12 Van Podman naar Kubernetes

12.1 Inleiding

In dit hoofdstuk wordt stap voor stap uitgelegd hoe je een bestaande Podman-pod kunt omzetten naar een Kubernetes-manifest met behulp van podman generate kube.

We tonen niet alleen hoe je de YAML-bestanden kunt genereren, maar ook hoe je ze moet aanpassen zodat ze compatibel zijn met Kubernetes (met name k3d-clusters).

Daarnaast gaan we dieper in op veelvoorkomende problemen bij de omzetting en tonen we hoe je deze kunt oplossen.

We zullen in een later stadium ook gebruik maken van deployments.i.p.v. pods

We gebruiken als casestudy Wordpress.

12.2 Podman pods voor Wordpress

Eerst zullen we alle clusters van Kubernetes verwijderen die aanwezig zijn.

```
student@virt-k8s-XX:~$ kubectl config get-clusters

student@virt-k8s-XX:~$ k3d cluster delete [naam cluster]
```

Voer het laatste commando uit totdat alle clusters verdwenen zijn.

We zullen nu eerst Wordpress installeren via een podman pod zoals we reeds gezien hebben in hoofdstuk 8. Voor meer info hierover verwijs ik dan ook graag door naar hoofdstuk 8.

```
student@virt-k8s-XX:~$ podman pod create --name wp-pod -p
8080:80

student@virt-k8s-XX:~$ podman run -d --restart=always --pod=wp-
pod -e MYSQL_ROOT_PASSWORD="dbpass" -e MYSQL_DATABASE="wp" -
eMYSQL_USER="wordpress" -e MYSQL_PASSWORD="wppass" --name=wp-db
mariadb

student@virt-k8s-XX:~$ podman run -d --restart=always --pod=wp-
pod -e WORDPRESS_DB_NAME="wp" -e WORDPRESS_DB_USER="wordpress" -
e WORDPRESS_DB_PASSWORD="wppass" -e
WORDPRESS_DB_HOST="127.0.0.1" --name wp-web wordpress

student@virt-k8s-XX:~$ podman ps -a --pod

…
```

12.3 Genereren van een Kubernetes-manifest

We zetten nu het bestaande Podman-pod automatisch om naar een Kubernetes YAML-bestand.

```
student@virt-k8s-XX:~$ podman generate kube -s wp-pod >
wordpress.yaml
```

We bekijken nu de inhoud van dit bestand.

```
student@virt-k8s-XX:~$ cat wordpress.yaml

# Save the output of this file and use kubectl create -f to
import

# it into Kubernetes.

#

# Created with podman-5.4.0

apiVersion: v1

kind: Service

metadata:

  creationTimestamp: "2025-11-10T10:19:38Z"

  labels:

    app: wp-pod

  name: wp-pod

spec:

  ports:

  - name: "80"

    nodePort: 30389
```

```
    port: 80
    targetPort: 80
  selector:
    app: wp-pod
  type: NodePort
---
apiVersion: v1
kind: Pod
metadata:
  annotations:
    io.kubernetes.cri-o.SandboxID/wp-db:
3d823a7bbffbfff0b6f2ca346327dc86c580045eaee1a459a03e2bc72be7547f
    io.kubernetes.cri-o.SandboxID/wp-web:
3d823a7bbffbfff0b6f2ca346327dc86c580045eaee1a459a03e2bc72be7547f
  creationTimestamp: "2025-11-10T10:19:38Z"
  labels:
    app: wp-pod
  name: wp-pod
spec:
  containers:
  - args:
    - mariadbd
    env:
    - name: MYSQL_ROOT_PASSWORD
      value: dbpass
    - name: MYSQL_USER
      value: wordpress
```

```
    - name: MYSQL_DATABASE
      value: wp
    - name: MYSQL_PASSWORD
      value: wppass
    image: docker.io/library/mariadb:latest
    name: wp-db
    ports:
    - containerPort: 80
    volumeMounts:
    - mountPath: /var/lib/mysql
      name:
13db45d1db6fb6a94cd40dffca9c7b4f4aaeded0ebb91ad5d9a86316283e12e5
-pvc
  - args:
    - apache2-foreground
    env:
    - name: WORDPRESS_DB_USER
      value: wordpress
    - name: WORDPRESS_DB_PASSWORD
      value: wppass
    - name: WORDPRESS_DB_HOST
      value: 127.0.0.1
    - name: WORDPRESS_DB_NAME
      value: wp
    image: docker.io/library/wordpress:latest
    name: wp-web
    volumeMounts:
```

```
    - mountPath: /var/www/html

      name:
fe3e9358634f981ebcbb20597b3a11b0be33be227ed32671e8df2684e40d4788
-pvc

  volumes:

  - name:
13db45d1db6fb6a94cd40dffca9c7b4f4aaeded0ebb91ad5d9a86316283e12e5
-pvc

    persistentVolumeClaim:

      claimName:
13db45d1db6fb6a94cd40dffca9c7b4f4aaeded0ebb91ad5d9a86316283e12e5

  - name:
fe3e9358634f981ebcbb20597b3a11b0be33be227ed32671e8df2684e40d4788
-pvc

    persistentVolumeClaim:

      claimName:
fe3e9358634f981ebcbb20597b3a11b0be33be227ed32671e8df2684e40d4788
```

Er is output gegenereerd.

Als je de output analyseert zal je zien dat volgende is aangemaakt:

- Service met de naam wp-pod
- Pod met de naam wp-pod
    - Container met de naam wp-db
    - Container met de naam wp-web

12.4 Toepassen van manifestbestand

We verwijderen nu eerst de podman pod met de bijhorende containers.

```
student@virt-k8s-XX:~$ podman pod rm wp-pod -f

d2402d6d61aad6720c457526f05cbbcb8c29b8aad448c40d838b70fd673a148d

student@virt-k8s-XX:~$ podman ps --pod

CONTAINER ID  IMAGE       COMMAND     CREATED     STATUS …
```

Uiteraard is de pod nu verdwenen.

We maken nu een cluster aan met onderstaande instellingen.

Let op:
De poort 30689 in het Service-manifest moet overeenkomen met de poort die je in het k3d-cluster hebt doorgestuurd met de -p optie (8080:30689@server:0).
Dat betekent dat verkeer van poort 8080 op je lokale machine wordt doorgestuurd naar poort 30677 in de Kubernetes-service. Bij het genereren van het manifestbestand is een willekeurige nodePort gekozen!

```
student@virt-k8s-XX:~$ cat wordpress.yaml
```

…

```
  ports:
  - name: "80"
    nodePort: 30389
    port: 80
    targetPort: 80
```

…

```
student@virt-k8s-XX:~$ k3d cluster create mijncluster --servers
1 --agents 1 -p "8080:30389@server:0"
…
```

We trachten nu het Kubernetes-manifest toe te passen.

```
student@virt-k8s-XX:~$ kubectl apply -f wordpress.yaml

service/wp-pod created

The Pod "wp-pod" is invalid:

* spec.volumes[0].name: Invalid value:
"13db45d1db6fb6a94cd40dffca9c7b4f4aaeded0ebb91ad5d9a86316283e12e
5-pvc": must be no more than 63 characters

* spec.volumes[1].name: Invalid value:
"fe3e9358634f981ebcbb20597b3a11b0be33be227ed32671e8df2684e40d478
8-pvc": must be no more than 63 characters
```

```
* spec.containers[0].volumeMounts[0].name: Not found:
"13db45d1db6fb6a94cd40dffca9c7b4f4aaeded0ebb91ad5d9a86316283e12e
5-pvc"

* spec.containers[1].volumeMounts[0].name: Not found:
"fe3e9358634f981ebcbb20597b3a11b0be33be227ed32671e8df2684e40d478
8-pvc"
```

Jammer genoeg werkt het dus niet 'out of the box'.

12.5 Back-up maken en manifestbestand aanpassen

Ik raad nu aan om eerst een kopie te maken van het originele manifestbestand aangezien we wijzigingen zullen moeten aanbrengen om het werkend te krijgen.

```
student@virt-k8s-XX:~$ cp wordpress.yaml wordpress.yaml~
```

Een ~ na een bestandsnaam in Linux betekent meestal dat het gaat om een back-upbestand of tijdelijke kopie van een ander bestand.

We zullen nu het manifestbestand aanpassen zodat het wel werkt.

Allereerst: er zijn een aantal regels die je zonder problemen mag verwijderen uit dit bestand. Ik heb het dan over regels met creationTimestamp en annotations met io.kubernetes.cri-o.SandboxID/wp-db. Ook de regels die commentaar zijn (beginnend met #) mag je verwijderen. Dit is uiteraard nog geen oplossing voor het probleem maar zo maak je het bestand in ieder geval al beter leesbaar.

Verwijder daarom volgende regels:

```
# Save the output of this file and use kubectl create -f to
import

# it into Kubernetes.

#

# Created with podman-5.4.0

  creationTimestamp: "2025-11-10T10:19:38Z"

  annotations:

    io.kubernetes.cri-o.SandboxID/wp-db:
    3d823a7bbffbfff0b6f2ca346327dc86c580045eaee1a459a03e2bc72be7547
    f

    io.kubernetes.cri-o.SandboxID/wp-web:
    3d823a7bbffbfff0b6f2ca346327dc86c580045eaee1a459a03e2bc72be7547
    f
```

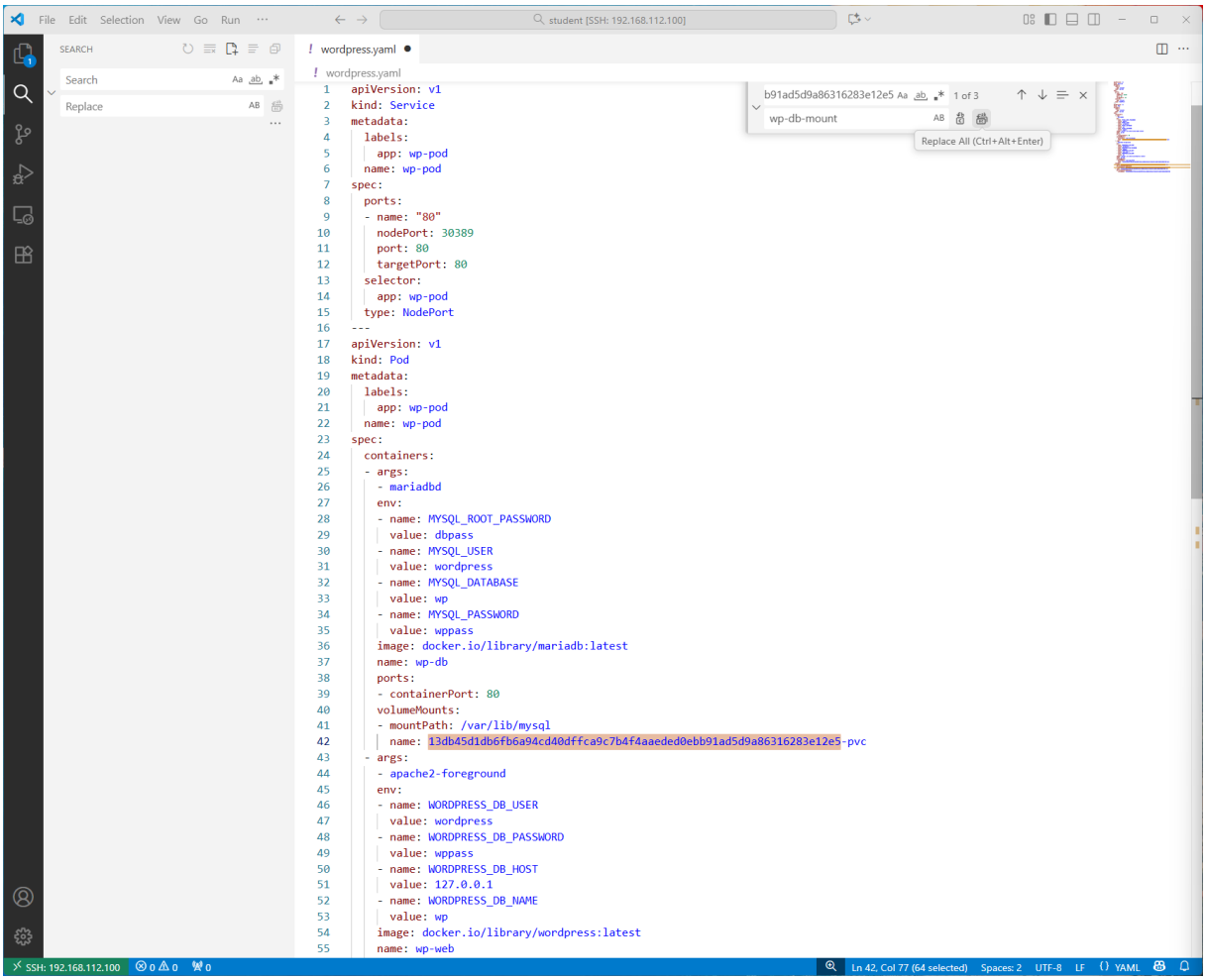
~~creationTimestamp: "2025-11-10T10:19:38Z"~~

Lange namen mogen niet gebruik worden in een manifestbestand. Kubernetes verwacht namen van maximaal 63 tekens. We lossen dit op.

```
    volumeMounts:
    - mountPath: /var/lib/mysql
      name: 13db45d1db6fb6a94cd40dffca9c7b4f4aaeded0ebb91ad5d9a86316283e12e5-pvc
  - args:
    - apache2-foreground
    env:
    - name: WORDPRESS_DB_USER
      value: wordpress
    - name: WORDPRESS_DB_PASSWORD
      value: wppass
    - name: WORDPRESS_DB_HOST
      value: 127.0.0.1
    - name: WORDPRESS_DB_NAME
      value: wp
    image: docker.io/library/wordpress:latest
    name: wp-web
    volumeMounts:
    - mountPath: /var/www/html
      name: fe3e9358634f981ebcbb20597b3a11b0be33be227ed32671e8df2684e40d4788-pvc
  volumes:
  - name: 13db45d1db6fb6a94cd40dffca9c7b4f4aaeded0ebb91ad5d9a86316283e12e5-pvc
    persistentVolumeClaim:
      claimName: 13db45d1db6fb6a94cd40dffca9c7b4f4aaeded0ebb91ad5d9a86316283e12e5
  - name: fe3e9358634f981ebcbb20597b3a11b0be33be227ed32671e8df2684e40d4788-pvc
    persistentVolumeClaim:
      claimName: fe3e9358634f981ebcbb20597b3a11b0be33be227ed32671e8df2684e40d4788
```

```
      name: fe3e9358634f981ebcbb20597b3a11b0be33be227ed32671e8df2684e40d4788-pvc
  volumes:
  - name: 13db45d1db6fb6a94cd40dffca9c7b4f4aaeded0ebb91ad5d9a86316283e12e5-pvc
    persistentVolumeClaim:
      claimName: 13db45d1db6fb6a94cd40dffca9c7b4f4aaeded0ebb91ad5d9a86316283e12e5
  - name: fe3e9358634f981ebcbb20597b3a11b0be33be227ed32671e8df2684e40d4788-pvc
    persistentVolumeClaim:
      claimName: fe3e9358634f981ebcbb20597b3a11b0be33be227ed32671e8df2684e40d4788
```

Ik vervang in mijn manifestbestand 13db45d1db6fb6a94cd40dffca9c7b4f4aaeded0ebb91ad5d9a86316283e12e5 door wp-db-mount en fe3e9358634f981ebcbb20597b3a11b0be33be227ed32671e8df2684e40d4788 door wp-web-mount.



Uiteindelijk krijg je onderstaand yaml-bestand.

```
student@virt-k8s-XX:~$ cat wordpress.yaml
apiVersion: v1
kind: Service
metadata:
  labels:
    app: wp-pod
  name: wp-pod
spec:
  ports:
  - name: "80"
    nodePort: 30389
```

```
    port: 80
    targetPort: 80
  selector:
    app: wp-pod
  type: NodePort
---
apiVersion: v1
kind: Pod
metadata:
  labels:
    app: wp-pod
  name: wp-pod
spec:
  containers:
  - args:
    - mariadbd
    env:
    - name: MYSQL_ROOT_PASSWORD
      value: dbpass
    - name: MYSQL_USER
      value: wordpress
    - name: MYSQL_DATABASE
      value: wp
    - name: MYSQL_PASSWORD
      value: wppass
    image: docker.io/library/mariadb:latest
```

```
    name: wp-db
    ports:
    - containerPort: 80
    volumeMounts:
    - mountPath: /var/lib/mysql
      name: wp-db-mount-pvc
  - args:
    - apache2-foreground
    env:
    - name: WORDPRESS_DB_USER
      value: wordpress
    - name: WORDPRESS_DB_PASSWORD
      value: wppass
    - name: WORDPRESS_DB_HOST
      value: 127.0.0.1
    - name: WORDPRESS_DB_NAME
      value: wp
    image: docker.io/library/wordpress:latest
    name: wp-web
    volumeMounts:
    - mountPath: /var/www/html
      name: wp-web-mount-pvc
  volumes:
  - name: wp-db-mount-pvc
    persistentVolumeClaim:
      claimName: wp-db-mount
```

```
    - name: wp-web-mount-pvc
      persistentVolumeClaim:
        claimName: wp-web-mount
```

Als je het grondig analyseert zal je constateren dat hierin een duidelijke fout staat…

```
…
name: wp-db
ports:
- containerPort: 80
…
```

De database-server luistert op poort 3306 zoals we reeds vaak hebben besproken.

We passen het manifestbestand dan ook als volgt aan.

```
…
name: wp-db
ports:
- containerPort: 3306
…
```

Bij de container wp-web zie dan weer geen poort. Voeg dat ook volgende regens toe onder name: wp-web.

```
  …
      ports:
      - containerPort: 80
  …
```

Nu heeft wordpress.yaml onderstaande inhoud.

```
apiVersion: v1
kind: Service
metadata:
    labels:
```

```
      app: wp-pod
    name: wp-pod
spec:
  ports:
  - name: "80"
    nodePort: 30389
    port: 80
    targetPort: 80
  selector:
    app: wp-pod
  type: NodePort
---
apiVersion: v1
kind: Pod
metadata:
  labels:
    app: wp-pod
  name: wp-pod
spec:
  containers:
  - args:
    - mariadbd
    env:
    - name: MYSQL_PASSWORD
      value: wppass
    - name: MYSQL_DATABASE
```

```
      value: wp
    - name: MYSQL_USER
      value: wordpress
    - name: MYSQL_ROOT_PASSWORD
      value: dbpass
    image: docker.io/library/mariadb:latest
    name: wp-db
    ports:
    - containerPort: 3306
    volumeMounts:
    - mountPath: /var/lib/mysql
      name: wp-db-mount-pvc
  - args:
    - apache2-foreground
    env:
    - name: WORDPRESS_DB_PASSWORD
      value: wppass
    - name: WORDPRESS_DB_HOST
      value: 127.0.0.1
    - name: WORDPRESS_DB_NAME
      value: wp
    - name: WORDPRESS_DB_USER
      value: wordpress
    image: docker.io/library/wordpress:latest
    name: wp-web
    ports:
```

```
    - containerPort: 80
    volumeMounts:
    - mountPath: /var/www/html
      name: wp-web-mount-pvc
  volumes:
  - name: wp-web-mount-pvc
    persistentVolumeClaim:
      claimName: wp-web-mount
  - name: wp-db-mount-pvc
    persistentVolumeClaim:
      claimName: wp-db-mount
```

12.6 PVC-manifest aanmaken

Zoals je reeds geleerd hebt moet je ook een PVC-manifest aanmaken. Dit moeten we dus nog zelf aanmaken.

```
student@virt-k8s-XX:~$ cat opslag.yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: wp-db-mount
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
---
```

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: wp-web-mount
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
```

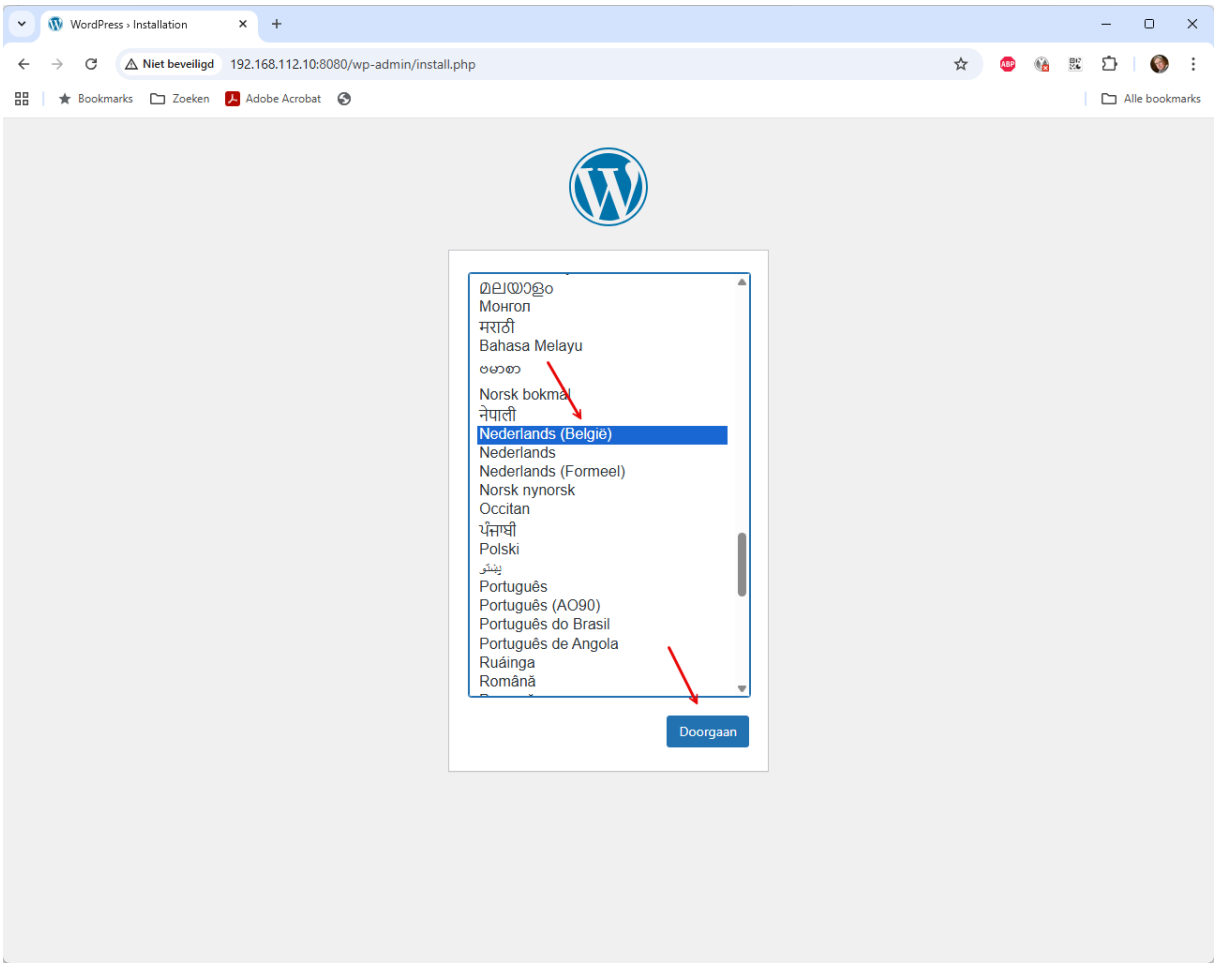
12.7 Toepassen van manifestbestanden

Voer nu onderstaande uit.

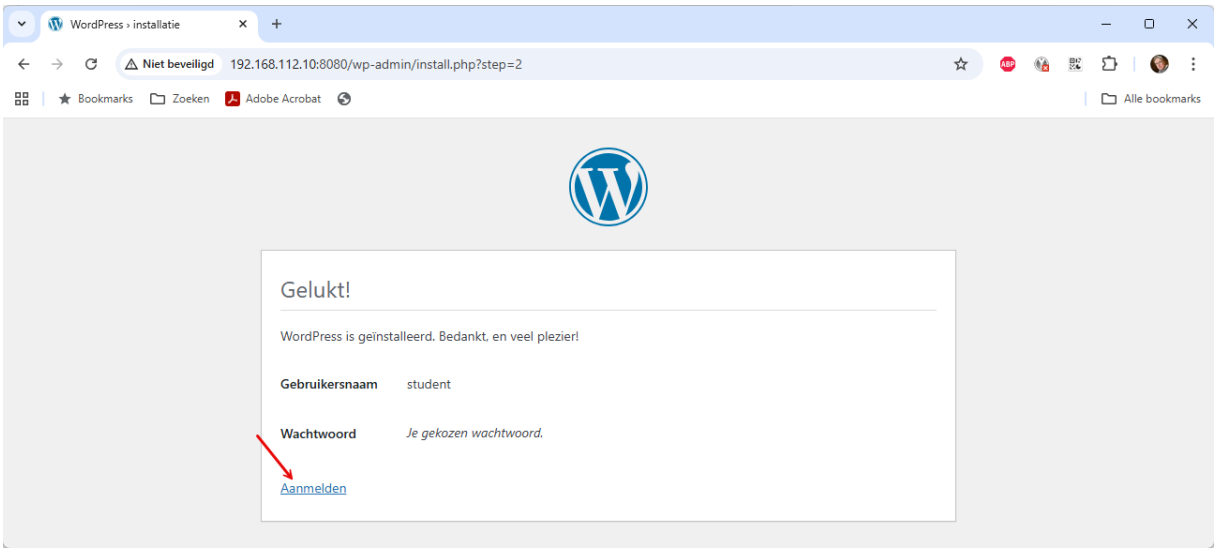
```
student@virt-k8s-XX:~$ kubectl apply -f opslag.yaml
persistentvolumeclaim/wp-db-mount created
persistentvolumeclaim/wp-web-mount created
student@virt-k8s-XX:~$ kubectl apply -f wordpress.yaml
service/wp-pod created
pod/wp-pod created
```

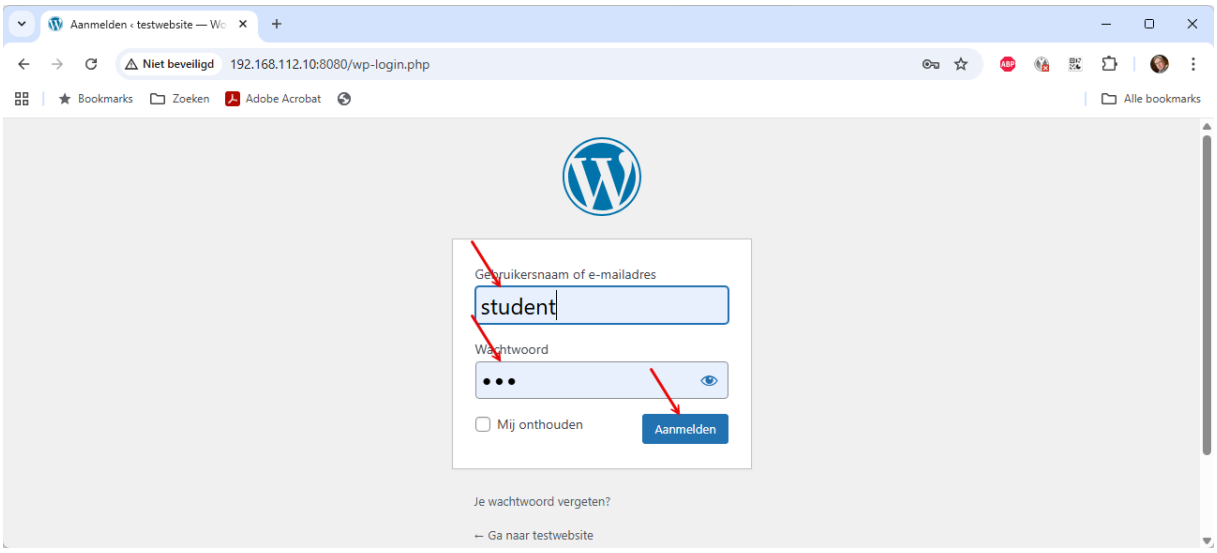
12.8 Instellen Wordpress

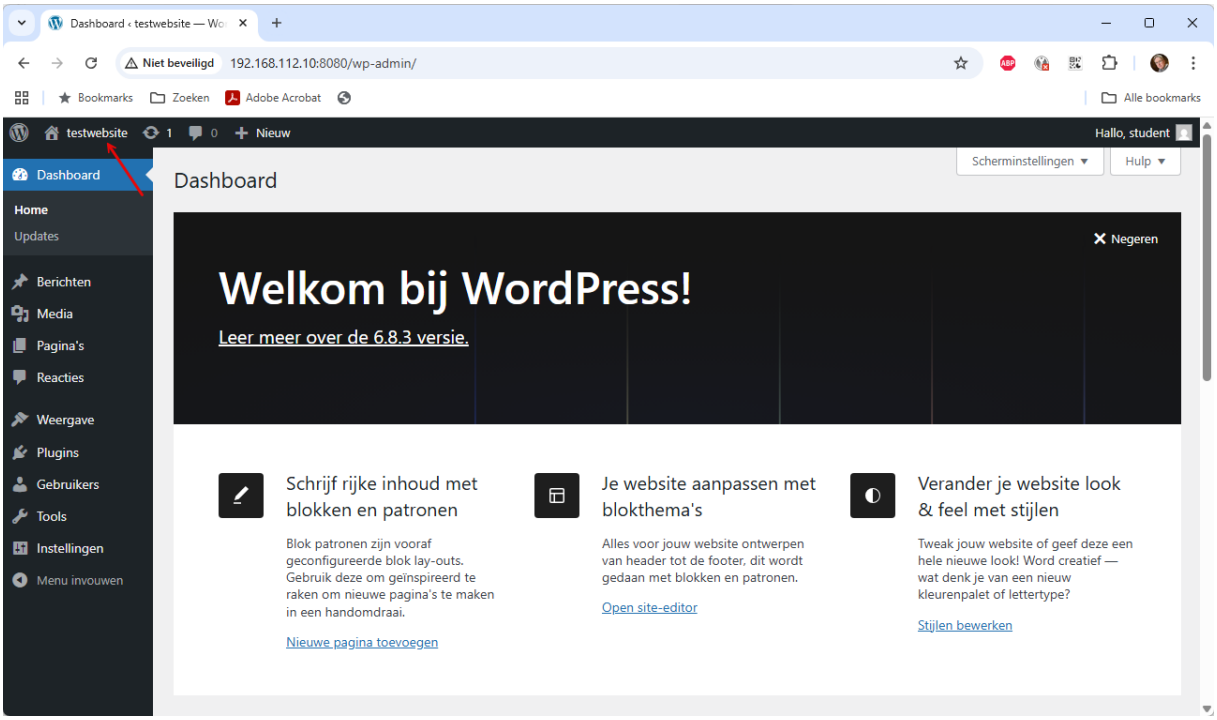
Je kan nu op een andere computer op het 192.168.112.0-netwerk onderstaande uitvoeren.

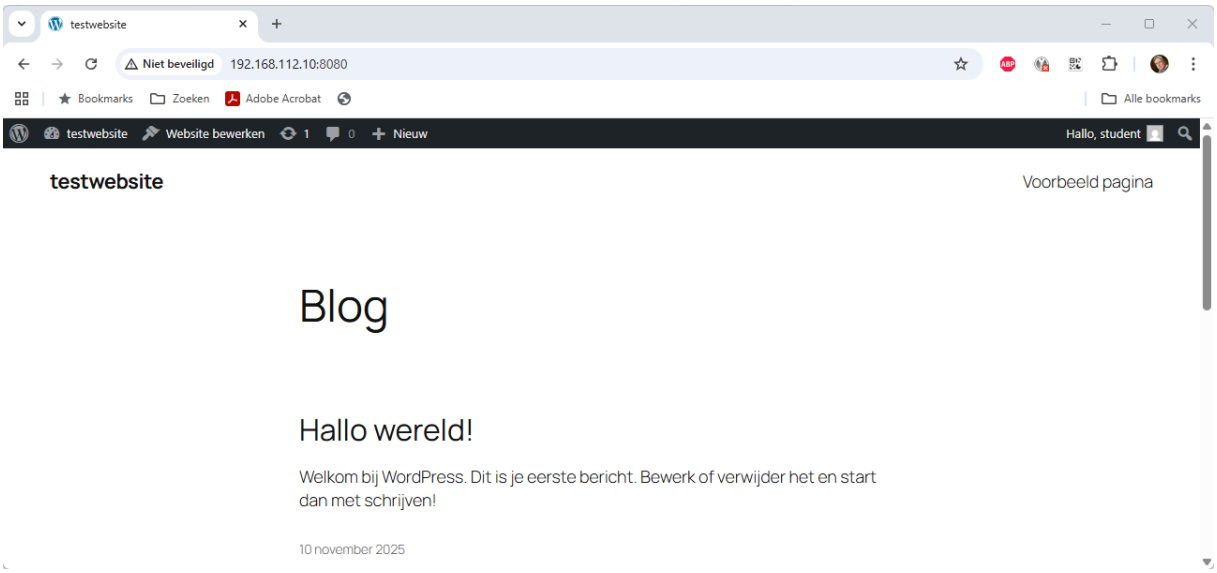

WordPress › Installation
Niet beveiligd
192.168.112.10:8080/wp-admin/install.php
Bookmarks
Zoeken
Adobe Acrobat
Alle bookmarks
മലയാളം
Монгол
मराठी
Bahasa Melayu
ဗမာစာ
Norsk bokmål
नेपाली
Nederlands (België)
Nederlands
Nederlands (Formeel)
Norsk nynorsk
Occitan
ਪੰਜਾਬੀ
Polski
پښتو
Português
Português (AO90)
Português do Brasil
Português de Angola
Ruáinga
Română
Doorgaan


Welkom
Welkom bij het bekende vijf minuten installatieproces van WordPress! Vul gewoon de informatie hieronder in en je bent klaar om het meeste krachtige en uitbreidbare publicatieplatform van de wereld te gebruiken.
Benodigde informatie
De volgende informatie invoeren. Maak je geen zorgen, deze instellingen kunnen steeds worden gewijzigd.
Website titel
testwebsite
Gebruikersnaam
student
Gebruikersnamen mogen alleen alfanumerieke karakters, spaties, underscores, koppeltekens, punten en het @ symbool bevatten.
Wachtwoord
123
Verbergen
Erg zwak
Belangrijk: Dit wachtwoord is nodig om in te loggen. Zorg ervoor dat je het bewaart op een geheime locatie.
Bevestig wachtwoord
Bevestig het gebruik van een zwak wachtwoord
Je e-mailadres
stijn.jacobs@pxl.be
Controleer zorgvuldig of je het e-mailadres goed hebt ingevuld voordat je verder gaat.
Zoekmachine zichtbaarheid
Blokkeer zoekmachines deze website te indexeren
Het is aan de zoekmachines of ze gehoor geven aan dit verzoek.
WordPress installeren


WordPress › installatie
Niet beveiligd
192.168.112.10:8080/wp-admin/install.php?step=2
Bookmarks
Zoeken
Adobe Acrobat
Alle bookmarks
Gelukt!
WordPress is geïnstalleerd. Bedankt, en veel plezier!
Gebruikersnaam
student
Wachtwoord
Je gekozen wachtwoord.
Aanmelden


Aanmelden ‹ testwebsite — Wo
Niet beveiligd
192.168.112.10:8080/wp-login.php
Bookmarks
Zoeken
Adobe Acrobat
Alle bookmarks
Gebruikersnaam of e-mailadres
student
Wachtwoord
Mij onthouden
Aanmelden
Je wachtwoord vergeten?
← Ga naar testwebsite


Dashboard ‹ testwebsite — Wo
Niet beveiligd
192.168.112.10:8080/wp-admin/
Bookmarks
Zoeken
Adobe Acrobat
Alle bookmarks
testwebsite
1
0
Nieuw
Hallo, student
Dashboard
Home
Updates
Berichten
Media
Pagina's
Reacties
Weergave
Plugins
Gebruikers
Tools
Instellingen
Menu invouwen
Scherminstellingen
Hulp
Dashboard
Negeren
Welkom bij WordPress!
Leer meer over de 6.8.3 versie.
Schrijf rijke inhoud met blokken en patronen
Blok patronen zijn vooraf geconfigureerde blok lay-outs. Gebruik deze om geïnspireerd te raken om nieuwe pagina's te maken in een handomdraai.
Nieuwe pagina toevoegen
Je website aanpassen met blokthema's
Alles voor jouw website ontwerpen van header tot de footer, dit wordt gedaan met blokken en patronen.
Open site-editor
Verander je website look & feel met stijlen
Tweak jouw website of geef deze een hele nieuwe look! Word creatief — wat denk je van een nieuw kleurenpalet of lettertype?
Stijlen bewerken



12.9 Check persistant storage in cluster

We vragen de service, pods en pvc op.

```
student@virt-k8s-XX:~$ kubectl get service
NAME         TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)
AGE
kubernetes   ClusterIP   10.43.0.1      <none>        443/TCP        26m
wp-pod       NodePort    10.43.53.246   <none>        80:30389/TCP   25m
```

PS De service kubernetes wordt automatisch door de cluster zelf aangemaakt. Het is de interne service die verwijst naar de API-server van je Kubernetes-cluster.

```
student@virt-k8s-XX:~$ kubectl get pods
NAME     READY   STATUS    RESTARTS   AGE
wp-pod   2/2     Running   0          19m
```

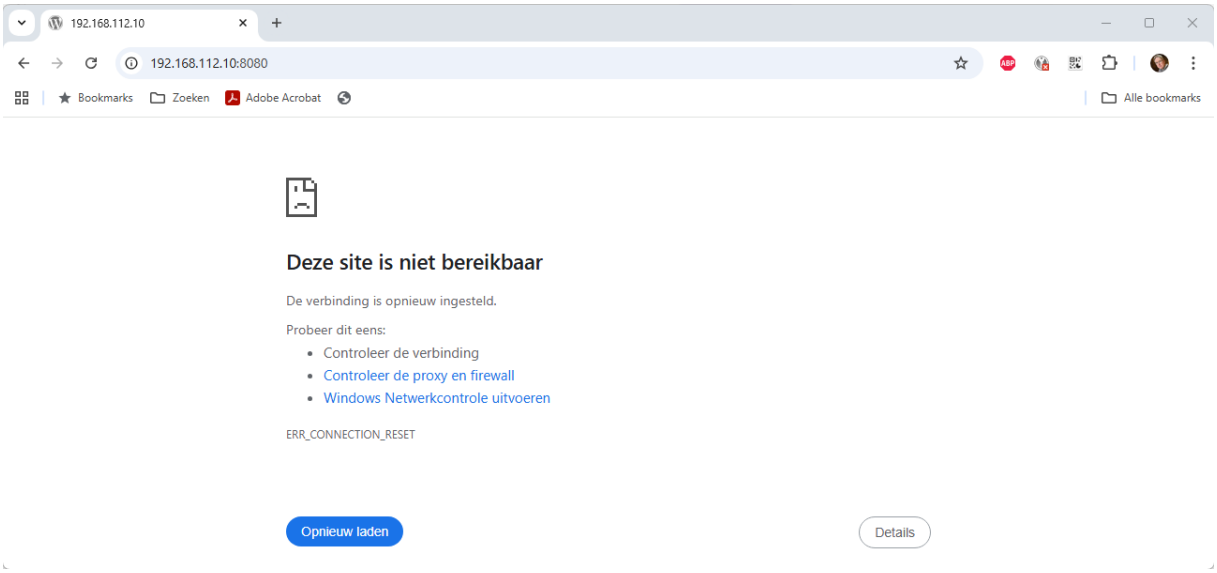
Je ziet 2/2 omdat wp-pod 2 containers bevat.

```
student@virt-k8s-XX:~$ kubectl get pvc
NAME           STATUS   VOLUME                                     …
wp-db-mount    Bound    pvc-aa4a6e08-0aa5-40ca-9123-ffd3fe5aa8f0   …
wp-web-mount   Bound    pvc-bb7b9a94-72bb-4ee9-b40a-740681eeb613   …
```

Om te checken of de gegevens in de cluster zijn opgeslagen verwijderen we de pods en de service. We verwijderen hiervoor de service en de pod.

```
student@virt-k8s-XX:~$ kubectl delete service wp-pod
service "wp-pod" deleted
student@virt-k8s-XX:~$ kubectl delete pod wp-pod
pod "wp-pod" deleted
```

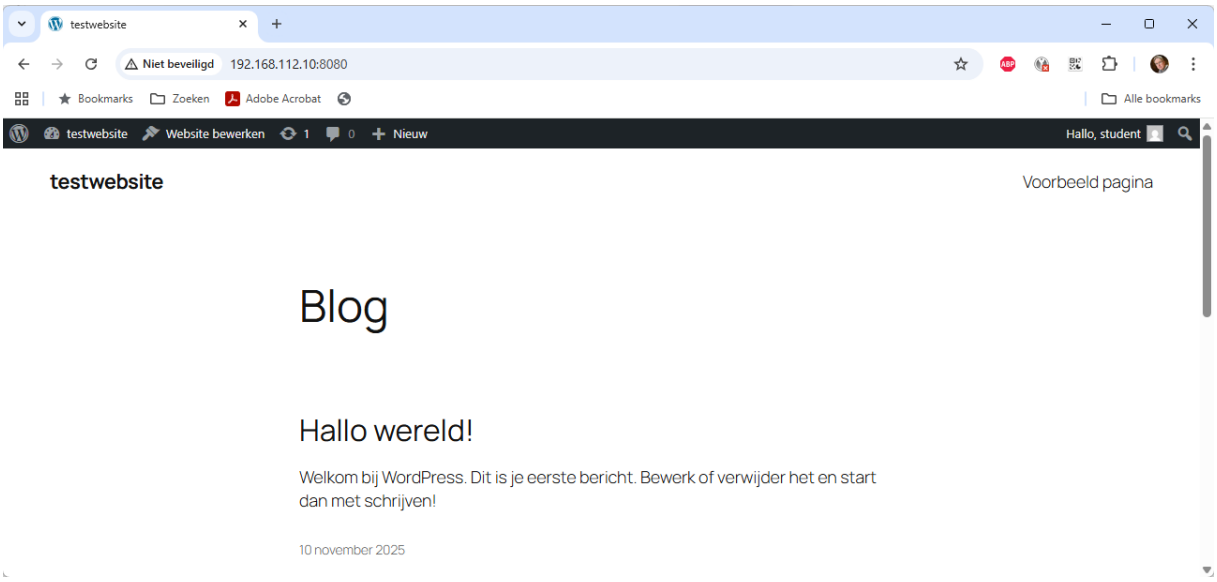
Uiteraard is nu de website niet meer beschikbaar.



We maken de service en de pod nu terug aan aan de hand van het manifestbestand.

```
student@virt-k8s-XX:~$ kubectl apply -f wordpress.yaml
service/wp-pod created
pod/wp-pod created
```

Als je nu de webpagina terug opvraagt zal je zien dat deze terug beschikbaar is.



12.10 Meerdere manifestbestanden

Grotere YAML-bestanden, zoals het bestand dat we net gebruikten (*wordpress.yaml*), worden al snel onoverzichtelijk. Voor schaalbaarheid is het bovendien beter om verschillende YAML-bestanden te gebruiken — bijvoorbeeld één per service of component.

We maken eerst een map aan en kopiëren de inhoud van wordpress.yaml ernaartoe.

```
student@virt-k8s-XX:~$ mkdir wordpress

student@virt-k8s-XX:~$ cp wordpress.yaml wordpress

student@virt-k8s-XX:~$ cd wordpress
```

We doen dit als volgt. Alles boven --- hoort bij service. Verwijder dus alles onder --- uit servicewp.yaml die je hieronder aanmaakt.

```
student@virt-k8s-XX:~/wordpress$ cp wordpress.yaml
servicewp.yaml

student@virt-k8s-XX:~/wordpress$ nano servicewp.yaml

apiVersion: v1

kind: Service

metadata:

    labels:
```

```
      app: wp-pod
    name: wp-pod
spec:
  ports:
  - name: "80"
    nodePort: 30389
    port: 80
    targetPort: 80
  selector:
    app: wp-pod
  type: NodePort
```

Alles onder --- hoort bij pod. Verwijder in podwp.yaml die je hieronder aanmaakt aldus alles boven ---.

```
student@virt-k8s-XX:~/wordpress$ cp wordpress.yaml podwp.yaml
student@virt-k8s-XX:~/wordpress$ nano podwp.yaml
student@virt-k8s-XX:~/wordpress$ cat podwp.yaml
apiVersion: v1
kind: Pod
metadata:
  labels:
    app: wp-pod
  name: wp-pod
spec:
  containers:
  - args:
    - mariadbd
    env:
```

```
    - name: MYSQL_PASSWORD
      value: wppass
    - name: MYSQL_DATABASE
      value: wp
    - name: MYSQL_USER
      value: wordpress
    - name: MYSQL_ROOT_PASSWORD
      value: dbpass
    image: docker.io/library/mariadb:latest
    name: wp-db
    ports:
    - containerPort: 3306
    volumeMounts:
    - mountPath: /var/lib/mysql
      name: wp-db-mount-pvc
  - args:
    - apache2-foreground
    env:
    - name: WORDPRESS_DB_PASSWORD
      value: wppass
    - name: WORDPRESS_DB_HOST
      value: 127.0.0.1
    - name: WORDPRESS_DB_NAME
      value: wp
    - name: WORDPRESS_DB_USER
      value: wordpress
```

```
    image: docker.io/library/wordpress:latest
    name: wp-web
    ports:
    - containerPort: 80
    volumeMounts:
    - mountPath: /var/www/html
      name: wp-web-mount-pvc
  volumes:
  - name: wp-web-mount-pvc
    persistentVolumeClaim:
      claimName: wp-web-mount
  - name: wp-db-mount-pvc
    persistentVolumeClaim:
      claimName: wp-db-mount
```

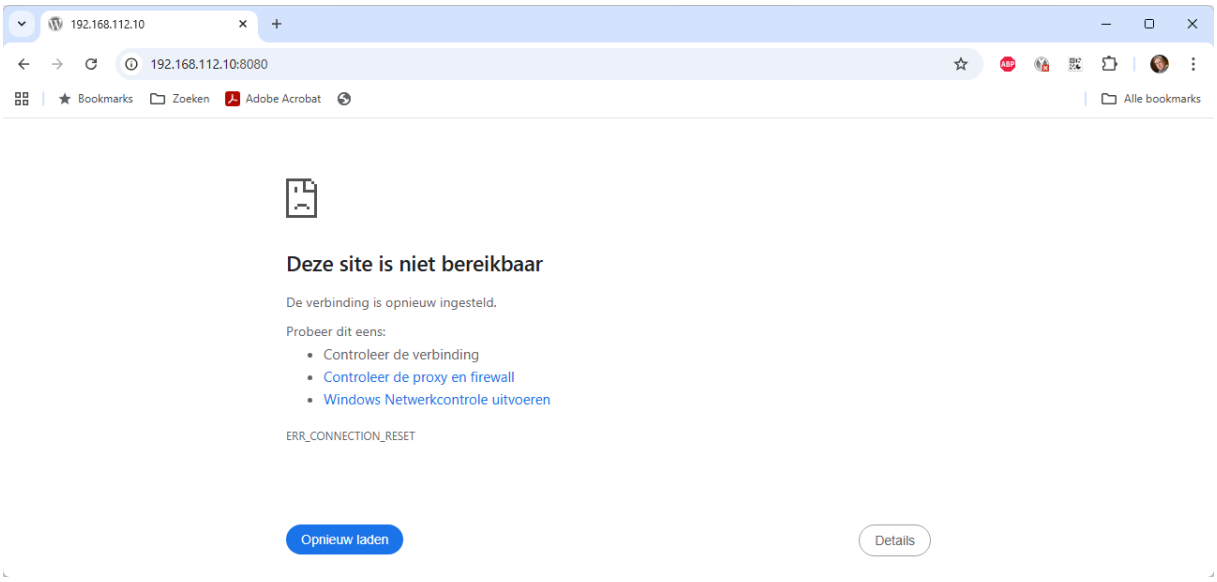
We kunnen nu uiteraard wordpress.yaml verwijderen aangezien we die niet meer nodig hebben.

```
student@virt-k8s-XX:~/wordpress$ rm wordpress.yaml
```

Het is een goed idee om te checken of nu met deze 2 maniefestbestand hetzelfde resultaat bekomt.

```
student@virt-k8s-XX:~/wordpress$ kubectl delete pod wp-pod
pod "wp-pod" deleted
student@virt-k8s-XX:~/wordpress$ kubectl delete service wp-pod
service "wp-pod" deleted
```

Uiteraard is de website nu niet meer beschikbaar.



We proberen nu de website op te zetten aan de hand van de 2 juist aangemaakte manifestbestanden.

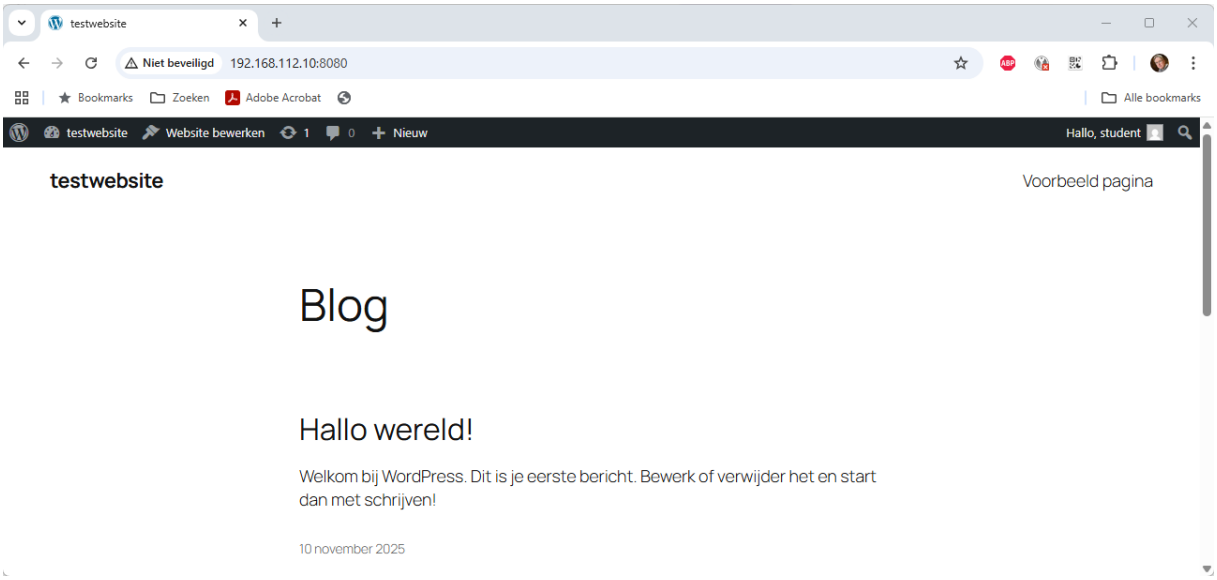
```
student@virt-k8s-XX:~/wordpress$ kubectl apply -f servicewp.yaml

service/wp-pod created

student@virt-k8s-XX:~/wordpress$ kubectl apply -f podwp.yaml

pod/wp-pod created
```

We zullen constateren dat de website nu beschikbaar is.



We kunnen podwp.yaml beter ook nog in 2 splitsen door de database (MariaDB) en WordPress webserver elk in hun eigen Pod te zetten. Eén service per pod wordt om volgende redenen aanbevolen:

- Losse componenten (DB ↔ web) kun je onafhankelijk herstarten of schalen.
- Kleiner manifest per functie = beter beheer.
- Kubernetes best practice: één hoofdcontainer per Pod (niet twee containers samenzetten).

student@virt-k8s-XX:~/wordpress$ cp podwp.yaml podwordpress.yaml

student@virt-k8s-XX:~/wordpress$ cp podwp.yaml poddatabase.yaml

Pas poddatabase.yaml nu aan zodat het moet onderstaande overeenkomt.

```
student@virt-k8s-XX:~/wordpress$ nano poddatabase.yaml
```

```
apiVersion: v1
kind: Pod
metadata:
  labels:
    app: database-pod
  name: database-pod
spec:
  containers:
  - args:
    - mariadbd
    env:
    - name: MYSQL_PASSWORD
      value: wppass
    - name: MYSQL_DATABASE
      value: wp
    - name: MYSQL_USER
      value: wordpress
```

```
    - name: MYSQL_ROOT_PASSWORD
      value: dbpass
    image: docker.io/library/mariadb:latest
    name: wp-db
    ports:
    - containerPort: 3306
    volumeMounts:
    - mountPath: /var/lib/mysql
      name: wp-db-mount-pvc
  volumes:
  - name: wp-db-mount-pvc
    persistentVolumeClaim:
      claimName: wp-db-mount
```

Pas podwordpress.yaml nu aan zodat het moet onderstaande overeenkomt.

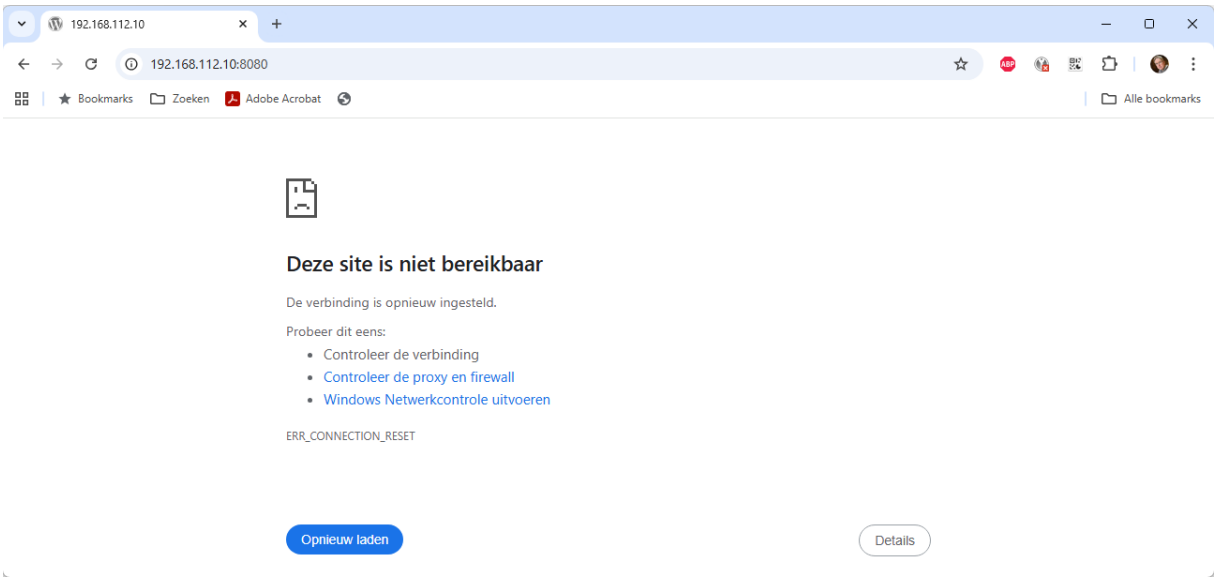
```
student@virt-k8s-XX:~/wordpress$ nano podwordpress.yaml
apiVersion: v1
kind: Pod
metadata:
  labels:
    app: wp-pod
  name: wordpress-pod
spec:
  containers:
  - args:
    - apache2-foreground
    env:
```

```
    - name: WORDPRESS_DB_PASSWORD
      value: wppass
    - name: WORDPRESS_DB_HOST
      value: database-service
    - name: WORDPRESS_DB_NAME
      value: wp
    - name: WORDPRESS_DB_USER
      value: wordpress
    image: docker.io/library/wordpress:latest
    name: wp-web
    ports:
    - containerPort: 80
    volumeMounts:
    - mountPath: /var/www/html
      name: wp-web-mount-pvc
  volumes:
  - name: wp-web-mount-pvc
    persistentVolumeClaim:
      claimName: wp-web-mount
```

We verwijderen nu de pod met de 2 containers, genaamd wp-pod.

```
student@virt-k8s-XX:~/wordpress$ kubectl delete pod wp-pod
pod "wp-pod" deleted
```

Uiteraard is de website nu niet meer beschikbaar.



De service draait nog. We checken dit.

```
student@virt-k8s-XX:~/wordpress$ kubectl get service wp-pod

NAME      TYPE        CLUSTER-IP      EXTERNAL-IP    …

wp-pod    NodePort    10.43.109.19    <none>         …
```

We starten nu de 2 pods met telkens 1 container.

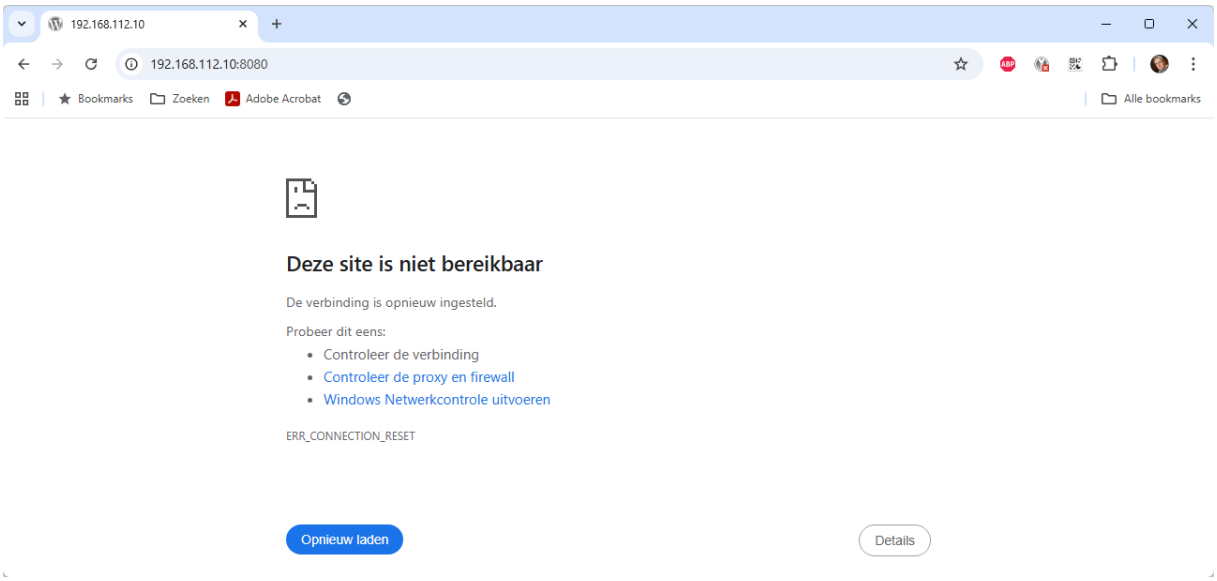
```
student@virt-k8s-XX:~/wordpress$ kubectl apply -f
poddatabase.yaml

pod/database-pod created

student@virt-k8s-XX:~/wordpress$ kubectl apply -f
podwordpress.yaml

pod/wordpress-pod created
```

Als je nu de website opvraagt krijg je onderstaande fout…



Er kan geen verbinding tot stand gebracht worden omdat servicewp.yaml wijst naar wp-pod en niet naar wordpress-pod.

```
student@virt-k8s-XX:~/wordpress$ cp servicewp.yaml
servicewordpress.yaml
```

Zorg dat servicewordpress.yaml onderstaande inhoud krijgt.

```
student@virt-k8s-XX:~/wordpress$ nano servicewordpress.yaml

apiVersion: v1

kind: Service

metadata:

    labels:

      app: wp

    name: wordpress-service

spec:

  ports:

  - name: "80"

    nodePort: 30389

    port: 80
```

```
    targetPort: 80
  selector:
    app: wordpress-pod
  type: NodePort
```

Stop de service wp-pod en pas de nieuwe service toe aan de hand van het manifestbestand.

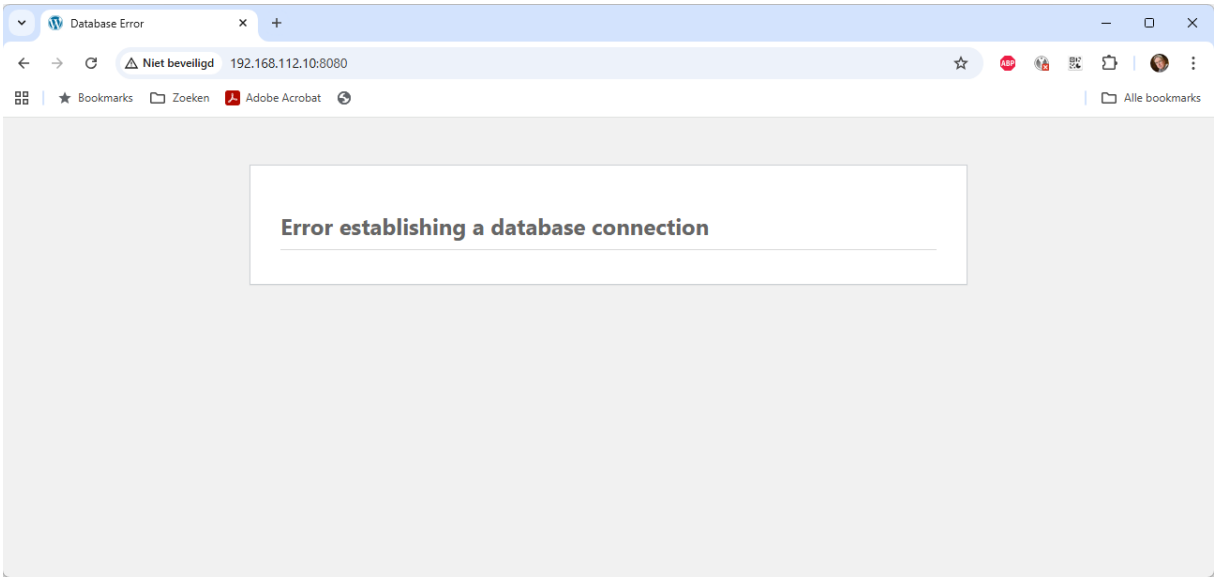
```
student@virt-k8s-XX:~/wordpress$ kubectl delete service wp-pod

service "wp-pod" deleted

student@virt-k8s-XX:~/wordpress$ kubectl apply -f
servicewordpress.yaml

service/wordpress-service created
```

Als we nu naar de website gaan krijg je onderstaande foutmelding.



Dit komt omdat pods enkel met elkaar kunnen praten via IP-adressen, maar die zijn veranderlijk. Dit hebben we reeds besproken in vorig hoofdstuk.

We moeten dus een service bijmaken zodat de database-service bereikbaar wordt vanuit Wordpress-pod.

```
student@virt-k8s-XX:~/wordpress$ nano servicedatabase.yaml

apiVersion: v1
```

```
kind: Service
metadata:
  labels:
      app: wp
  name: database-service
spec:
  selector:
    app: database-pod
  ports:
    - port: 3306
      targetPort: 3306
```

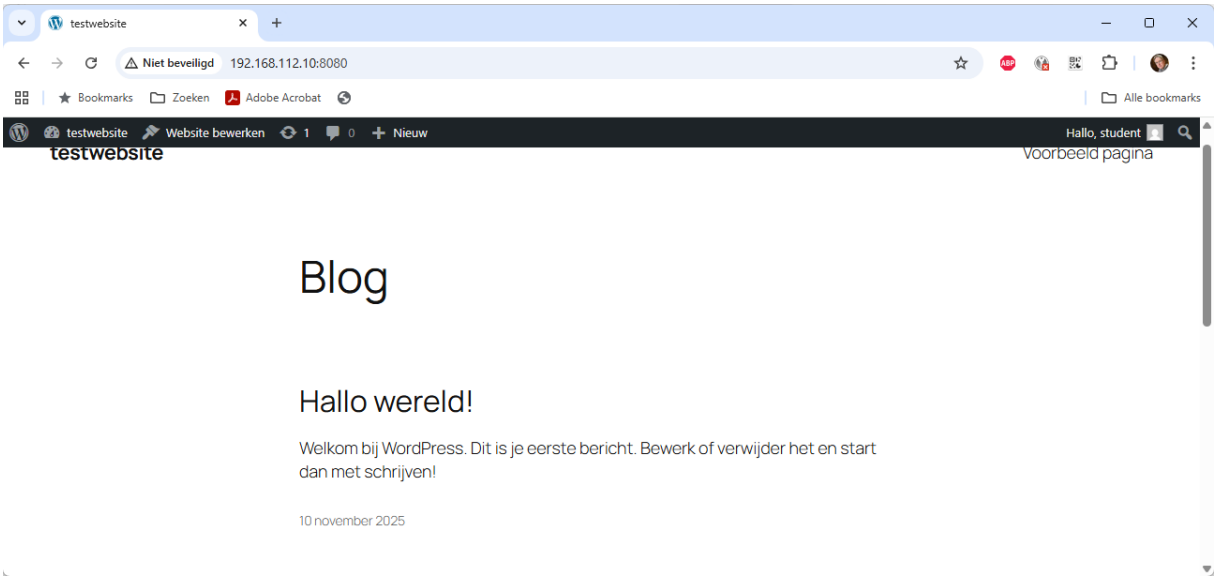
“TargetPort: 3306” is optioneel. Dit is de poort in de Pod waar verkeer naartoe wordt gestuurd Als je targetPort weglaat, gebruikt Kubernetes automatisch dezelfde waarde als 'port'

We starten nu de service aan de hand van het manifestbestand.

```
student@virt-k8s-XX:~/wordpress$ kubectl apply -f
servicedatabase.yaml

service/database-service created
```

De website is nu opnieuw beschikbaar!



12.11 Deployments

In plaats van pods kun je beter Deployments gebruiken.

Deployments zorgen ervoor dat pods automatisch herstarten als ze crashen, dat ze eenvoudig geschaald kunnen worden en dat updates gecontroleerd uitgerold kunnen worden. Voor onze WordPress-opstelling betekent dit dat zowel de MariaDB-pod als de WordPress-webserver in een Deployment geplaatst worden.

Het gebruik van deployments is trouwens best-practice…

Aan de hand van de pods maken we volgende deployments aan.

```
student@virt-k8s-XX:~/wordpress$ nano deployment-database.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: database-deployment
  labels:
    app: database-pod
spec:
  replicas: 1
  selector:
    matchLabels:
      app: database-pod
  template:
    metadata:
      labels:
        app: database-pod
    spec:
      containers:
```

```
      - name: wp-db
        image: docker.io/library/mariadb:latest
        args:
          - mariadbd
        env:
          - name: MYSQL_PASSWORD
            value: wppass
          - name: MYSQL_DATABASE
            value: wp
          - name: MYSQL_USER
            value: wordpress
          - name: MYSQL_ROOT_PASSWORD
            value: dbpass
        ports:
          - containerPort: 3306
        volumeMounts:
          - mountPath: /var/lib/mysql
            name: wp-db-mount-pvc
    volumes:
      - name: wp-db-mount-pvc
        persistentVolumeClaim:
          claimName: wp-db-mount
```

```
student@virt-k8s-XX:~/wordpress$ nano deployment-wordpress.yaml
apiVersion: apps/v1
kind: Deployment
```

```
metadata:
  name: wordpress-deployment
  labels:
    app: wp-pod
spec:
  replicas: 1
  selector:
    matchLabels:
      app: wp-pod
  template:
    metadata:
      labels:
        app: wp-pod
    spec:
      containers:
        - name: wp-web
          image: docker.io/library/wordpress:latest
          args:
            - apache2-foreground
          env:
            - name: WORDPRESS_DB_PASSWORD
              value: wppass
            - name: WORDPRESS_DB_HOST
              value: database-service
            - name: WORDPRESS_DB_NAME
              value: wp
```

```
        - name: WORDPRESS_DB_USER
          value: wordpress
      ports:
        - containerPort: 80
      volumeMounts:
        - mountPath: /var/www/html
          name: wp-web-mount-pvc
  volumes:
    - name: wp-web-mount-pvc
      persistentVolumeClaim:
        claimName: wp-web-mount
```

We verwijderen de pods en starten de deployments.

```
student@virt-k8s-XX:~/wordpress$ kubectl delete pod database-pod
pod "database-pod" deleted
student@virt-k8s-XX:~/wordpress$ kubectl delete pod wordpress-
pod
pod "wordpress-pod" deleted
student@virt-k8s-XX:~/wordpress$ kubectl apply -f deployment-
database.yaml
deployment.apps/database-deployment created
student@virt-k8s-XX:~/wordpress$ kubectl apply -f deployment-
wordpress.yaml
deployment.apps/wordpress-deployment created
```

De website is nu uiteraard ook beschikbaar.

We vragen nu de pods op.

```
student@virt-k8s-XX:~/wordpress$ kubectl get pods
NAME                                    READY   STATUS …
database-deployment-597cddbfc7-rvnk9    1/1     Running …
wordpress-deployment-5cc9fd7ffc-rg4s8   1/1     Running …
```

Je kan het aantal replicas (aantal pods) van een Deployment heel eenvoudig verhogen naar 2 op twee manieren: imperatief en declaratief.

Om het declaratief in te stellen open je een deployment YAML (we kiezen voor deployment-wordpress.yaml) en wijzigen bijvoorbeeld replicas: 1 naar replicas: 2 zodat er 2 i.p.v. 1 pod draait.

```
student@virt-k8s-XX:~/wordpress$ nano deployment-wordpress.yaml
…
spec:
  replicas: 2 # Pas dit aan
  selector:
…
```

Uiteraard dienen we de wijzigingen door te voeren.

```
student@virt-k8s-XX:~/wordpress$ kubectl apply -f deployment-
wordpress.yaml
```

```
deployment.apps/wordpress-deployment configured
```

Als je de pods nu opvraagt zie je dat er 2 pods van Wordpress draaien.

```
student@virt-k8s-XX:~/wordpress$ kubectl get pods
NAME                                    READY   STATUS
RESTARTS   AGE
database-deployment-597cddbfc7-rvnk9    1/1     Running   0
8m23s
wordpress-deployment-5cc9fd7ffc-clqln   1/1     Running   0
27s
wordpress-deployment-5cc9fd7ffc-rg4s8   1/1     Running   0
8m19s
```

Om het imperatief in te stellen zullen we het aantal pods van wordpress opschalen naar 3.

```
student@virt-k8s-XX:~/wordpress$ kubectl scale deployment
wordpress-deployment --replicas=3
deployment.apps/wordpress-deployment scaled
student@virt-k8s-XX:~/wordpress$ kubectl get deployments
NAME                   READY   UP-TO-DATE   AVAILABLE   AGE
database-deployment    1/1     1            1           14m
wordpress-deployment   2/3     3            2           14m
```

En na enige tijd…

```
student@virt-k8s-XX:~/wordpress$ kubectl get deployments
NAME                   READY   UP-TO-DATE   AVAILABLE   AGE
database-deployment    1/1     1            1           15m
wordpress-deployment   3/3     3            3           14m
```