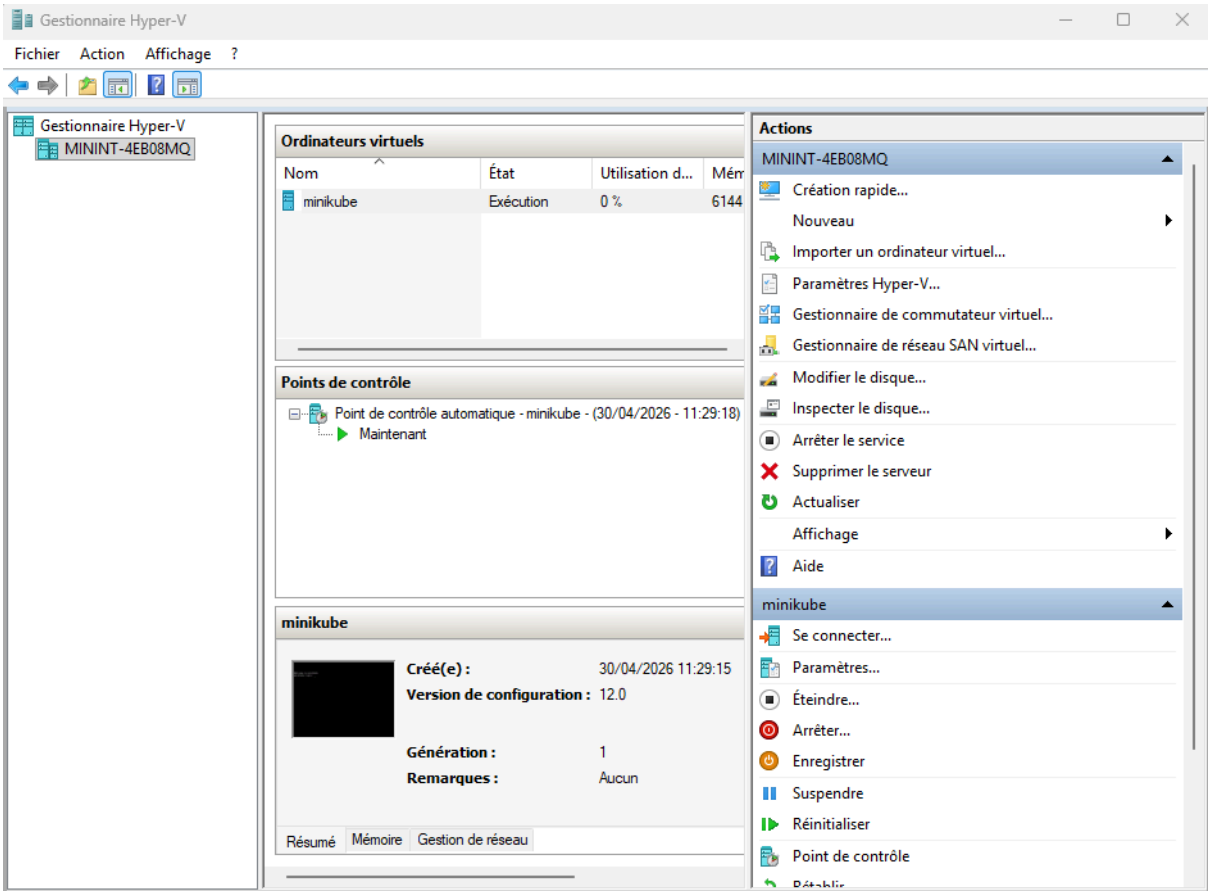
La commande **kubectl get nodes** permet de vérifier que le cluster est opérationnel.



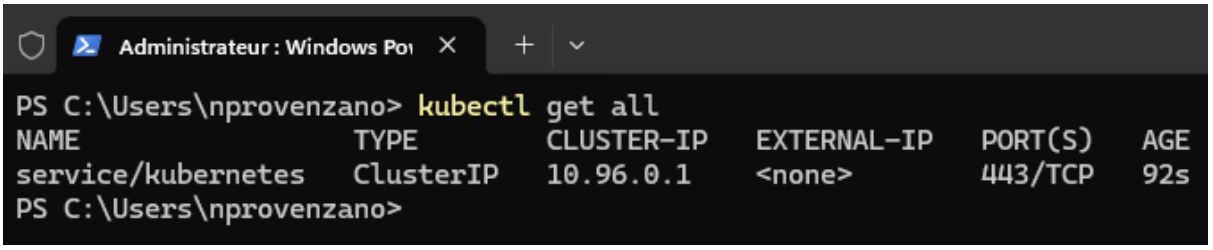
Ouverture du Gestionnaire Hyper-V :



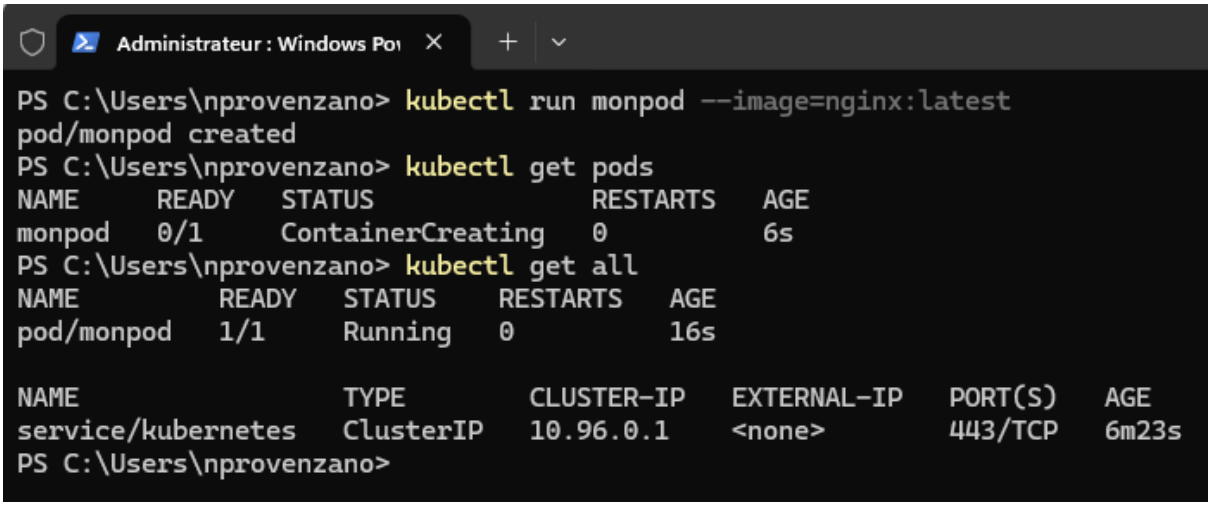
Le Gestionnaire Hyper-V permet de vérifier que la machine virtuelle Minikube est créée et en cours d'exécution.



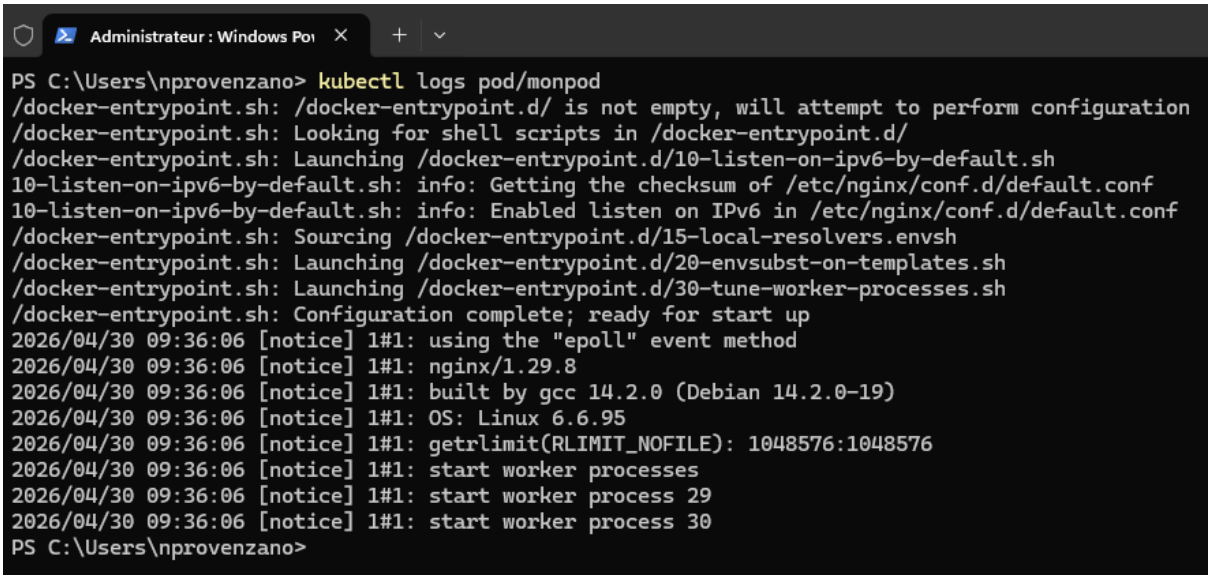
La commande **kubectl get all** permet d'afficher tous les objets du cluster.



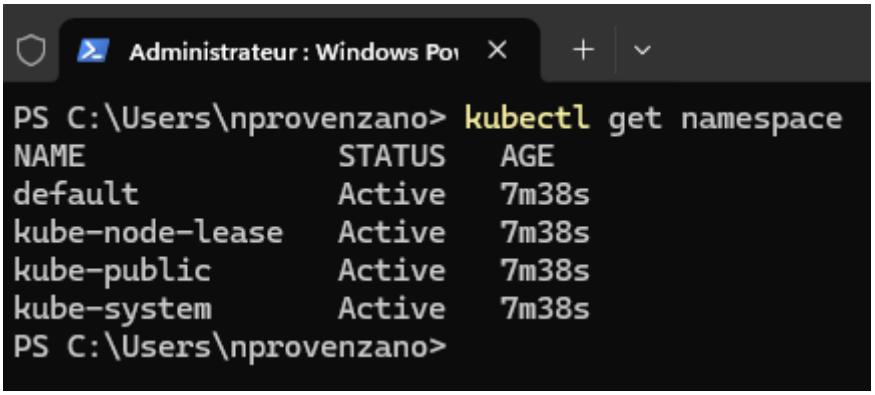
La commande **kubectl run** permet de créer un pod à partir d'une image Docker.
La commande **kubectl get pods** permet de lister les pods en cours d'exécution.



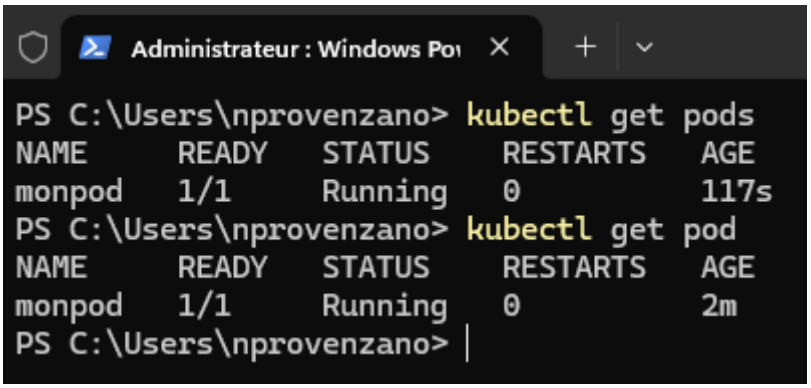
La commande **kubectl logs** permet d'afficher les journaux d'un pod.



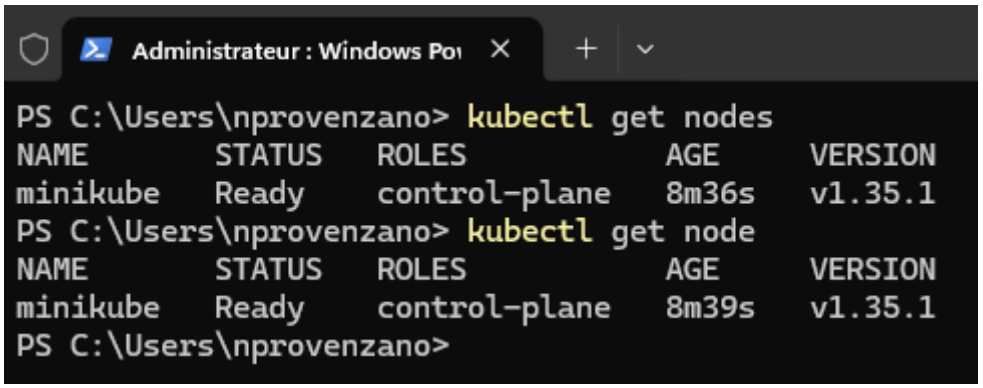
La commande **kubectl get namespace** permet d'afficher les espaces de noms.



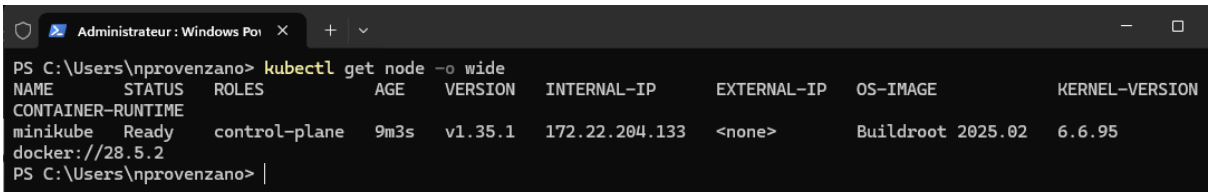
La commande **kubectl get pods** permet de lister les pods et de vérifier leur état. Le pod "**monpod**" est en statut Running, le conteneur est en cours d'exécution ce qui indique que le déploiement est fonctionnel.



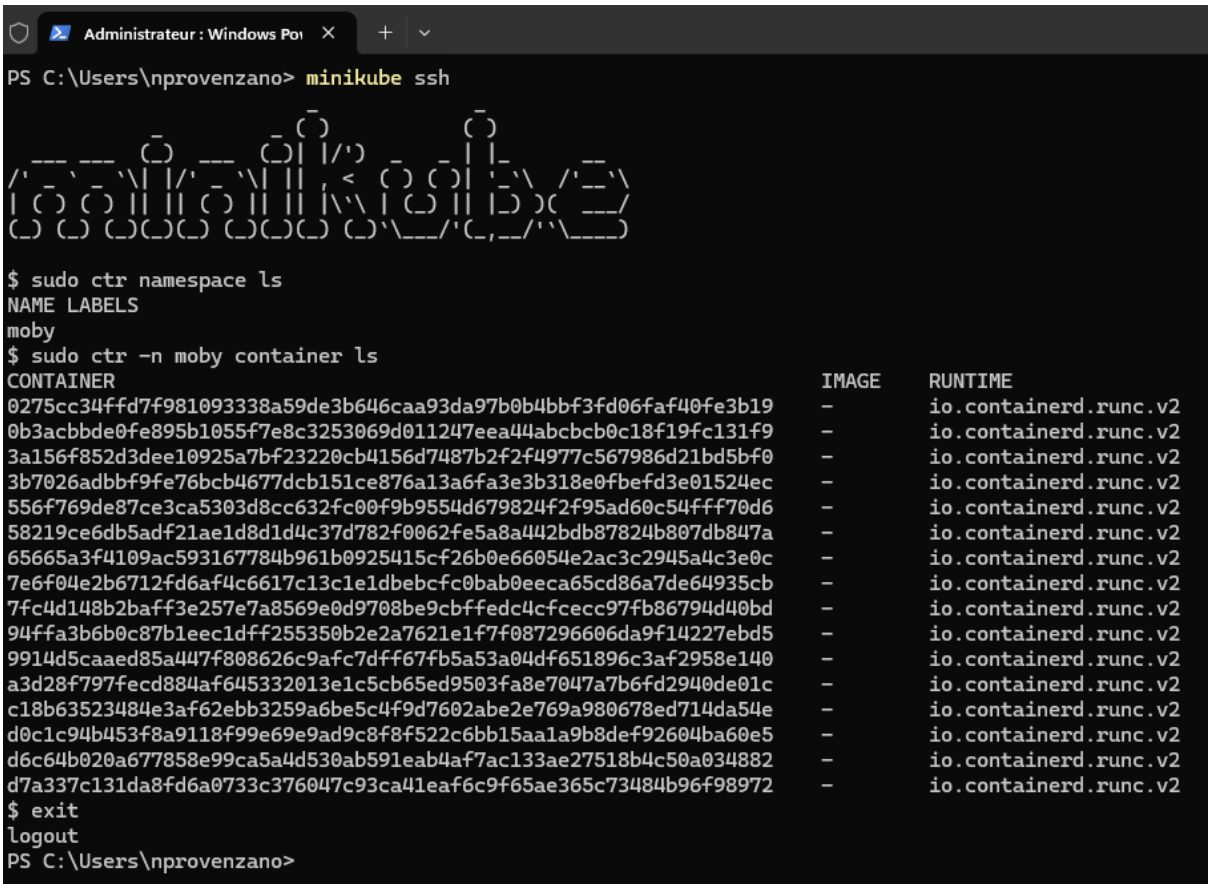
La commande **kubectl get nodes** permet de lister les nœuds du cluster et de vérifier leur état. Le résultat montre que le nœud "minikube" est en statut Ready avec le rôle control-plane.



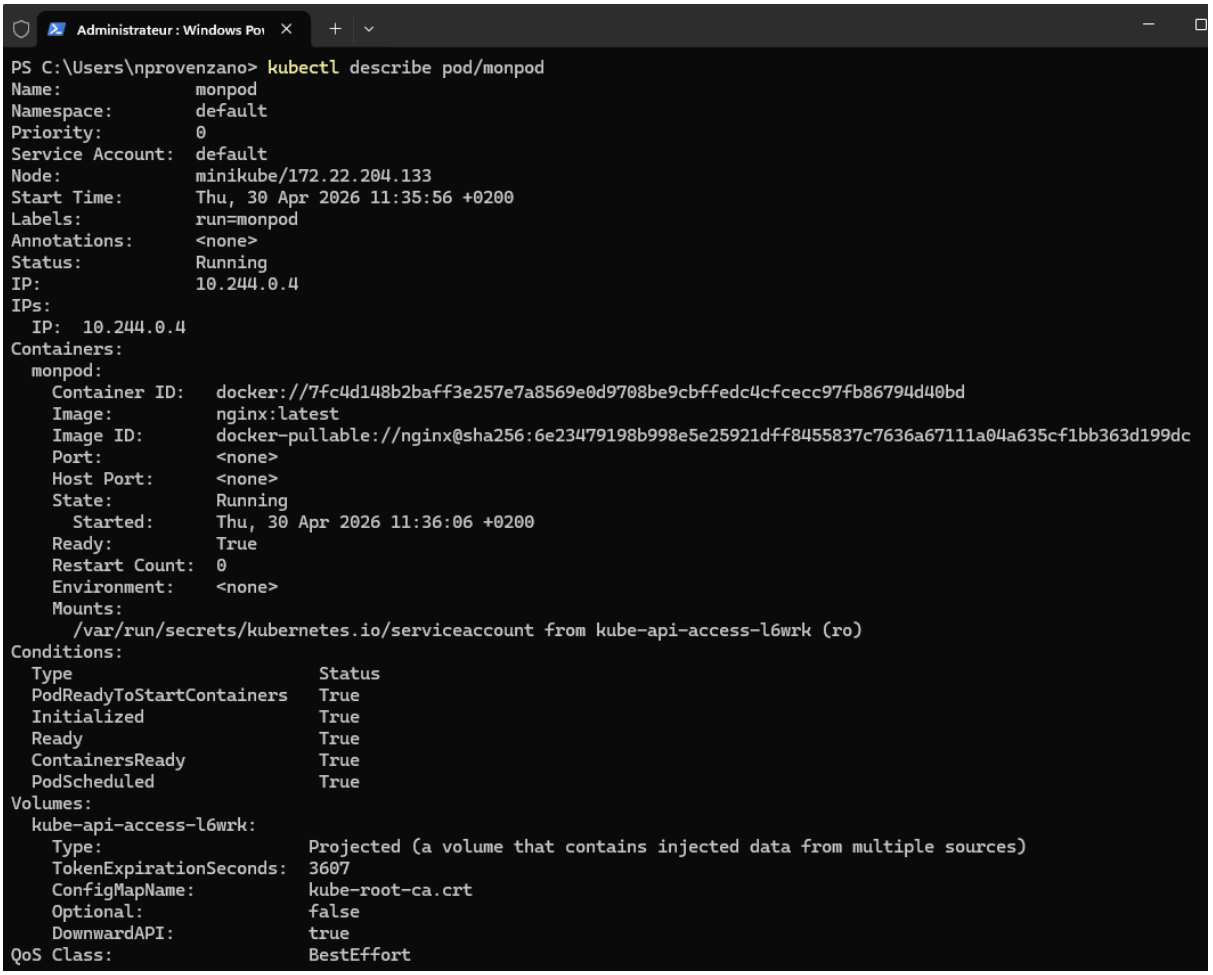
La commande **kubectl get node -o wide** permet d'afficher des informations détaillées sur les nœuds du cluster. Le résultat montre que le nœud "minikube" est en statut Ready, avec son adresse IP interne, sa version Kubernetes et son système d'exploitation.



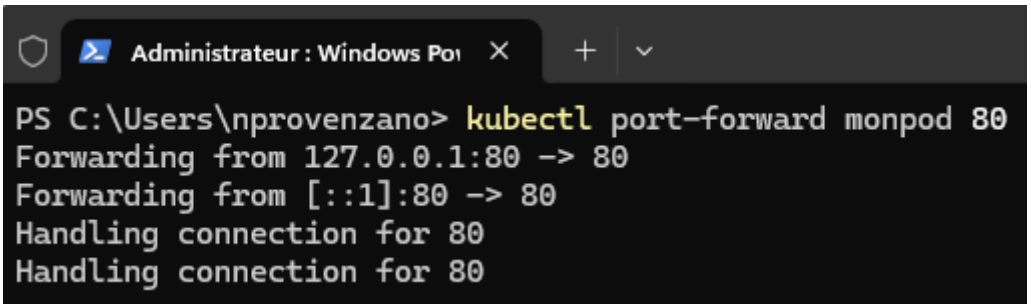
La commande **minikube ssh** permet d'accéder à la machine virtuelle Minikube.



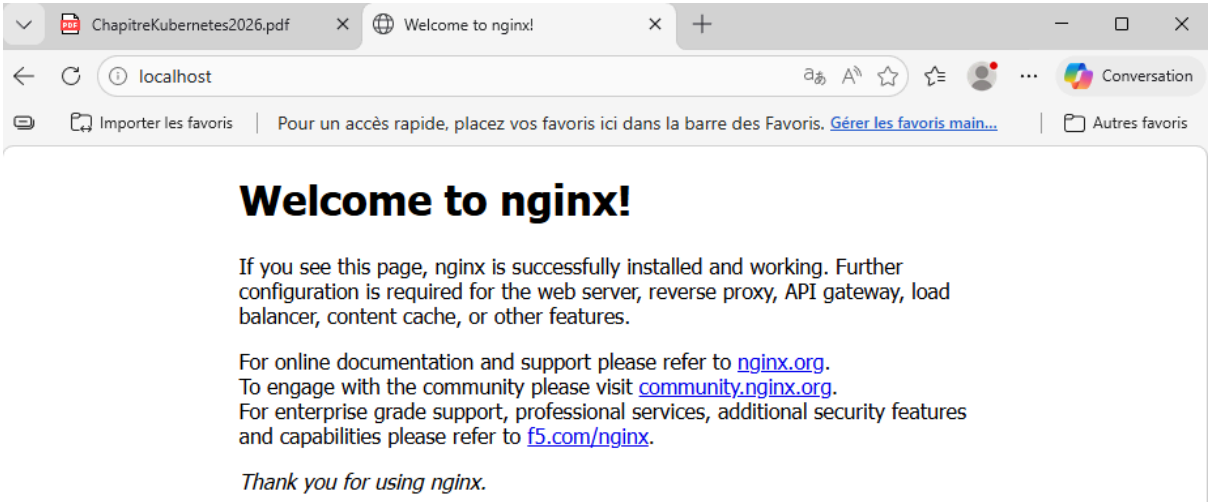
La commande **kubectl describe pod monpod** permet d'obtenir les détails d'un pod, sur son statut, sa configuration, ses conteneurs et les événements supplémentaires.



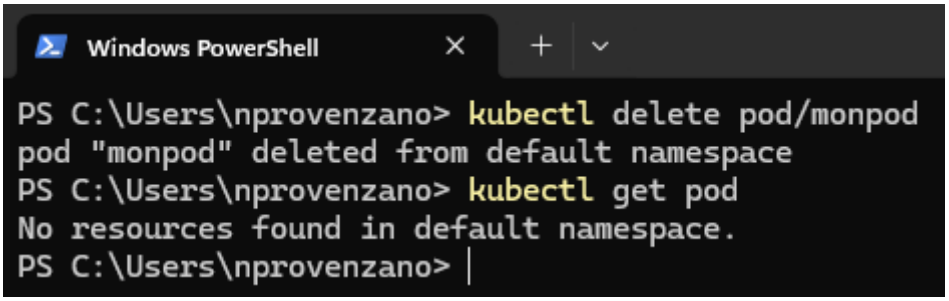
La commande **kubectl port-forward monpod 80** permet de faire un tunnel entre le pod et la machine locale pour accéder à un service interne du cluster via le port 80.



L'accès à localhost permet de vérifier que le pod Nginx est bien déployé et que le service est accessible depuis la machine locale.



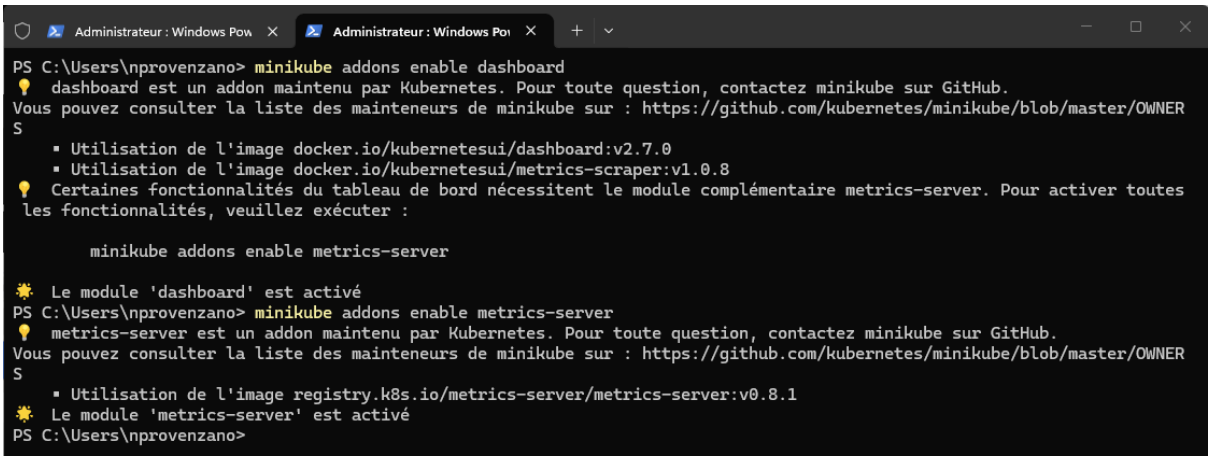
La commande **kubectl delete pod monpod** permet de supprimer un pod "monpod" du cluster Kubernetes.



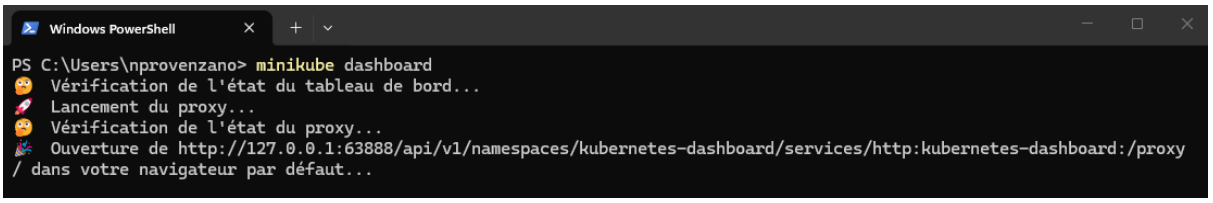
3. Le tableau de bord de Kubernetes

La commande **minikube addons enable dashboard** permet d'activer l'interface graphique de Kubernetes pour visualiser et gérer les ressources du cluster.

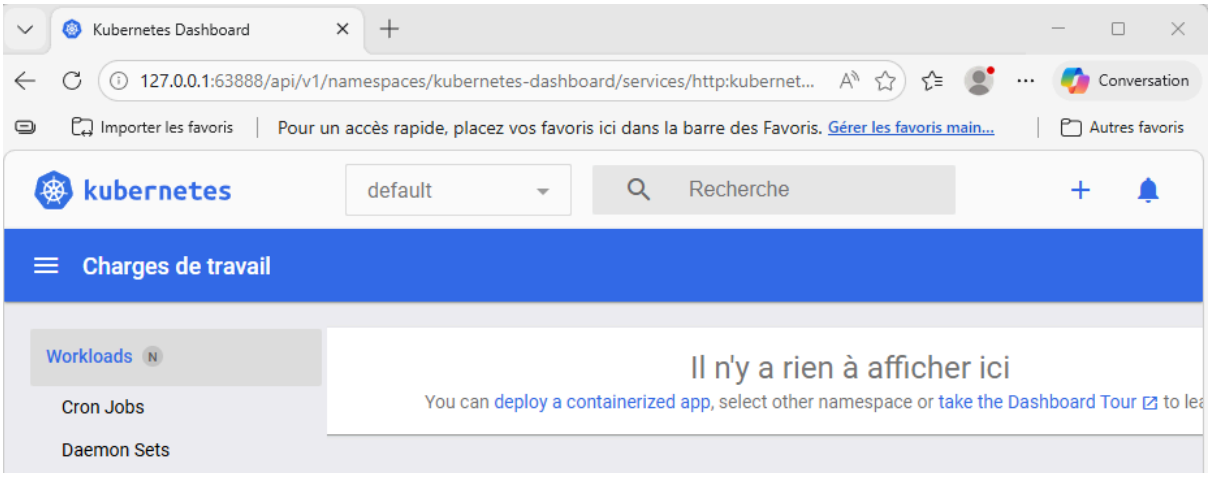
La commande **minikube addons enable metrics-server** permet d'activer le service qui collecte les métriques des ressources (CPU, mémoire) des pods et des nœuds.



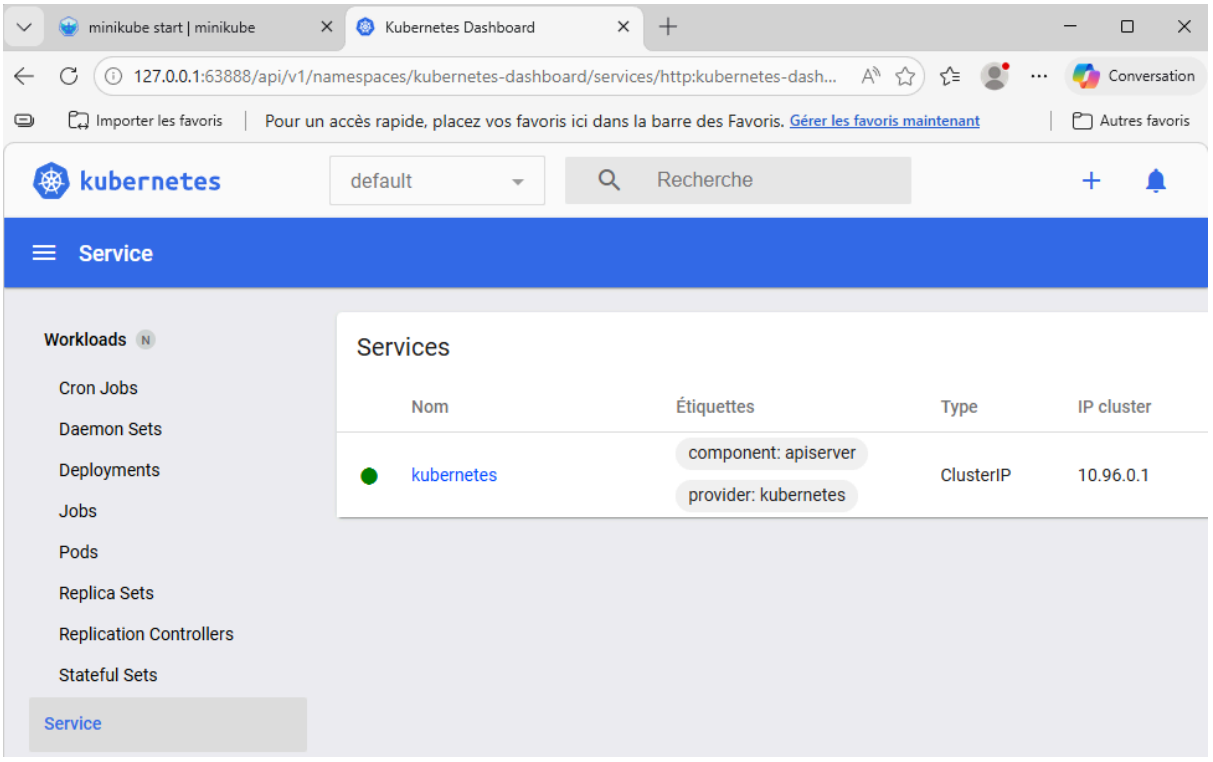
La commande **minikube dashboard** permet de lancer l'interface graphique de Kubernetes.



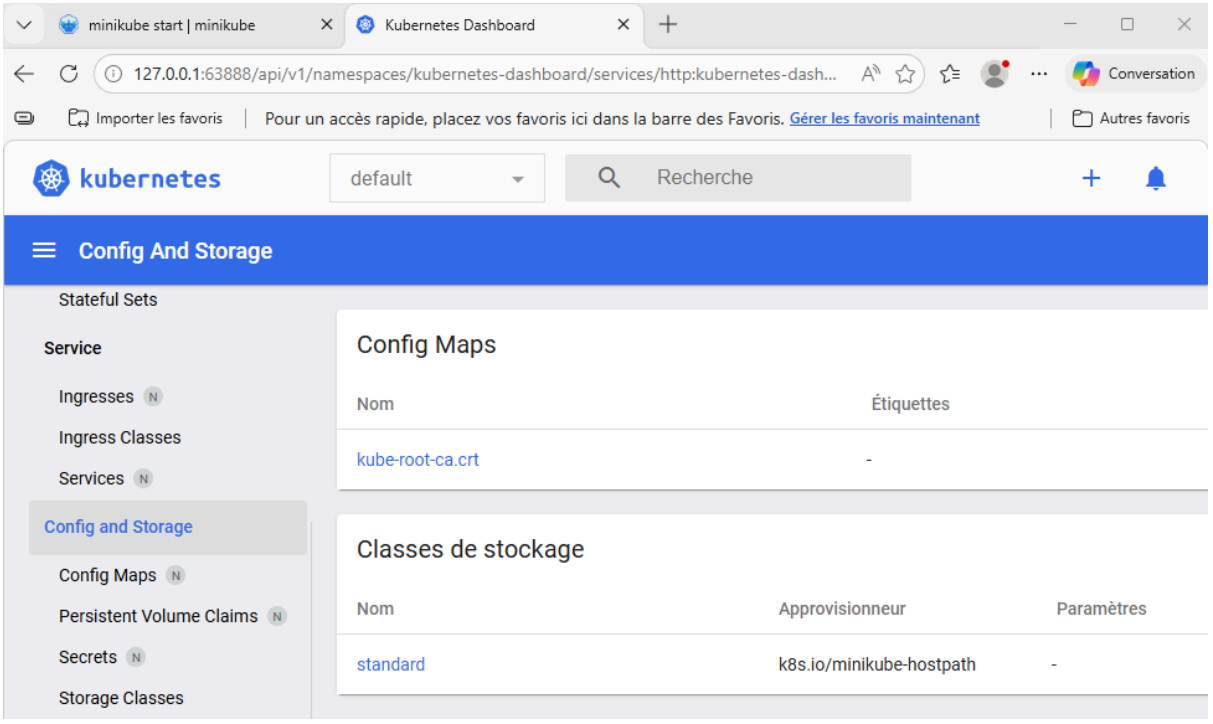
Visualisation du tableau de bord du cluster Kubernetes :



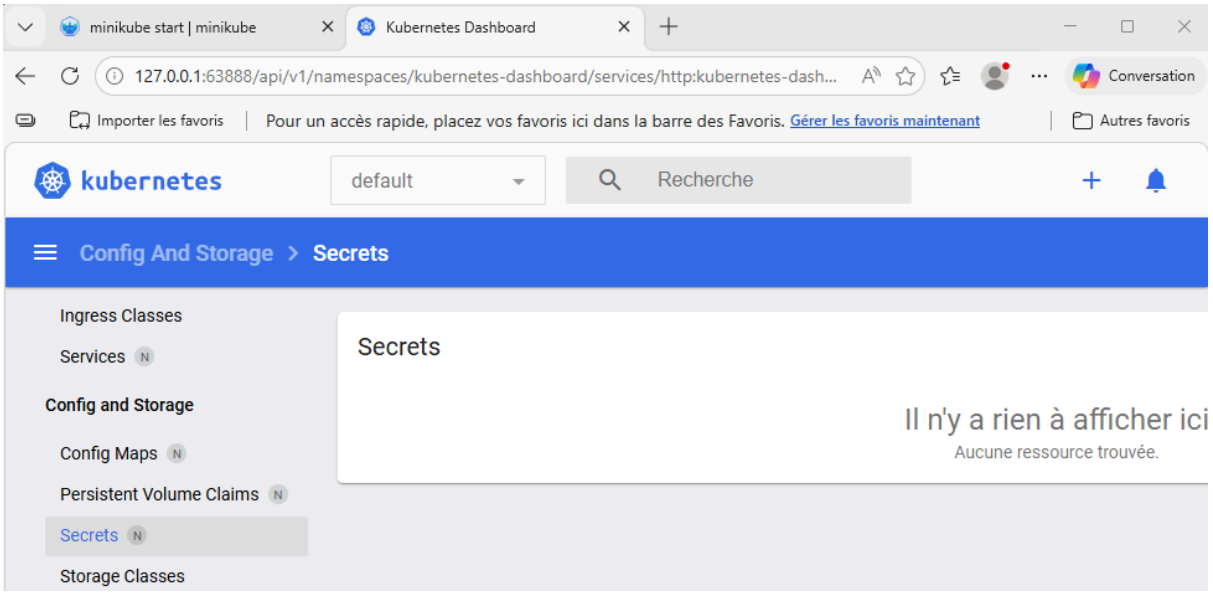
Affichage des Services du cluster :



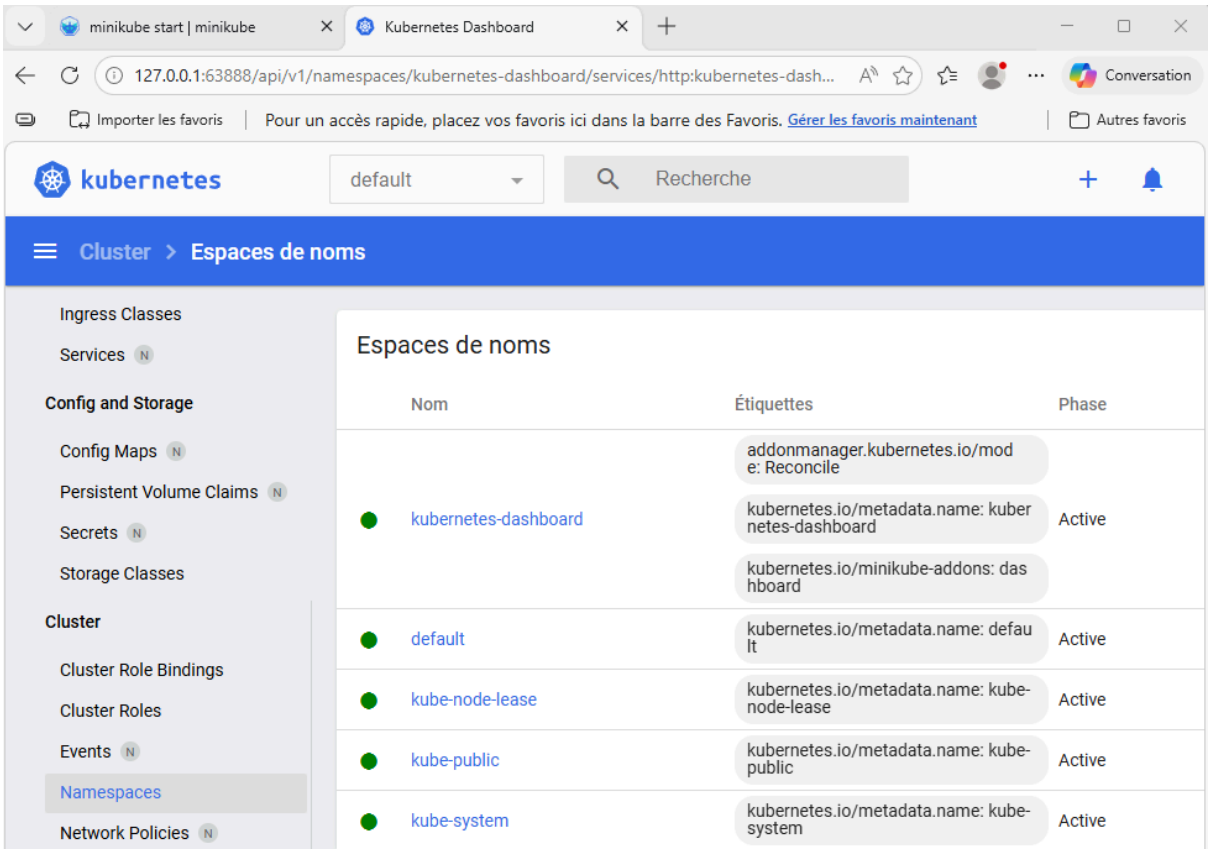
Affichage des ConfigMaps et du stockage :



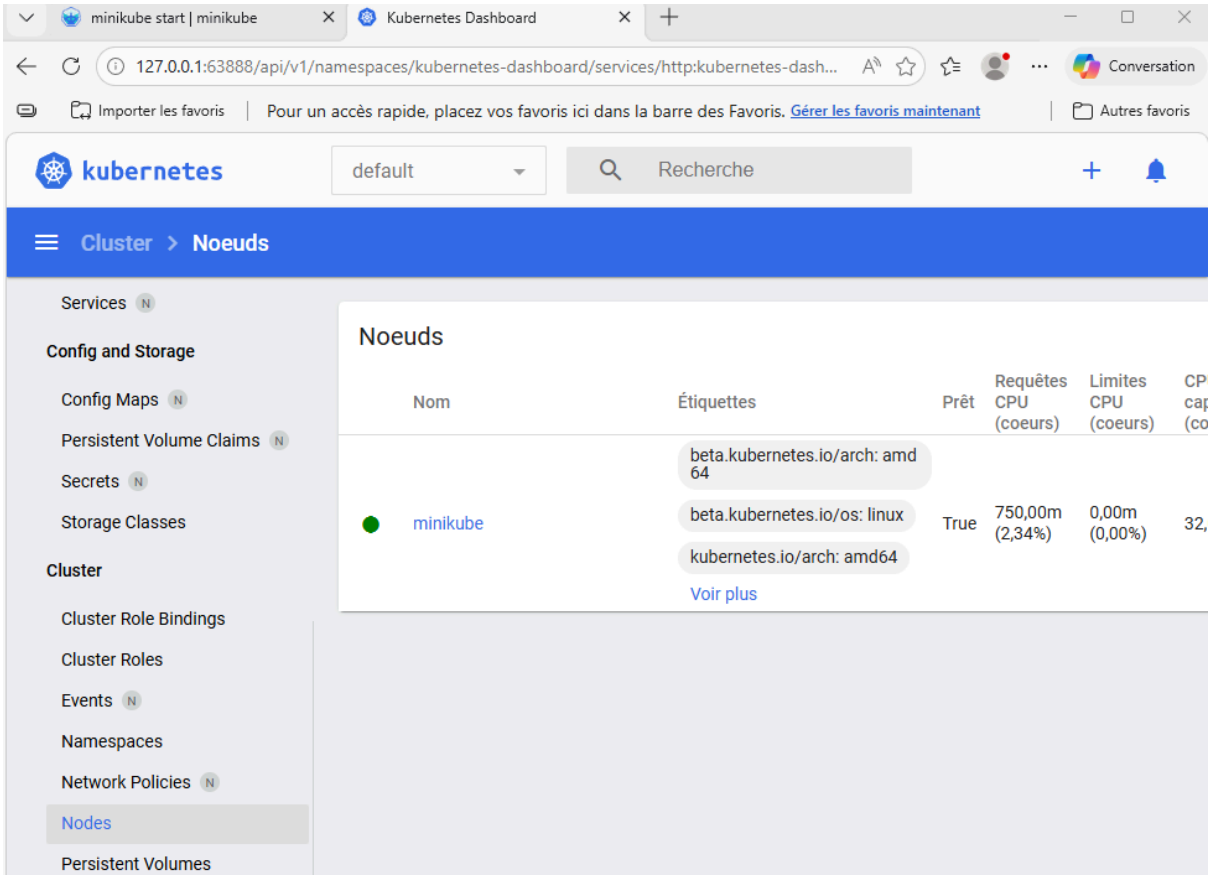
Affichage des Secrets :



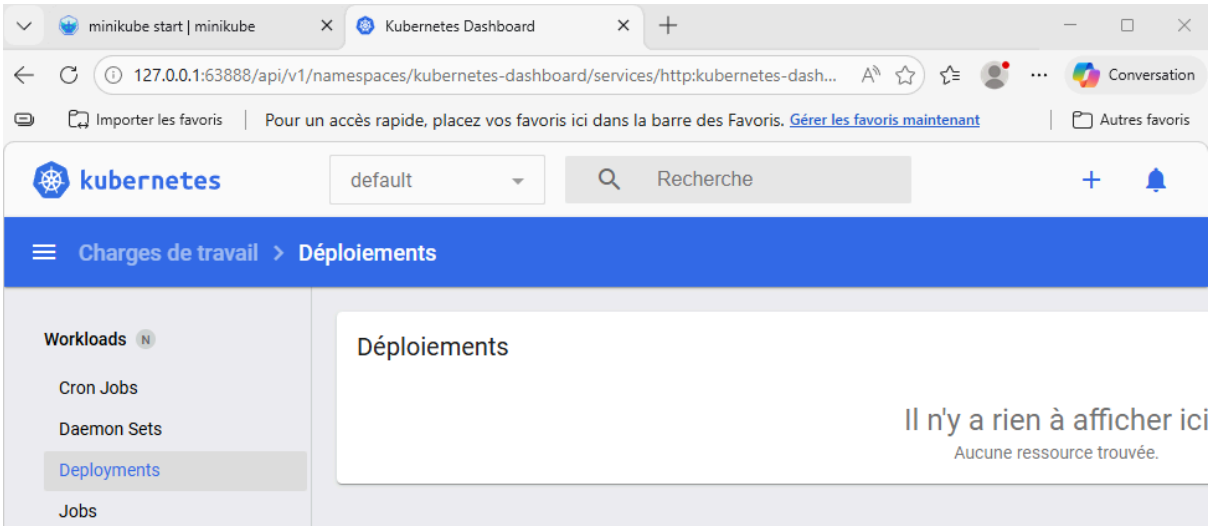
Affichage des Namespaces :



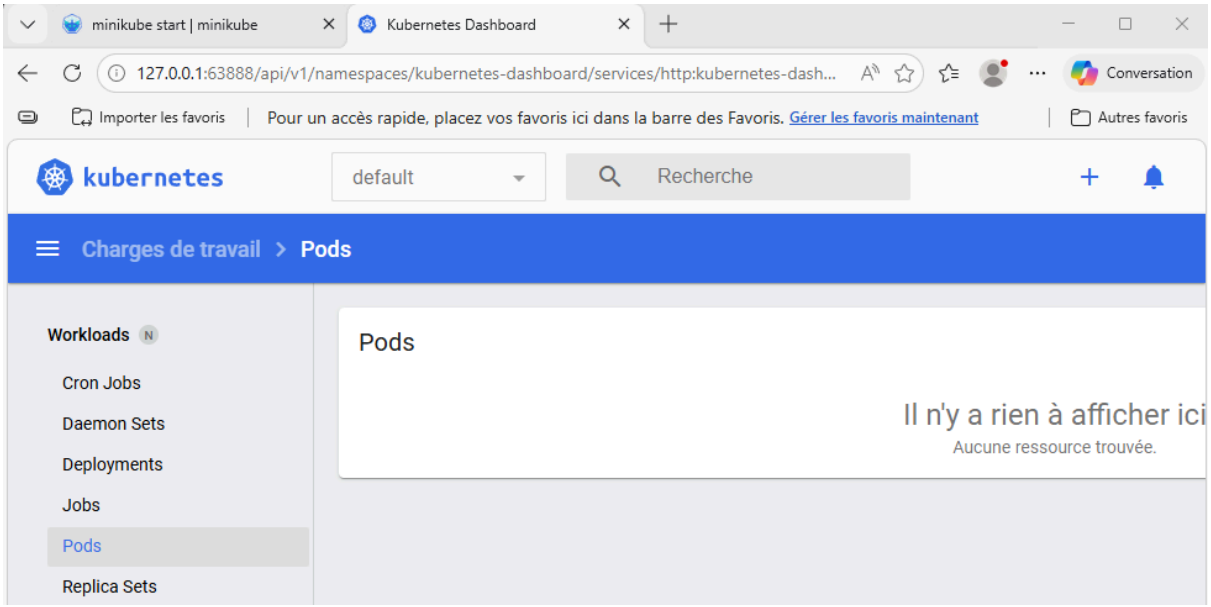
Affichage des Nodes (noeuds) :



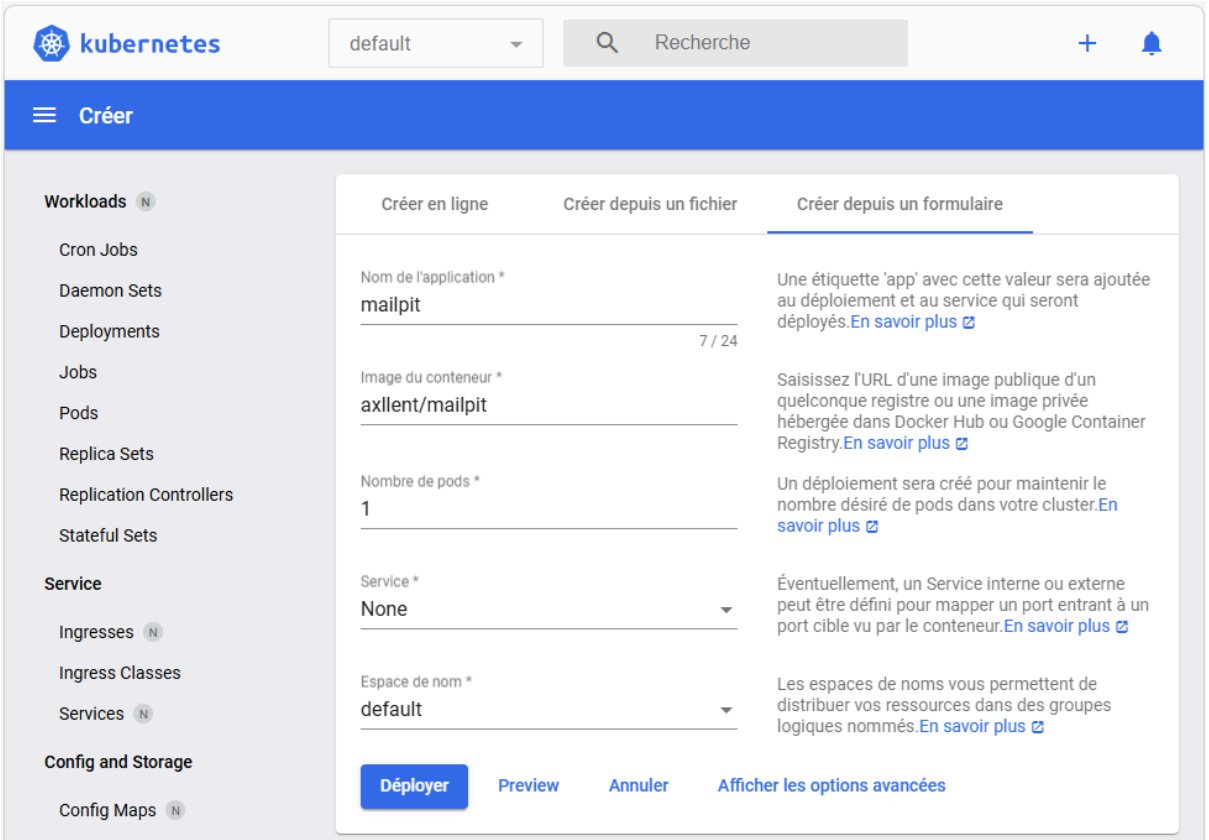
Affichage des Déploiements :



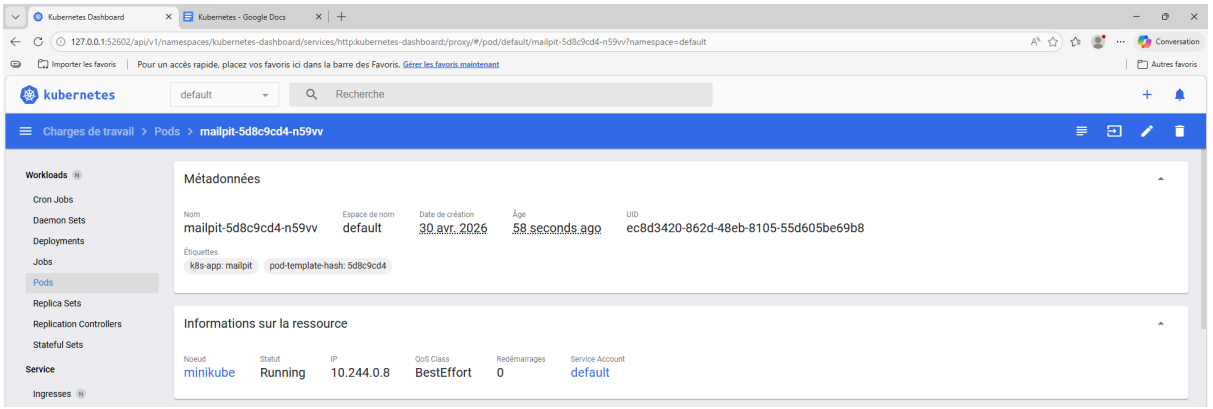
Affichage des Pods :



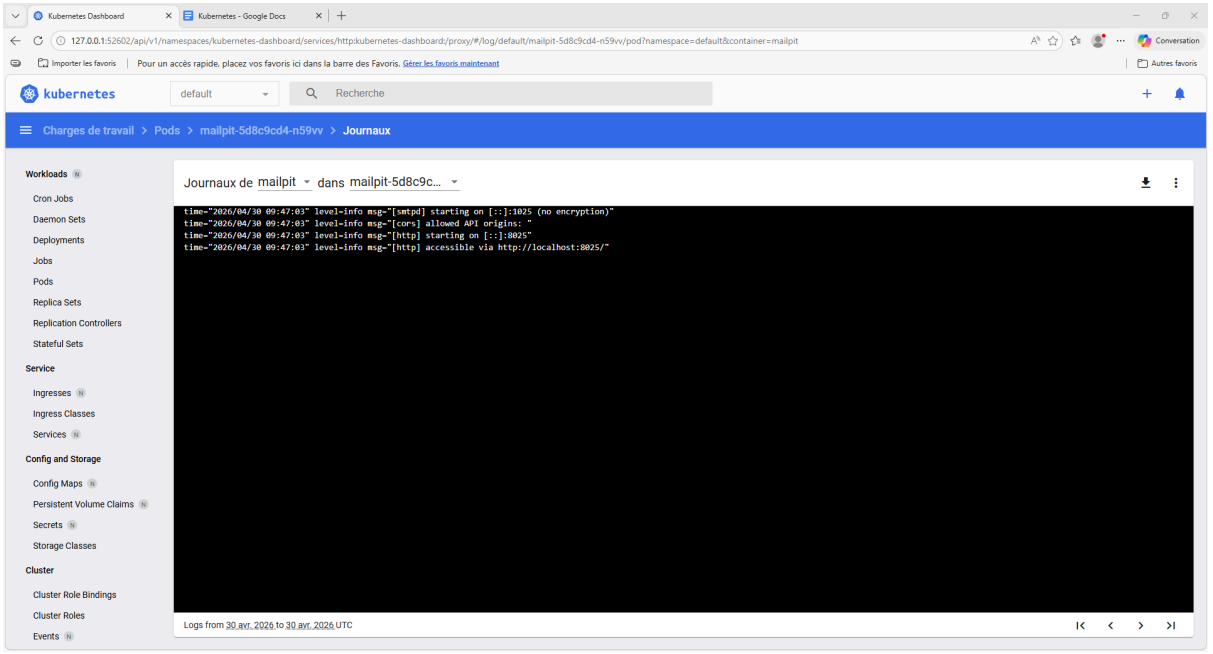
Déploiement de l'application Mailpit :



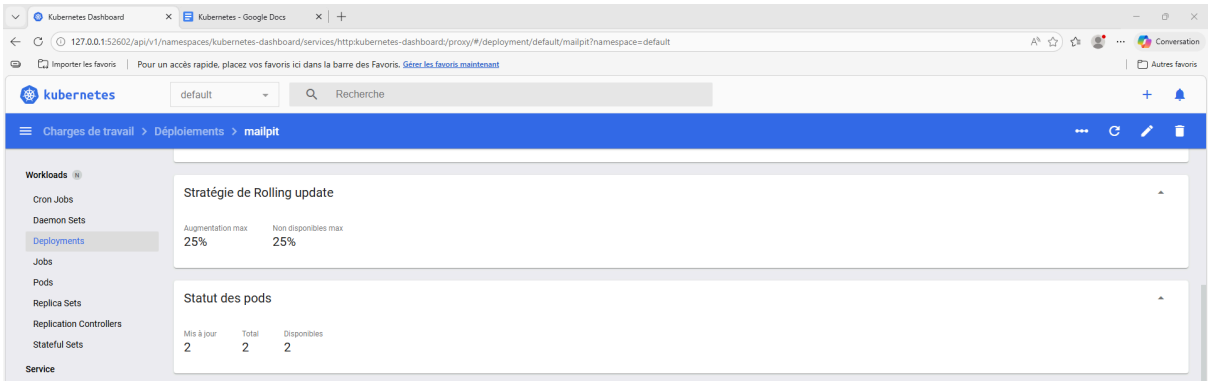
Affichage des détails du pod Mailpit :



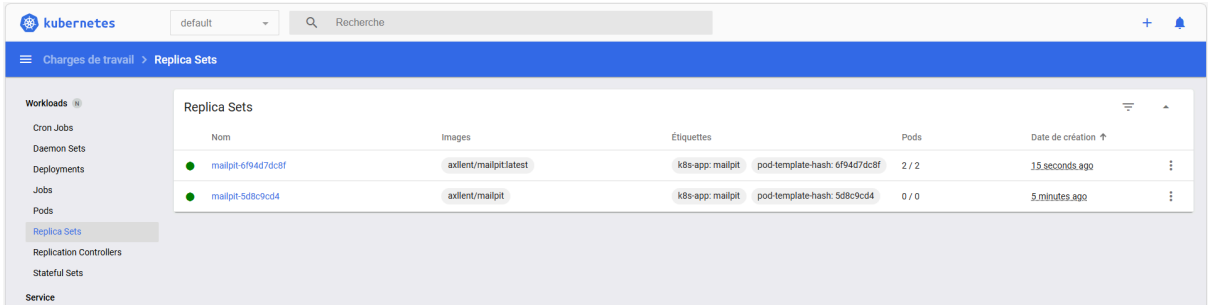
Affichage des logs du pod Mailpit :



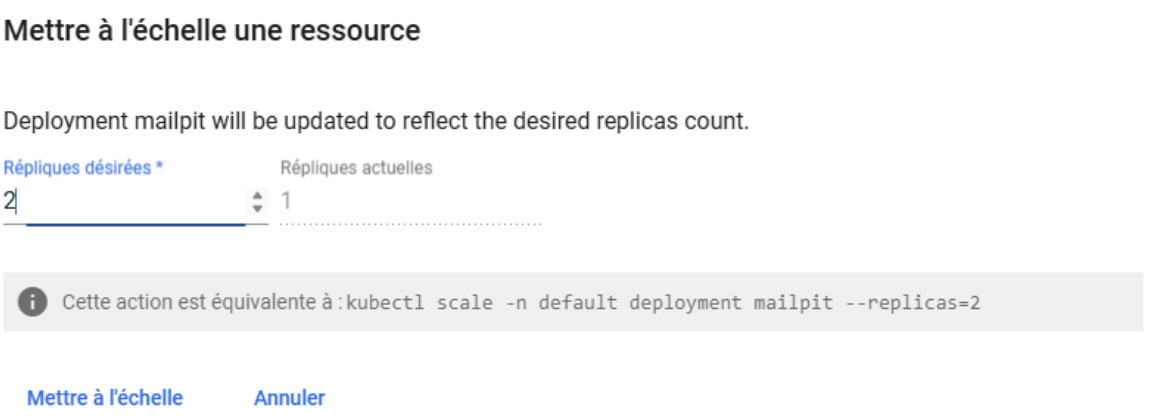
Affichage du déploiement Mailpit :



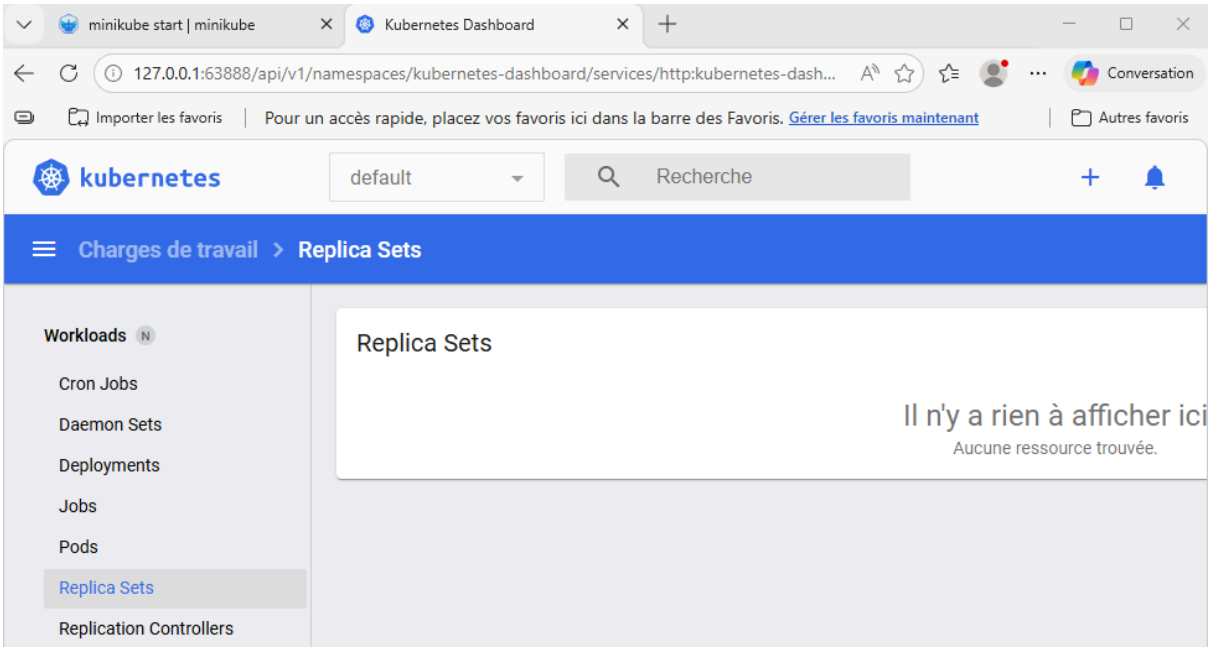
Affichage des ReplicaSets du cluster :



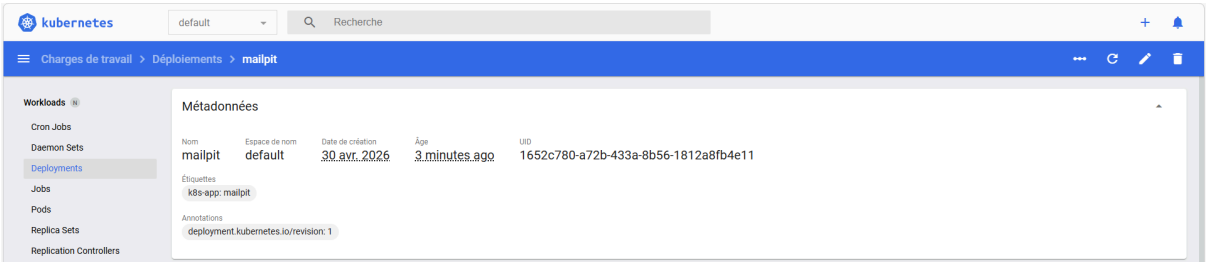
Mise à l'échelle du déploiement à deux répliques :



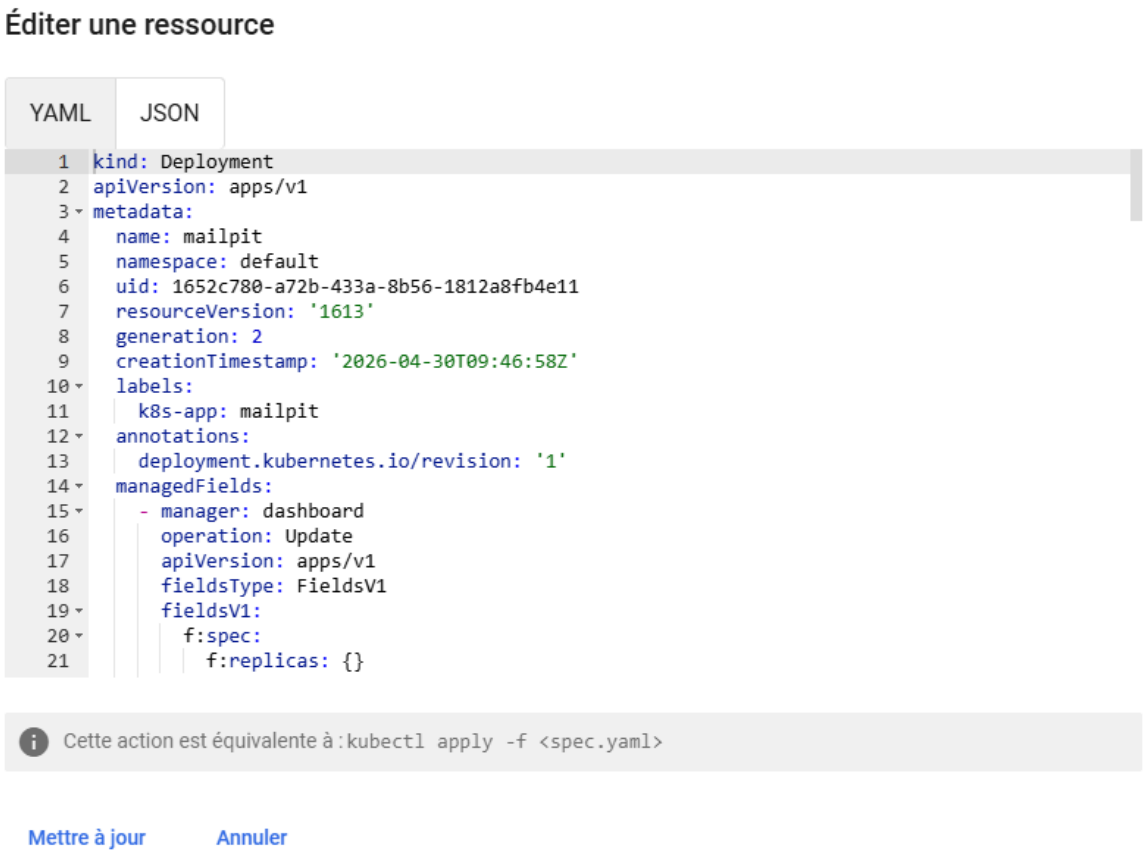
Affichage des ReplicaSets du cluster :



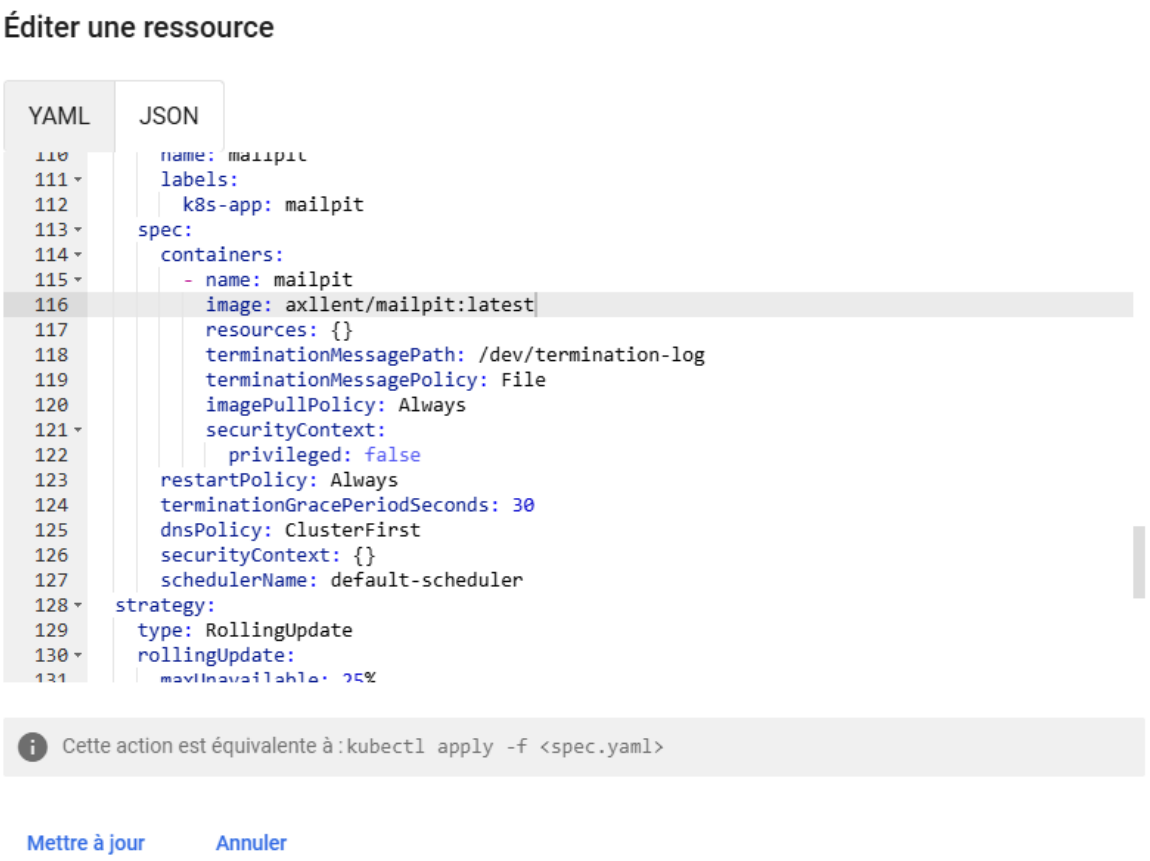
Mise à jour du déploiement Mailpit suite à la mise à l'échelle à 2 répliques.



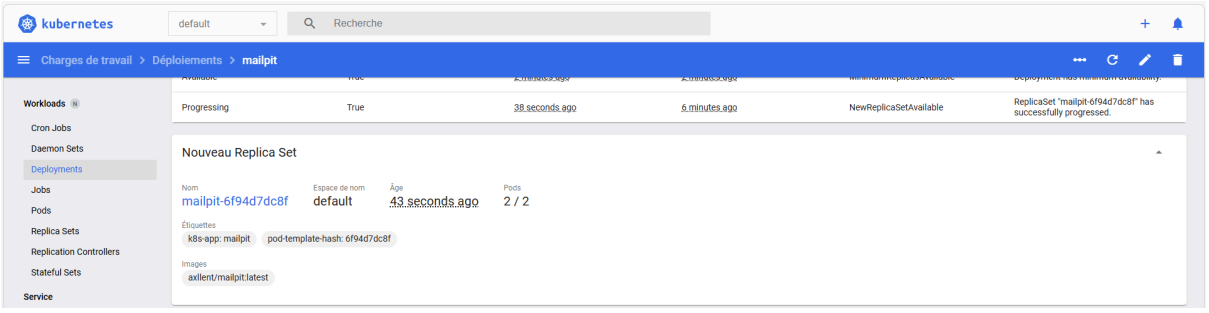
Avant modification du fichier YAML du déploiement Mailpit :



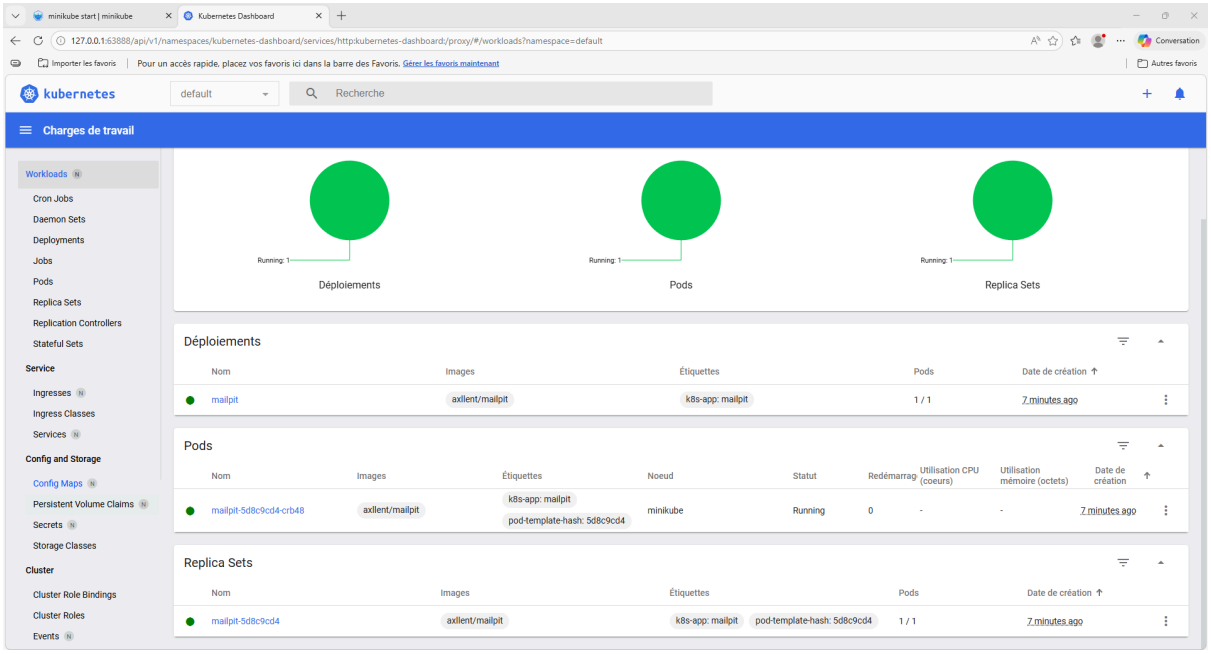
Après modification du fichier YAML du déploiement Mailpit pour mettre à jour la configuration de l'application.



La mise à jour du déploiement Mailpit crée un nouveau ReplicaSet avec les modifications appliquées.

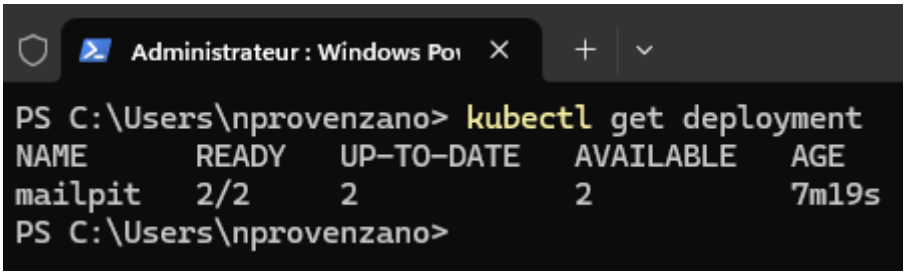


Le tableau de bord permet de visualiser l'état des ressources (déploiements, pods, replicats sets) du cluster Kubernetes.

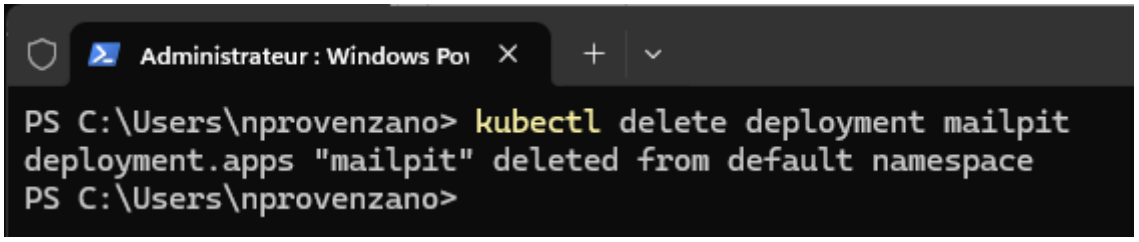


4. Créer un déploiement à l'aide de l'outil kubectl

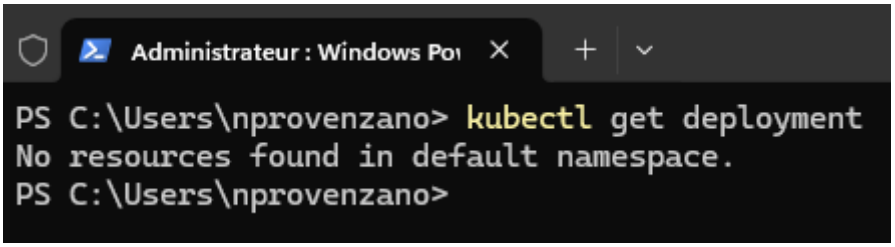
La commande **kubectl get deployment** permet d'afficher les déploiements du cluster et leur état.



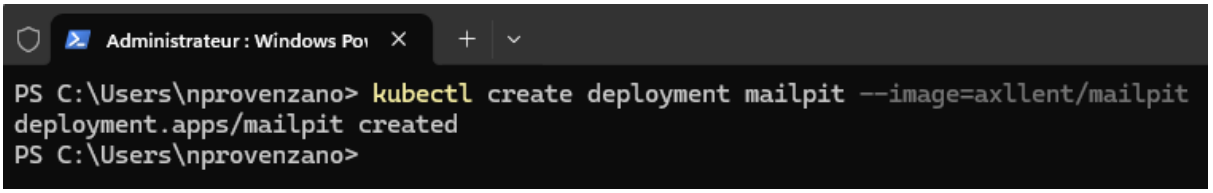
La commande **kubectl delete deployment mailpit** permet de supprimer le déploiement Mailpit, ce qui entraîne la suppression des pods et des ReplicaSets associés.



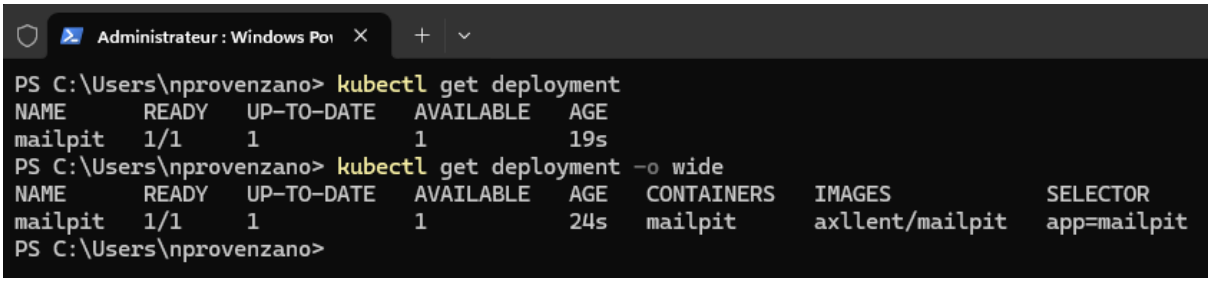
La commande **kubectl get deployment** permet de confirmer que le déploiement Mailpit a bien été supprimé du cluster.



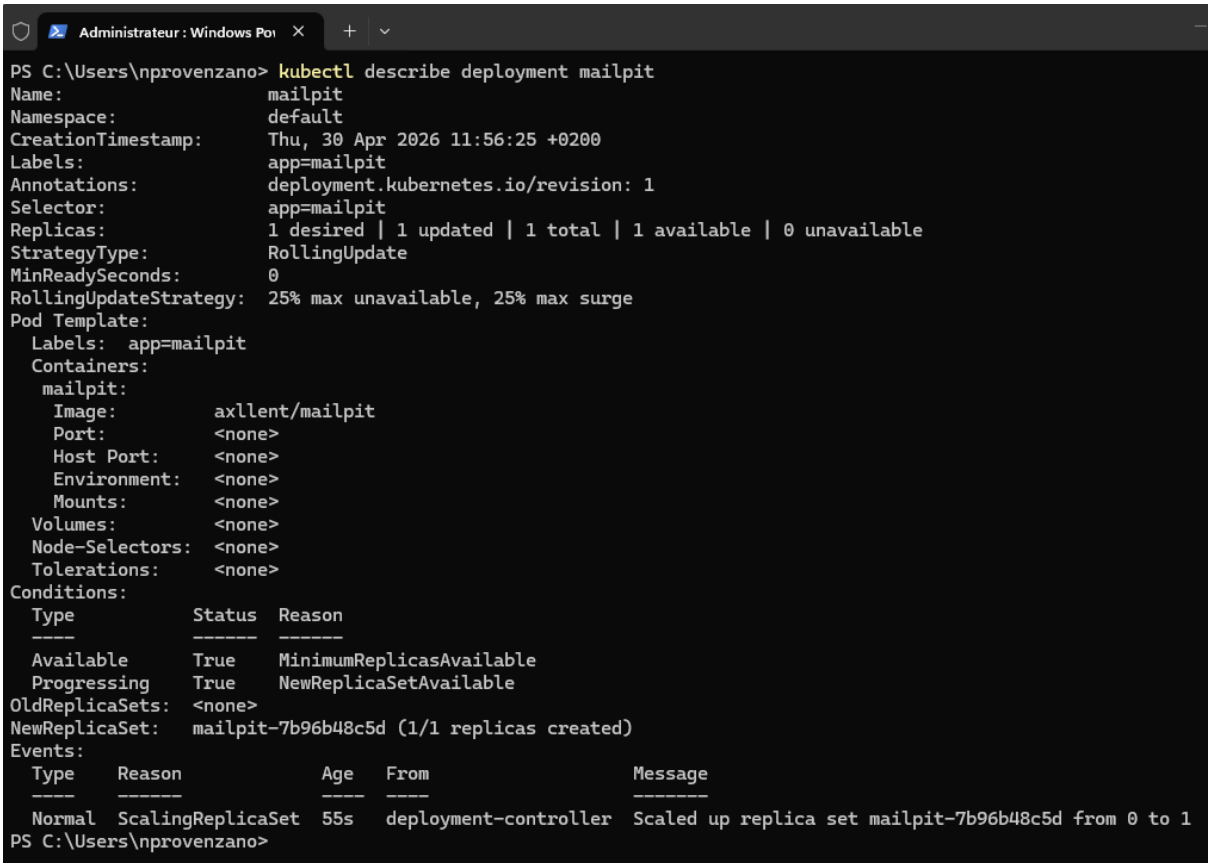
La commande **kubectl create deployment mailpit --image=axllent/mailpit** permet de créer un déploiement Mailpit à partir d'une image Docker.



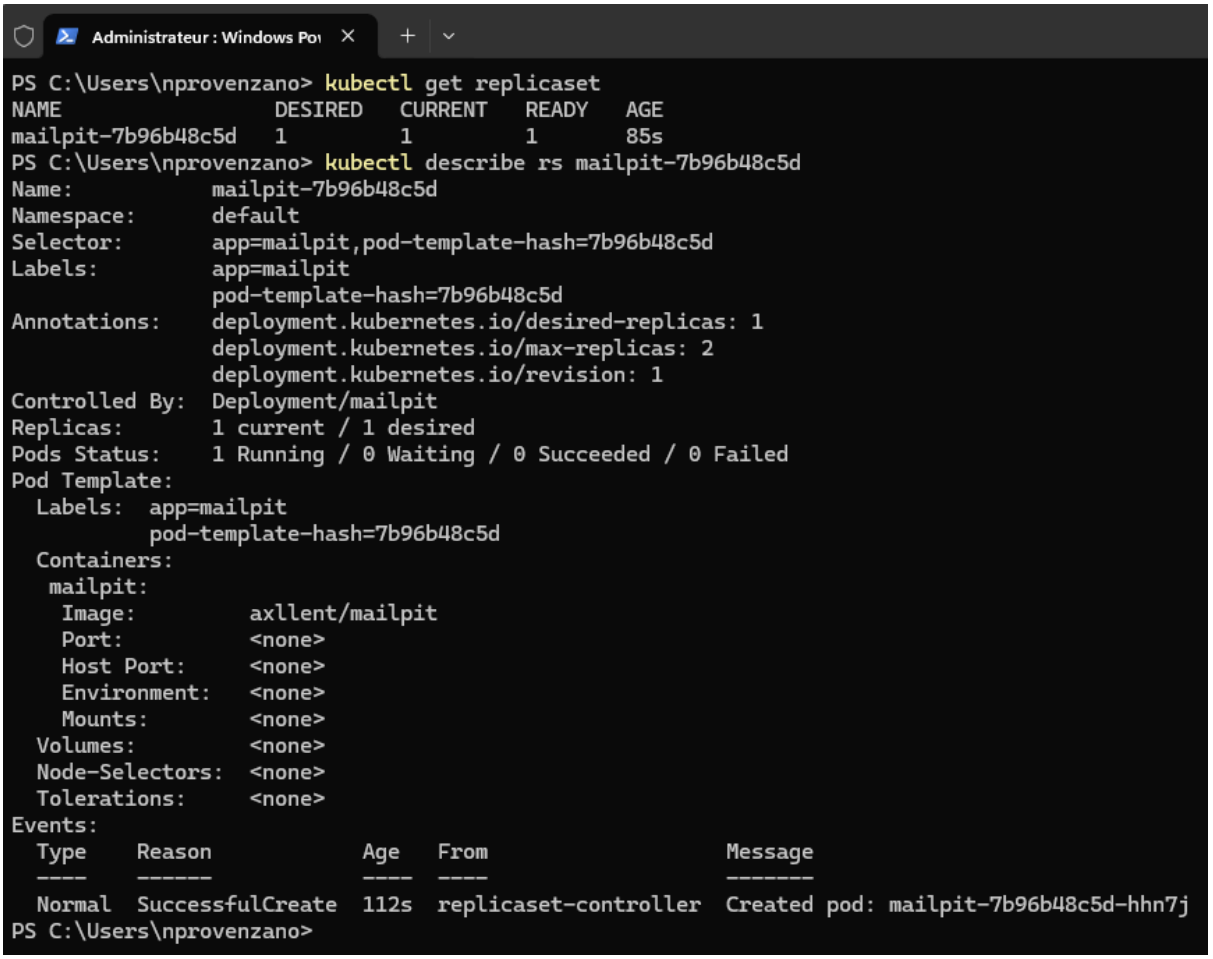
La commande **kubectl get deployment -o wide** permet d'afficher les détails du déploiement Mailpit.



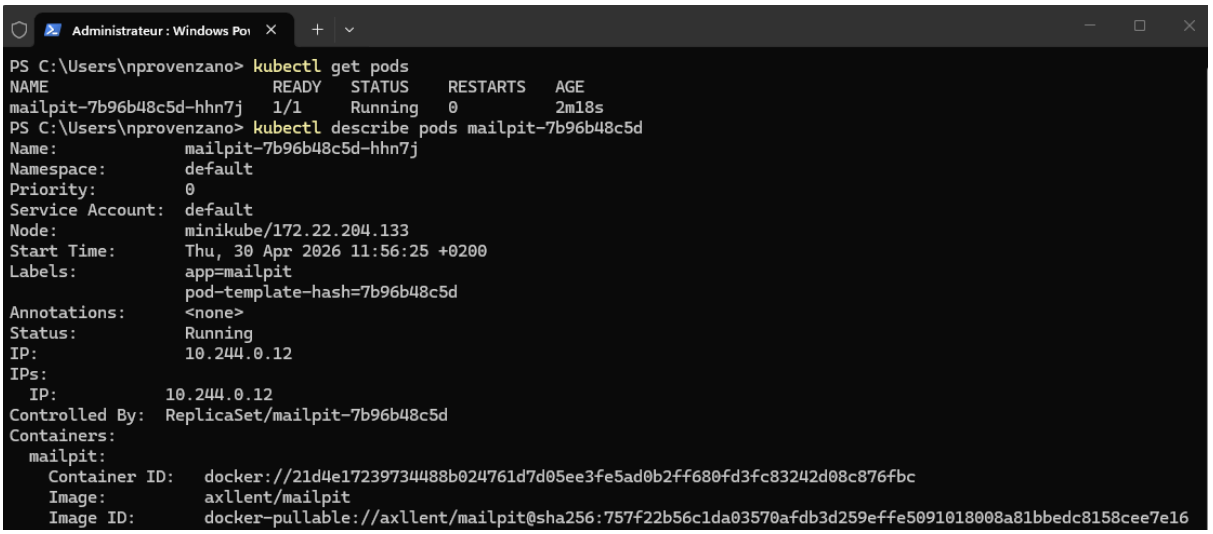
La commande **kubectl describe deployment mailpit** permet d'afficher les informations détaillées du déploiement Mailpit.



La commande **kubectl get replicaset** permet de lister les ReplicaSets, et **kubectl describe rs** permet d'afficher leurs détails comme les pods associés et la configuration.



La commande **kubectl get pods** permet de lister les pods, et **kubectl describe pod** permet d'afficher les détails comme l'état, l'IP et la configuration.



Suite de l'image :

```
Containers:
  mailpit:
    Container ID:   docker://21d4e17239734488b024761d7d05ee3fe5ad0b2ff680fd3fc83242d08c876fbc
    Image:          axllent/mailpit
    Image ID:       docker-pullable://axllent/mailpit@sha256:757f22b56c1da03570afdb3d259effe5091018008a81bbedc8158cee7e16
fdbc
    Port:           <none>
    Host Port:      <none>
    State:          Running
      Started:      Thu, 30 Apr 2026 11:56:26 +0200
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-wtrk7 (ro)
Conditions:
  Type                        Status
  PodReadyToStartContainers   True
  Initialized                 True
  Ready                       True
  ContainersReady             True
  PodScheduled                True
Volumes:
  kube-api-access-wtrk7:
```

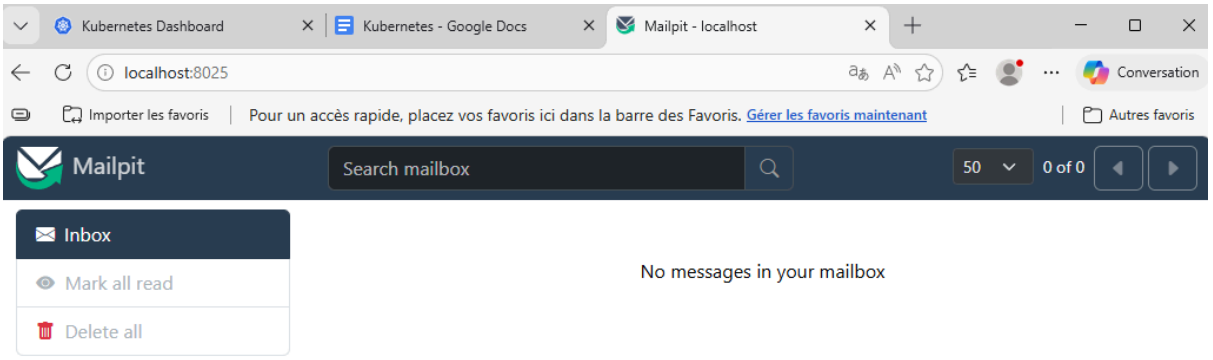
La commande **kubectl logs** permet de consulter les logs du pod Mailpit pour vérifier le démarrage et le fonctionnement du service.

```
Administrateur : Windows Po
PS C:\Users\nprovenzano> kubectl logs mailpit-7b96b48c5d-hhn7j
time="2026/04/30 09:56:26" level=info msg="[smtpd] starting on [::]:1025 (no encryption)"
time="2026/04/30 09:56:26" level=info msg="[cors] allowed API origins: "
time="2026/04/30 09:56:26" level=info msg="[http] starting on [::]:8025"
time="2026/04/30 09:56:26" level=info msg="[http] accessible via http://localhost:8025/"
PS C:\Users\nprovenzano> kubectl logs mailpit-7b96b48c5d-hhn7j -c mailpit
time="2026/04/30 09:56:26" level=info msg="[smtpd] starting on [::]:1025 (no encryption)"
time="2026/04/30 09:56:26" level=info msg="[cors] allowed API origins: "
time="2026/04/30 09:56:26" level=info msg="[http] starting on [::]:8025"
time="2026/04/30 09:56:26" level=info msg="[http] accessible via http://localhost:8025/"
PS C:\Users\nprovenzano>
```

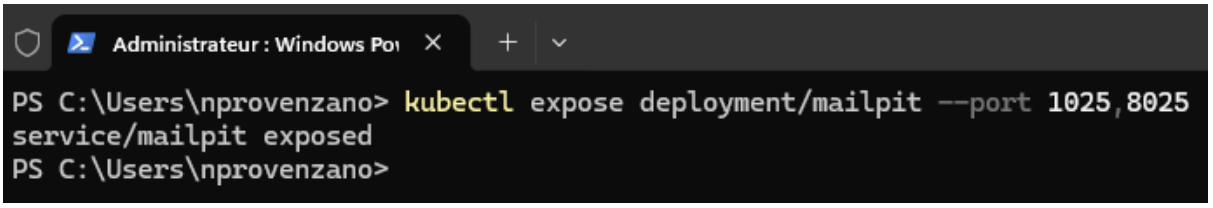
La commande **kubectl port-forward** permet d'accéder au service Mailpit en redirigeant le port du pod vers localhost.

```
Administrateur : Windows Po
PS C:\Users\nprovenzano> kubectl port-forward mailpit-7b96b48c5d-hhn7j 8025
Forwarding from 127.0.0.1:8025 -> 8025
Forwarding from [::1]:8025 -> 8025
Handling connection for 8025
Handling connection for 8025
Handling connection for 8025
```

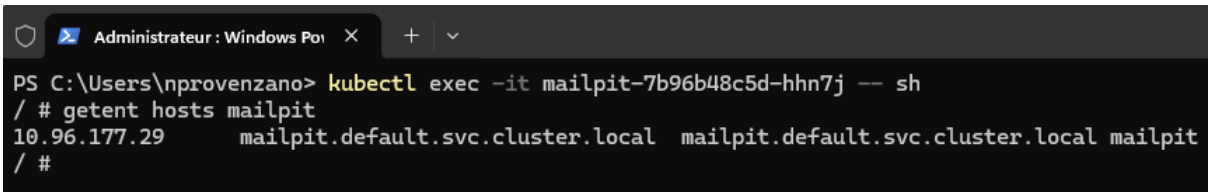
L'accès à **localhost:8025** permet de vérifier l'interface web de Mailpit.



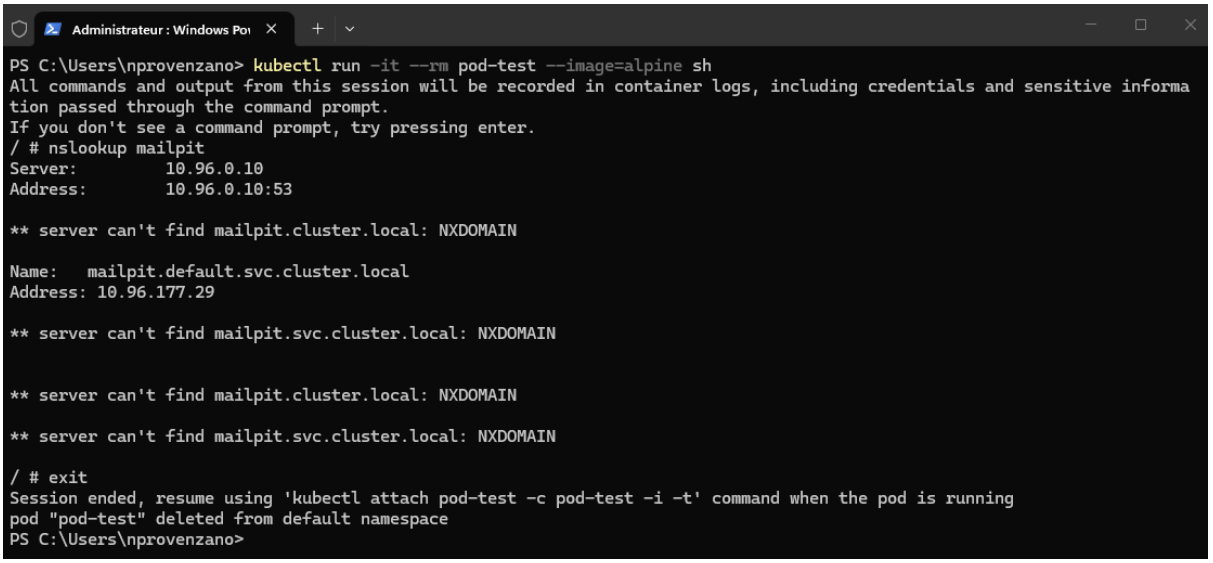
La commande **kubectl expose deployment mailpit** permet de créer un service pour exposer l'application Mailpit sur des ports accessibles.



La commande **kubectl exec -it ... -- sh** permet d'accéder au terminal du conteneur pour exécuter des commandes à l'intérieur du pod.

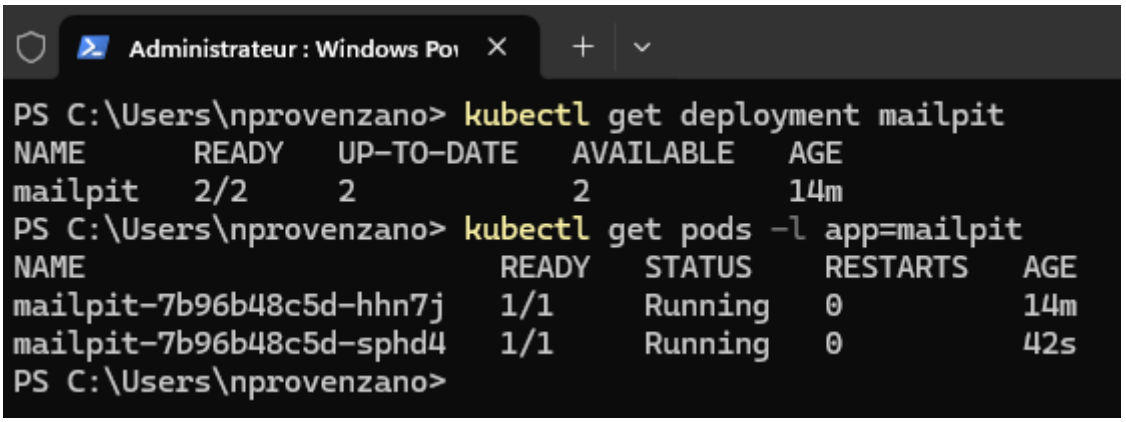


La commande **kubectl run** permet de créer un pod temporaire pour tester la résolution DNS du service Mailpit.

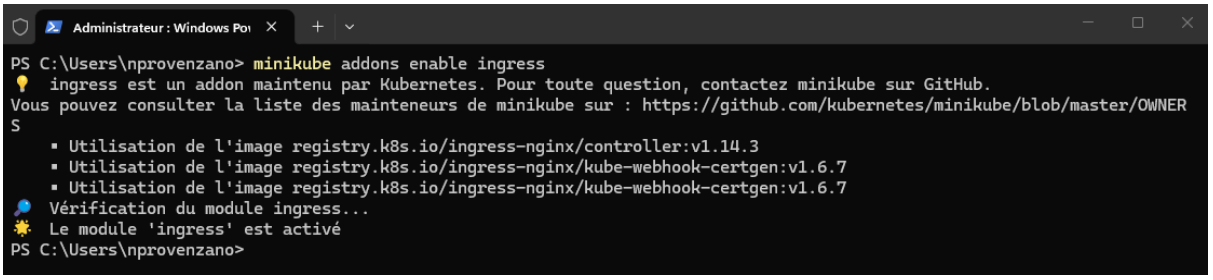


La commande **kubectl scale deployment mailpit --replicas=2** permet de mettre à l'échelle le déploiement en créant 2 répliques de l'application.

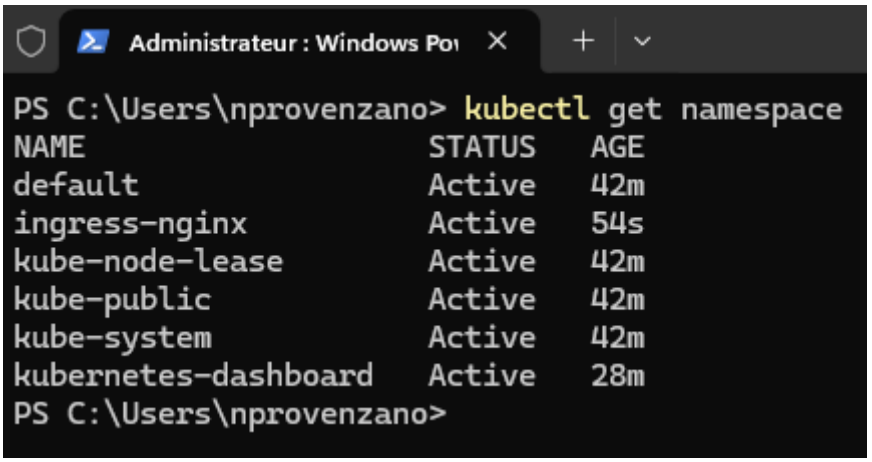
Les commandes **kubectl get deployment** et **kubectl get pods** permettent de vérifier que le déploiement Mailpit possède bien deux répliques en fonctionnement.



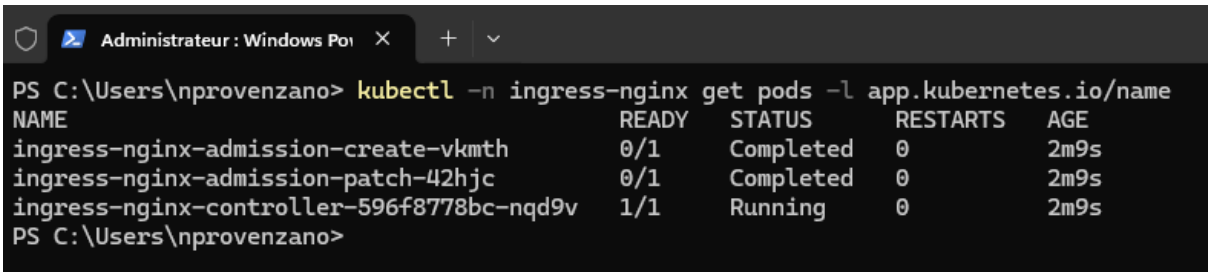
La commande **minikube addons enable ingress** permet d'activer le contrôleur Ingress dans le cluster Kubernetes.



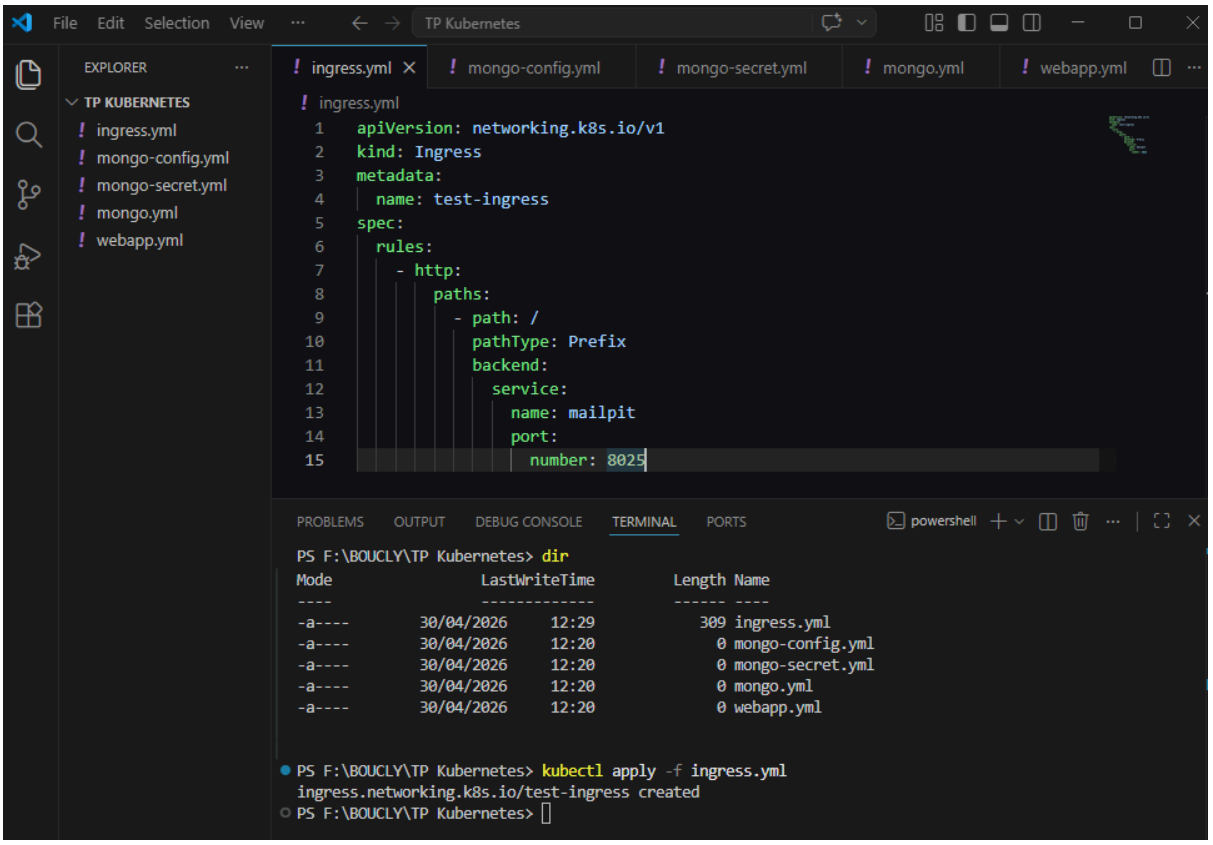
La commande **kubectl get namespace** permet d'afficher les différents espaces de noms du cluster Kubernetes.



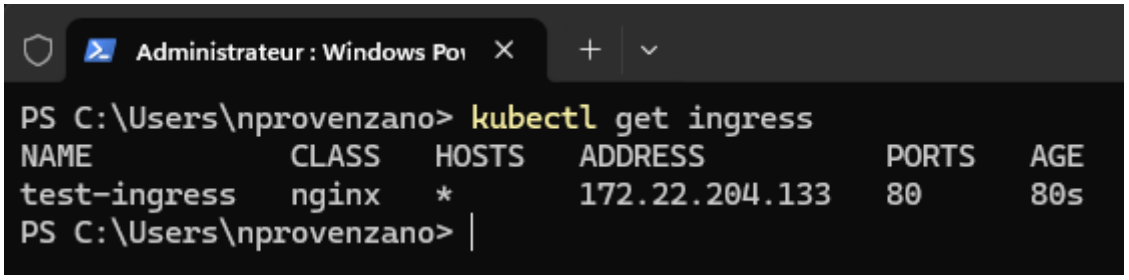
La commande **kubectl -n ingress-nginx get pods** permet de vérifier le fonctionnement des pods du contrôleur Ingress. Trois pods sont présents : deux terminés (Completed) et un pod principal en état Running, indique que le contrôleur Ingress est actif.



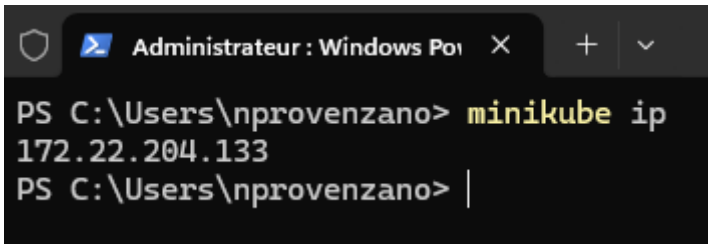
Le fichier **ingress.yml** permet de définir une règle Ingress pour rediriger les requêtes vers le service Mailpit.



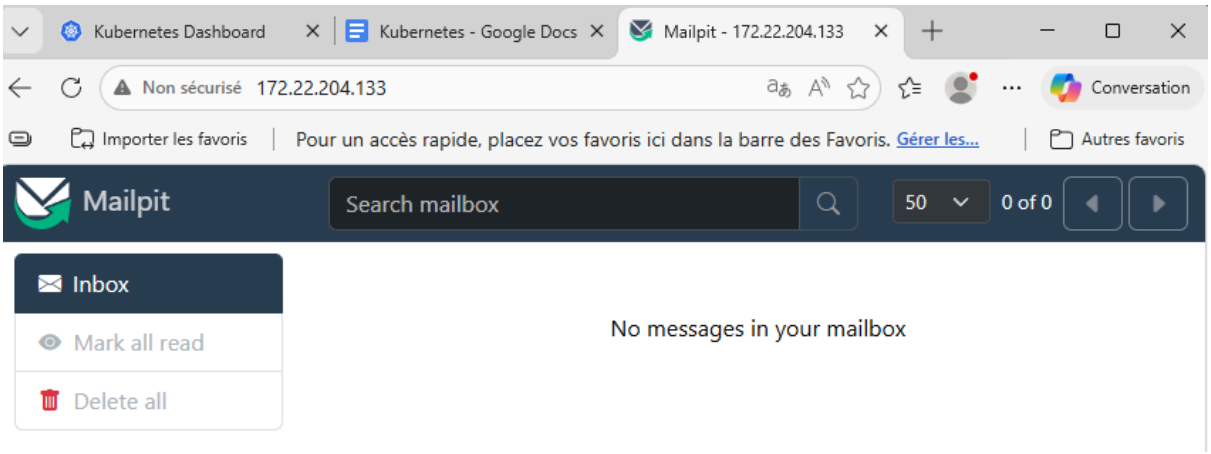
La commande **kubectl get ingress** permet de vérifier que la ressource Ingress a bien été créée et d'obtenir l'adresse d'accès à l'application.



La commande **minikube ip** permet d'afficher l'adresse IP du cluster Minikube pour accéder à l'interface de Mailpit.

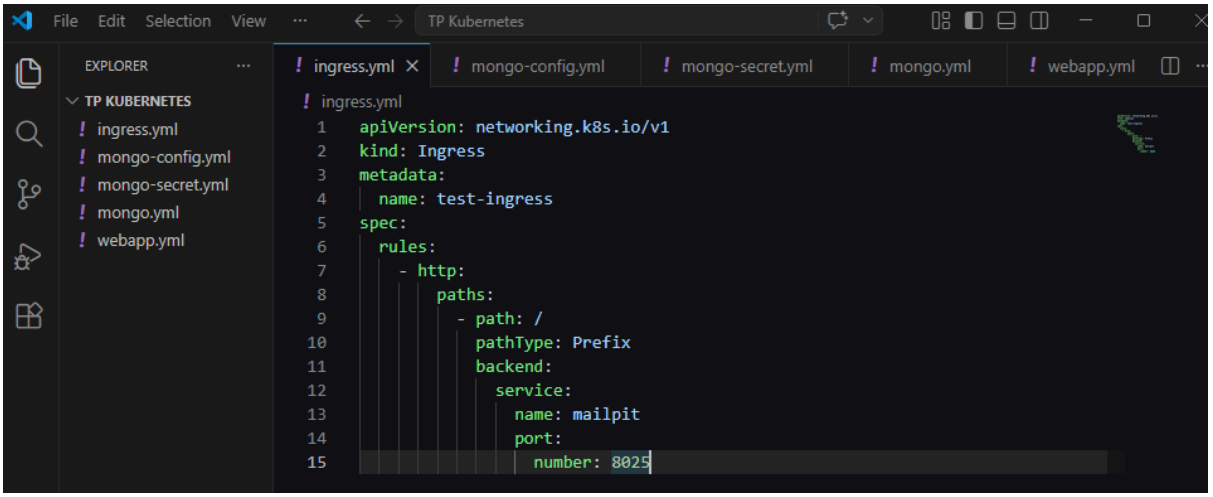


L'accès à l'adresse IP du cluster Minikube dans le navigateur permet de confirmer que l'application Mailpit est bien accessible depuis l'extérieur grâce à Ingress.



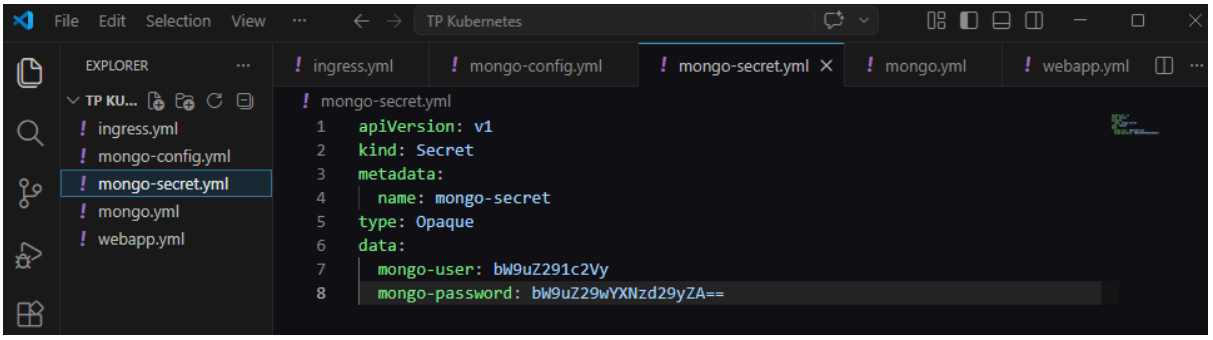
5. Automatisation de déploiement par fichier YAML

Le fichier **ingress.yml** permet de définir une règle Ingress redirigeant les requêtes vers le service Mailpit.

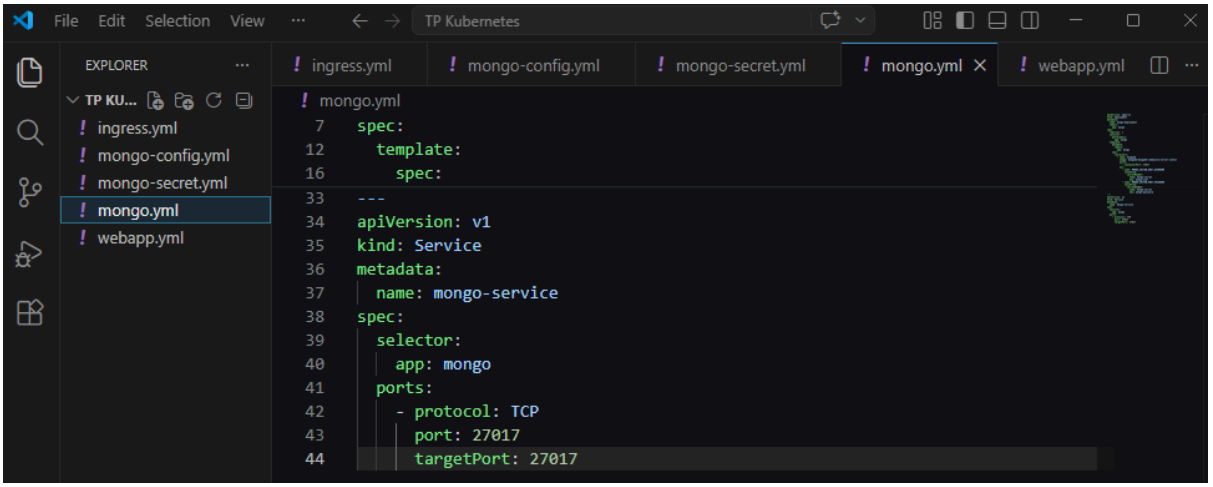


Le fichier **mongo-config.yml** permet de créer une ConfigMap contenant les paramètres de configuration de MongoDB.

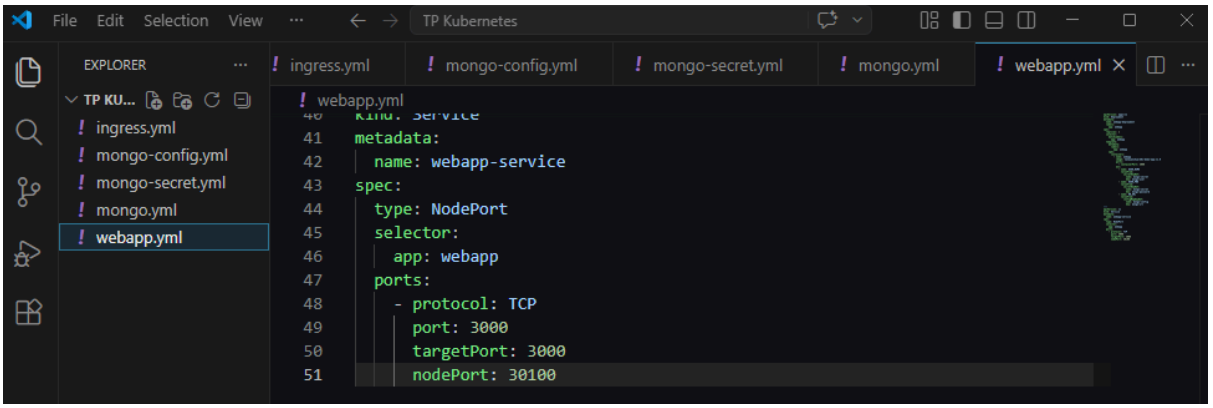
Le fichier **mongo-secret.yml** permet de créer un Secret afin de stocker de manière sécurisée les identifiants et mots de passe de MongoDB.



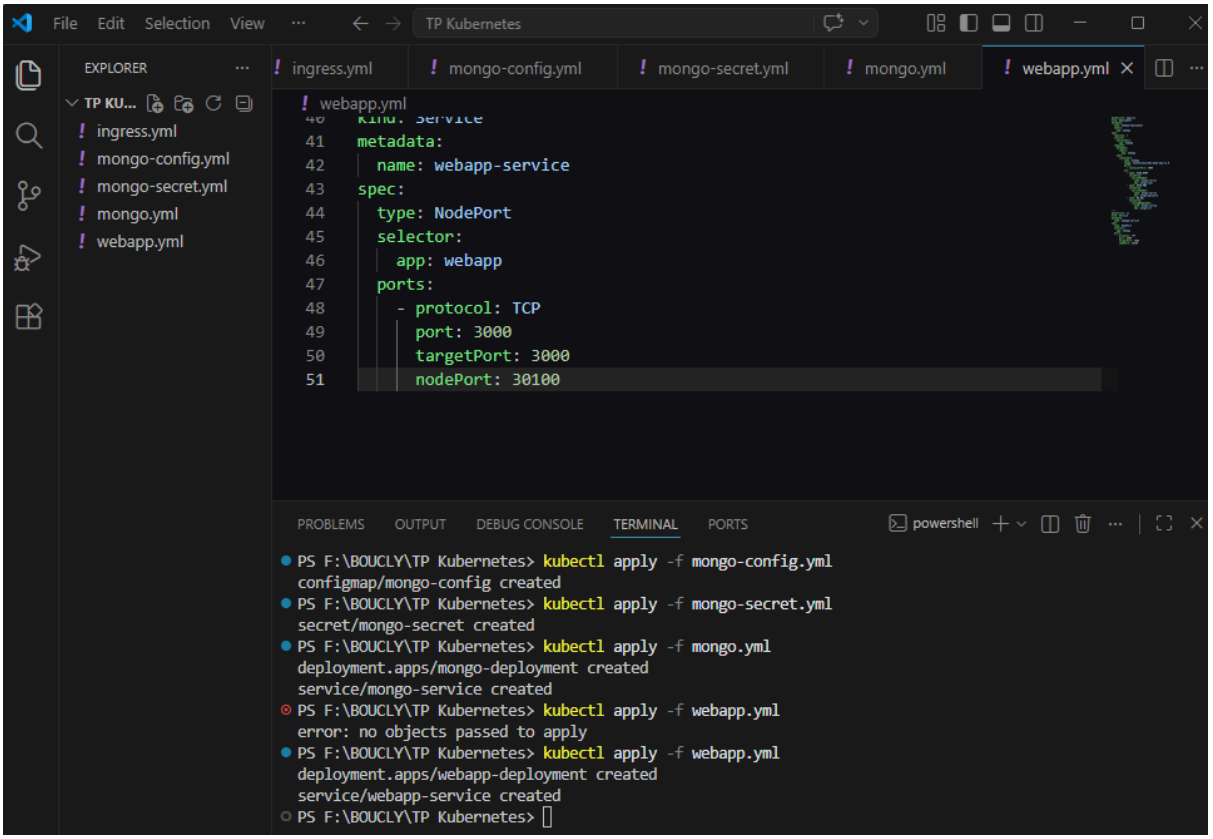
Le fichier **mongo.yml** permet de créer un déploiement MongoDB et un service associé pour permettre la communication avec les autres applications.



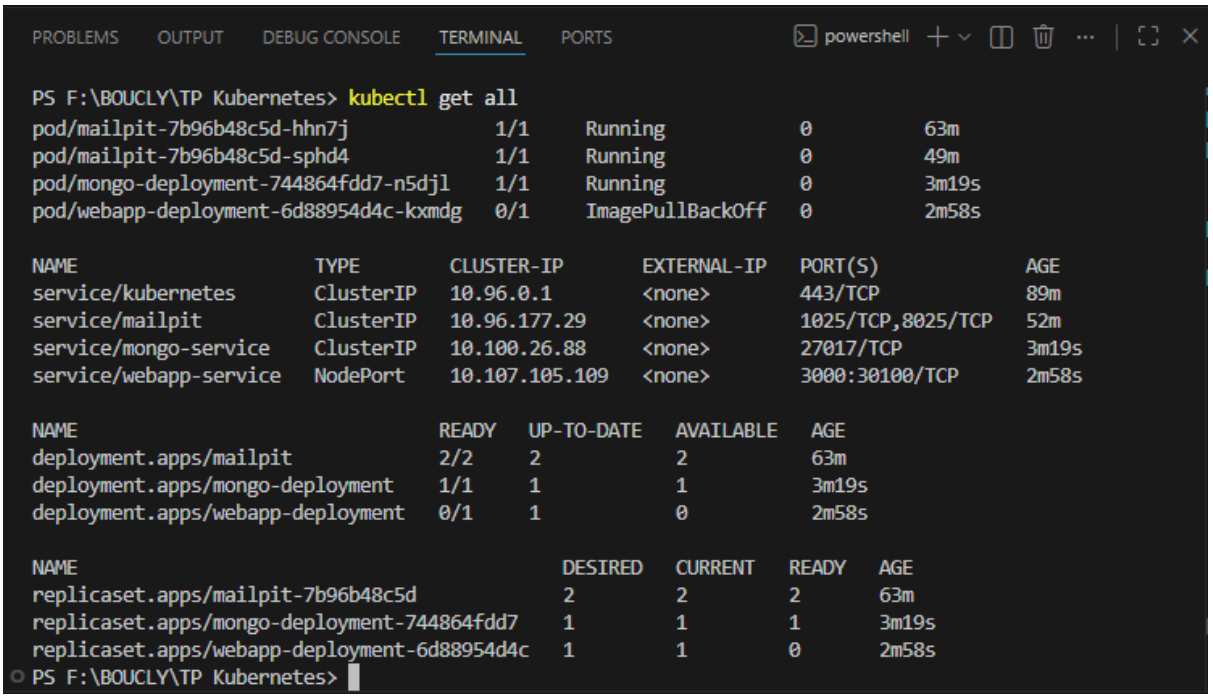
Le fichier **webapp.yml** permet de créer le déploiement de l'application web et un service NodePort afin de rendre l'application accessible depuis l'extérieur du cluster.



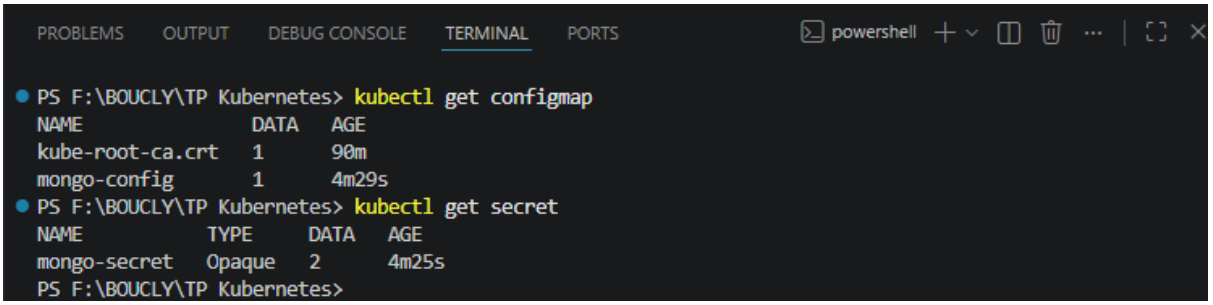
La commande **kubectl apply -f** permet de déployer les composants de l'application à partir des fichiers YAML.



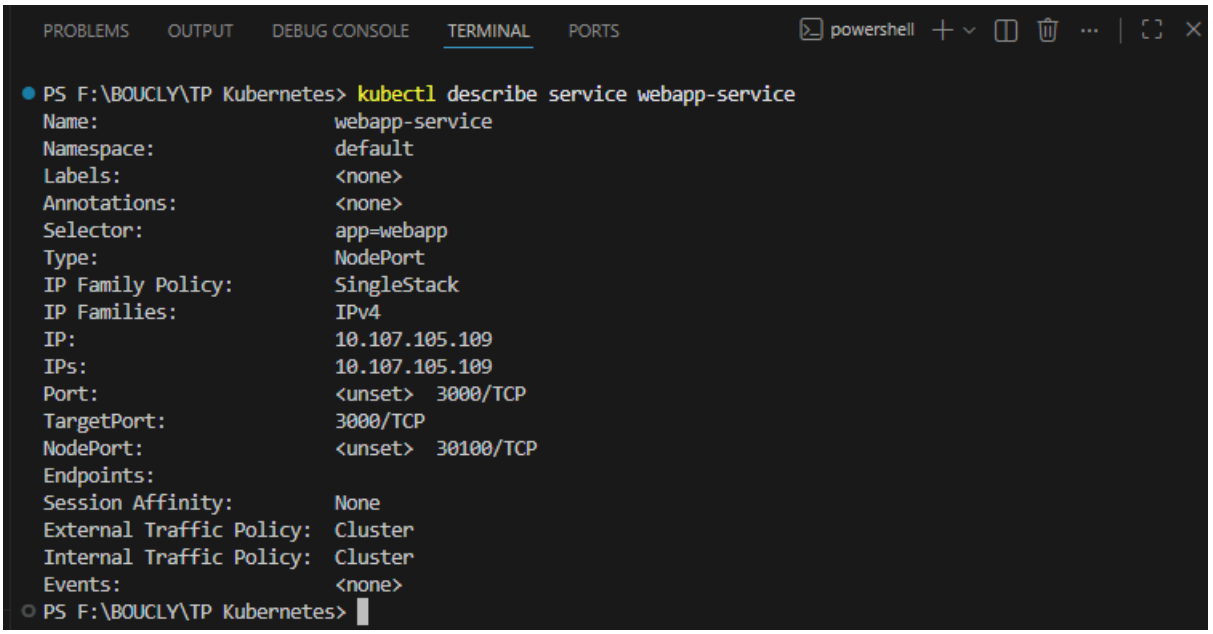
La commande **kubectl get all** permet d'afficher l'ensemble des ressources du cluster (pods, services, déploiements et ReplicaSets).



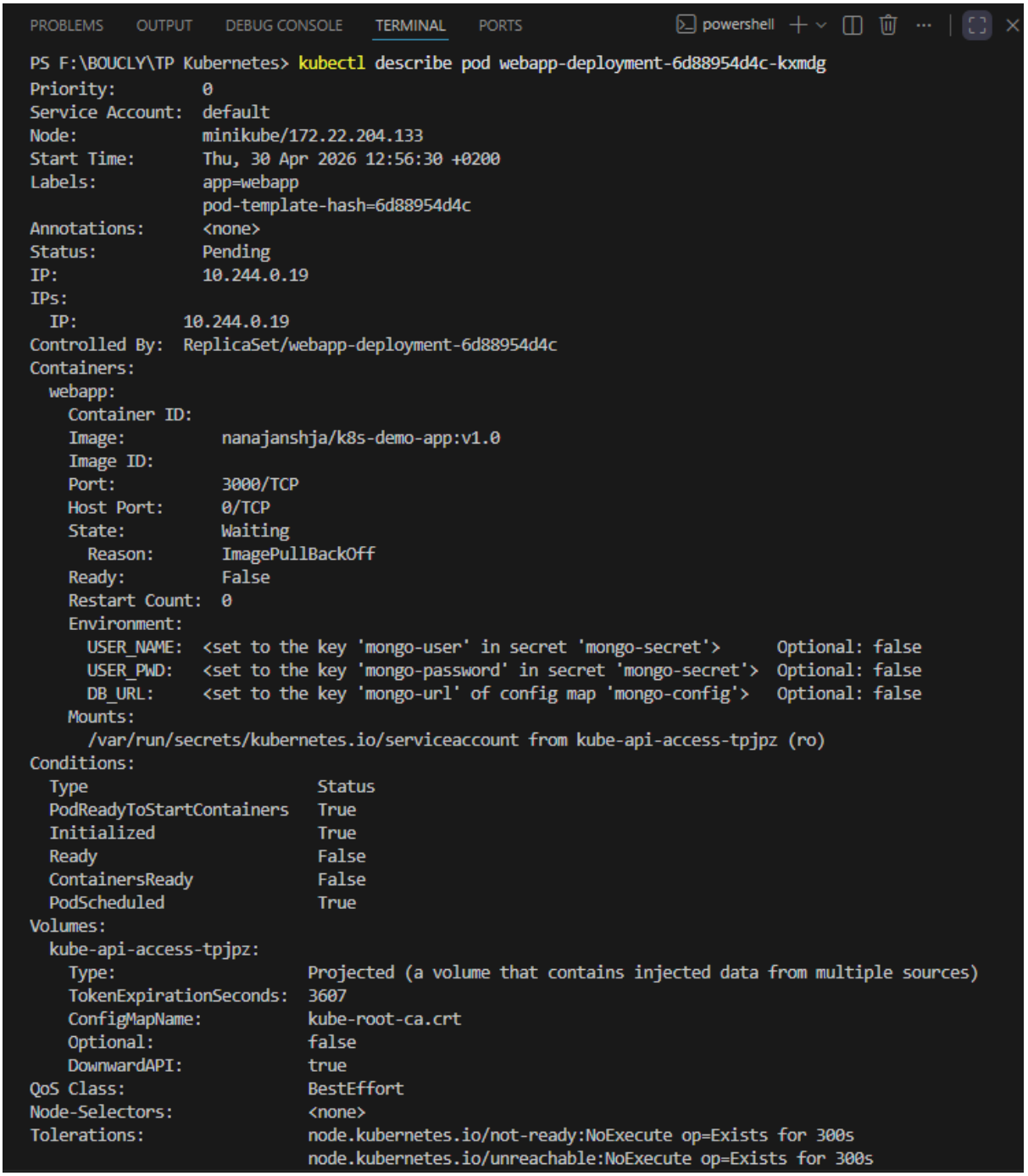
Les commandes **kubectl get configmap** et **kubectl get secret** permettent de vérifier la présence des configurations et des données sensibles utilisées par l'application.



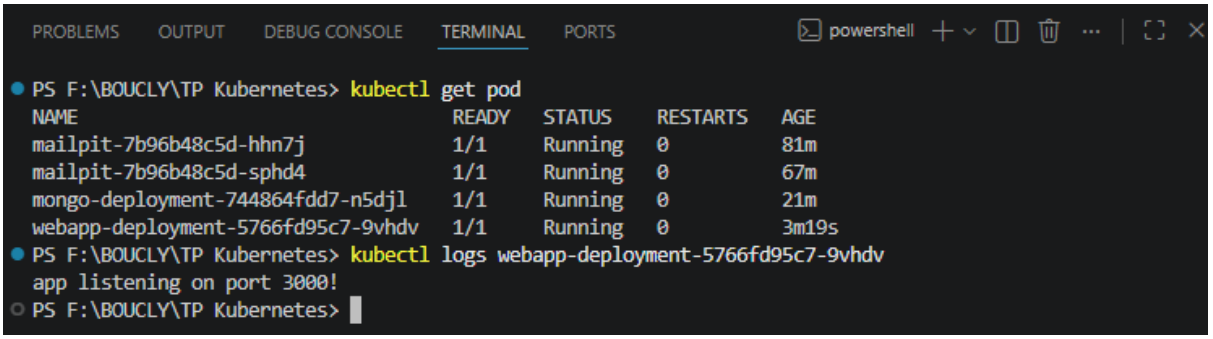
La commande **kubectl describe service webapp-service** permet d'afficher la configuration du service, notamment les ports, l'adresse IP et le type NodePort.



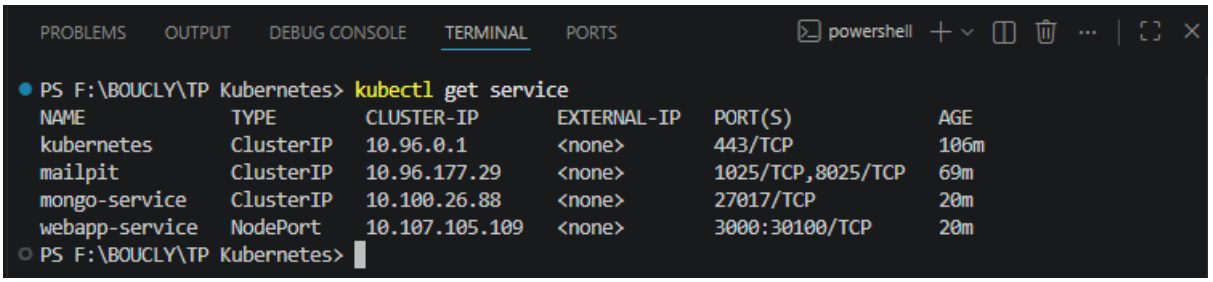
La commande **kubectl describe pod** permet d'afficher les détails du pod et de détecter une erreur de récupération de l'image empêchant son démarrage.



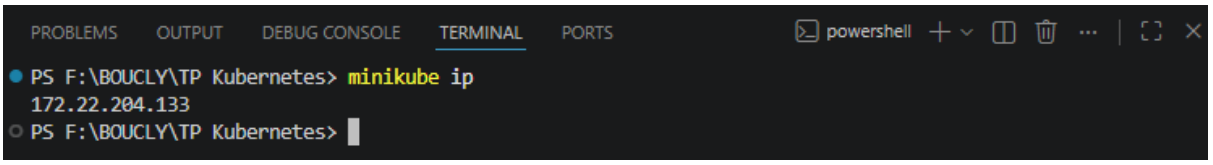
La commande **kubectl describe pod** permet d'analyser l'état du pod webapp et d'identifier une erreur de type ImagePullBackOff empêchant le démarrage du conteneur.



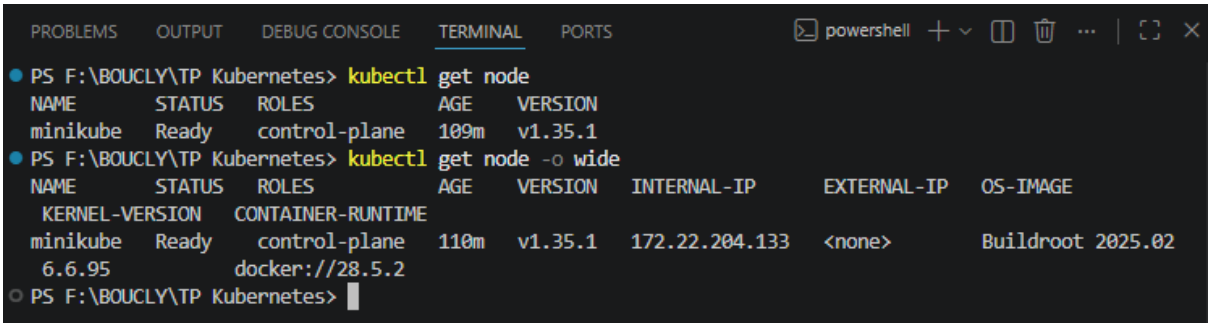
La commande **kubectl get service** permet de lister les services et de vérifier leurs types, adresses IP et ports.



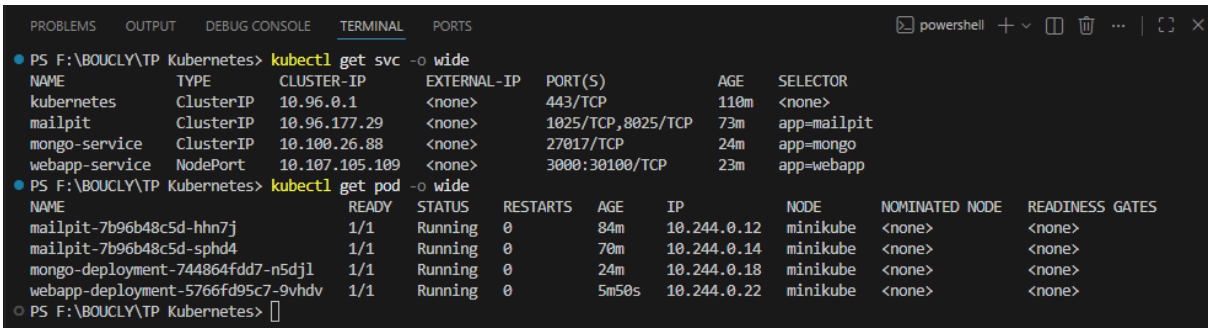
La commande **minikube ip** permet de récupérer l'adresse IP du cluster afin d'accéder aux services déployés.



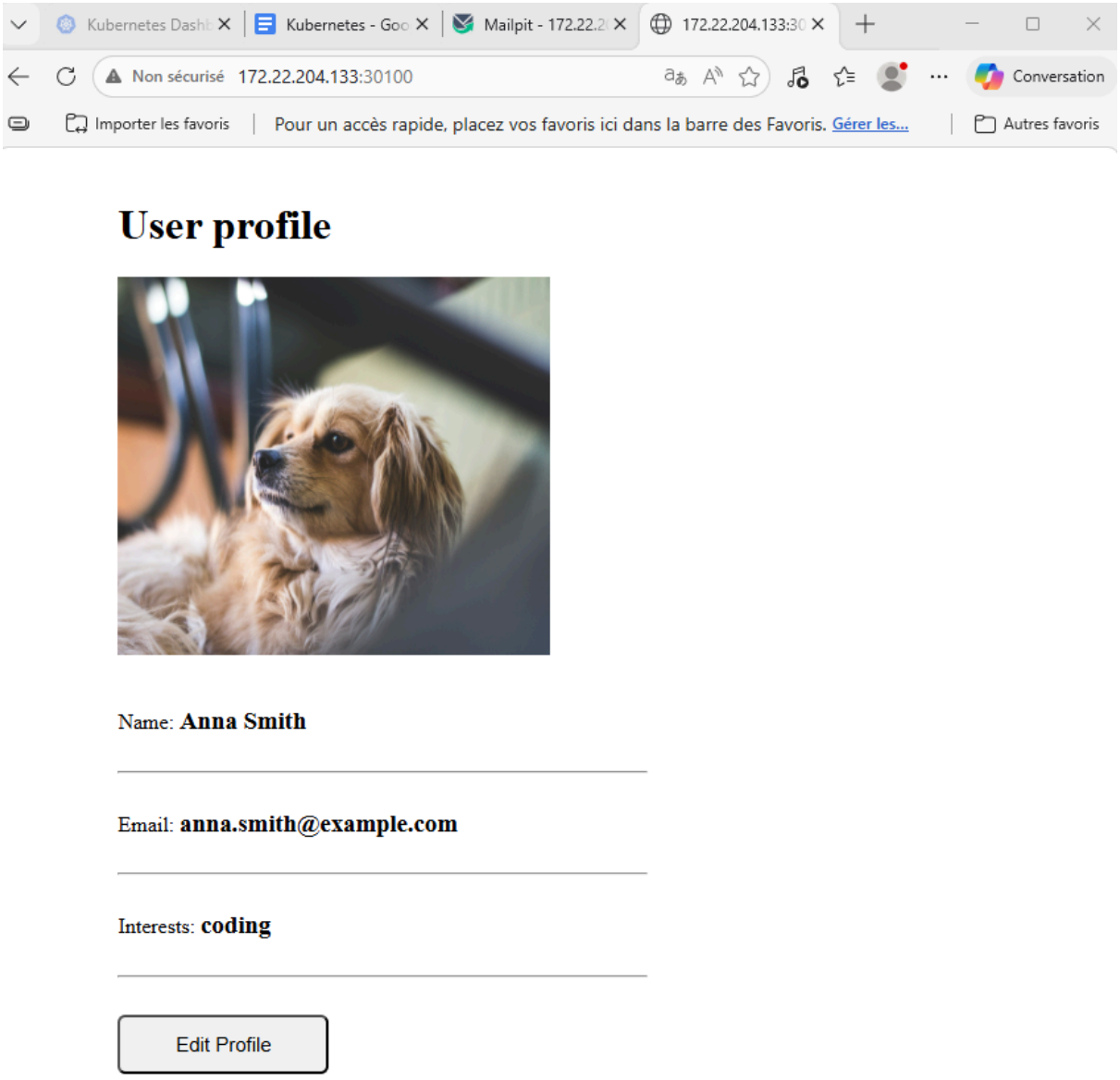
Les commandes **kubectl get node** et **kubectl get node -o wide** permettent de vérifier l'état du nœud ainsi que des informations détaillées comme l'adresse IP.



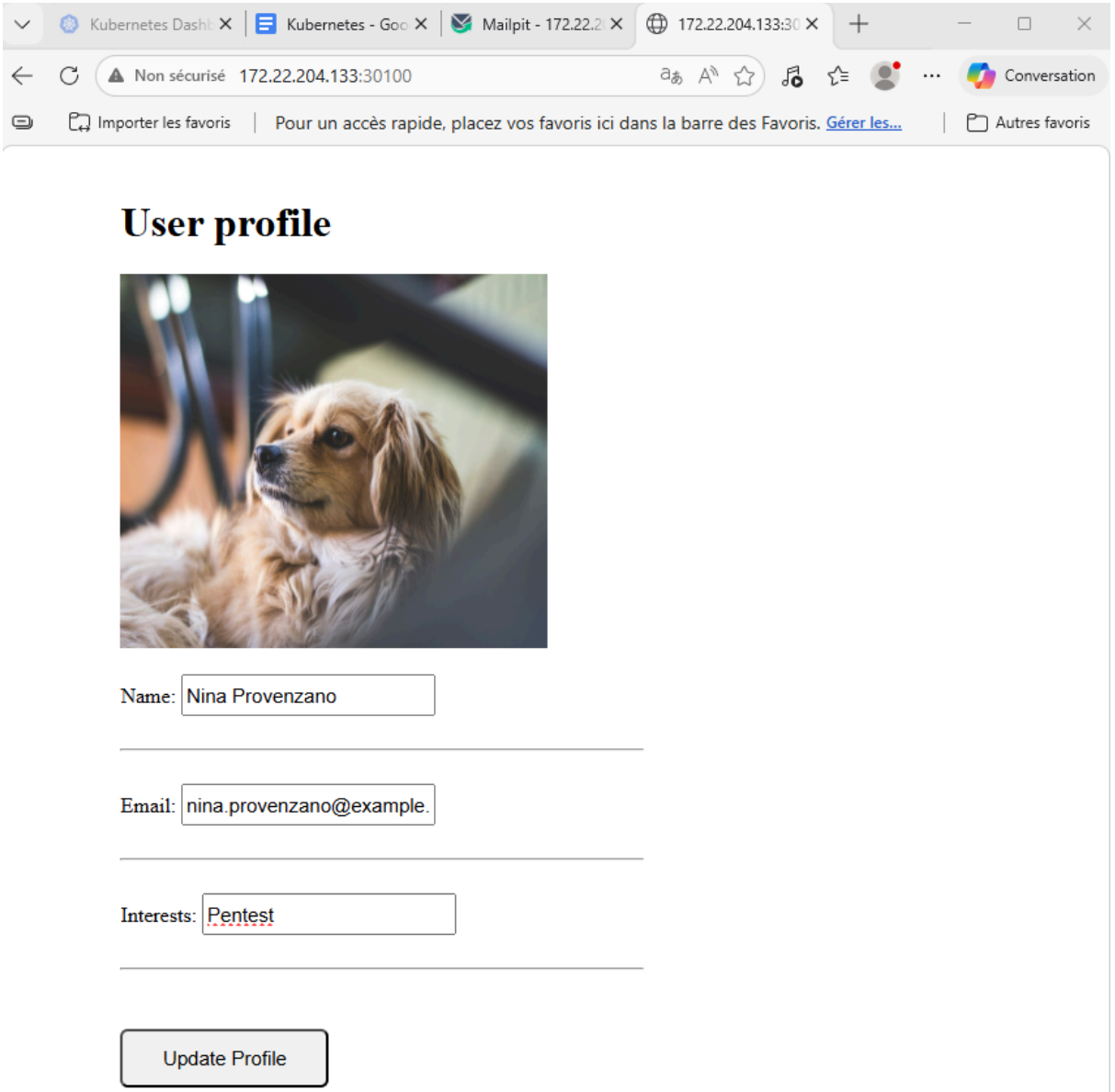
Les commandes **kubectl get svc -o wide** et **kubectl get pod -o wide** permettent de visualiser les services et les pods avec des détails supplémentaires comme les adresses IP et les nœuds associés.



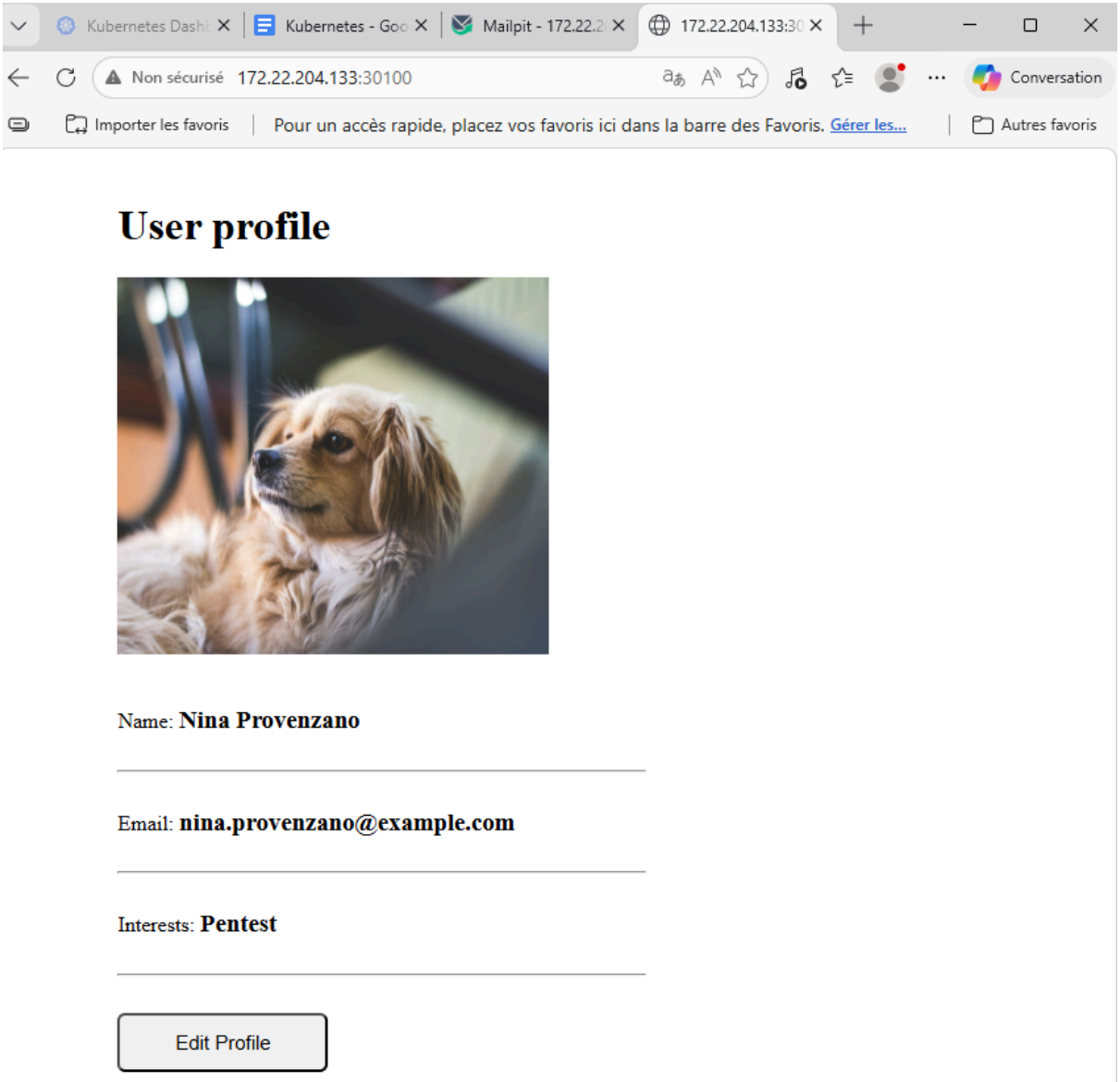
L’accès à l’adresse IP et au port du service permet d’afficher l’interface de l’application web.
> Edit Profile



La modification du profil permet de vérifier que les données de l'application web peuvent être mises à jour.



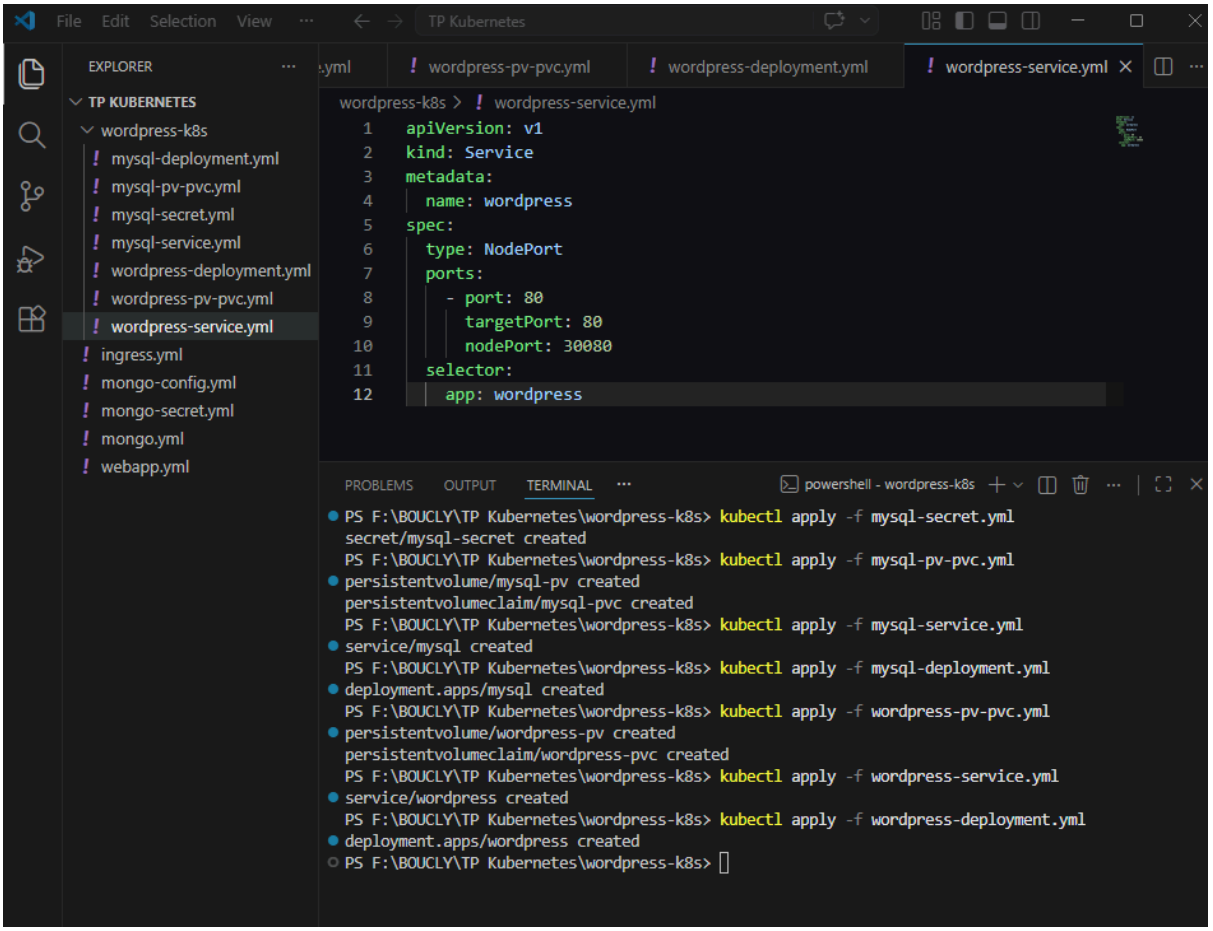
L’affichage du profil mis à jour permet de vérifier que les modifications ont bien été enregistrées dans l’application.



6. Application

Les commandes **mkdir** et **cd** permettent de créer un répertoire de travail et de s'y positionner pour stocker les fichiers du projet Kubernetes.

La commande **kubectl apply -f** permet de créer automatiquement les différents composants de l'application WordPress et MySQL (PVC, services et déploiements).



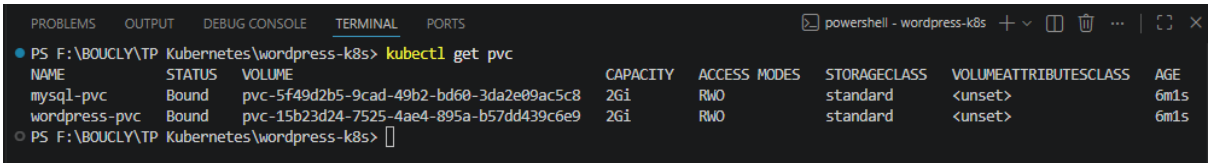
La commande **kubectl get all** permet de vérifier que toutes les ressources de l'application WordPress et MySQL sont correctement déployées.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
PS F:\BOUCLY\TP Kubernetes\wordpress-k8s> kubectl get all
NAME                                    READY   STATUS    RESTARTS   AGE
pod/mailpit-7b96b48c5d-hhn7j            1/1     Running   0          100m
pod/mailpit-7b96b48c5d-sphd4            1/1     Running   0          87m
pod/mongo-deployment-744864fdd7-n5djl   1/1     Running   0          40m
pod/mysql-66f86944b7-hd5b6              1/1     Running   0          4m22s
pod/webapp-deployment-5766fd95c7-9vhdv  1/1     Running   0          22m
pod/wordpress-59fcc67747-vll9f          1/1     Running   0          4m20s

NAME                      TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)              AGE
service/kubernetes        ClusterIP   10.96.0.1        <none>        443/TCP              127m
service/mailpit           ClusterIP   10.96.177.29     <none>        1025/TCP,8025/TCP    90m
service/mongo-service     ClusterIP   10.100.26.88     <none>        27017/TCP            40m
service/mysql             ClusterIP   10.107.152.182   <none>        3306/TCP             4m22s
service/webapp-service    NodePort    10.107.105.109   <none>        3000:30100/TCP       40m
service/wordpress         NodePort    10.108.202.173   <none>        80:30080/TCP         4m22s

NAME                                  READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/mailpit               2/2     2            2           100m
deployment.apps/mongo-deployment      1/1     1            1           40m
deployment.apps/mysql                 1/1     1            1           4m22s
deployment.apps/webapp-deployment     1/1     1            1           40m
deployment.apps/wordpress             1/1     1            1           4m20s

NAME                                             DESIRED   CURRENT   READY   AGE
replicaset.apps/mailpit-7b96b48c5d               2         2         2       100m
replicaset.apps/mongo-deployment-744864fdd7      1         1         1       40m
replicaset.apps/mysql-66f86944b7                 1         1         1       4m22s
replicaset.apps/webapp-deployment-5766fd95c7     1         1         1       22m
replicaset.apps/webapp-deployment-6d88954d4c     0         0         0       40m
replicaset.apps/wordpress-59fcc67747             1         1         1       4m20s
PS F:\BOUCLY\TP Kubernetes\wordpress-k8s>
```

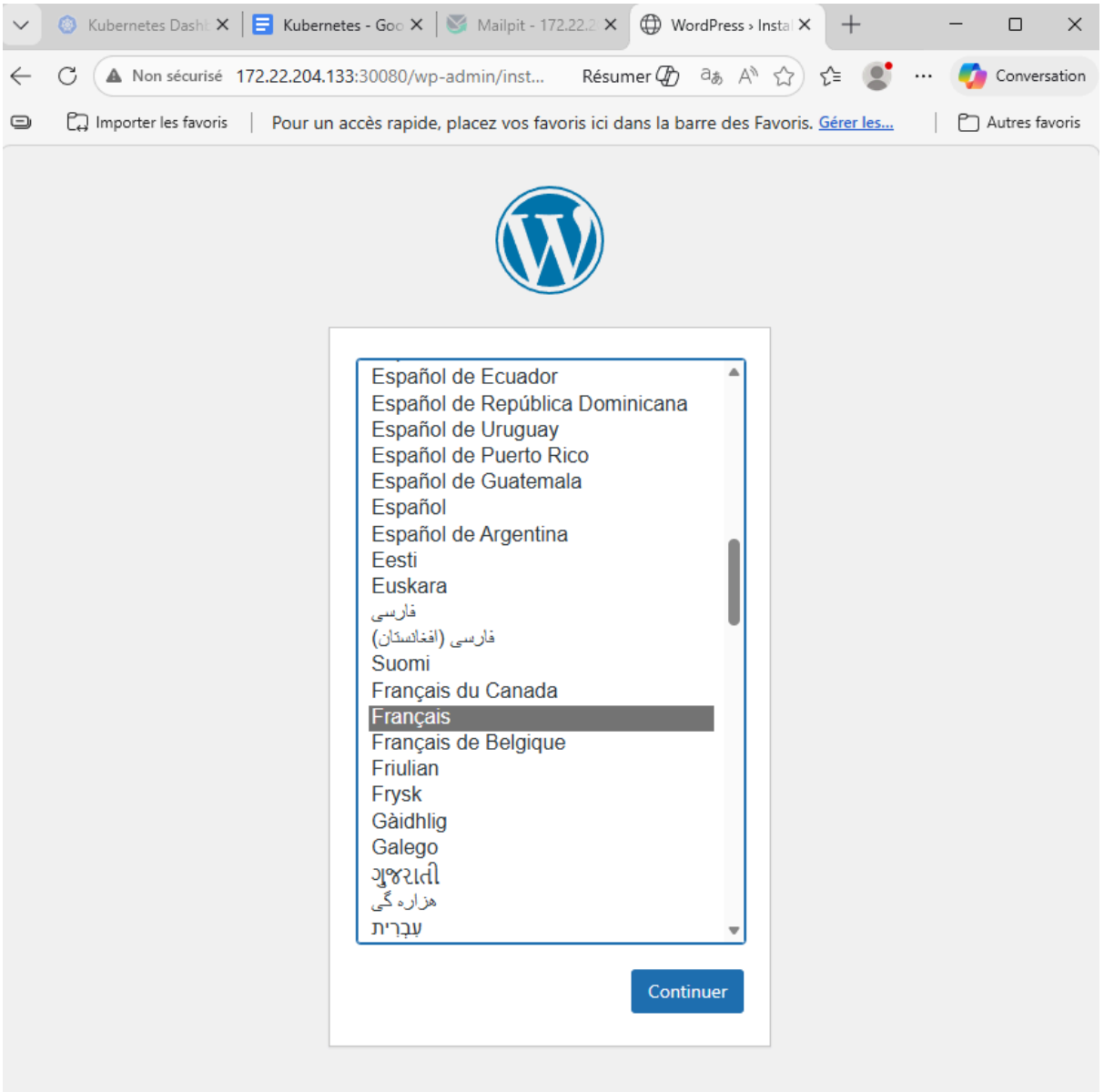
La commande **kubectl get pvc** permet d'afficher les volumes persistants utilisés par l'application.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS                    powershell - wordpress-k8s
PS F:\BOUCLY\TP Kubernetes\wordpress-k8s> kubectl get pv
NAME                                       CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM                   STORAGECLASS   VOLUMEATTRIBUTESCLASS   REASON   AGE
mysql-pv                                   2Gi        RWO            Retain           Available                                          <unset>                          4m43s
pvc-15b23d24-7525-4ae4-895a-b57dd439c6e9   2Gi        RWO            Delete           Bound       default/wordpress-pvc   standard       <unset>                          4m43s
pvc-5f49d2b5-9cad-49b2-bd60-3da2e09ac5c8   2Gi        RWO            Delete           Bound       default/mysql-pvc       standard       <unset>                          4m43s
wordpress-pv                               2Gi        RWO            Retain           Available                                          <unset>                          4m43s
PS F:\BOUCLY\TP Kubernetes\wordpress-k8s>
```

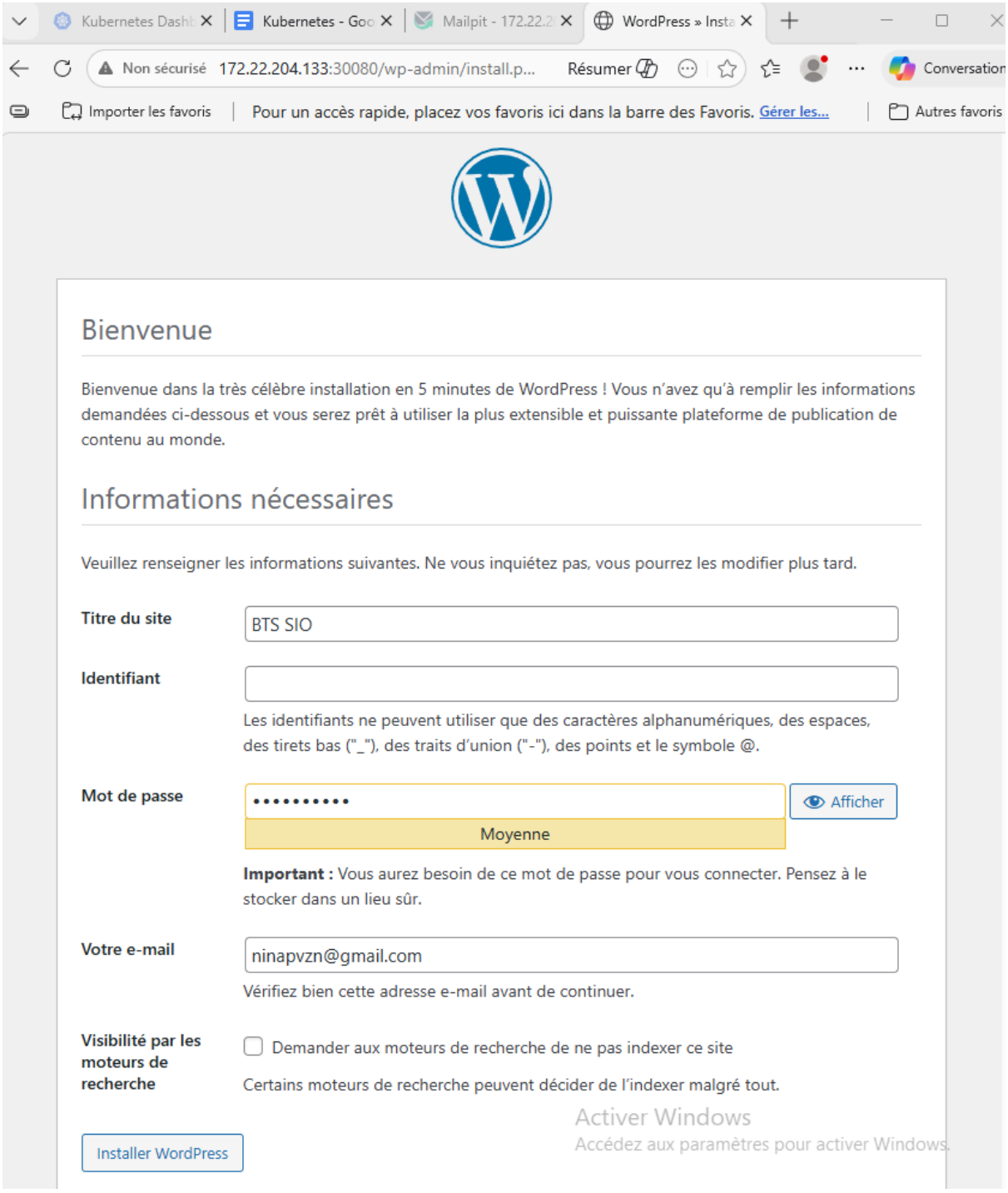
La commande **kubectl get pv** permet de vérifier les PersistentVolumes et leur état afin de s'assurer le fonctionnement du stockage.



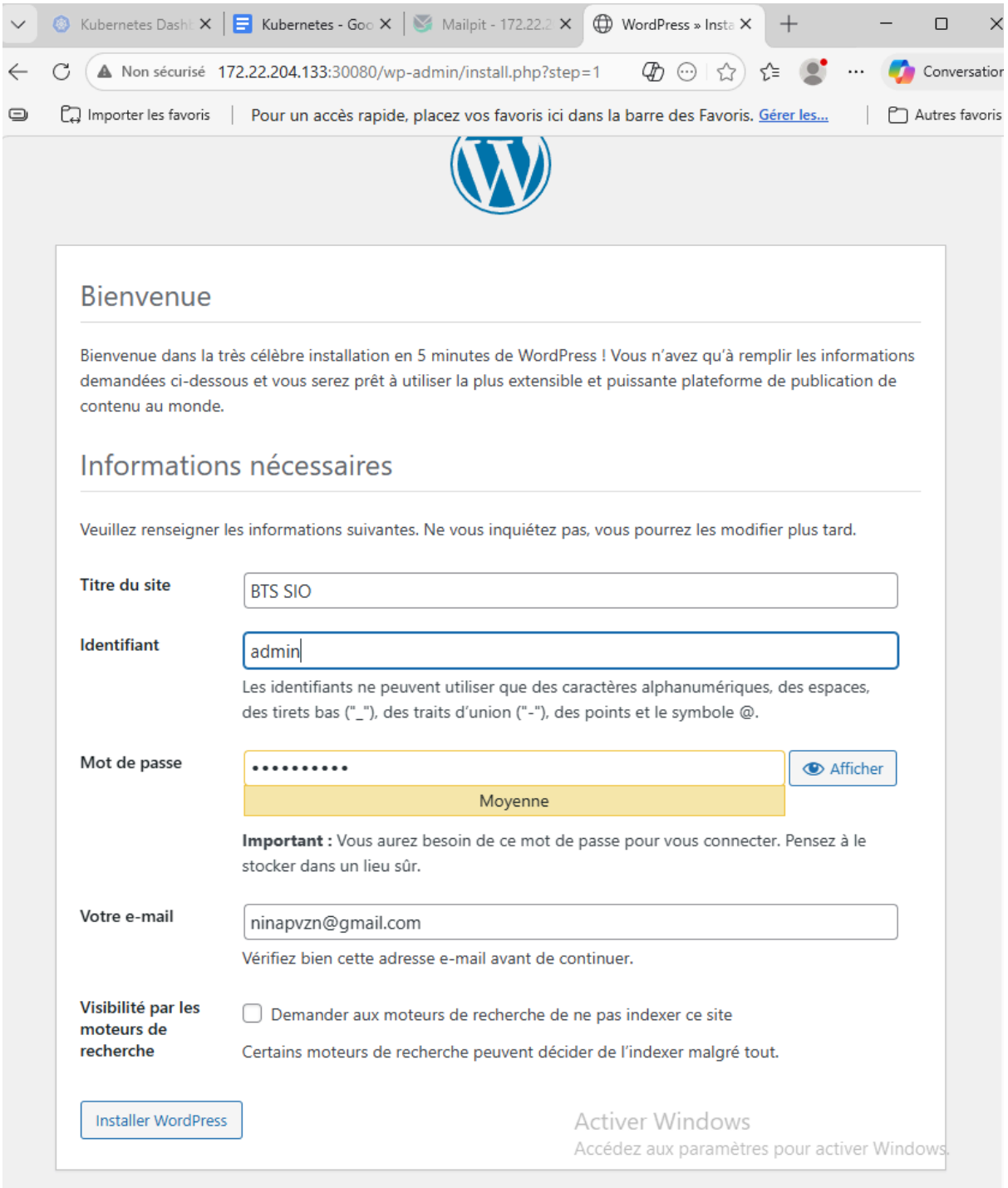
L'écran d'installation de WordPress permet de choisir la langue de l'application avant la configuration.



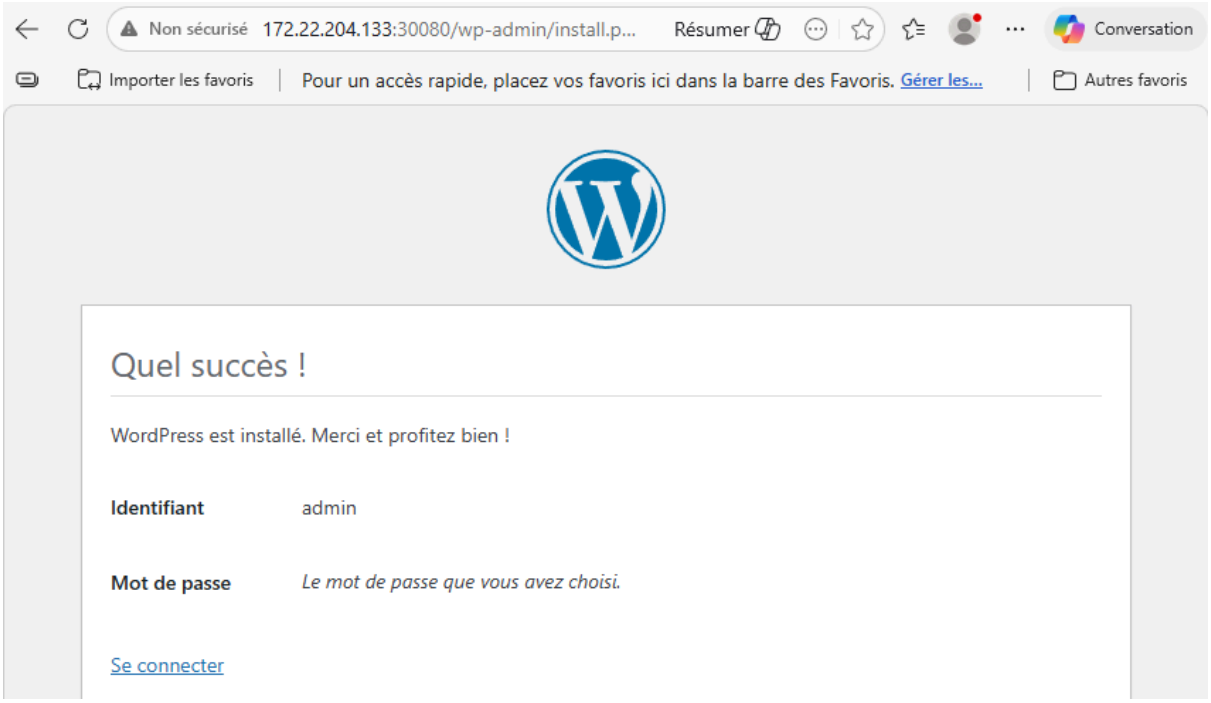
L'interface d'installation de WordPress permet de renseigner les informations nécessaires à la création du site.



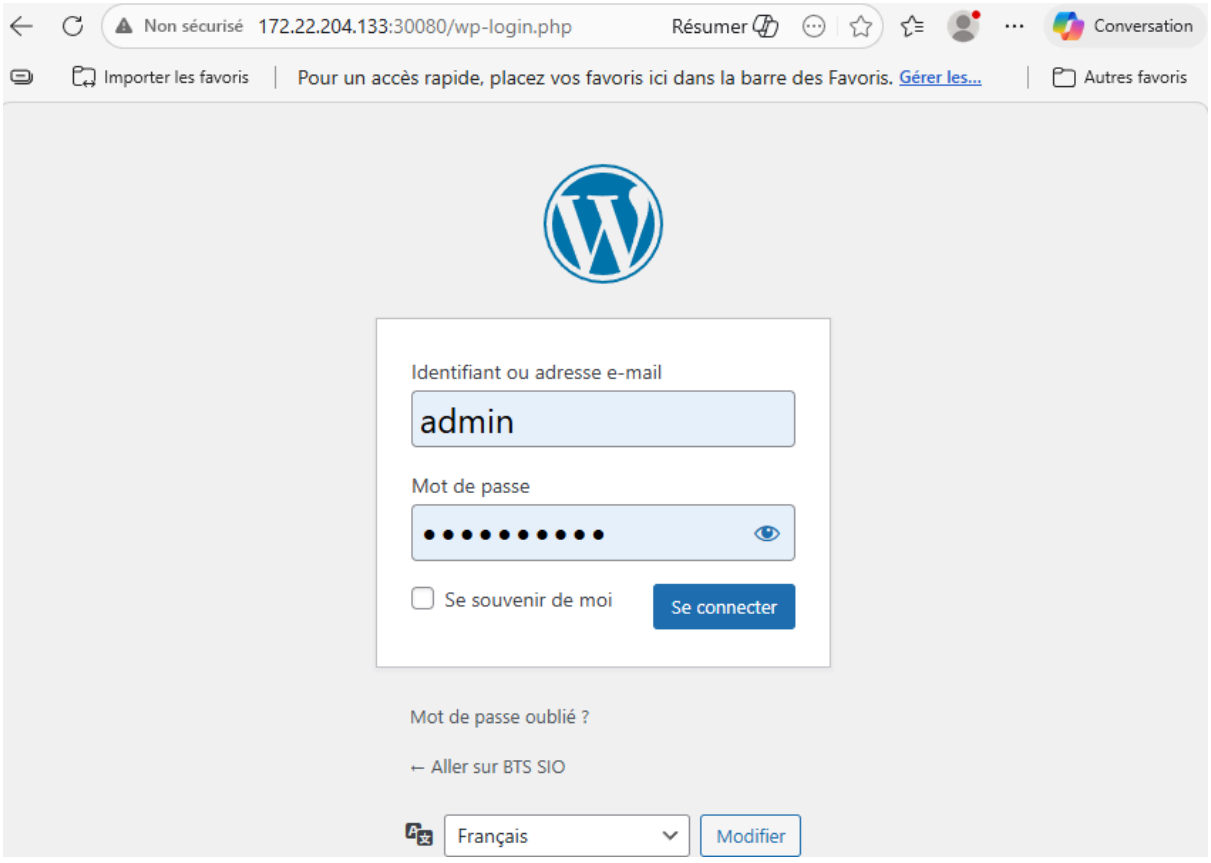
Saisie des informations pour configurer les paramètres administrateur du site WordPress.



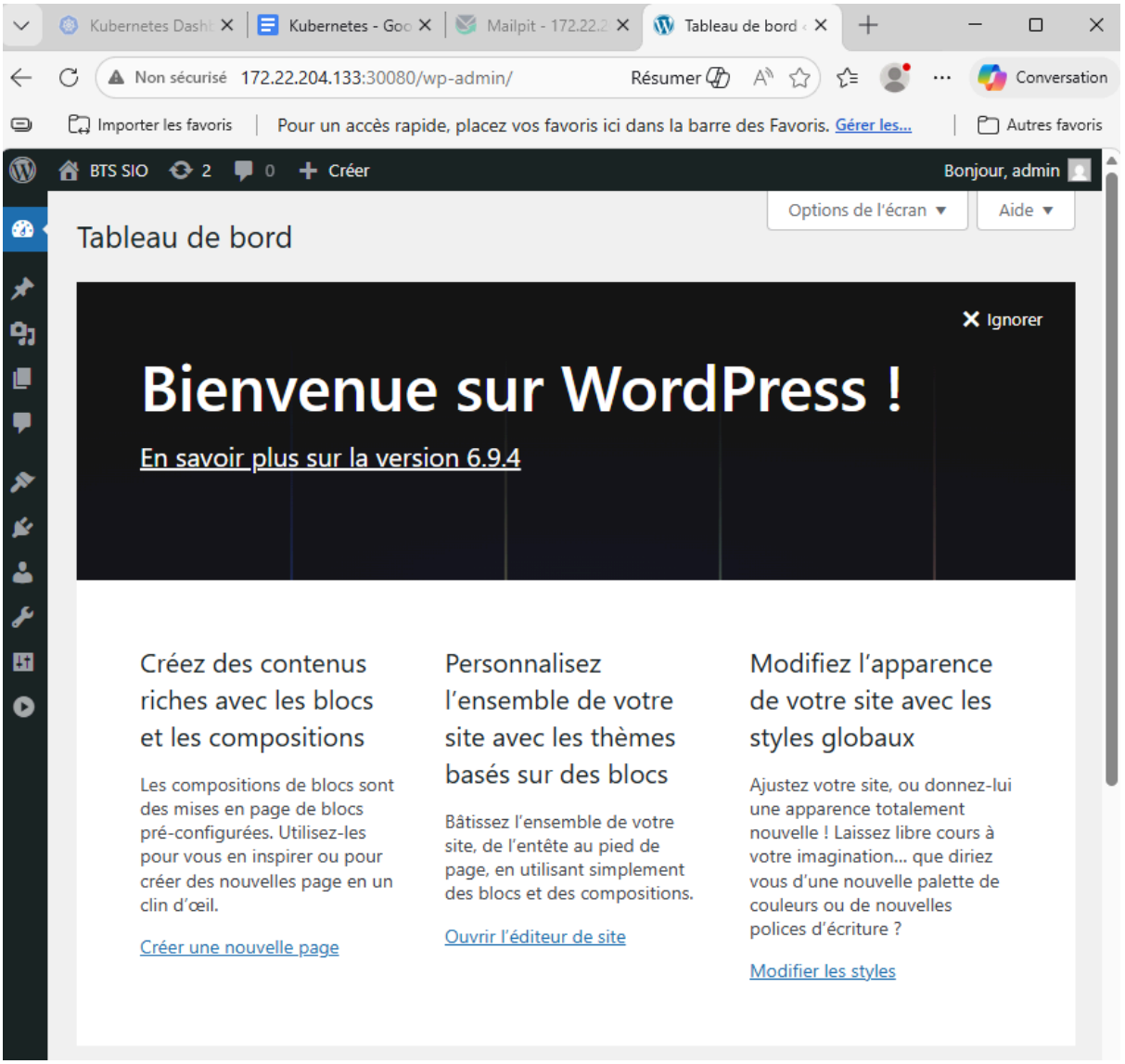
Confirmation que l'installation de WordPress a été réalisée avec succès.



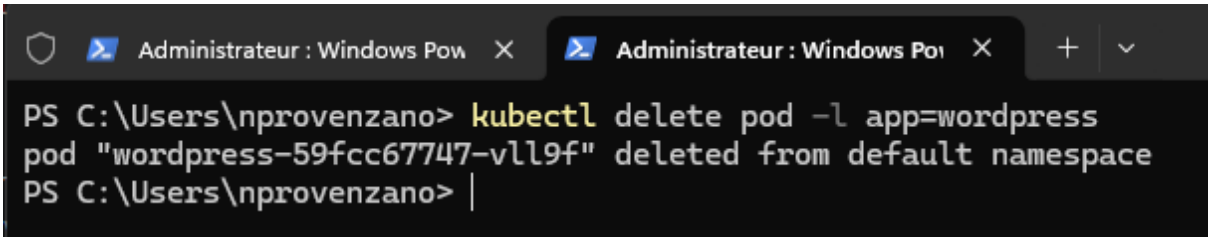
Page de connexion de l'administrateur :



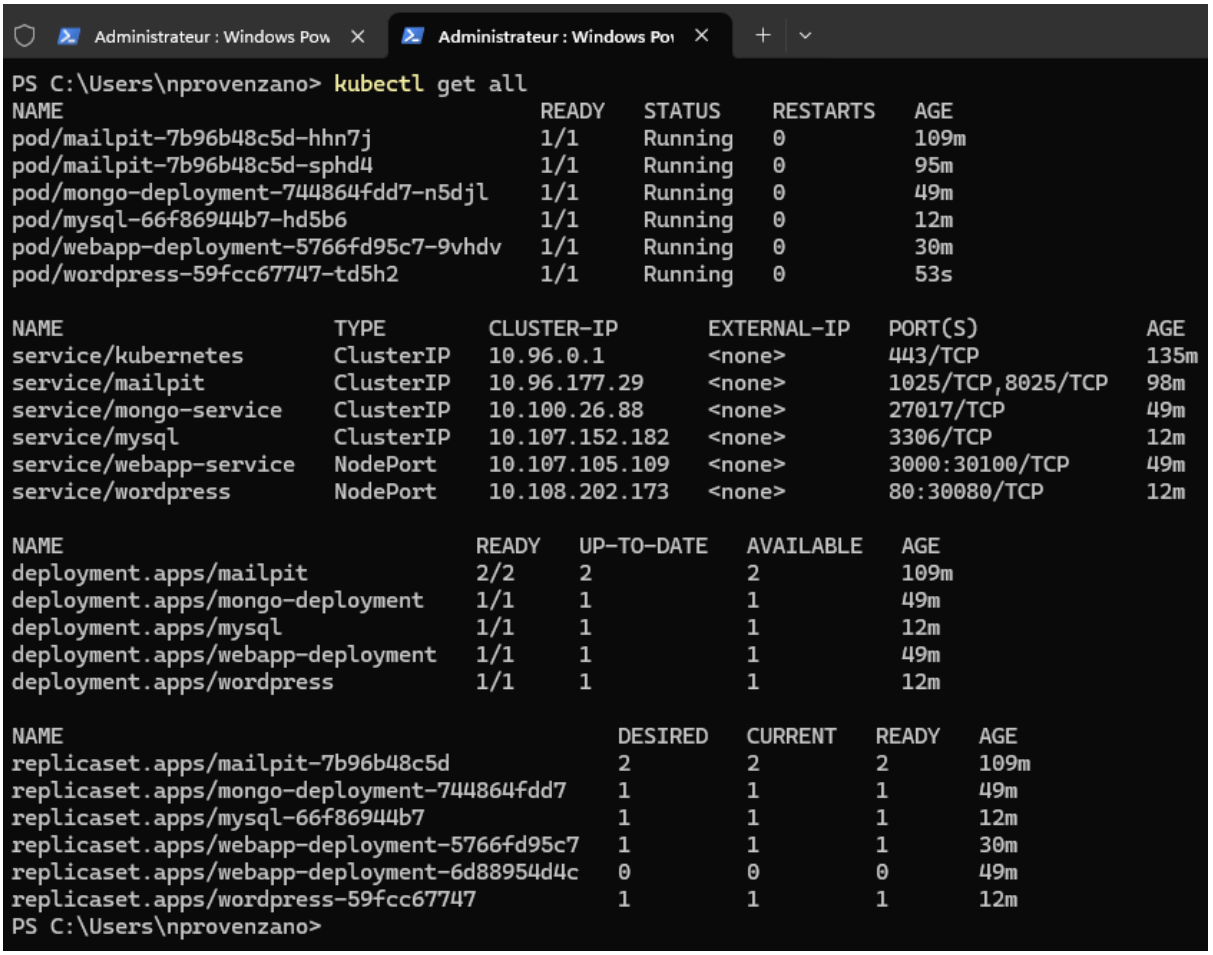
L'accès au tableau de bord WordPress permet d'administrer et de gérer le contenu du site.



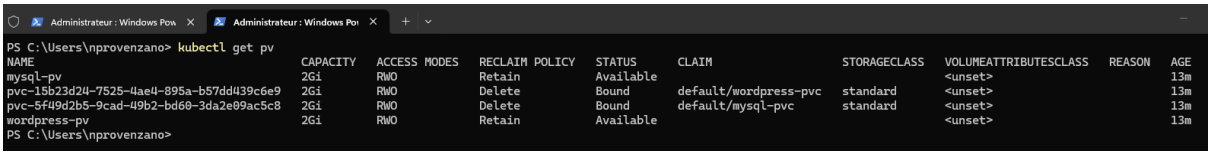
La commande **kubectl delete pod -l app=wordpress** permet de supprimer les pods WordPress.



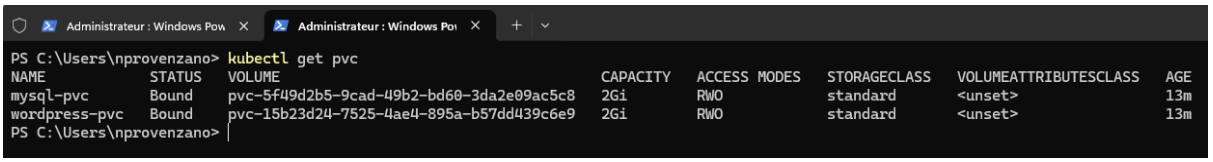
La commande **kubectl get all** permet de confirmer que les pods WordPress ont été recréés automatiquement par le déploiement, assurant la continuité du service.



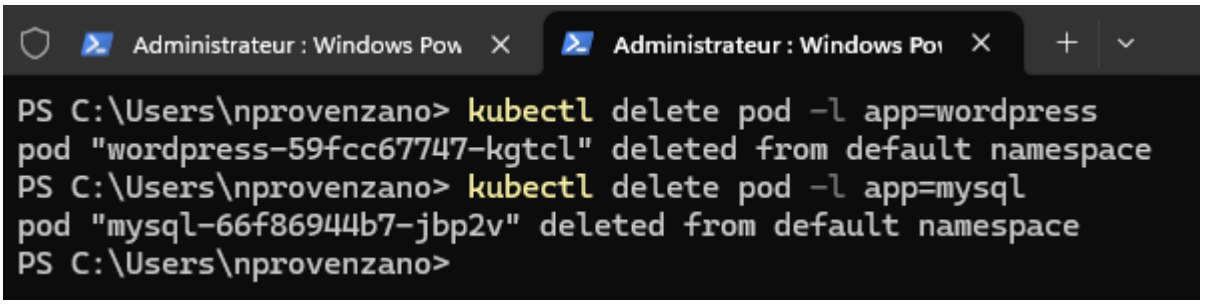
La commande **kubectl get pv** permet de confirmer que les données sont conservées grâce aux volumes persistants malgré la suppression et la recréation des pods.



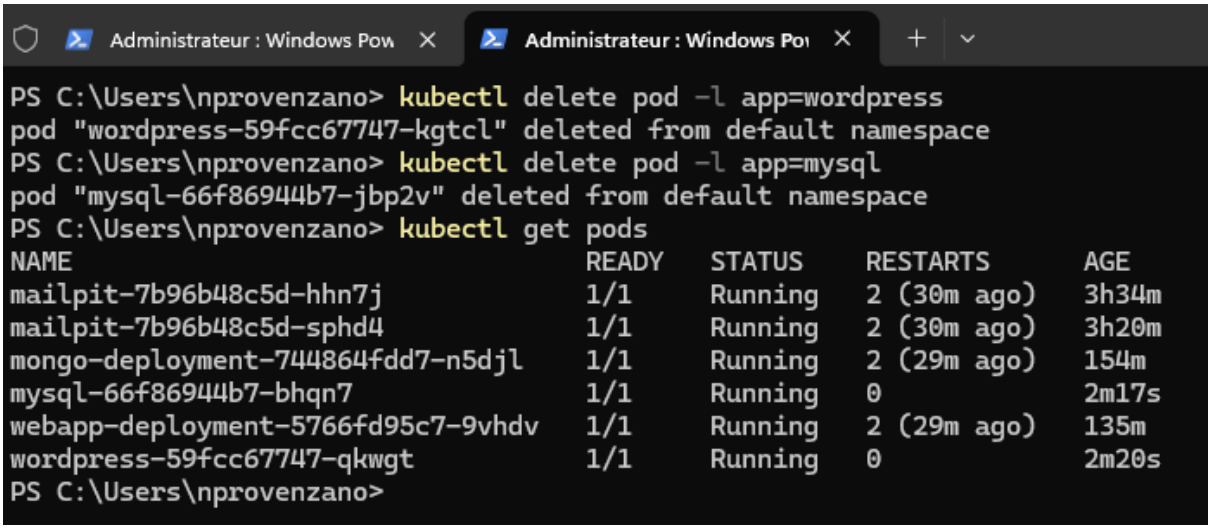
La commande **kubectl get pvc** permet de confirmer que les PersistentVolumeClaims sont toujours actifs et associés aux volumes de stockage.



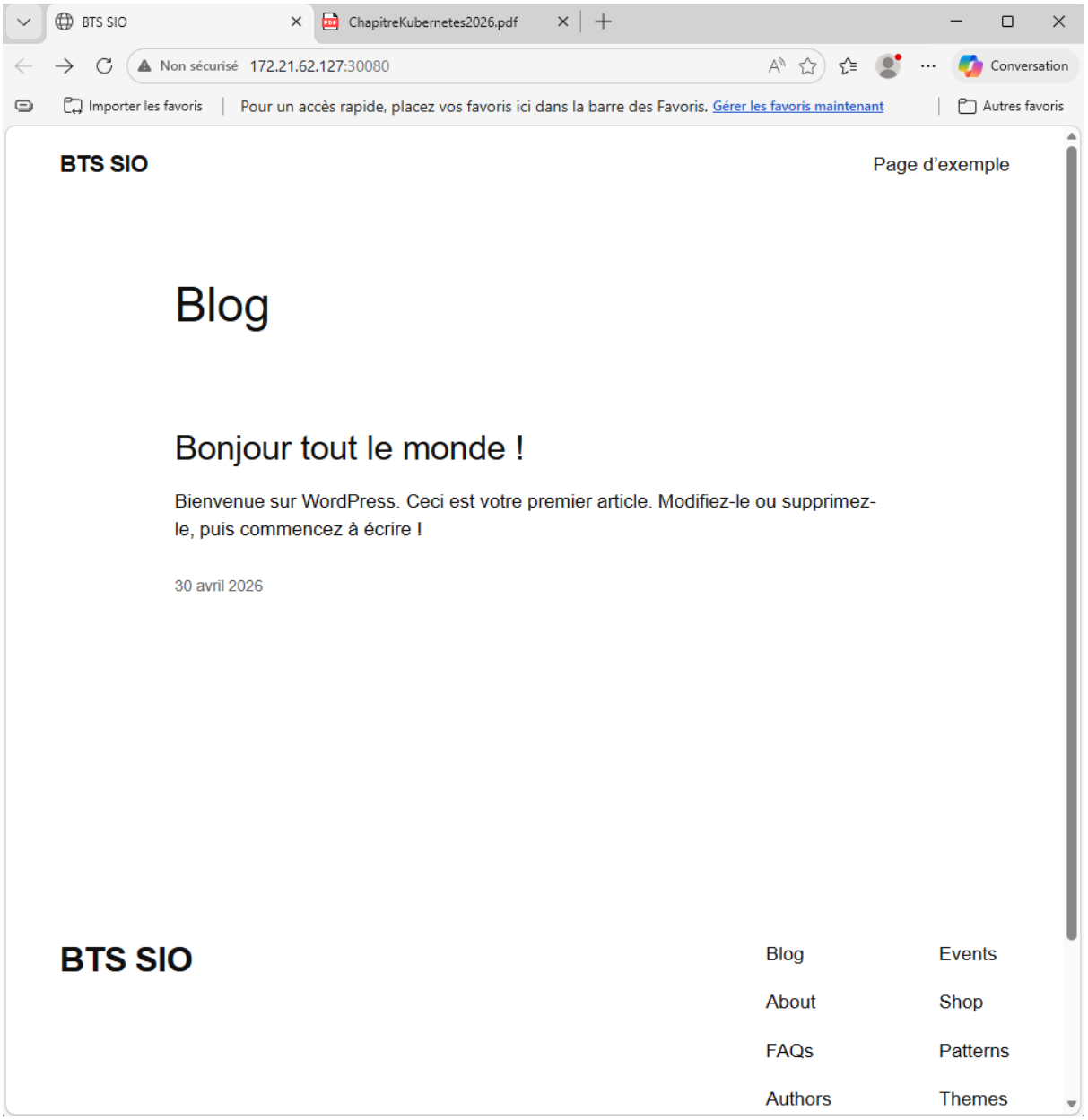
Les commandes **kubectl delete pod -l app=wordpress** et **kubectl delete pod -l app=mysql** permettent de supprimer les pods pour tester leur recréation automatique par les déploiements.



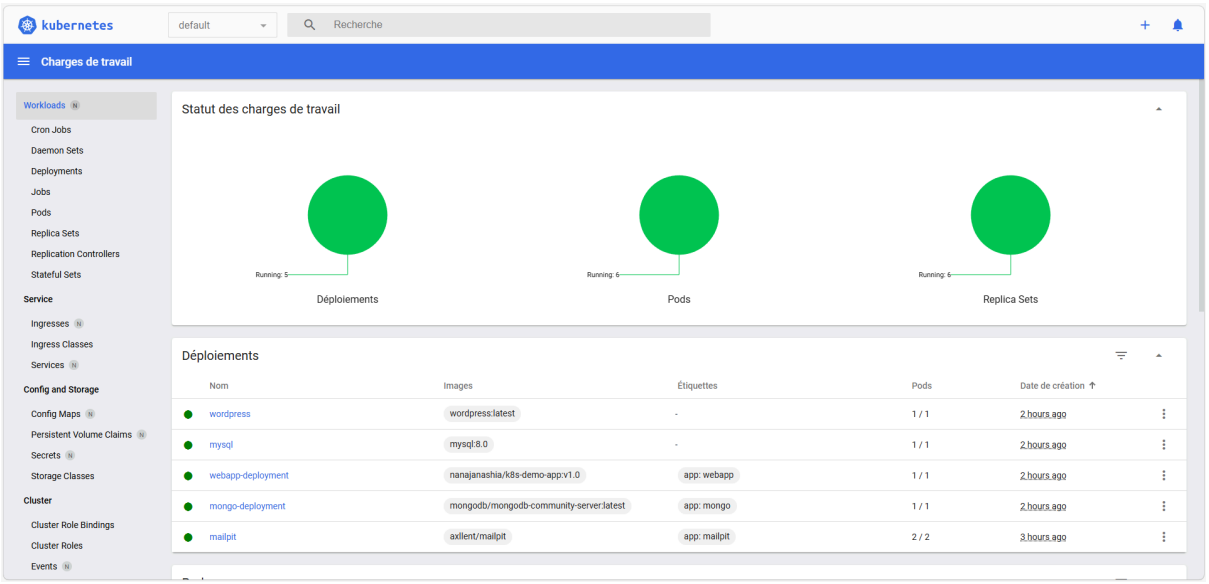
La commande **kubectl get pods** permet de confirmer que les pods WordPress et MySQL sont de nouveau en état Running grâce aux déploiements.



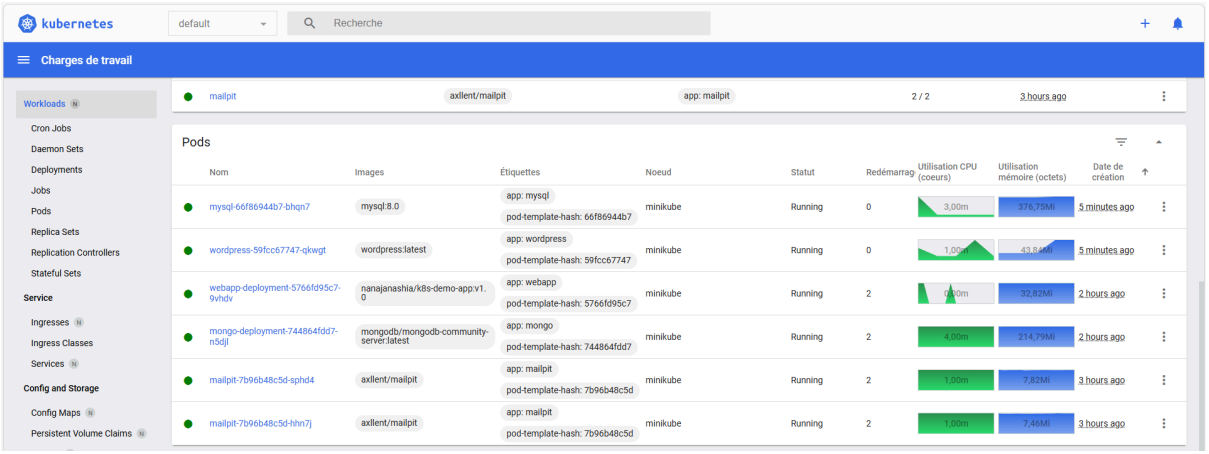
Vérification que le site WordPress est accessible depuis le navigateur et que l'application fonctionne avec l'affichage du contenu.



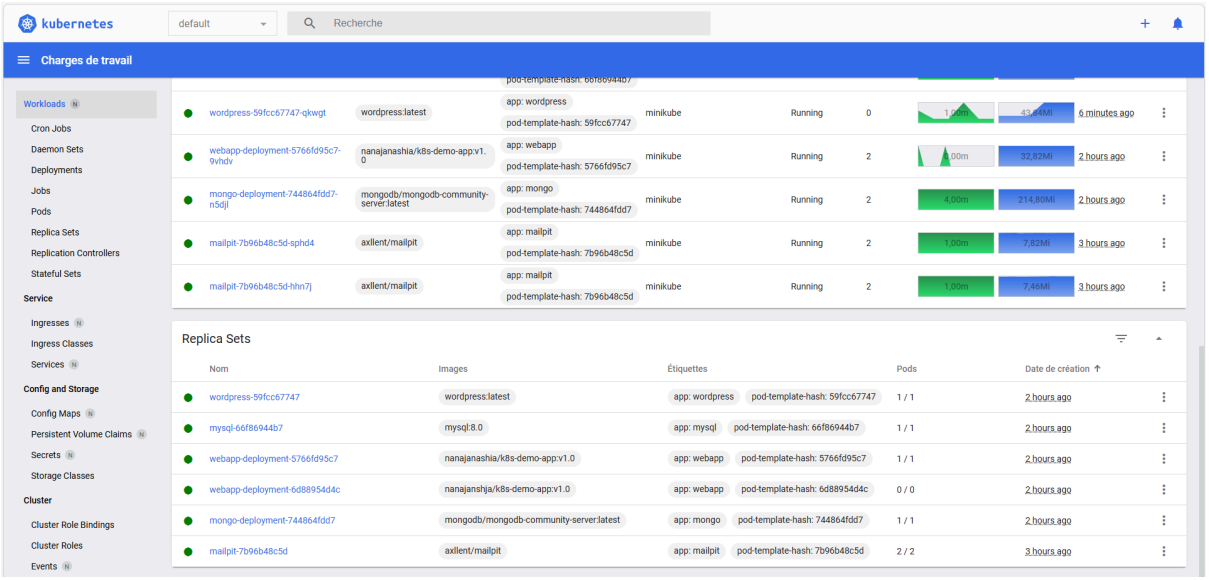
Le tableau de bord Kubernetes permet de vérifier que les déploiements, les pods et les ReplicaSets sont en état Running.



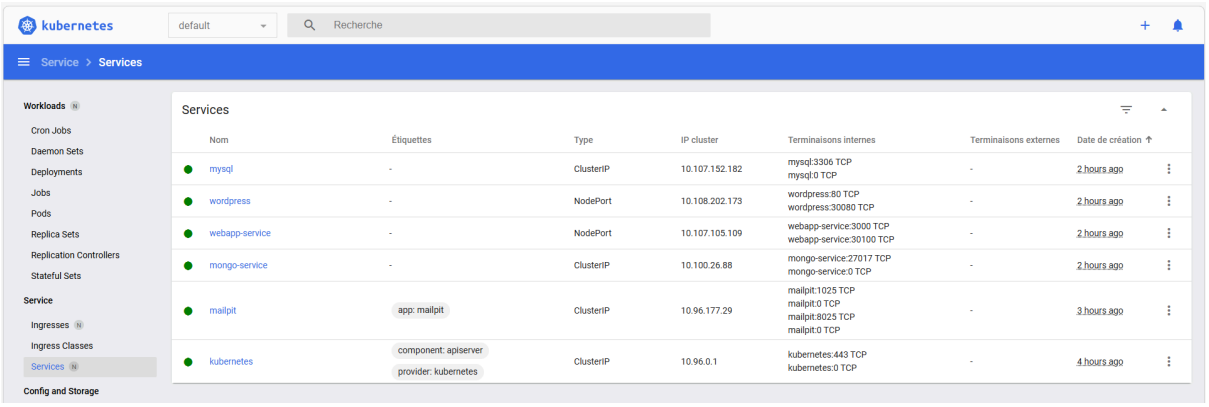
Affichage des pods et de leurs états :



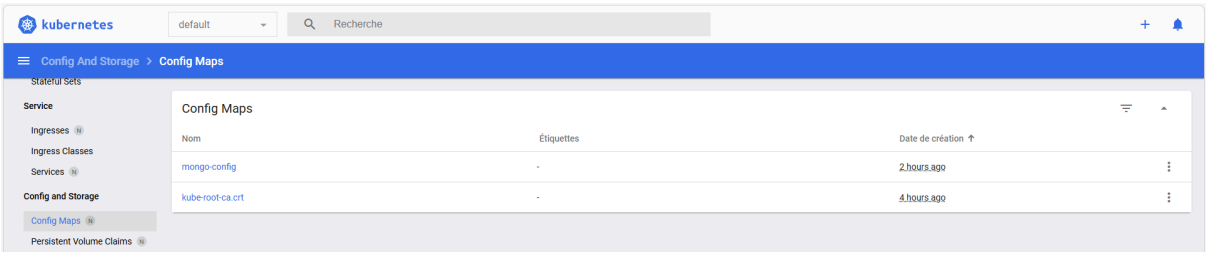
Affichage des Replicats Sets et de leurs états :



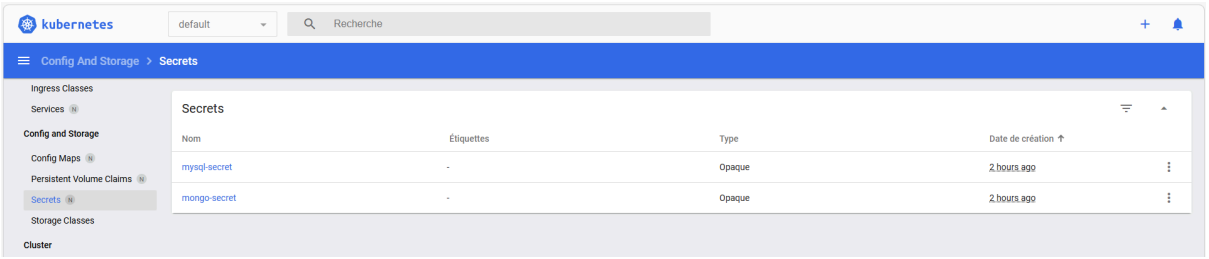
Affichage des Services :



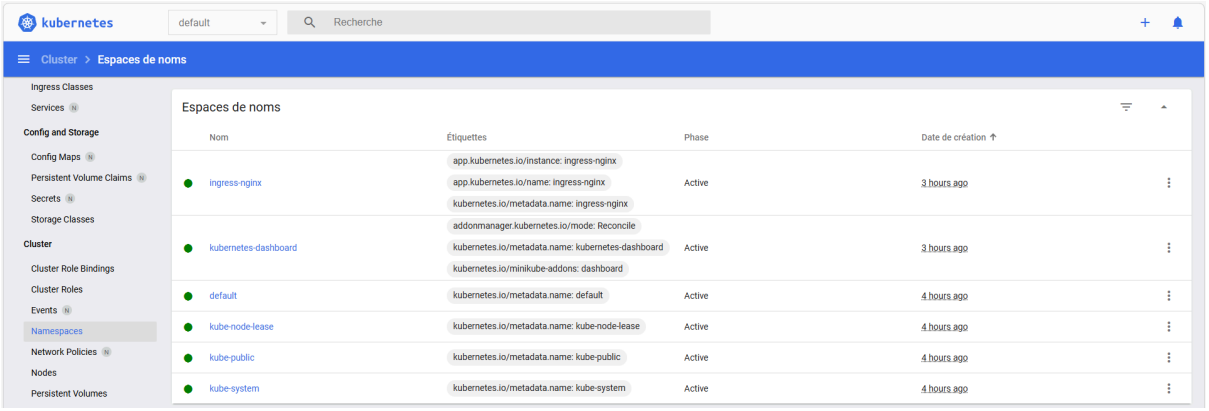
Affichage des ConfigMaps :



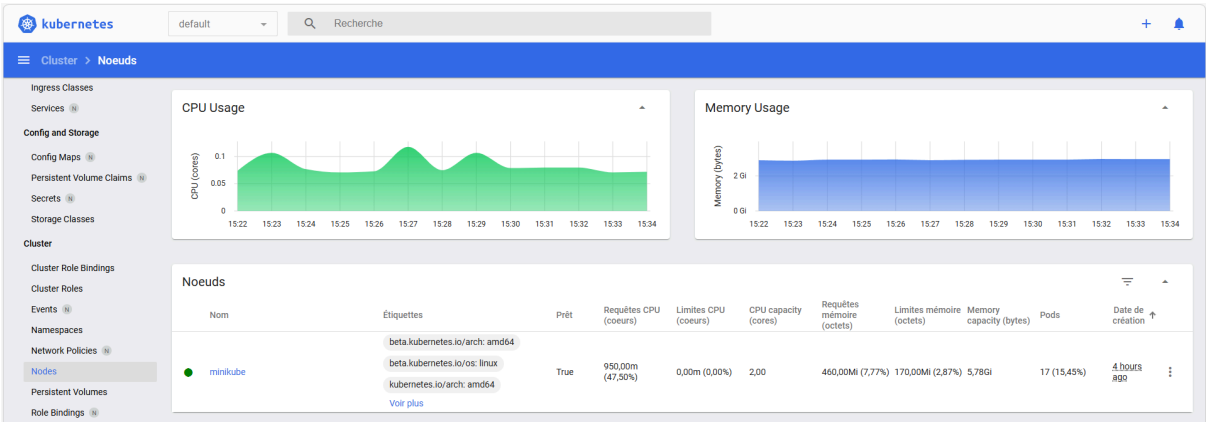
Affichage des Secrets :



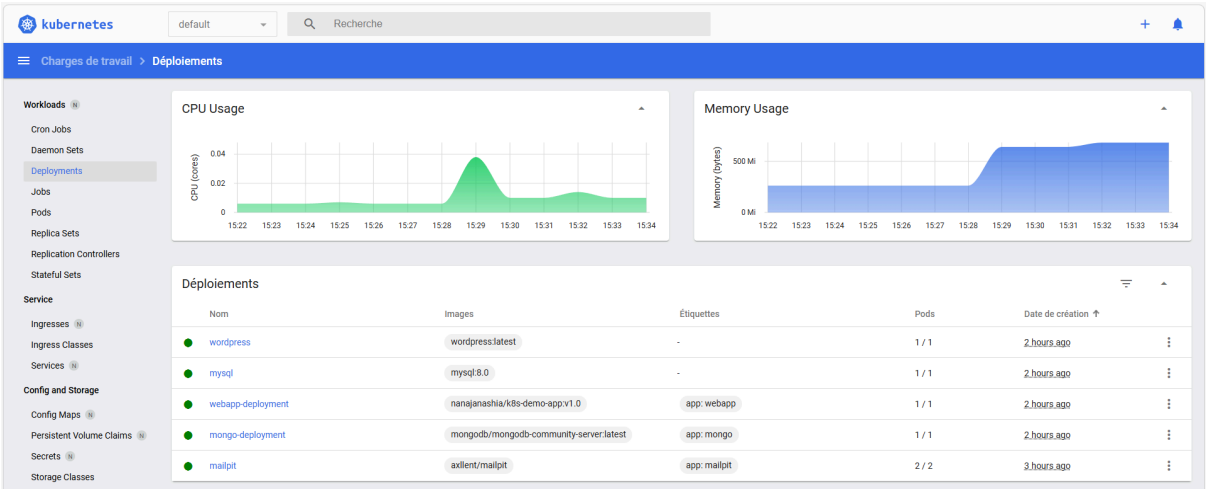
Affichage des Namespaces :



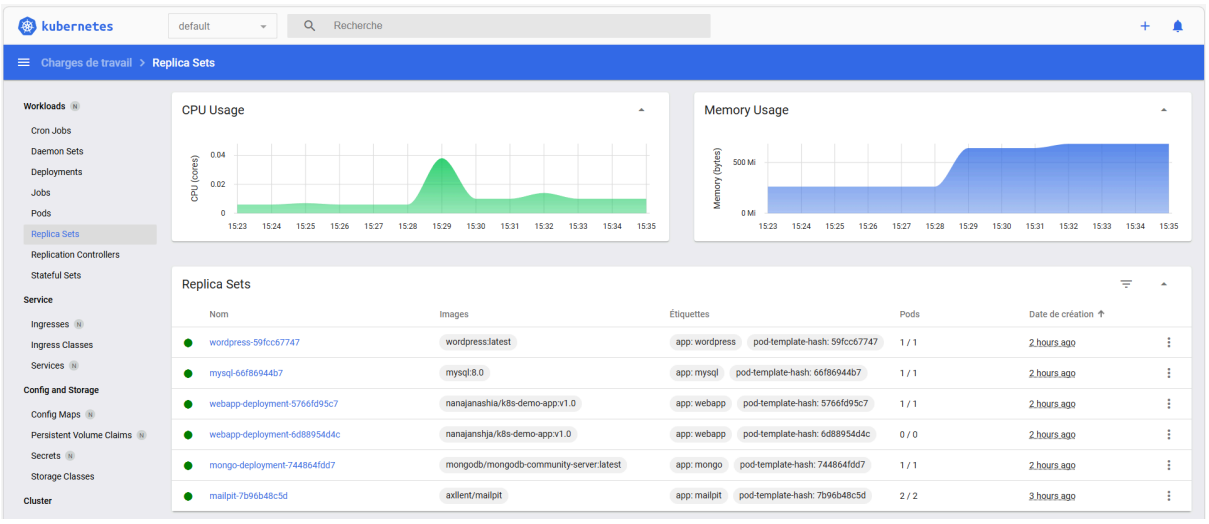
Affichage du Node (noeud) en cours d'exécution :



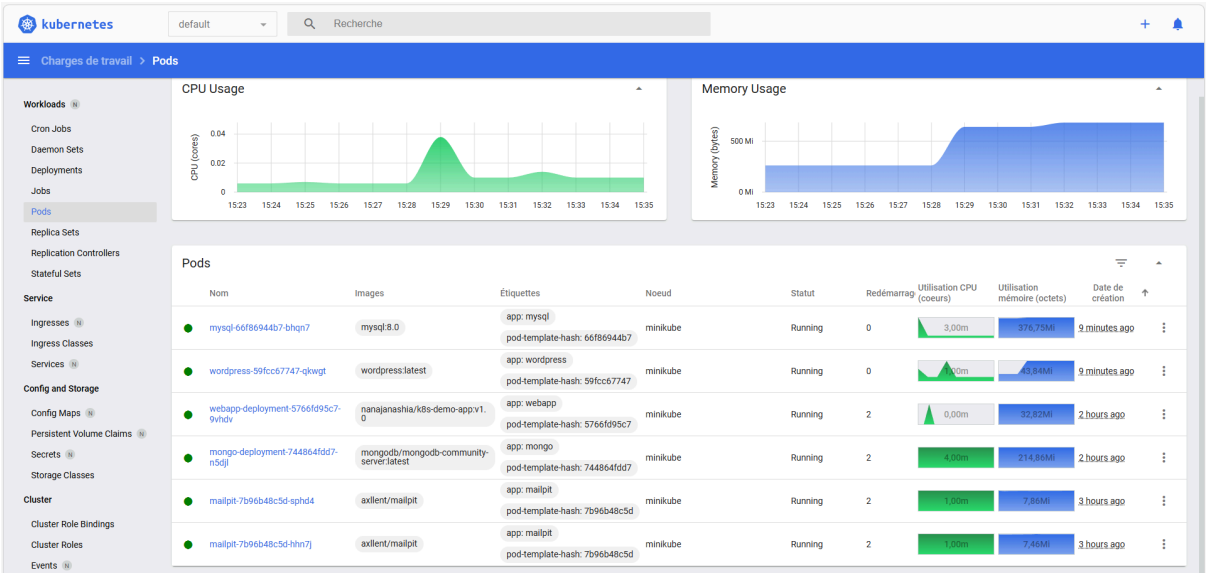
Affichage des Déploiements en cours d'exécution :



Affichage des ReplicaSets en cours d'exécution :



Affichage des Pods en cours d'exécution :

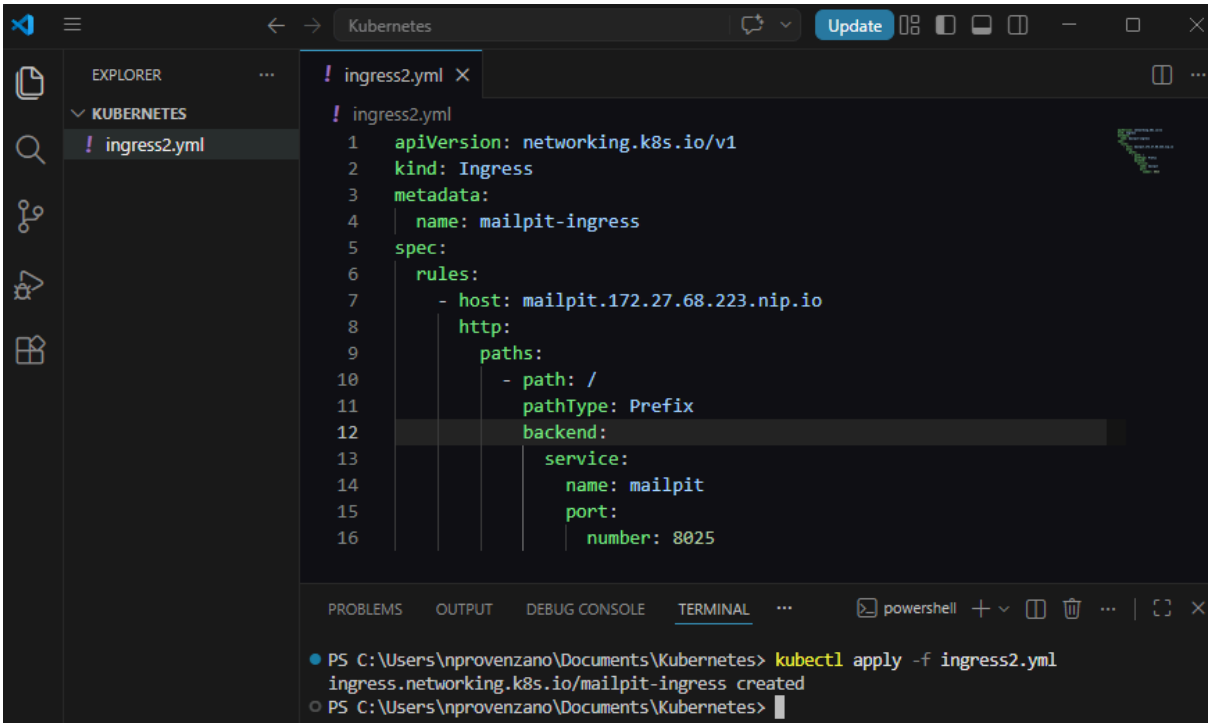


7. Suite Ingress (Mailpit) : hôtes virtuels et nom de domaine nip.io

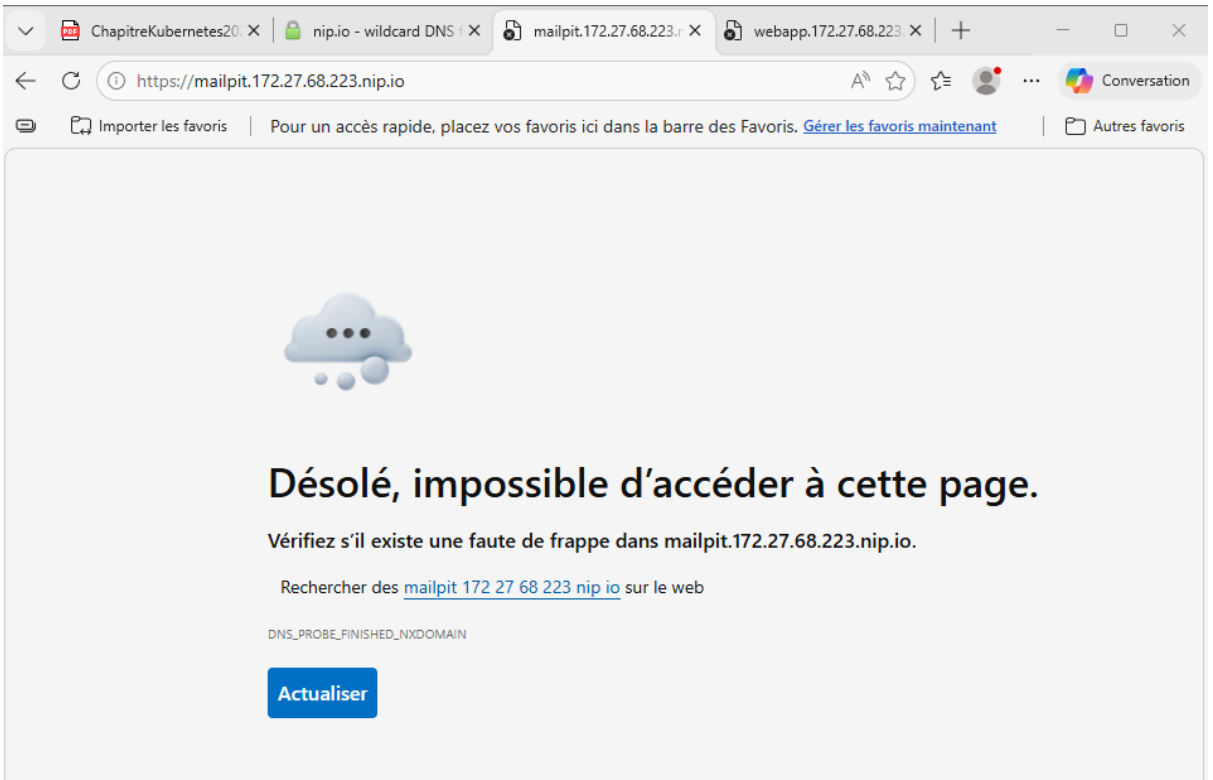
La commande

```
PS C:\Users\nprovenzano> kubectl get ingress
NAME           CLASS   HOSTS   ADDRESS   PORTS   AGE
test-ingress   nginx   *                 80      6d21h
PS C:\Users\nprovenzano> minikube ip
172.28.25.20
PS C:\Users\nprovenzano>
```

Fichier ingress2.yml

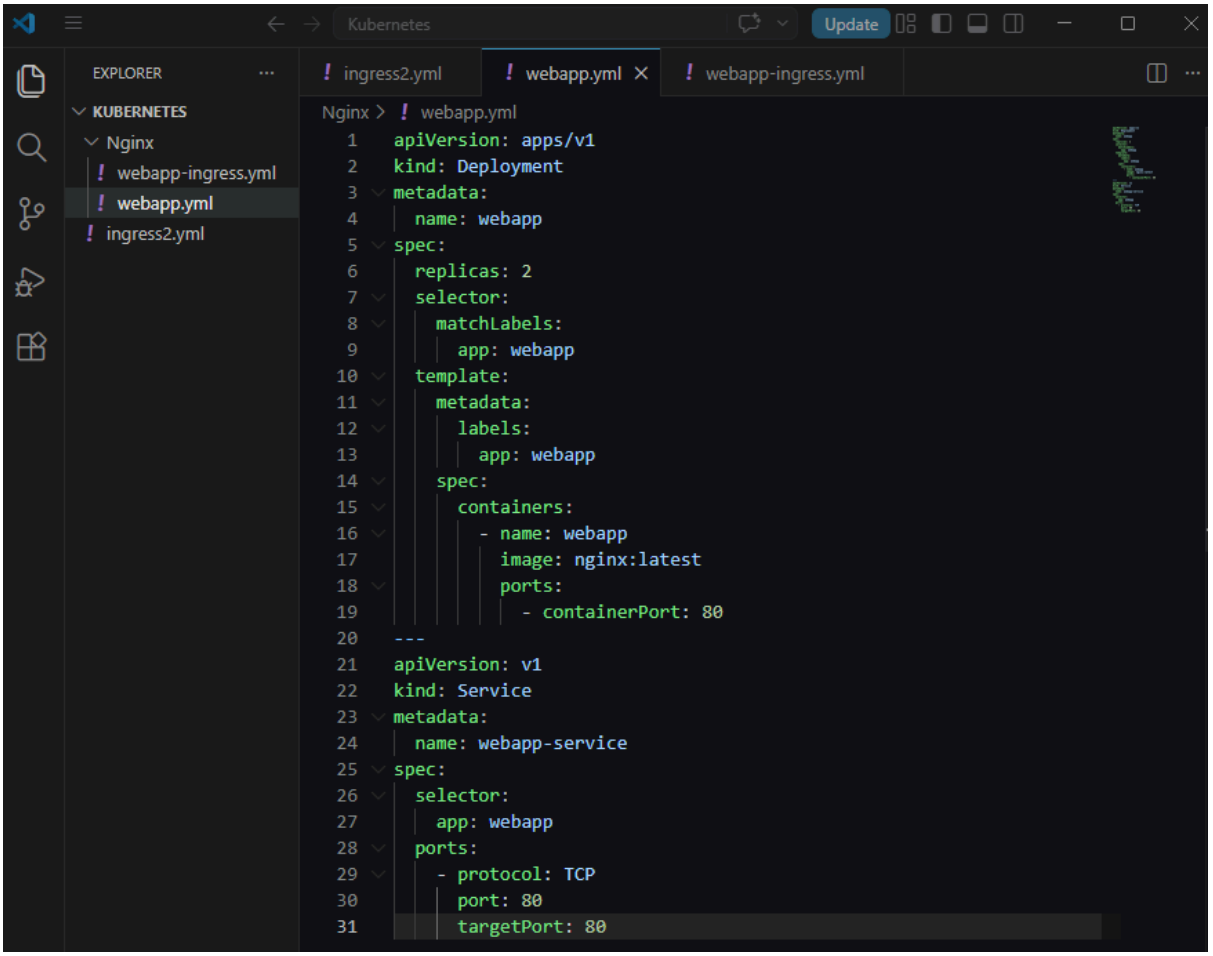


Le nom de domaine .nip.io étant bloqué par le serveur du lycée, la redirection ne s'est pas pu se faire.

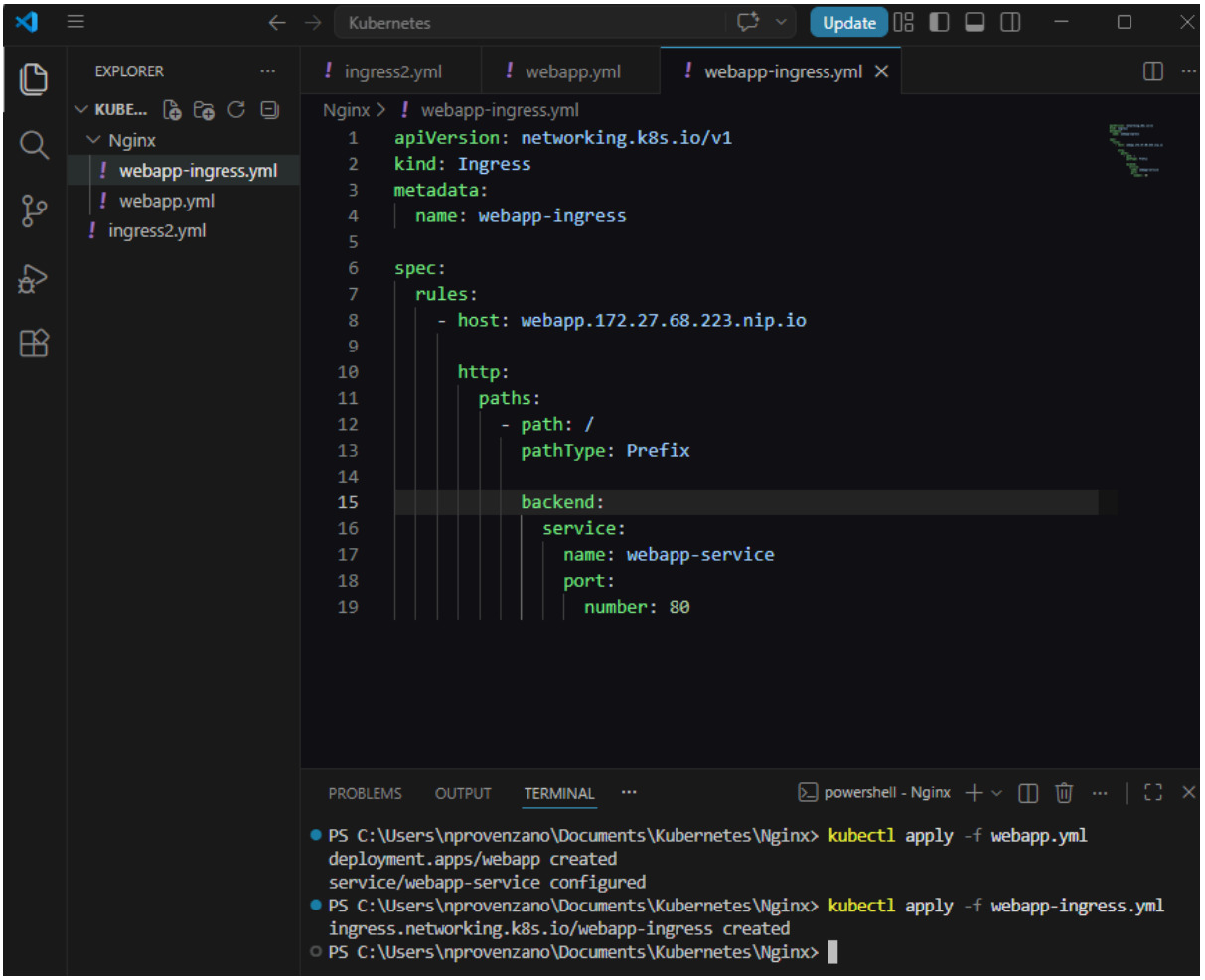


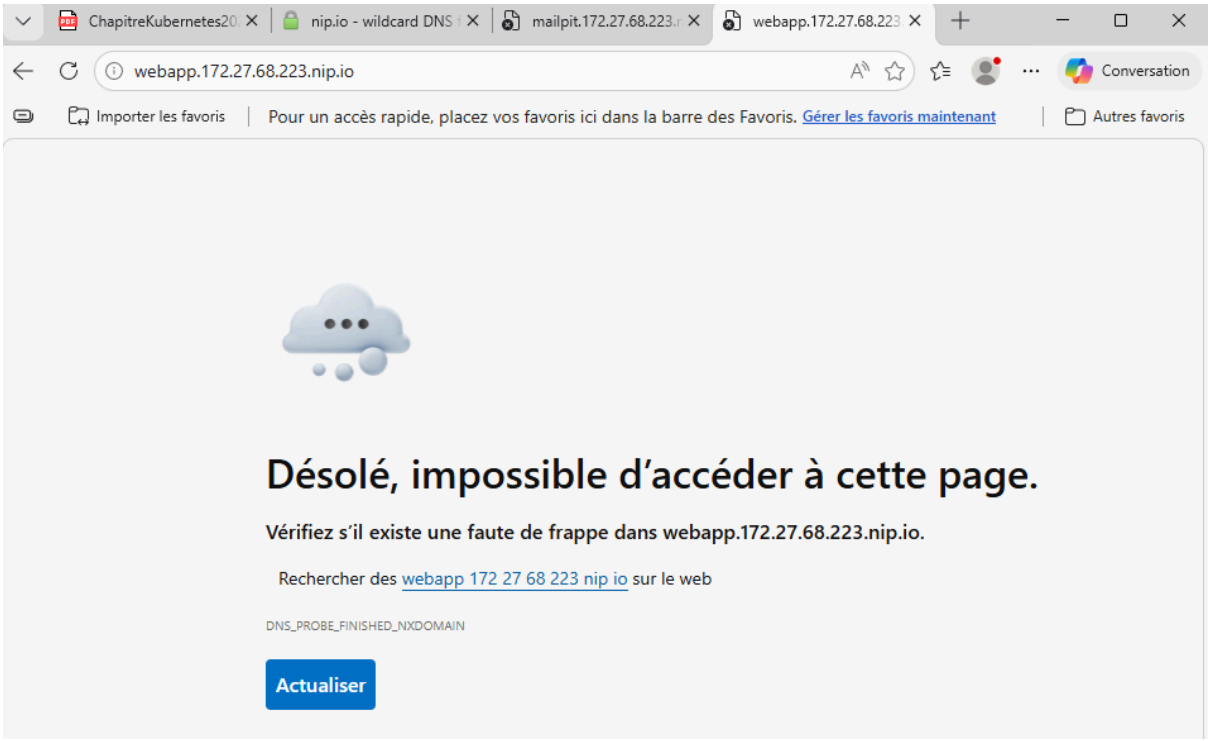
8. Hôtes virtuels et nom de domaine nip.io : autre exemple

Fichier webapp.yml



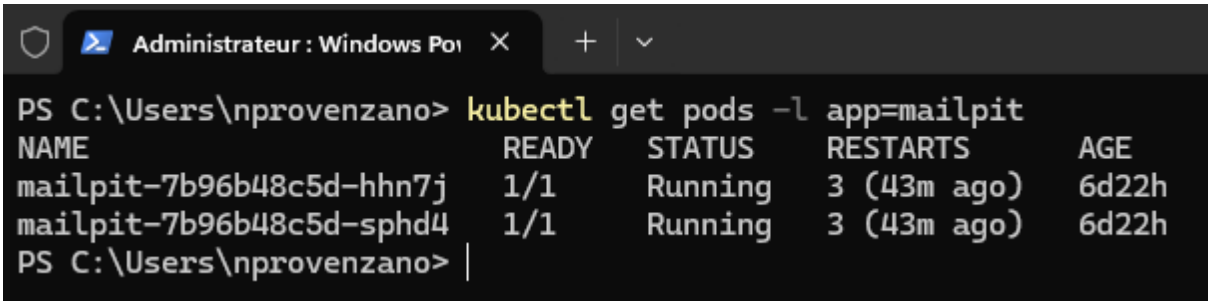
Fichier webapp-ingress.yml



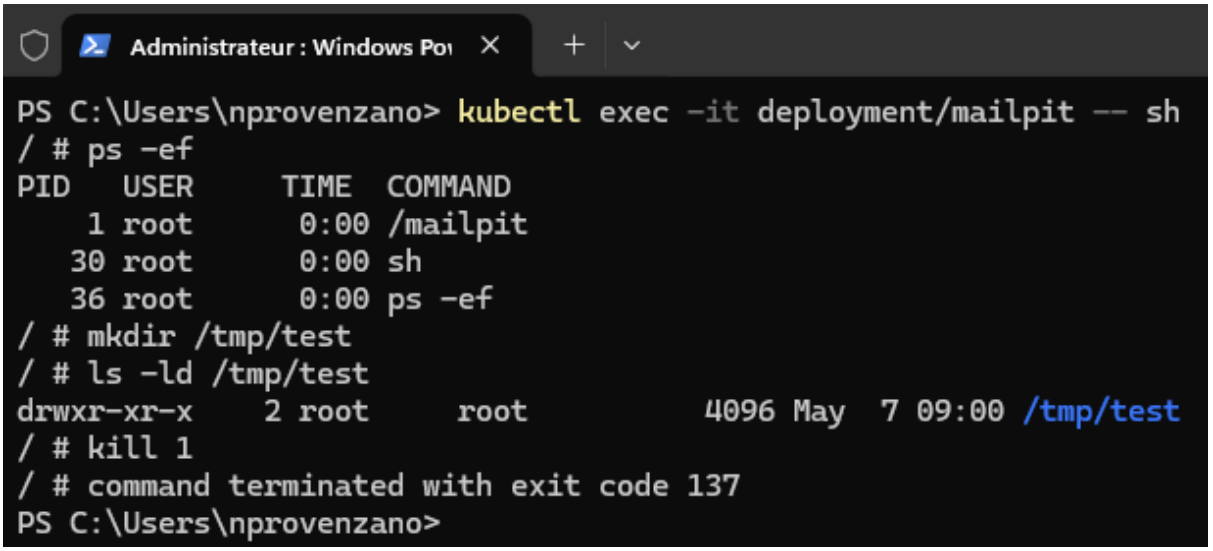


9. Cycle de vie d'un conteneur dans Kubernetes

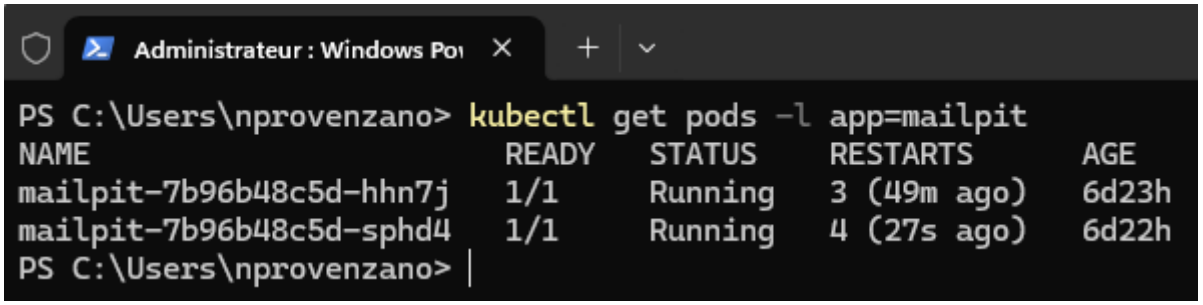
La commande **kubectl get pods -l app=mailpit** permet d'afficher les pods de l'application Mailpit.



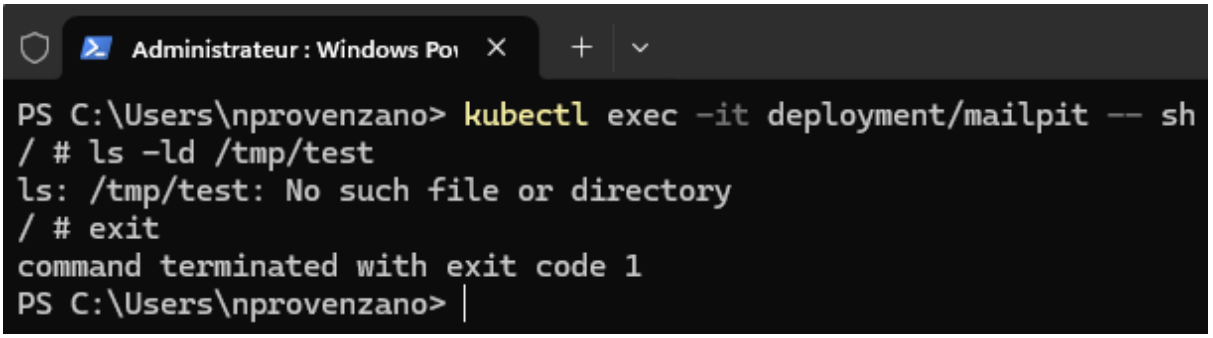
La commande **kubectl exec -it deployment/mailpit -- sh** permet d'accéder au terminal du conteneur Mailpit.



La commande **kubectl get pods -l app=mailpit** permet d'afficher les pods de l'application Mailpit et leur état.



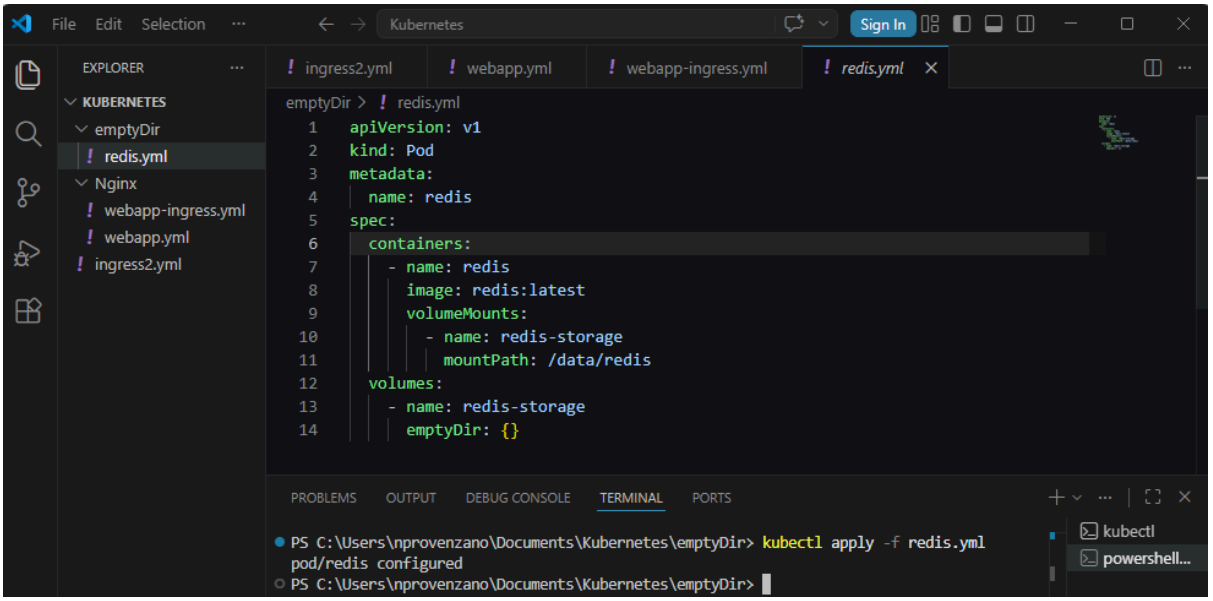
La commande kubectl **exec -it deployment/mailpit -- sh** permet d'accéder au terminal du conteneur Mailpit.
L'absence du répertoire **/tmp/test** après le redémarrage du conteneur montre que les données non persistantes ont été supprimées.



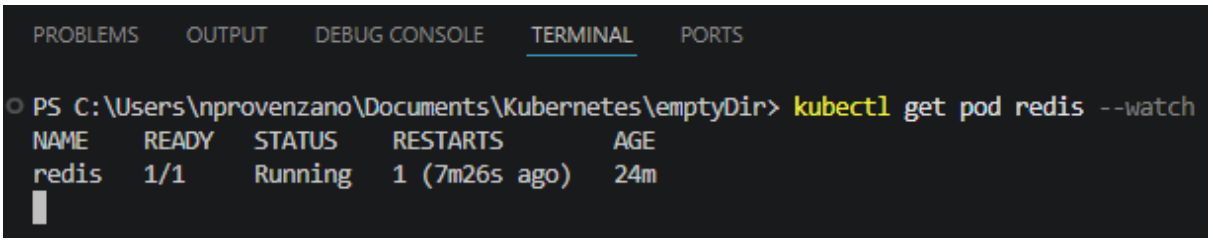
10. Persistence des données

1) Volume de type emptyDir

Le fichier **redis.yml** permet de créer un pod Redis utilisant un volume de type emptyDir.
La commande **kubectl apply -f redis.yml** permet de créer le pod Redis.



La commande **kubectl get pod redis --watch** permet de surveiller en temps réel l'état du pod Redis.



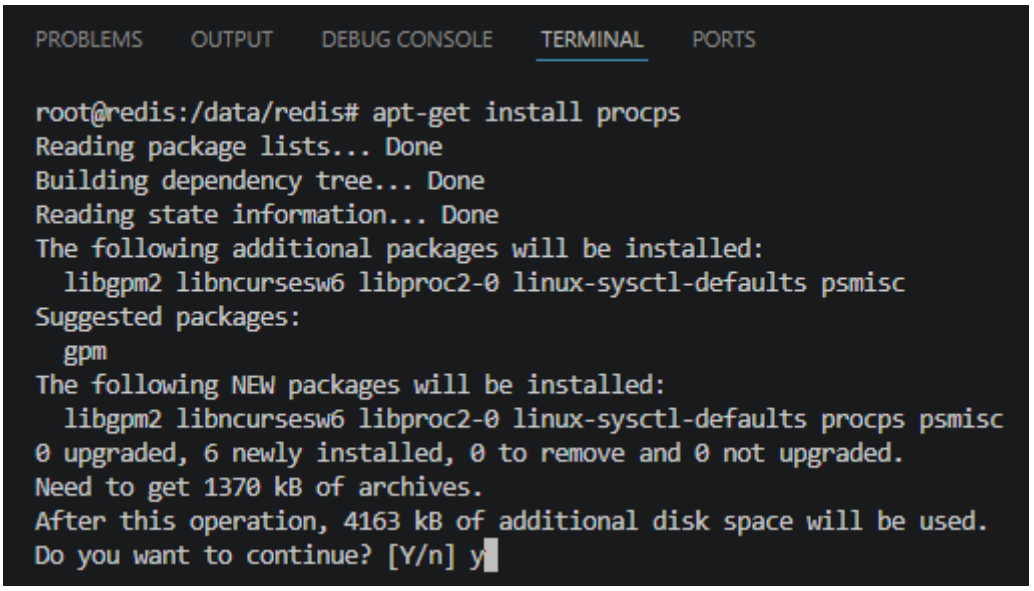
La commande **kubectl exec -it redis -- /bin/bash** permet d'accéder au terminal du conteneur Redis.
La création du fichier **test-file** dans **/data/redis** permet de tester la persistance du volume emptyDir.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\nprovenzano\Documents\Kubernetes\emptyDir> kubectl exec -it redis -- /bin/bash
root@redis:/data# cd /data/redis/
root@redis:/data/redis# echo Hello > test-file
root@redis:/data/redis#
```

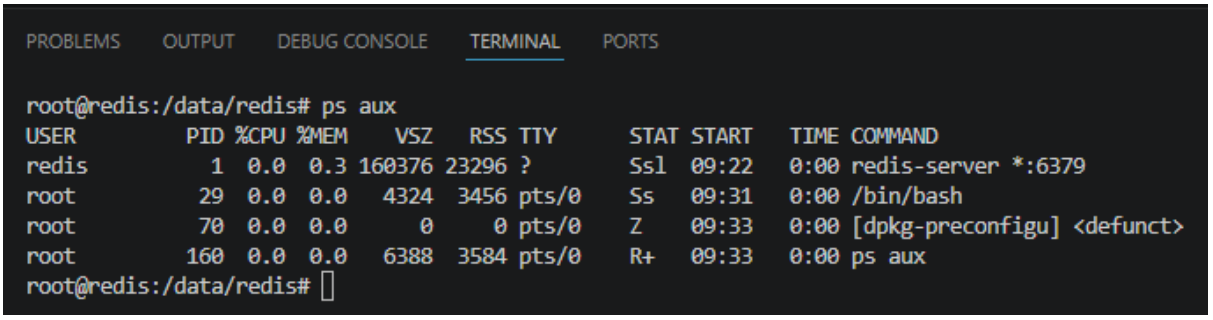
La commande **apt-get update** permet de mettre à jour la liste des paquets du conteneur Redis.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

root@redis:/data/redis# apt-get update
Get:1 http://deb.debian.org/debian trixie InRelease [140 kB]
Get:2 http://deb.debian.org/debian trixie-updates InRelease [47.3 kB]
Get:3 http://deb.debian.org/debian-security trixie-security InRelease [43.4 kB]
Get:4 http://deb.debian.org/debian trixie/main amd64 Packages [9671 kB]
Get:5 http://deb.debian.org/debian trixie-updates/main amd64 Packages [5412 B]
Get:6 http://deb.debian.org/debian-security trixie-security/main amd64 Packages [132 kB]
Fetched 10.0 MB in 1s (7056 kB/s)
Reading package lists... Done
root@redis:/data/redis#
```

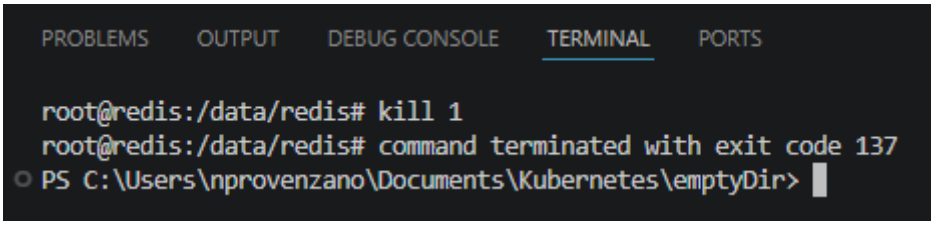
La commande **apt-get install procps** permet d'installer les outils nécessaires pour afficher les processus du conteneur.



La commande **ps aux** permet d'afficher les processus en cours d'exécution dans le conteneur Redis.



La commande **kill 1** permet d'arrêter le processus principal du conteneur Redis, provoquant son redémarrage automatique.



Le pod Redis redémarre automatiquement après l'arrêt du processus principal du conteneur.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\nprovenzano\Documents\Kubernetes> kubectl get pod redis --watch
NAME    READY   STATUS      RESTARTS       AGE
redis   1/1     Running     1 (22s ago)    17m
redis   0/1     Completed   1 (11m ago)    27m
redis   1/1     Running     2 (2s ago)     27m
```

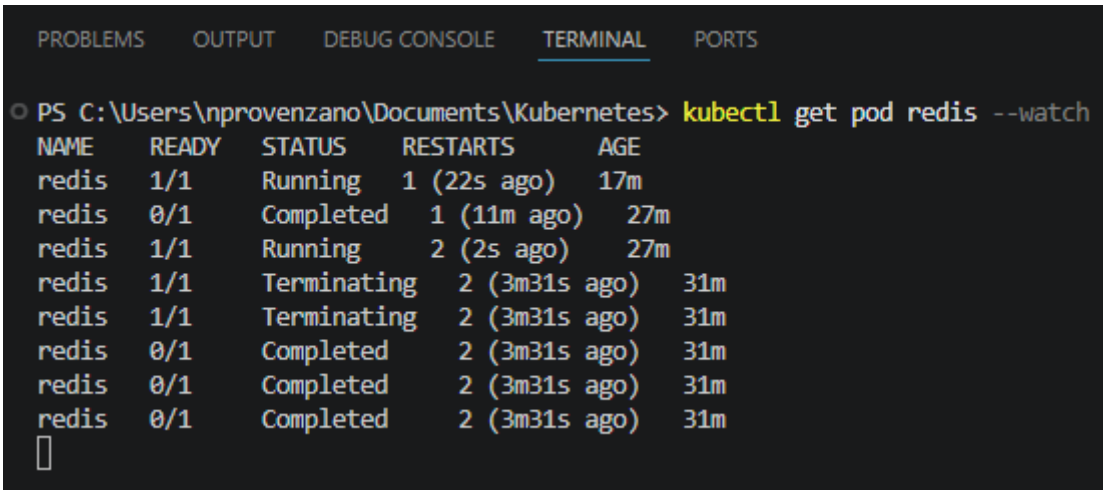
Le pod Redis redémarre automatiquement après l'arrêt du processus principal du conteneur.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\nprovenzano\Documents\Kubernetes\emptyDir> kubectl exec -it redis -- /bin/bash
root@redis:/data# cd /data/redis
root@redis:/data/redis# ls
test-file
root@redis:/data/redis# exit
exit
PS C:\Users\nprovenzano\Documents\Kubernetes\emptyDir>
```

La commande **kubectl delete pod redis** permet de supprimer le pod Redis du cluster Kubernetes.

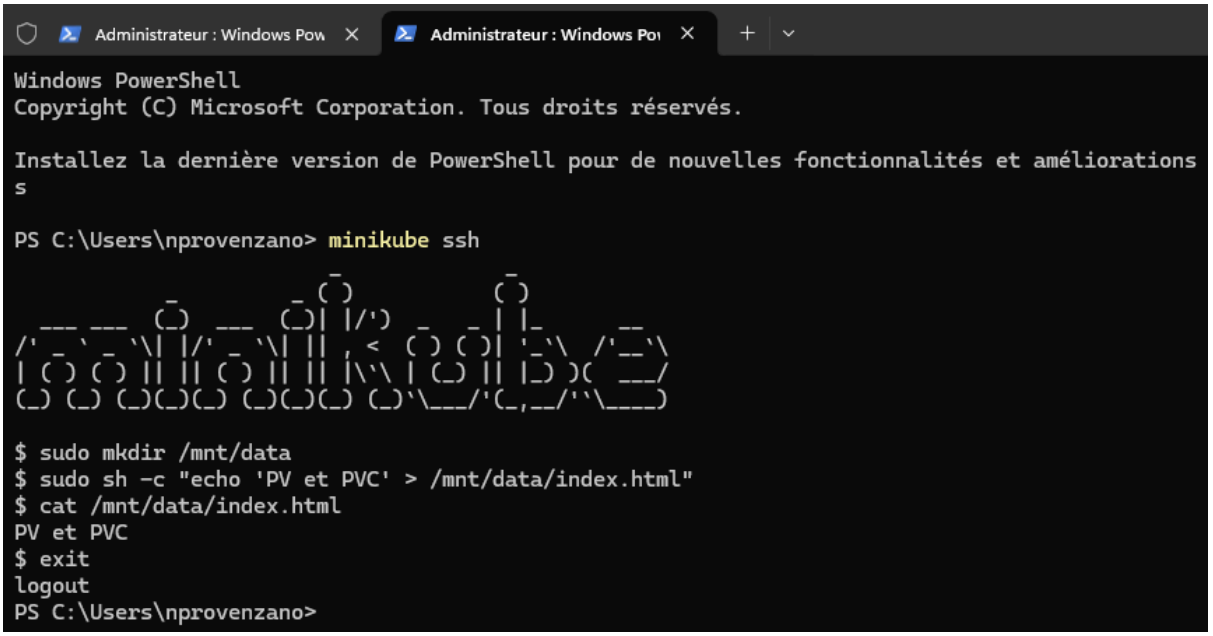
```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\nprovenzano\Documents\Kubernetes\emptyDir> kubectl delete pod redis
pod "redis" deleted from default namespace
PS C:\Users\nprovenzano\Documents\Kubernetes\emptyDir>
```

La suppression du pod Redis entraîne son arrêt définitif et la perte des données stockées dans le volume emptyDir.

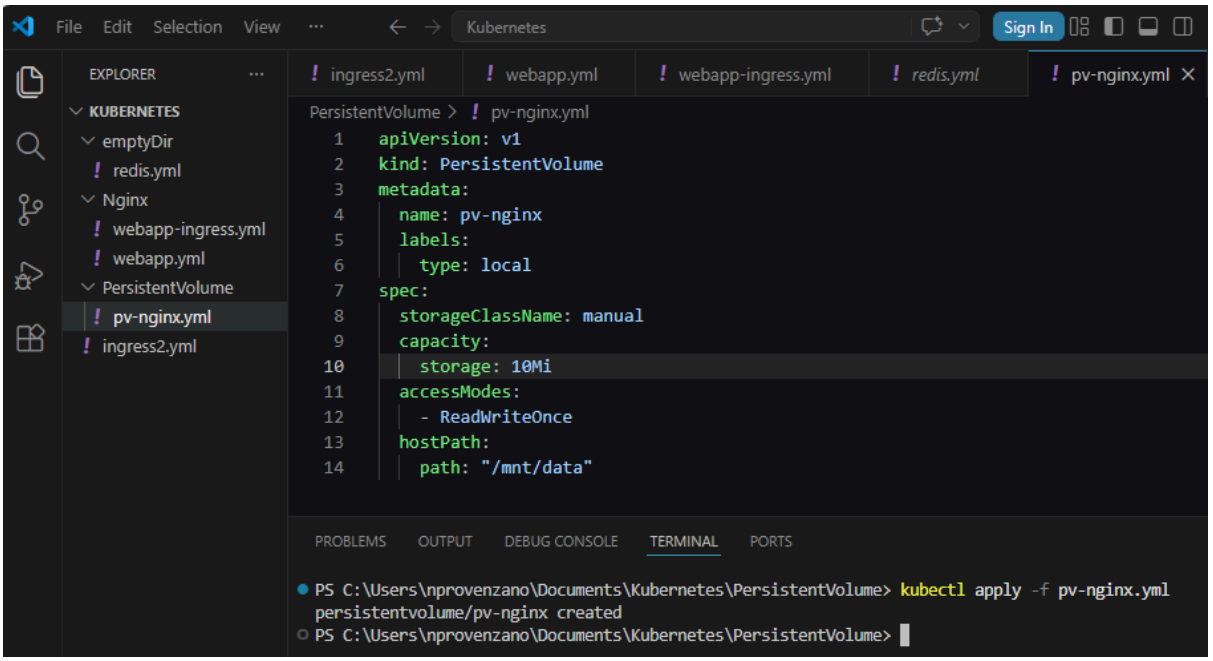


2) Volume de type PV (PersistentVolume) et PVC (PersistentVolumeClaim)

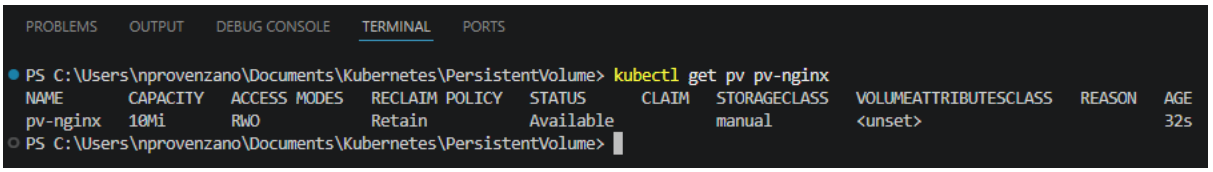
La commande **minikube ssh** permet d'accéder au nœud Minikube du cluster Kubernetes. La création du dossier **/mnt/data** et du fichier **index.html** permet de préparer le stockage persistant utilisé par le PersistentVolume.



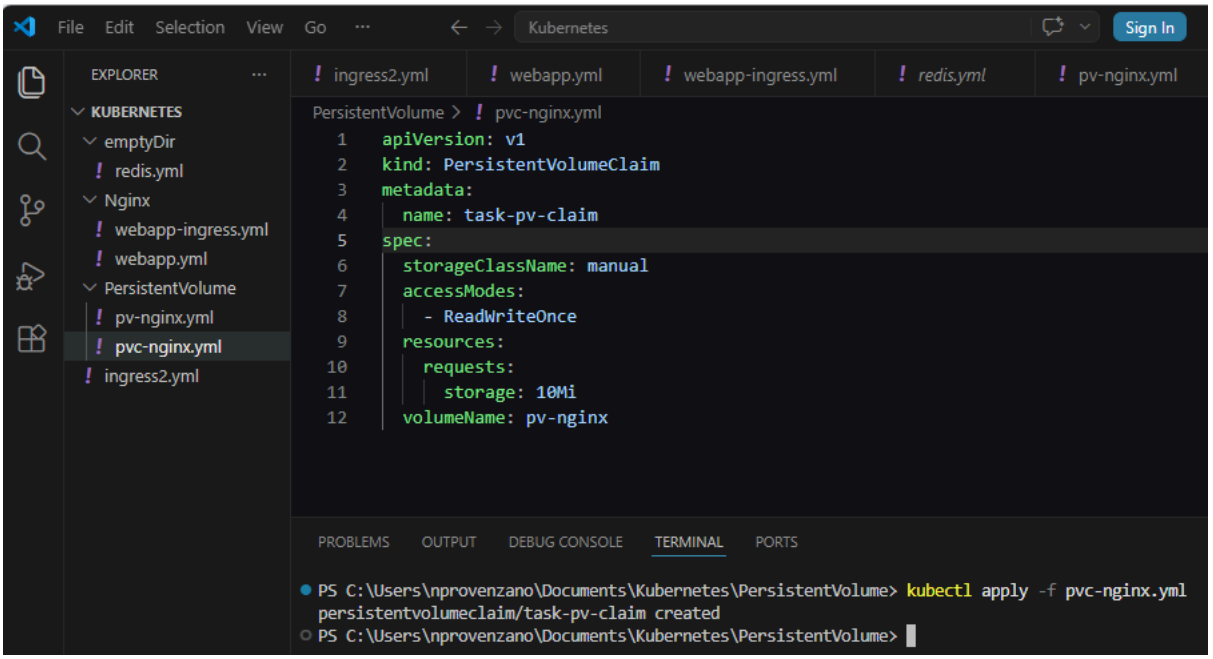
Le fichier **pv-nginx.yml** permet de créer un PersistentVolume de type hostPath pour le stockage persistant.
La commande **kubectl apply -f pv-nginx.yml** permet de créer le PersistentVolume pv-nginx.



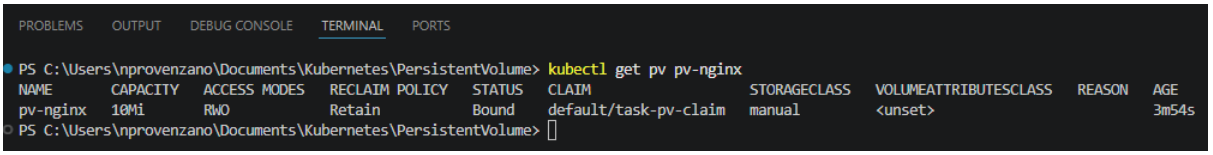
La commande **kubectl get pv pv-nginx** permet de vérifier que le PersistentVolume est créé et disponible.



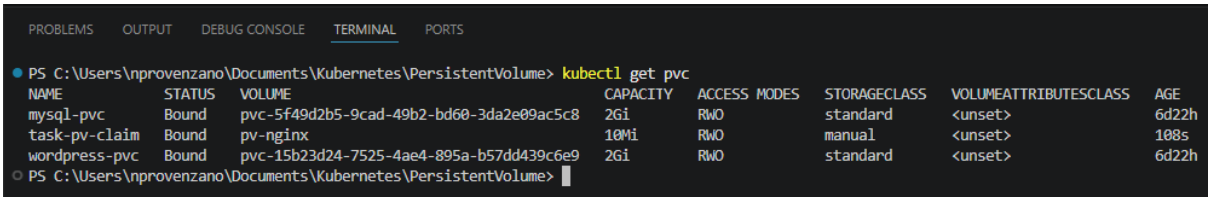
Le fichier **pvc-nginx.yml** permet de créer un PersistentVolumeClaim associé au volume persistant pv-nginx.
La commande **kubectl apply -f pvc-nginx.yml** permet de créer le PersistentVolumeClaim task-pv-claim.



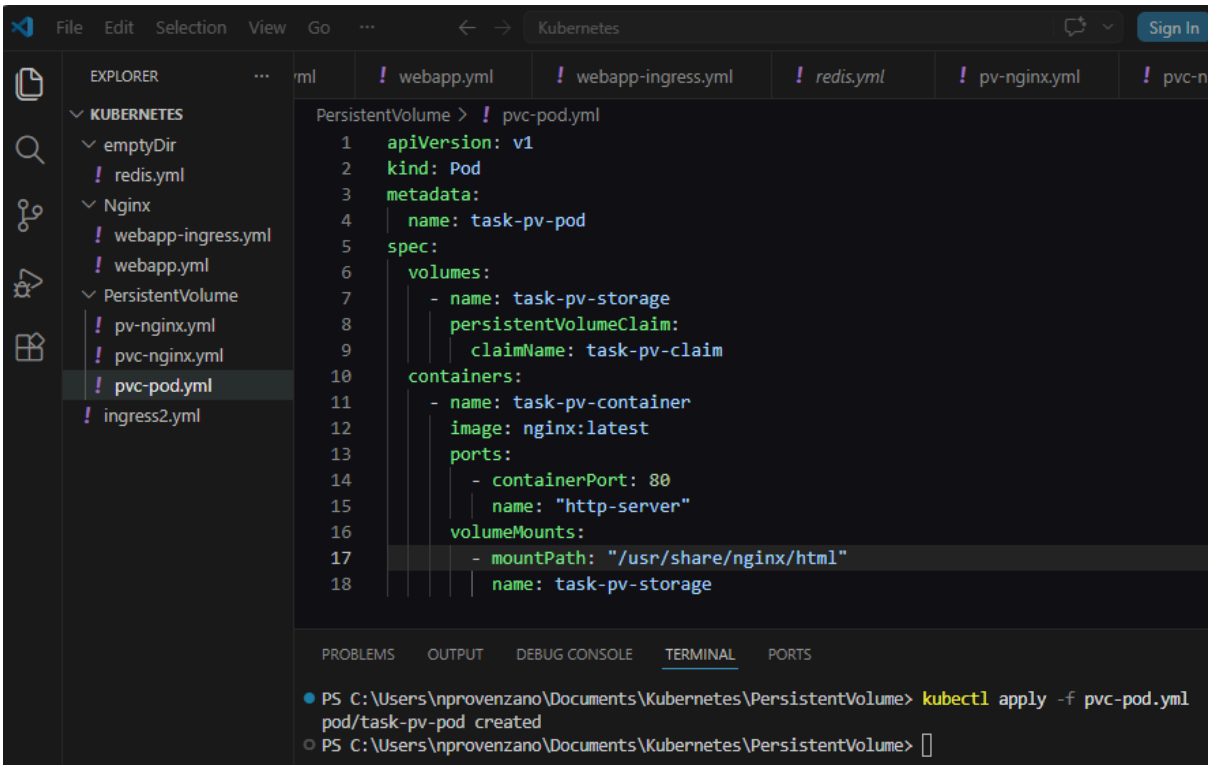
Le PersistentVolume pv-nginx est maintenant associé au PersistentVolumeClaim task-pv-claim avec le statut Bound.



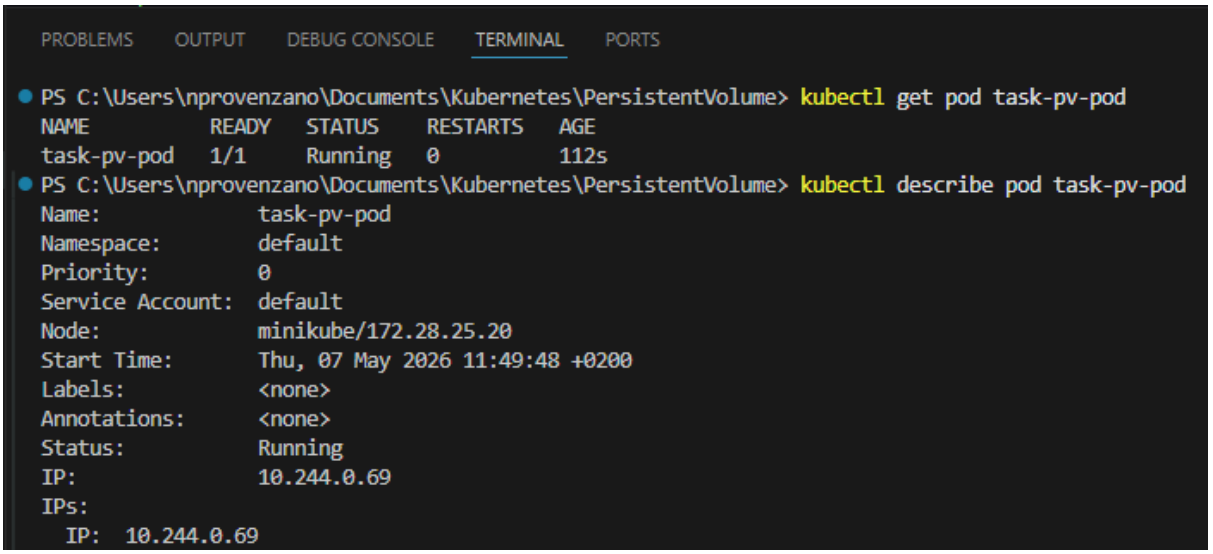
La commande **kubectl get pvc** permet d'afficher les PersistentVolumeClaims et leurs états dans le cluster Kubernetes.



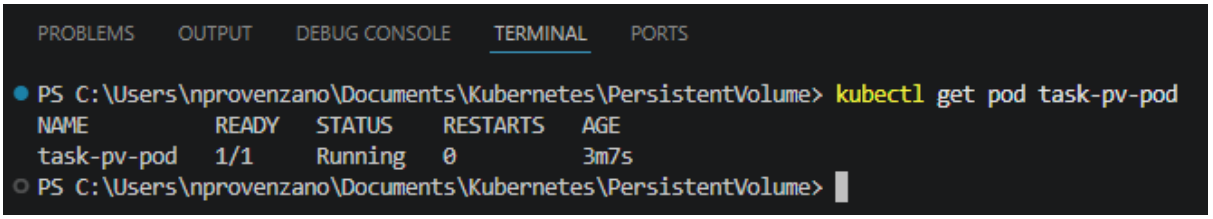
Le fichier **pvc-pod.yml** permet de créer un pod Nginx utilisant le PersistentVolumeClaim task-pv-claim.
La commande **kubectl apply -f pvc-pod.yml** permet de créer le pod task-pv-pod.



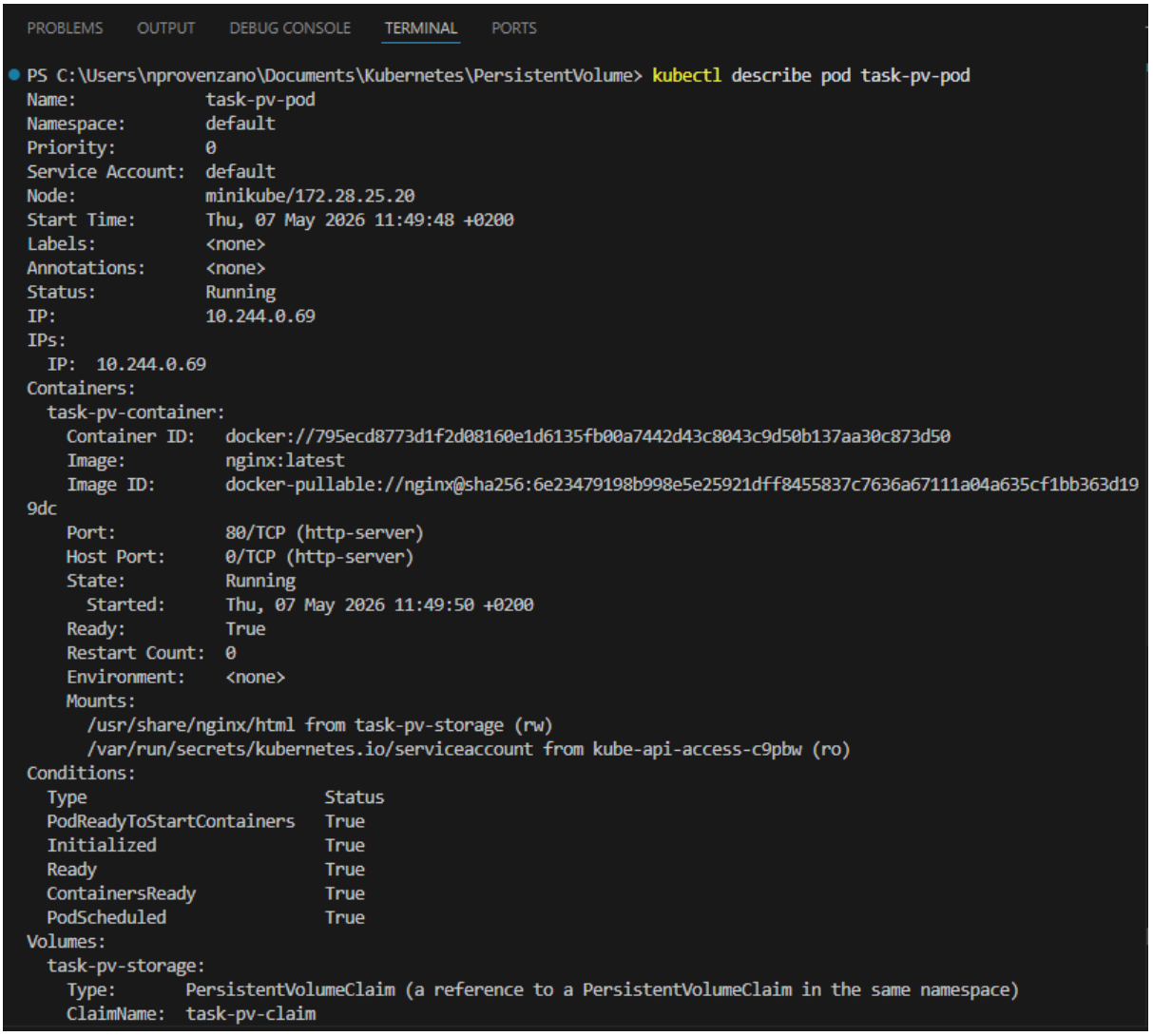
La commande **kubectl get pod task-pv-pod** permet de vérifier que le pod est en état Running.
La commande **kubectl describe pod task-pv-pod** permet d'afficher les informations détaillées du pod.



La commande **kubectl get pod task-pv-pod** permet de vérifier que le pod Nginx est en cours d'exécution.



La commande **kubectl describe pod task-pv-pod** permet d'afficher les détails du pod ainsi que le volume persistant utilisé.



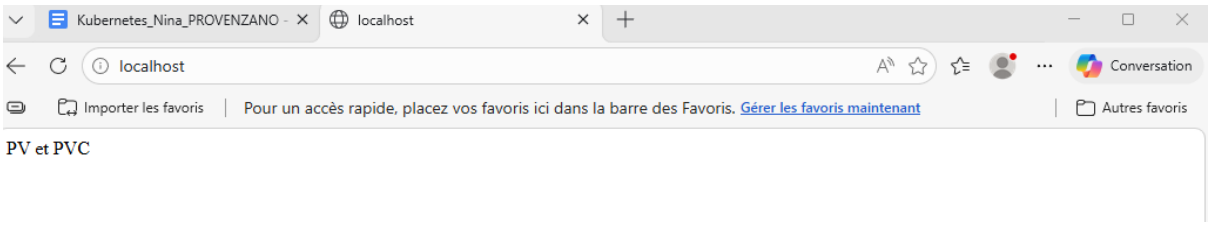
La commande **kubectl exec -it task-pv-pod -- /bin/bash** permet d'accéder au terminal du conteneur Nginx.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\nprovenzano\Documents\Kubernetes\PersistentVolume> kubectl exec -it task-pv-pod -- /bin/bash
root@task-pv-pod:/#
```

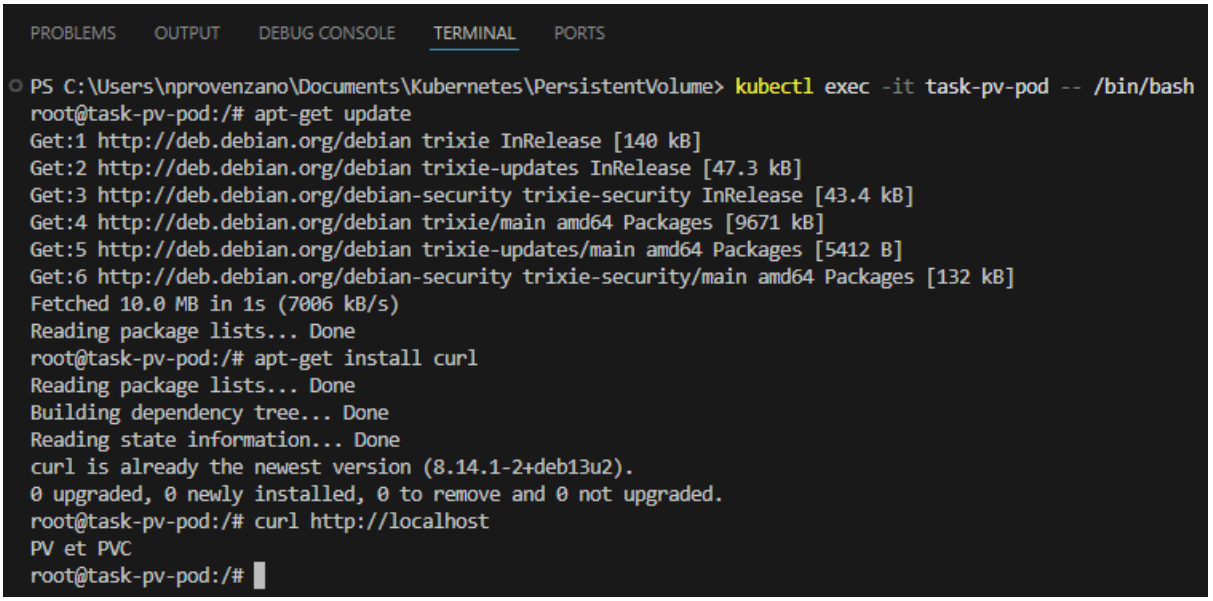
La commande **kubectl port-forward task-pv-pod 80** permet d'accéder au serveur Nginx du pod depuis localhost.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\nprovenzano\Documents\Kubernetes\PersistentVolume> kubectl port-forward task-pv-pod 80
Forwarding from 127.0.0.1:80 -> 80
Forwarding from [::1]:80 -> 80
Handling connection for 80
Handling connection for 80
```

L'accès à localhost permet de vérifier que le contenu stocké dans le volume persistant est bien affiché par le serveur Nginx.

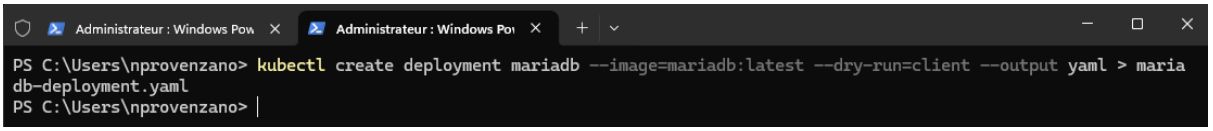


La commande **curl http://localhost** permet de vérifier que le serveur Nginx affiche le contenu du volume persistant.

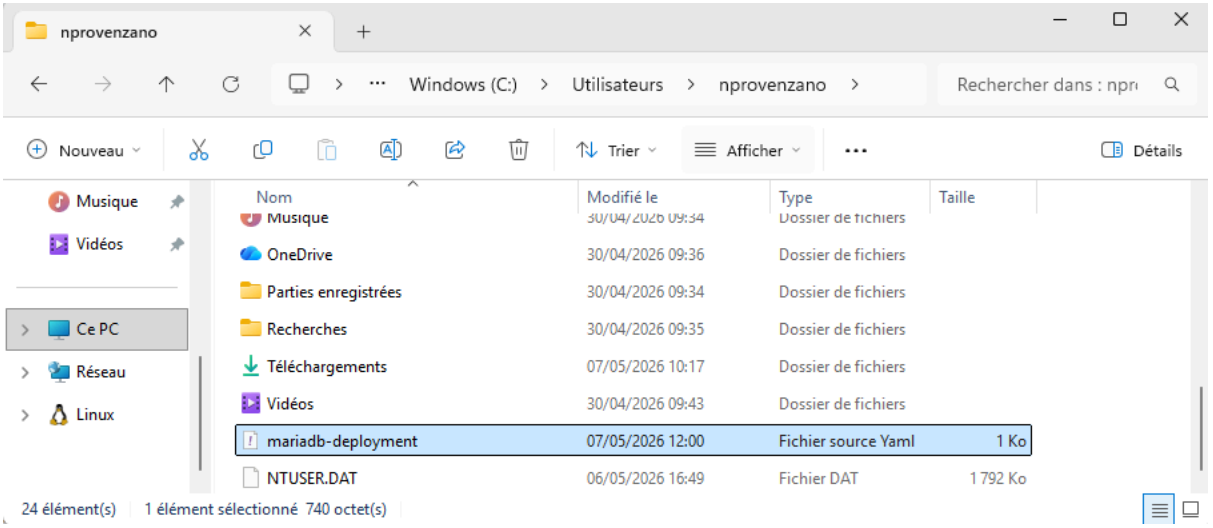


11. Hébergement d'un groupe de pods fonctionnant en cluster

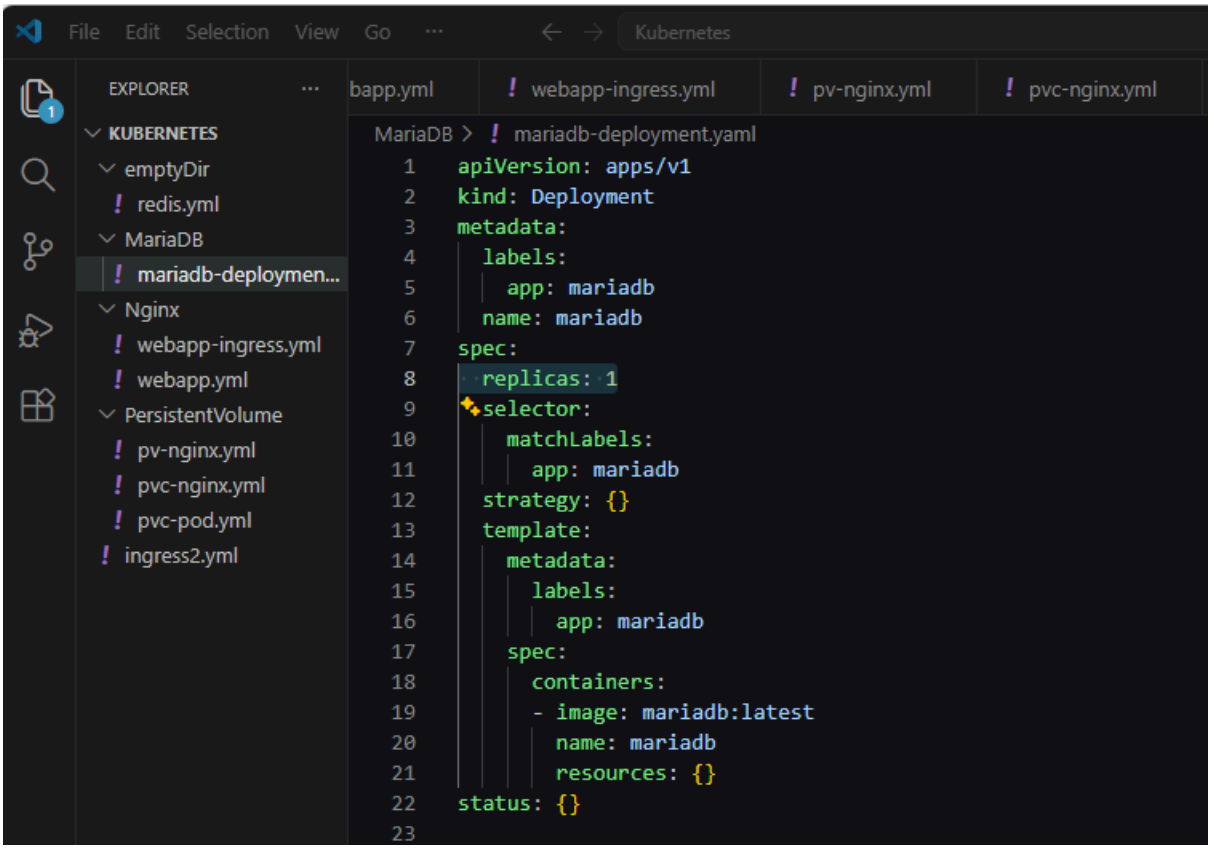
La commande **kubectl create deployment mariadb --image=mariadb:latest --dry-run=client --output yaml** permet de générer automatiquement le fichier YAML du déploiement MariaDB.



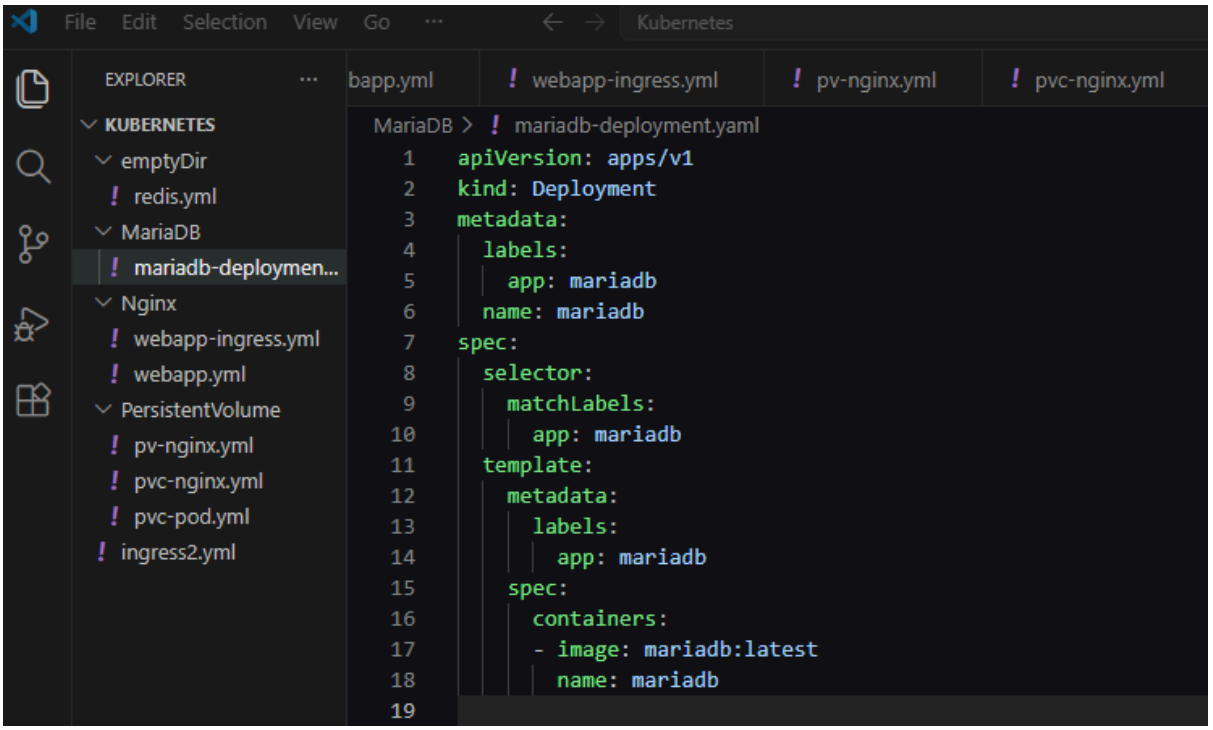
Le fichier **mariadb-deployment.yaml** contient la configuration du déploiement MariaDB générée automatiquement par Kubernetes.



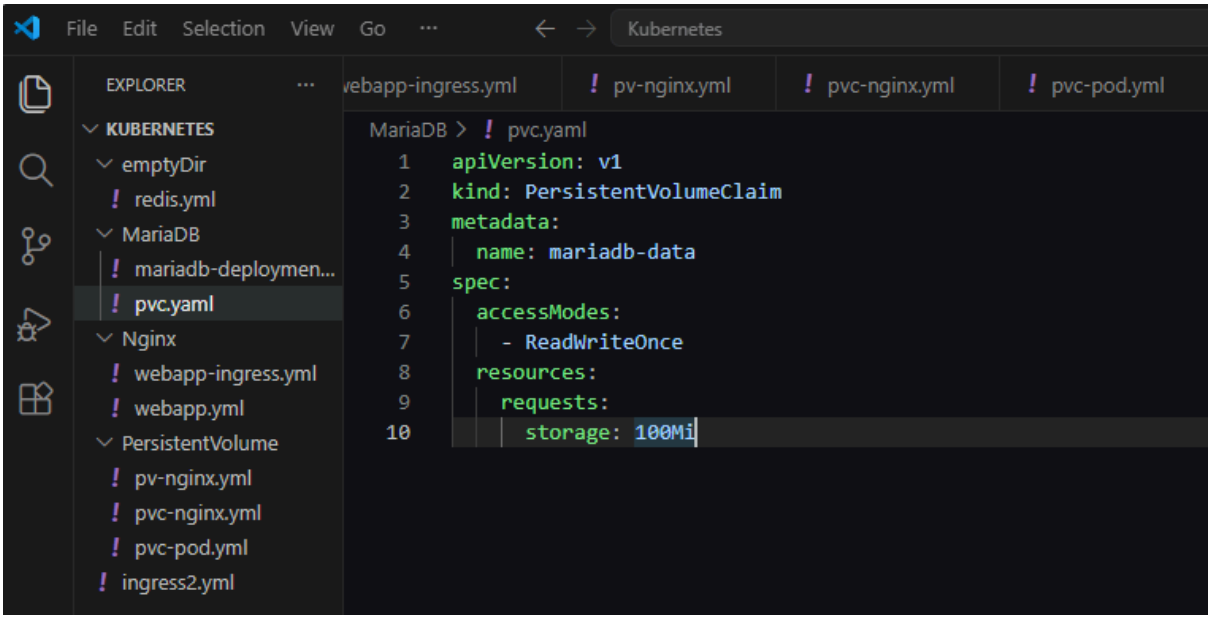
Le fichier **mariadb-deployment.yaml** contient la configuration du déploiement MariaDB avec l'image Docker mariadb:latest.



La modification du fichier **mariadb-deployment.yaml** permet de simplifier la configuration du déploiement MariaDB.



Le fichier **pvc.yaml** permet de créer un PersistentVolumeClaim pour stocker les données de MariaDB.



Le PersistentVolumeClaim mariadb-data est associé à un volume persistant avec le statut Bound.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
PS C:\Users\nprovenzano\Documents\Kubernetes\MariaDB> kubectl get pvc mariadb-data
NAME           STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS   VOLUMEATTRIBUTESCLASS   AGE
mariadb-data   Bound    pvc-8dbc6c9b-3f4f-43e2-8ea9-cb9854be3735   100Mi      RWO            standard       <unset>                 13m
PS C:\Users\nprovenzano\Documents\Kubernetes\MariaDB>
```

La commande **kubectl apply -f pvc.yaml** permet de créer le PersistentVolumeClaim mariadb-data.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\nprovenzano\Documents\Kubernetes\MariaDB> kubectl apply -f pvc.yaml
persistentvolumeclaim/mariadb-data created
PS C:\Users\nprovenzano\Documents\Kubernetes\MariaDB>
```

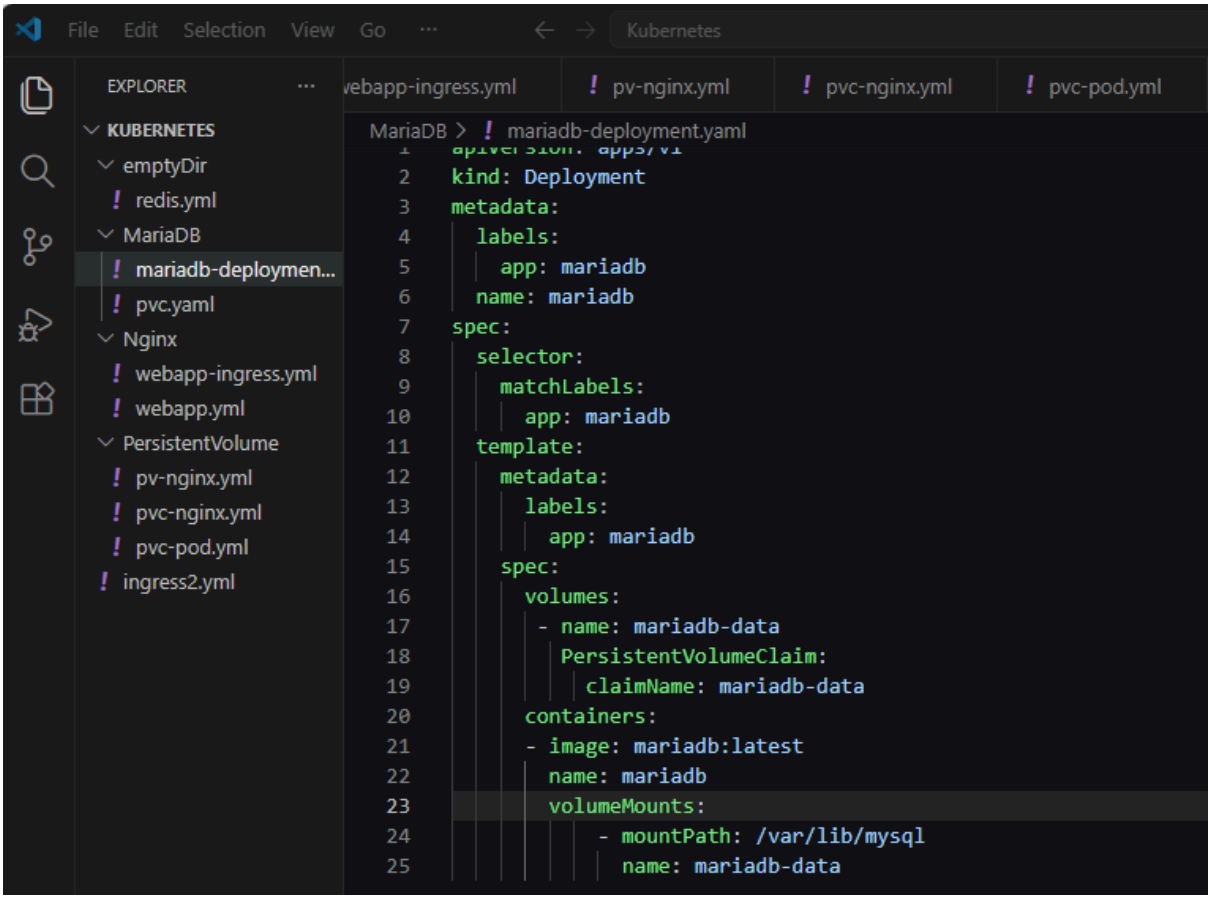
Le pod MariaDB est en état CrashLoopBackOff, indiquant un échec du démarrage du conteneur suivi de redémarrages automatiques.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
PS C:\Users\nprovenzano\Documents\Kubernetes\MariaDB> kubectl get pod -l app=mariadb
NAME                       READY   STATUS             RESTARTS      AGE
mariadb-5c476b5c79-256d2   0/1     CrashLoopBackOff   1 (8s ago)   11s
PS C:\Users\nprovenzano\Documents\Kubernetes\MariaDB>
```

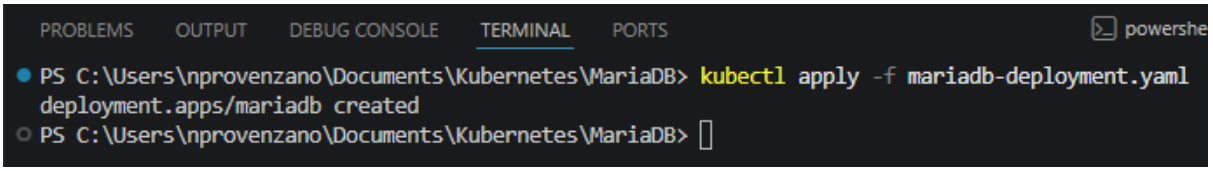
Le pod MariaDB est en état Error, indiquant un problème empêchant le démarrage correct du conteneur.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
PS C:\Users\nprovenzano\Documents\Kubernetes\MariaDB> kubectl get pod -l app=mariadb
NAME                       READY   STATUS    RESTARTS      AGE
mariadb-5c476b5c79-256d2   0/1     Error     3 (38s ago)   55s
PS C:\Users\nprovenzano\Documents\Kubernetes\MariaDB>
```

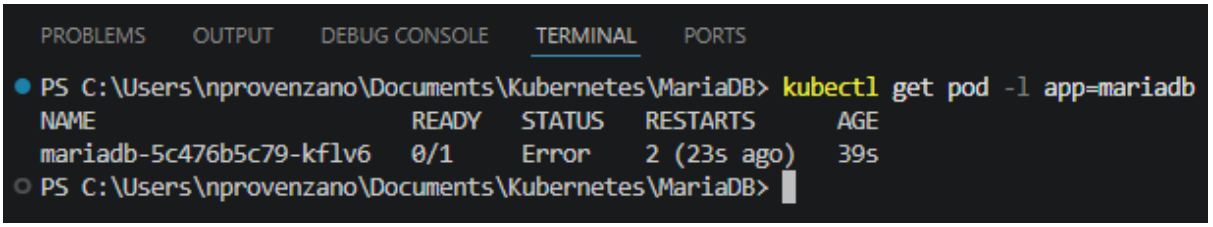
La modification du fichier **mariadb-deployment.yaml** permet d'ajouter un volume persistant au conteneur MariaDB.



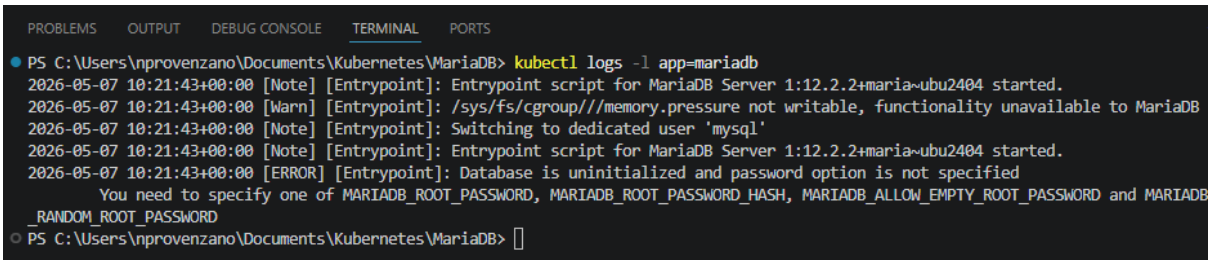
La commande **kubectl apply -f mariadb-deployment.yaml** permet de créer le déploiement MariaDB avec le volume persistant.



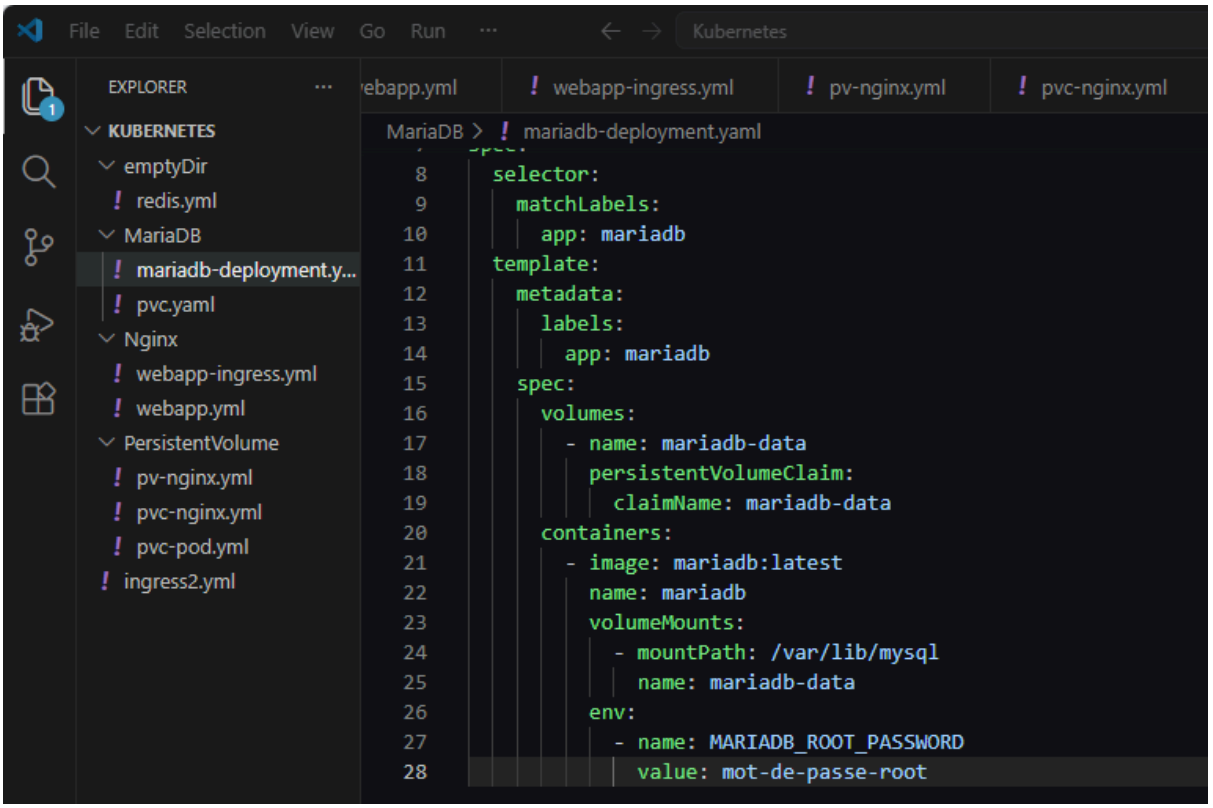
Le pod MariaDB reste en état Error, indiquant que le conteneur ne démarre toujours pas correctement.



Les logs du pod MariaDB indiquent que le mot de passe administrateur n'est pas défini, ce qui empêche le démarrage du conteneur.



La modification du fichier **mariadb-deployment.yaml** permet d'ajouter la variable MARIADB_ROOT_PASSWORD nécessaire au démarrage de MariaDB.



La commande **kubectl apply -f mariadb-deployment.yaml** permet de mettre à jour le déploiement MariaDB.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
PS C:\Users\nprovenzano\Documents\Kubernetes\MariaDB> kubectl apply -f mariadb-deployment.yaml
deployment.apps/mariadb configured
PS C:\Users\nprovenzano\Documents\Kubernetes\MariaDB>
```

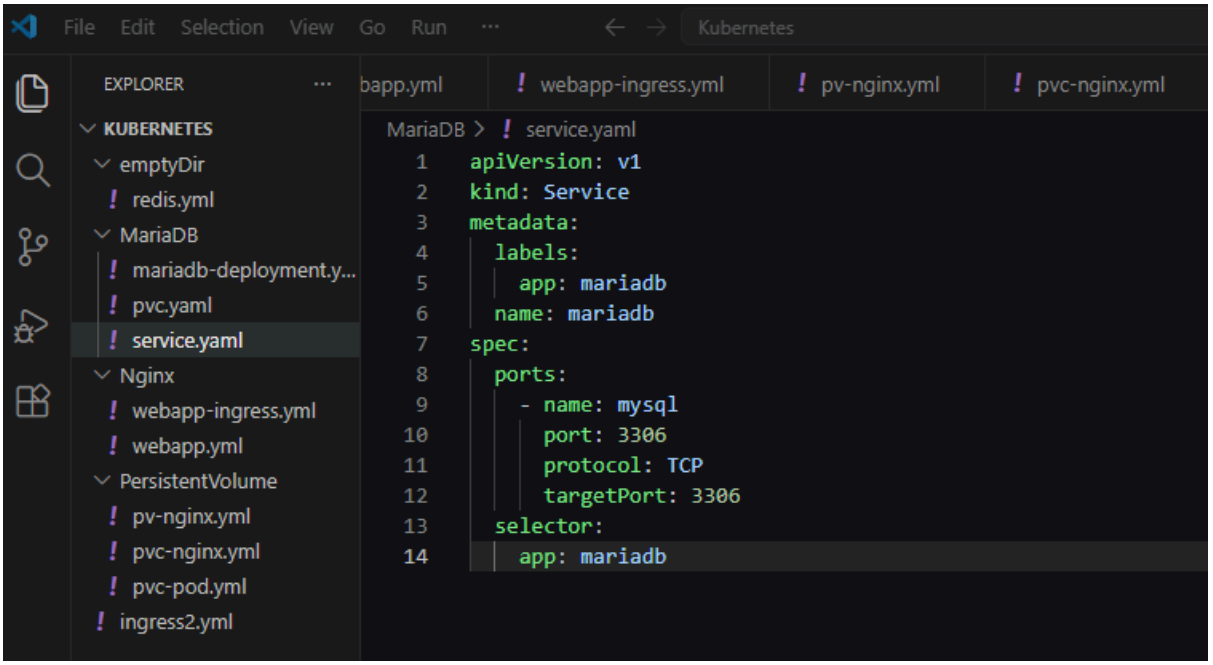
Le pod MariaDB est maintenant en état Running, indiquant que le conteneur fonctionne correctement.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
PS C:\Users\nprovenzano\Documents\Kubernetes\MariaDB> kubectl get pod -l app=mariadb
NAME                       READY   STATUS    RESTARTS   AGE
mariadb-5bbb9c9b6d-llktt   1/1     Running   0          28s
PS C:\Users\nprovenzano\Documents\Kubernetes\MariaDB>
```

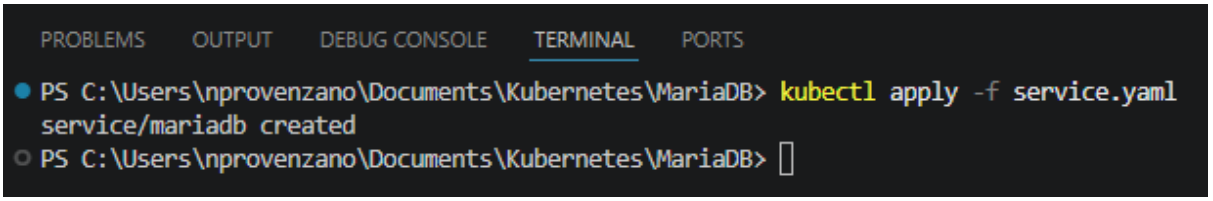
La commande **mariadb-admin status** permet de vérifier que le serveur MariaDB fonctionne correctement dans le conteneur.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
PS C:\Users\nprovenzano\Documents\Kubernetes\MariaDB> kubectl exec -it deployment/mariadb -- bash
root@mariadb-5bbb9c9b6d-llktt:/# mariadb-admin status -p$MARIADB_ROOT_PASSWORD
Uptime: 127  Threads: 1  Questions: 1  Slow queries: 0  Opens: 17  Open tables: 10  Queries per second avg: 0.007
root@mariadb-5bbb9c9b6d-llktt:/# exit
exit
PS C:\Users\nprovenzano\Documents\Kubernetes\MariaDB>
```

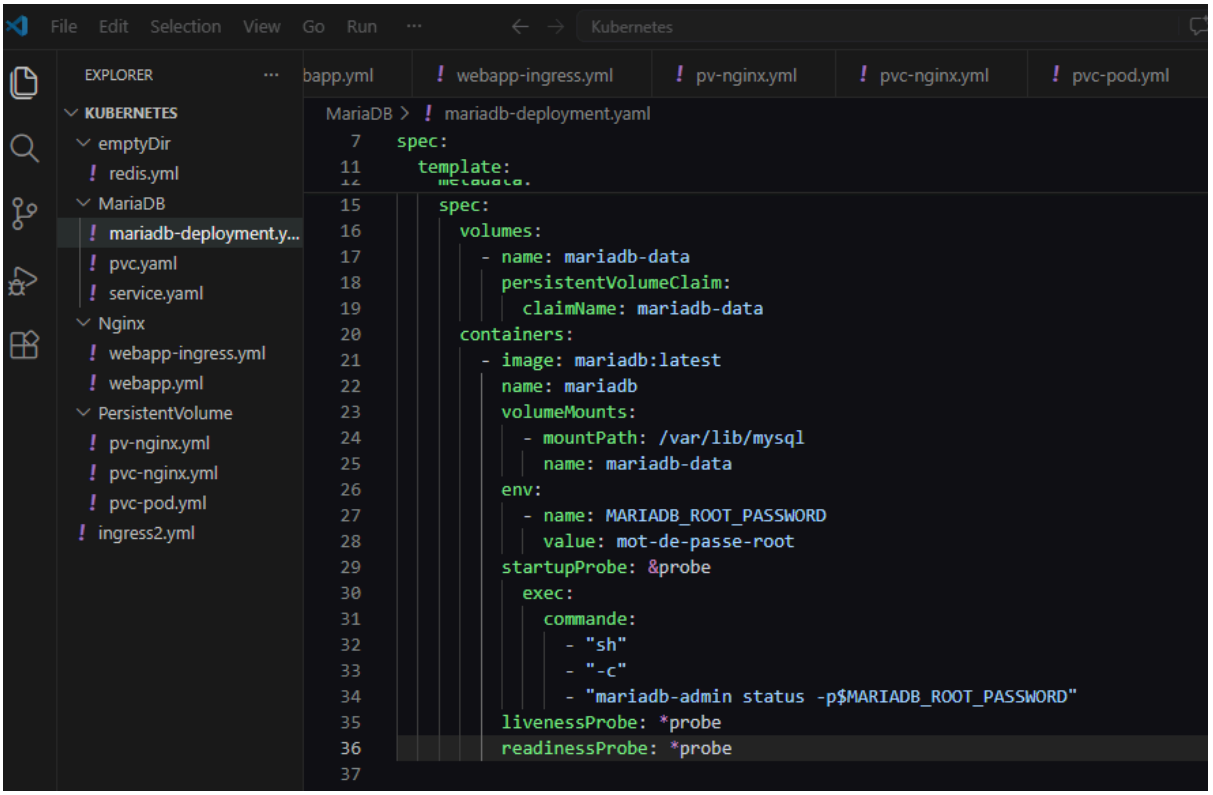
Le fichier **service.yaml** permet de créer un service Kubernetes pour exposer MariaDB sur le port 3306.



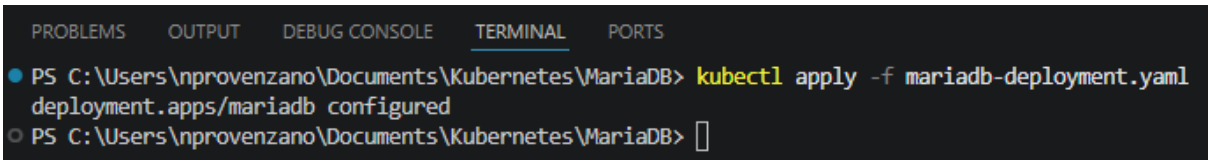
La commande **kubectl apply -f service.yaml** permet de créer le service MariaDB.



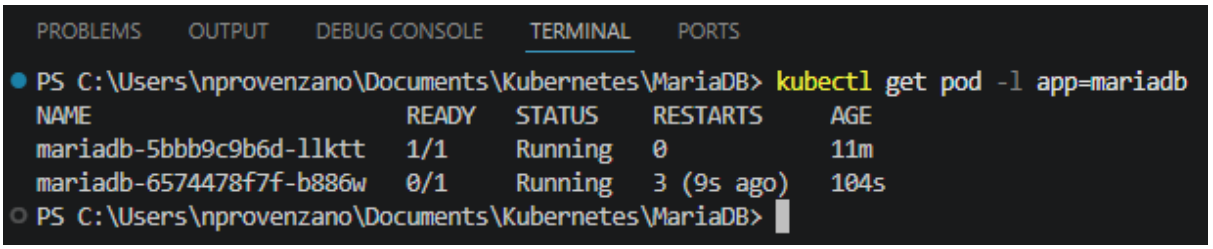
La configuration des probes permet à Kubernetes de surveiller l'état et le bon fonctionnement du conteneur MariaDB.



La commande **kubectl apply -f mariadb-deployment.yaml** permet d'appliquer les modifications du déploiement MariaDB.



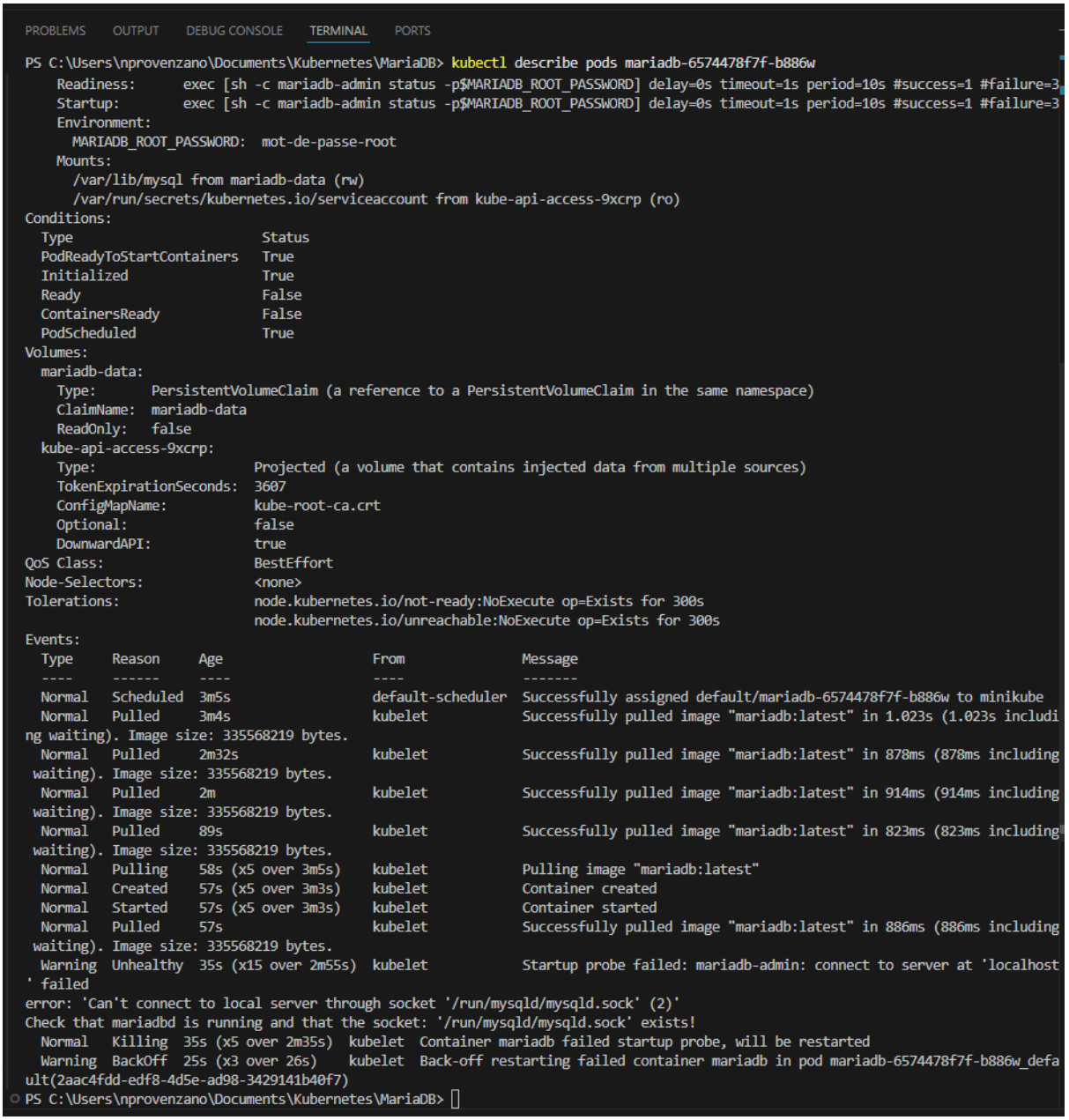
Les pods MariaDB sont en état Running, indiquant que les probes fonctionnent correctement.



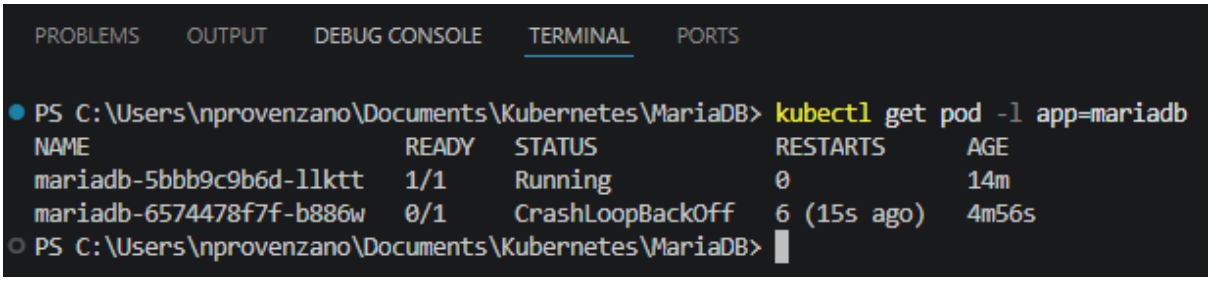
La commande **kubectl describe pods mariadb-6574478f7f-b080w** permet d'afficher les détails du pod ainsi que les informations des probes Kubernetes.

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

PS C:\Users\nprovenzano\Documents\Kubernetes\MariaDB> kubectl get pod -l app=mariadb
NAME                       READY   STATUS    RESTARTS      AGE
mariadb-5bbb9c9b6d-llktt   1/1     Running   0             12m
mariadb-6574478f7f-b886w   0/1     Running   4 (10s ago)   2m17s
PS C:\Users\nprovenzano\Documents\Kubernetes\MariaDB> kubectl describe pods mariadb-6574478f7f-b886w
Name:             mariadb-6574478f7f-b886w
Namespace:        default
Priority:         0
Service Account:  default
Node:             minikube/172.28.25.20
Start Time:       Thu, 07 May 2026 12:34:19 +0200
Labels:           app=mariadb
                  pod-template-hash=6574478f7f
Annotations:      <none>
Status:           Running
IP:               10.244.0.73
IPs:
  IP:           10.244.0.73
Controlled By:  ReplicaSet/mariadb-6574478f7f
Containers:
  mariadb:
    Container ID:   docker://b01ed11f2b0b3731f5f6255031a25a8303a263b7cb3f1caa046c6f241cc1bd66
    Image:          mariadb:latest
    Image ID:       docker-pullable://mariadb@sha256:e0236fc6386e7eacd9359e59d0a078bd7aa0d18280d36d13061121bedeaee903
    Port:           <none>
    Host Port:      <none>
    State:          Waiting
      Reason:       CrashLoopBackOff
    Last State:     Terminated
      Reason:       Error
      Exit Code:    1
      Started:      Thu, 07 May 2026 12:36:27 +0200
      Finished:     Thu, 07 May 2026 12:36:58 +0200
    Ready:          False
    Restart Count:  4
    Liveness:       exec [sh -c mariadb-admin status -p$MARIADB_ROOT_PASSWORD] delay=0s timeout=1s period=10s #success=1 #failure=3
    Readiness:      exec [sh -c mariadb-admin status -p$MARIADB_ROOT_PASSWORD] delay=0s timeout=1s period=10s #success=1 #failure=3
    Startup:        exec [sh -c mariadb-admin status -p$MARIADB_ROOT_PASSWORD] delay=0s timeout=1s period=10s #success=1 #failure=3
    Environment:
      MARIADB_ROOT_PASSWORD:  mot-de-passe-root
    Mounts:
      /var/lib/mysql from mariadb-data (rw)
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-9xcrp (ro)
```

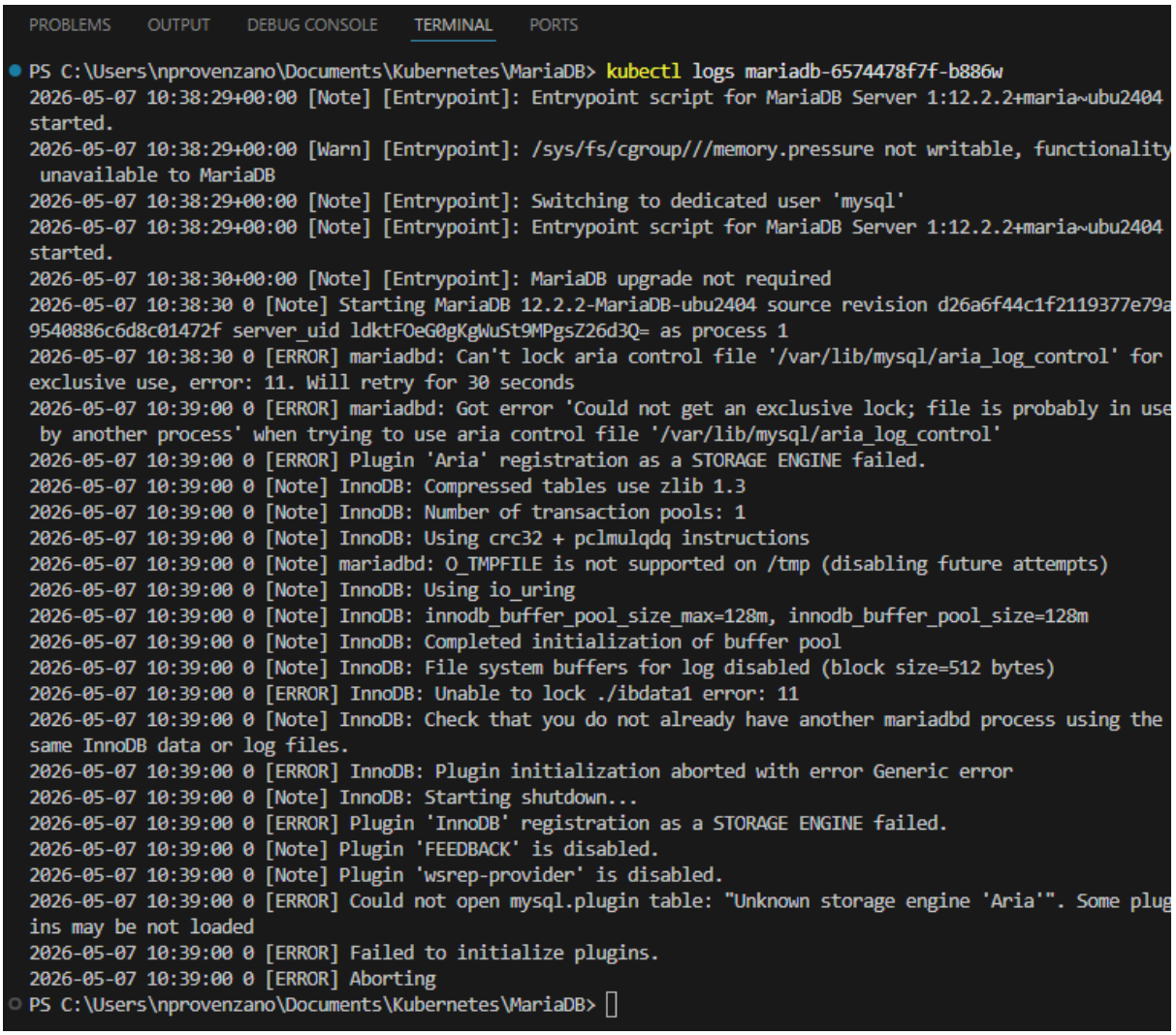
Les événements du pod indiquent que la startupProbe a échoué car le serveur MariaDB n'était pas encore prêt au démarrage.



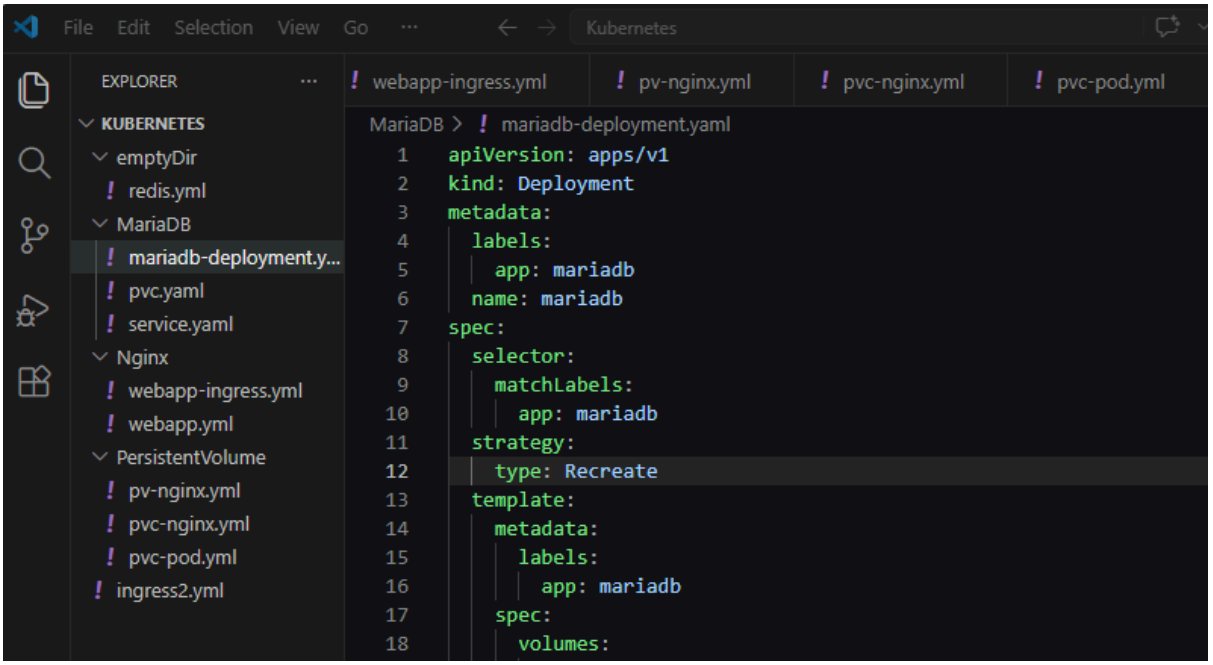
Le nouveau pod MariaDB fonctionne correctement tandis que l'ancien pod reste en état CrashLoopBackOff.



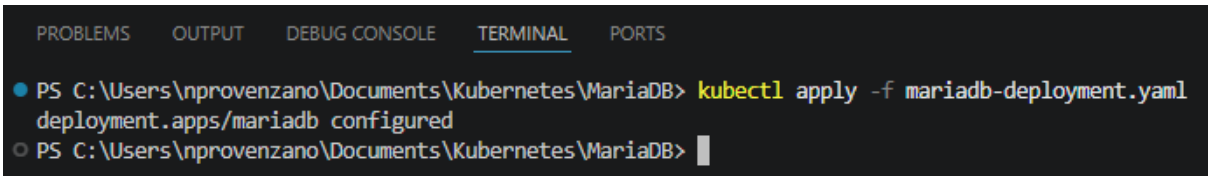
Les logs indiquent un conflit d'accès aux fichiers de la base de données, empêchant le démarrage correct du second pod MariaDB.



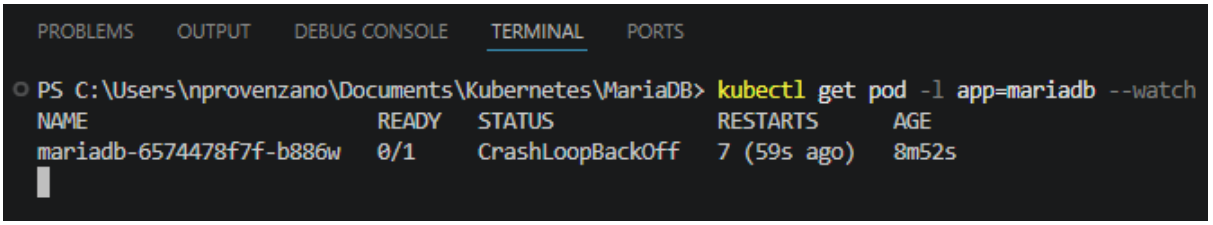
La stratégie Recreate permet d'éviter le démarrage simultané de plusieurs pods MariaDB utilisant le même volume persistant.



La commande **kubectl apply -f mariadb-deployment.yaml** permet d'appliquer la stratégie Recreate au déploiement MariaDB.



Le pod MariaDB redémarre automatiquement après un échec détecté par Kubernetes.



Le pod MariaDB est toujours en état CrashLoopBackOff malgré l'application de la stratégie Recreate.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\nprovenzano\Documents\Kubernetes\MariaDB> kubectl get pod -l app=mariadb
NAME                       READY   STATUS             RESTARTS        AGE
mariadb-6574478f7f-b886w   0/1     CrashLoopBackOff   7 (2m36s ago)   10m
PS C:\Users\nprovenzano\Documents\Kubernetes\MariaDB>
```

La commande **kubectl delete pod -l app=mariadb** permet de supprimer les pods MariaDB du déploiement.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\nprovenzano\Documents\Kubernetes\MariaDB> kubectl delete pod -l app=mariadb
pod "mariadb-6574478f7f-b886w" deleted from default namespace
PS C:\Users\nprovenzano\Documents\Kubernetes\MariaDB>
```

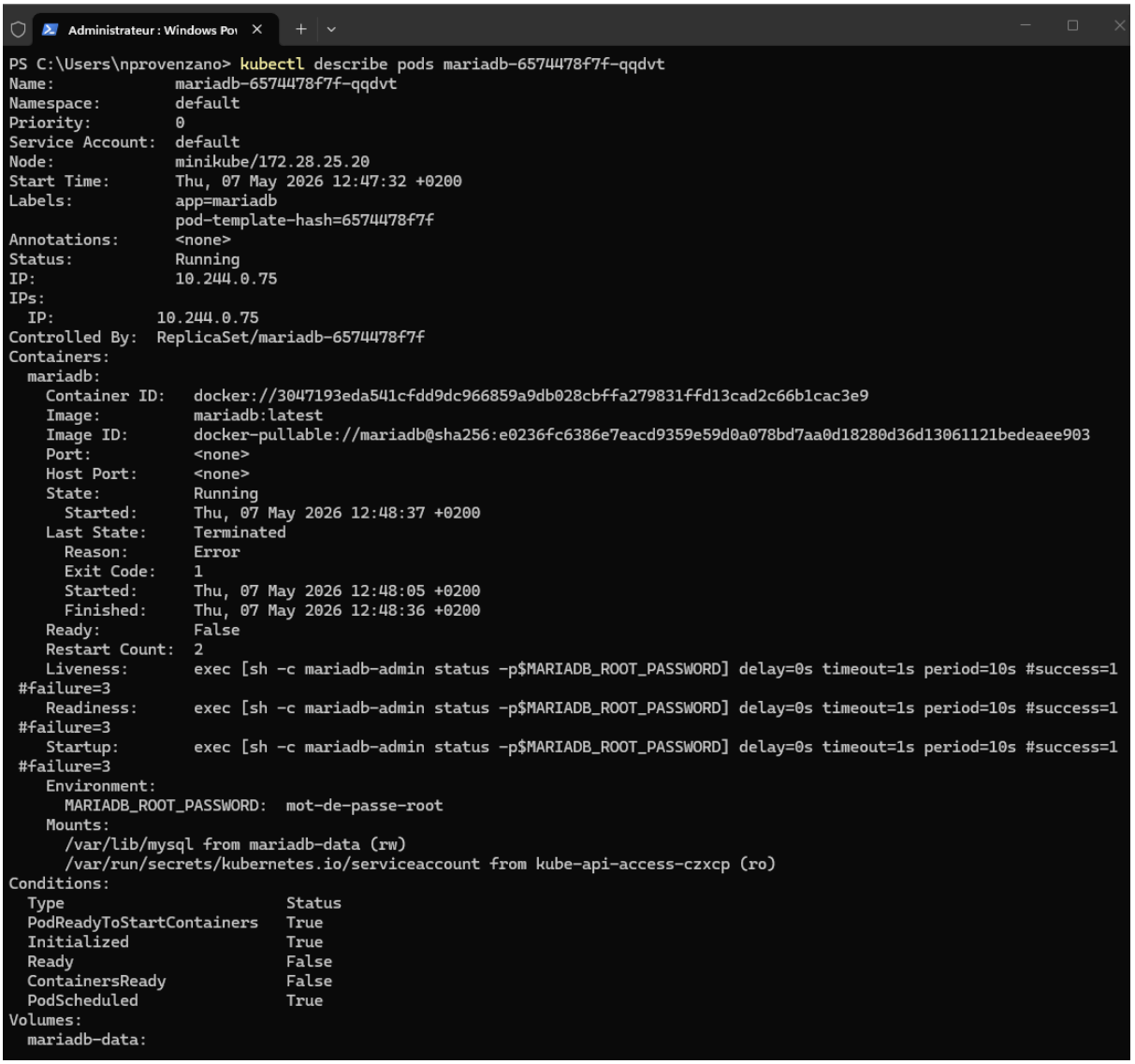
Le nouveau pod MariaDB est maintenant en état Running après la recréation du déploiement.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\nprovenzano\Documents\Kubernetes\MariaDB> kubectl get pod -l app=mariadb
NAME                       READY   STATUS    RESTARTS   AGE
mariadb-6574478f7f-65tkj   1/1     Running   0          22s
PS C:\Users\nprovenzano\Documents\Kubernetes\MariaDB>
```

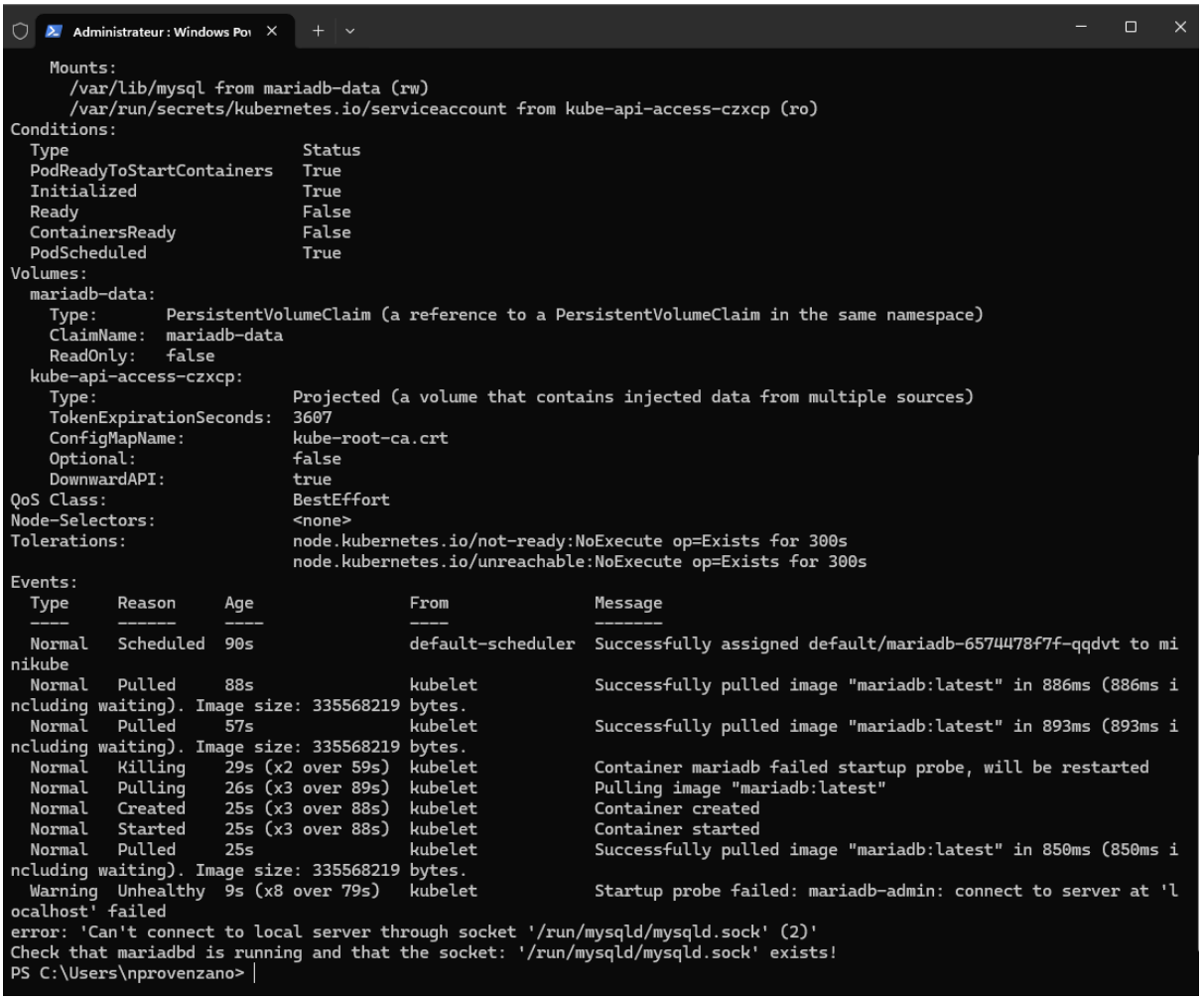
La commande **kubectl scale deployment mariadb --replicas=2** permet d'augmenter le nombre de pods du déploiement MariaDB.
Le second pod MariaDB démarre mais rencontre des problèmes liés au partage du même volume persistant.

```
Administrateur : Windows Po

PS C:\Users\nprovenzano> kubectl scale deployment mariadb --replicas=2
deployment.apps/mariadb scaled
PS C:\Users\nprovenzano> kubectl get pods -l app=mariadb --watch
NAME                       READY   STATUS    RESTARTS      AGE
mariadb-6574478f7f-65tkj   1/1     Running   0             2m33s
mariadb-6574478f7f-qqdvt   0/1     Running   0             20s
mariadb-6574478f7f-qqdvt   0/1     Running   1 (2s ago)    34s
```

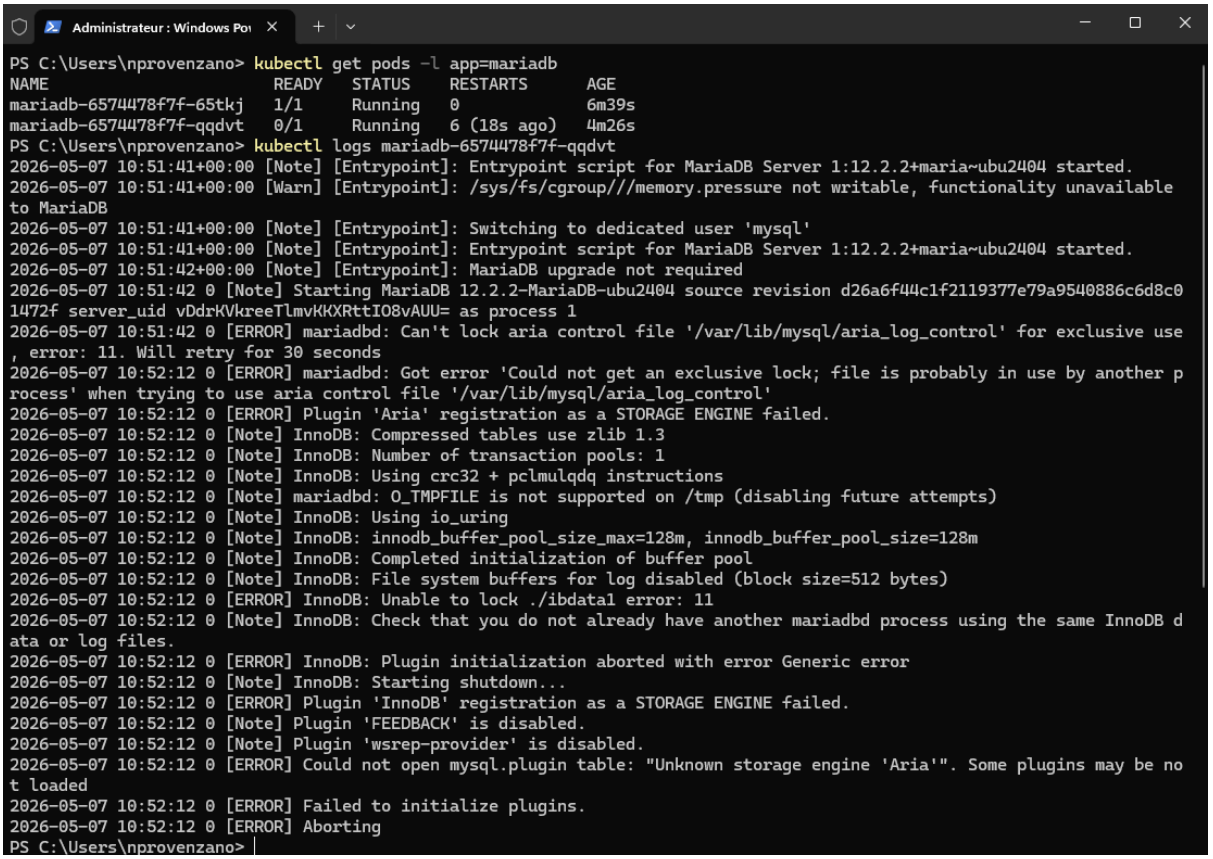
La commande **kubectl describe pods mariadb-6574478f7f-qqdvt** permet d'afficher les informations détaillées et l'état des probes du pod MariaDB.



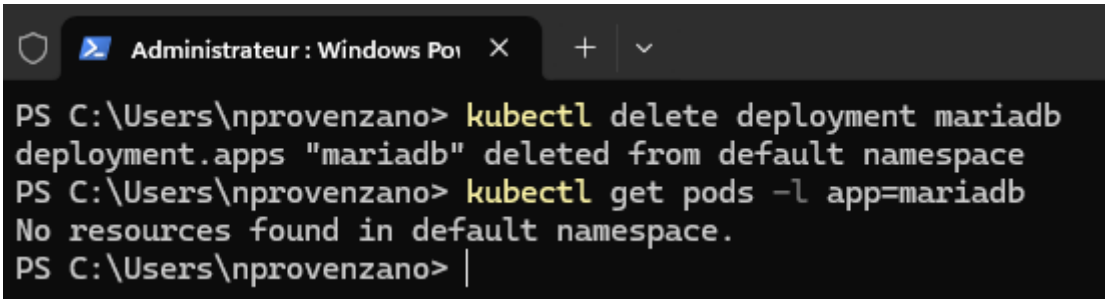
Les événements du pod indiquent un échec de la startupProbe empêchant le démarrage correct du second pod MariaDB.



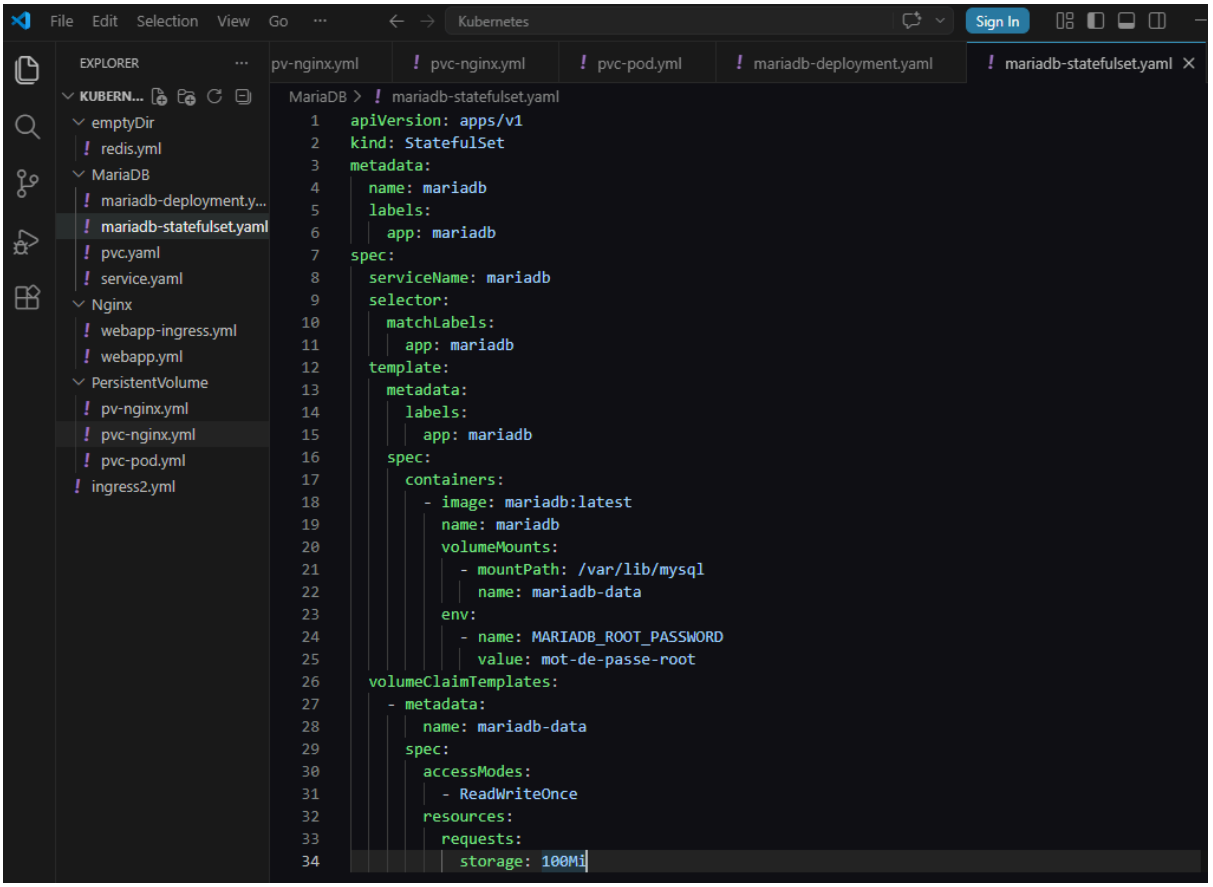
Les logs du second pod MariaDB indiquent un conflit d'accès au volume persistant déjà utilisé par un autre pod.



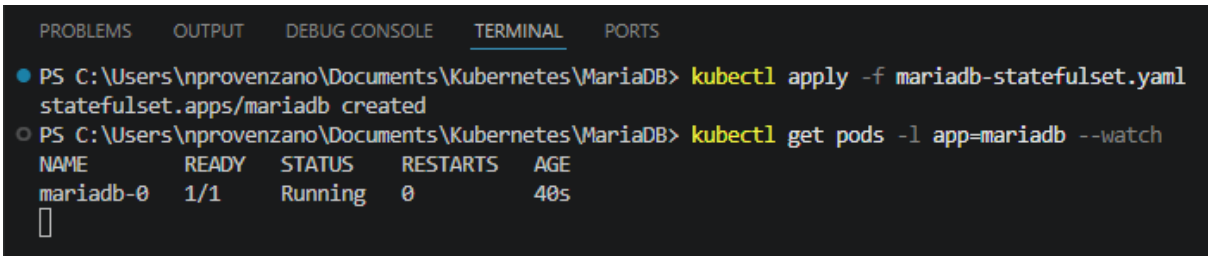
La suppression du déploiement MariaDB entraîne également la suppression des pods associés.



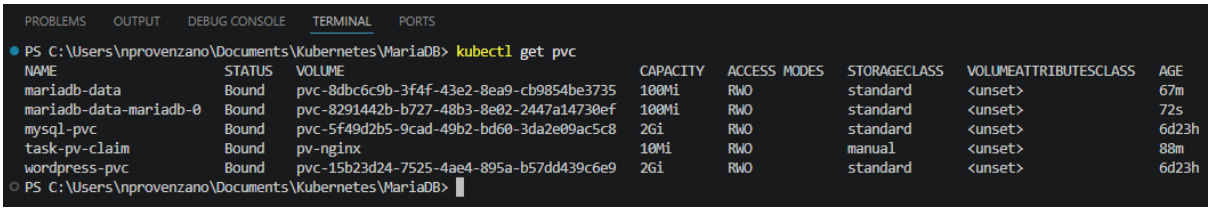
Le fichier **mariadb-statefulset.yaml** permet de créer un StatefulSet pour gérer les pods MariaDB avec un stockage persistant dédié.



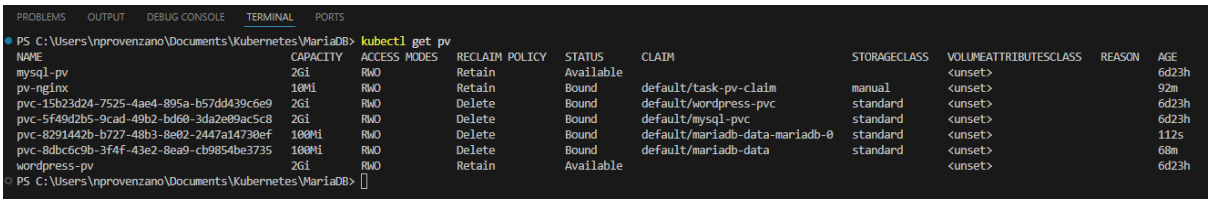
La commande **kubectl apply -f mariadb-statefulset.yaml** permet de créer le StatefulSet MariaDB. Le pod mariadb-0 démarre correctement avec le StatefulSet.



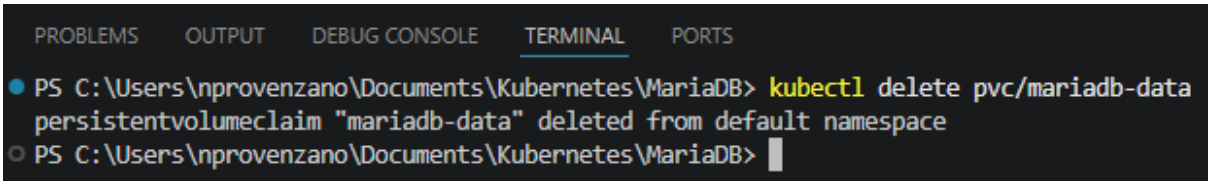
La commande **kubectl get pvc** permet d'afficher les PersistentVolumeClaims créés automatiquement par le StatefulSet MariaDB.



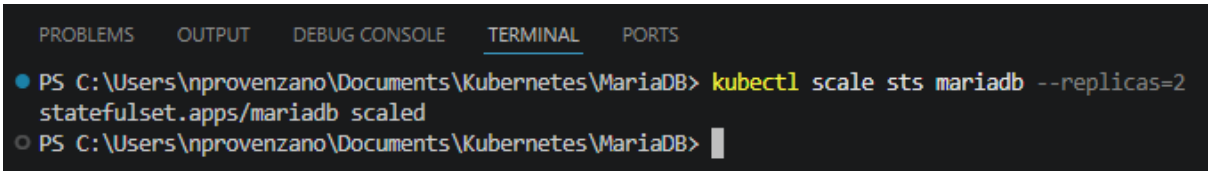
Le StatefulSet crée automatiquement un volume persistant dédié pour chaque pod MariaDB.



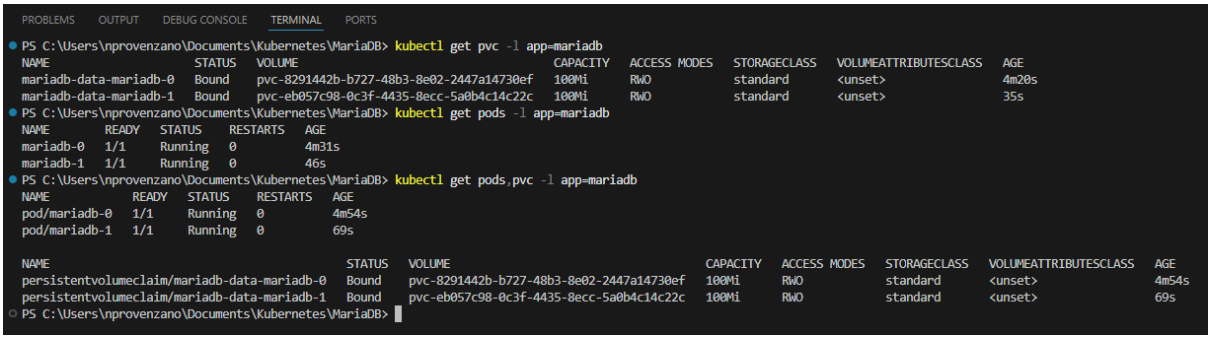
La commande **kubectl delete pvc/mariadb-data** permet de supprimer l'ancien PersistentVolumeClaim du déploiement MariaDB.



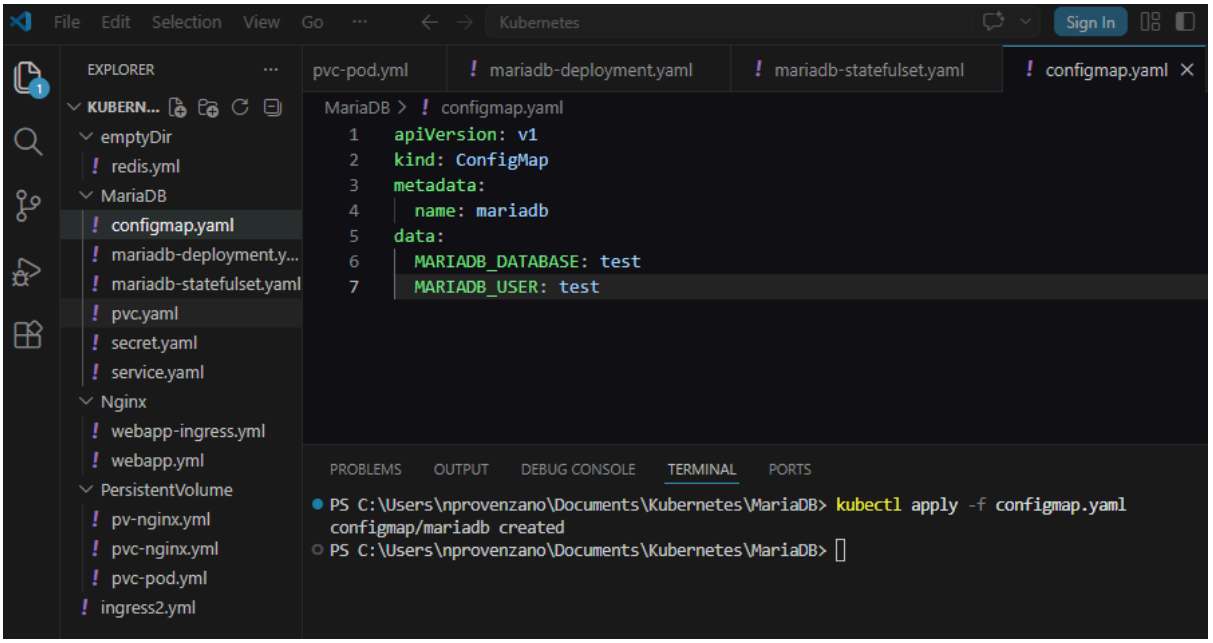
La commande **kubectl scale sts mariadb --replicas=2** permet d'augmenter le nombre de pods du StatefulSet MariaDB.



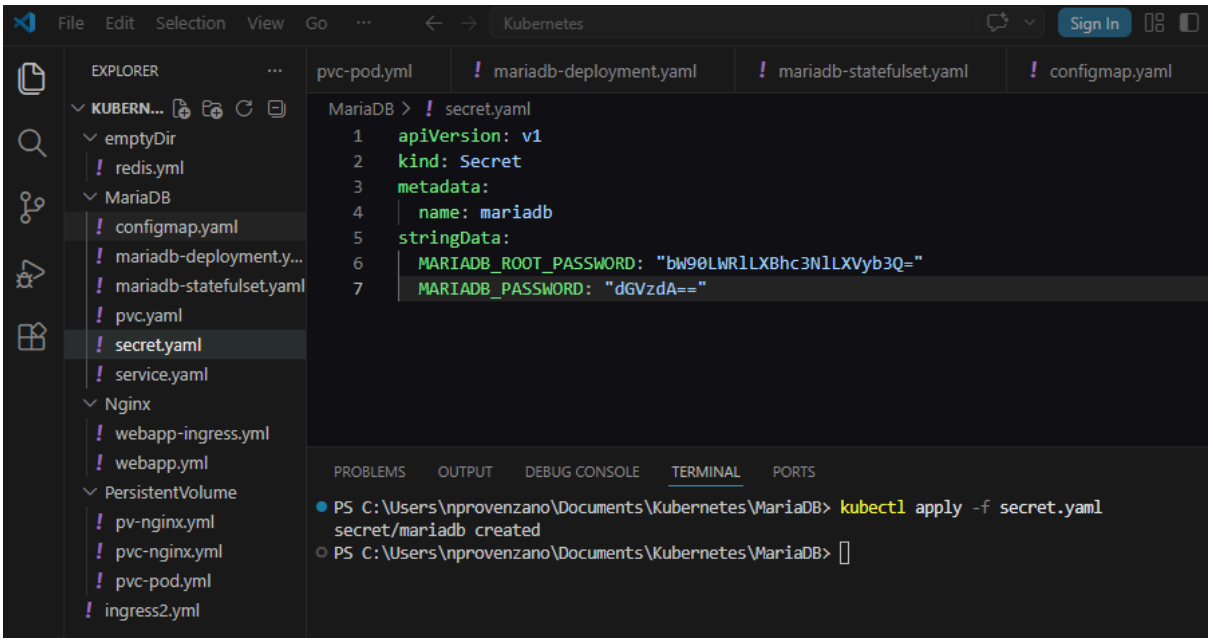
Les pods mariadb-0 et mariadb-1 sont en état Running avec leurs propres volumes persistants créés automatiquement par le StatefulSet.



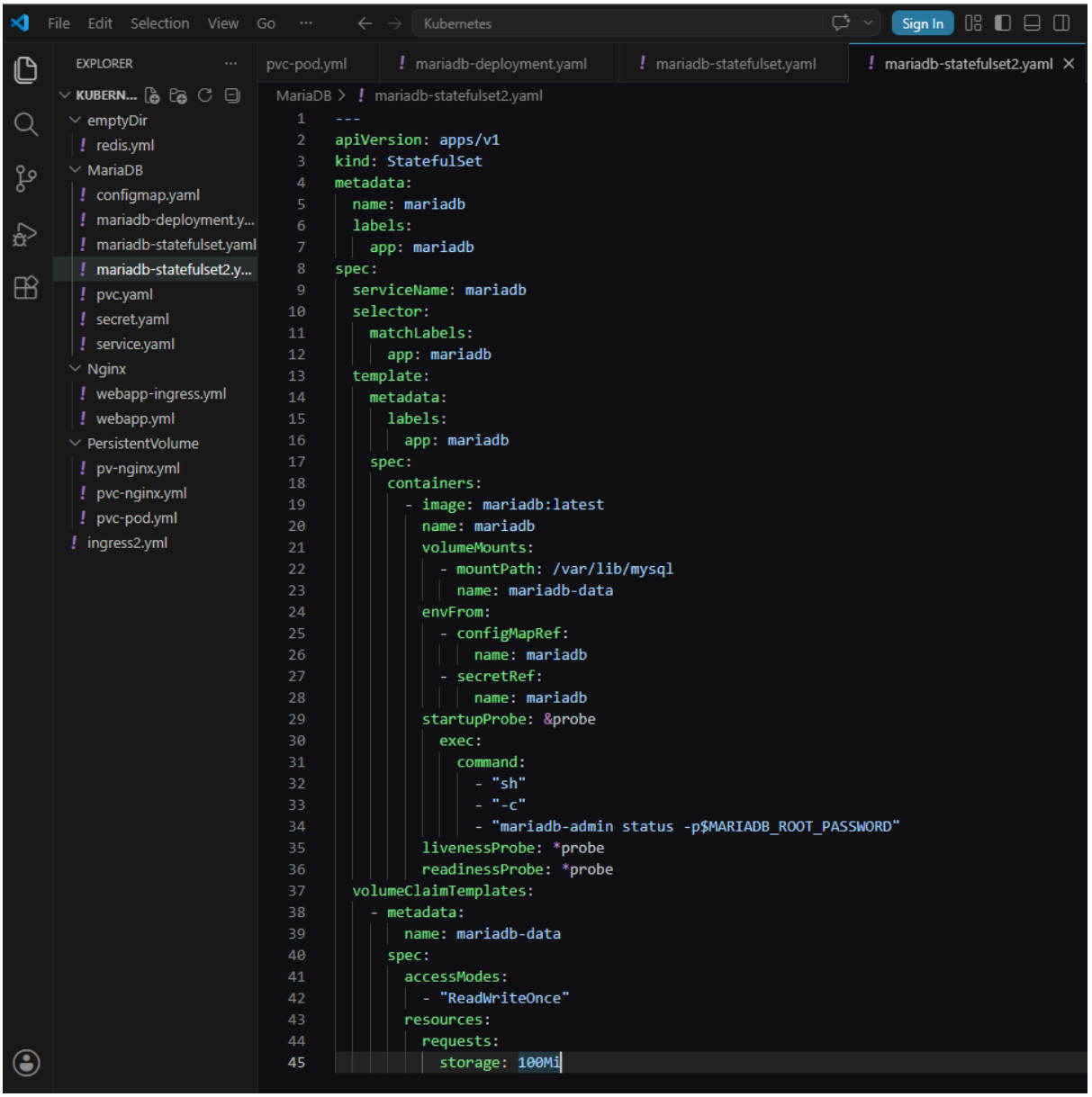
Le fichier **configmap.yaml** permet de stocker les variables de configuration utilisées par MariaDB.
La commande **kubectl apply -f configmap.yaml** permet de créer le ConfigMap mariadb.



Le fichier **secret.yaml** permet de stocker les mots de passe MariaDB de manière sécurisée.
La commande **kubectl apply -f secret.yaml** permet de créer le Secret mariadb.



La modification du fichier **mariadb-statefulset.yaml** permet d'utiliser les variables du ConfigMap et du Secret dans le conteneur MariaDB.



La commande **kubectl apply -f mariadb-statefulset2.yaml** permet de mettre à jour le StatefulSet MariaDB avec le ConfigMap et le Secret.