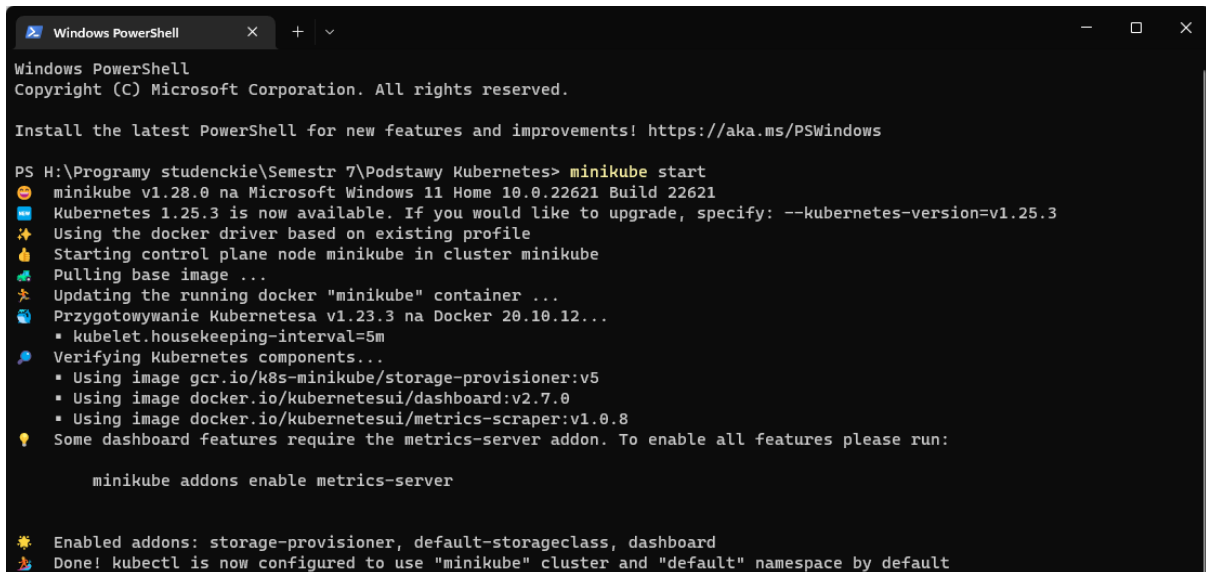


# Podstawy Kubernetes - projekt zaliczeniowy

Jakub Podsiadły  
nr albumu 200865  
rok IV ISI-1

Na początku instaluję dockera i minikube.  
Uruchamiam minikube:



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

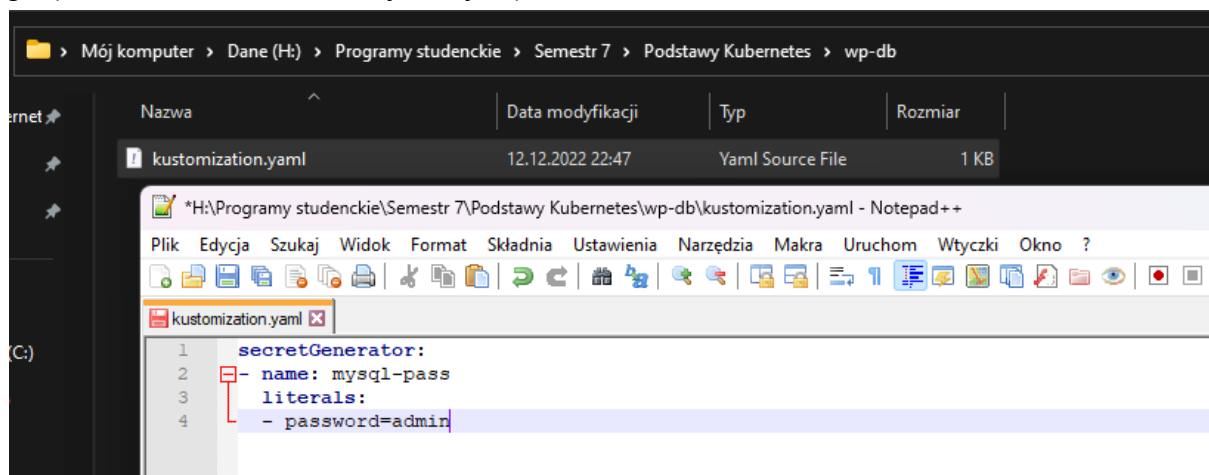
Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS H:\Programy studenckie\Semestr 7\Podstawy Kubernetes> minikube start
minikube v1.28.0 na Microsoft Windows 11 Home 10.0.22621 Build 22621
Kubernetes 1.25.3 is now available. If you would like to upgrade, specify: --kubernetes-version=v1.25.3
Using the docker driver based on existing profile
Starting control plane node minikube in cluster minikube
Pulling base image ...
Updating the running docker "minikube" container ...
Przygotowywanie Kubernetesa v1.23.3 na Docker 20.10.12...
  kubelet.housekeeping-interval=5m
Verifying Kubernetes components...
  Using image gcr.io/k8s-minikube/storage-provisioner:v5
  Using image docker.io/kubernetes/dashboard:v2.7.0
  Using image docker.io/kubernetes/metrics-scraper:v1.0.8
Some dashboard features require the metrics-server addon. To enable all features please run:

    minikube addons enable metrics-server

Enabled addons: storage-provisioner, default-storageclass, dashboard
Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
```

Tworzę folder do bazy danych a w nim tworzę plik kustomization.yaml i konfiguruję go (ustawiam hasło do bazy danych):



File Explorer path: `Mój komputer > Dane (H:) > Programy studenckie > Semestr 7 > Podstawy Kubernetes > wp-db`

File list:

Nazwa	Data modyfikacji	Typ	Rozmiar
kustomization.yaml	12.12.2022 22:47	Yaml Source File	1 KB

Notepad++ content:

```
*H:\Programy studenckie\Semestr 7\Podstawy Kubernetes\wp-db\kustomization.yaml - Notepad++
Plik Edycja Szukaj Widok Format Składnia Ustawienia Narzędzia Makra Uruchom Wtyczki Okno ?

1 secretGenerator:
2   name: mysql-pass
3   literals:
4     - password=admin
```

Dodaję także pliki konfiguracyjne mysql-deployment.yaml.

```
kustomization.yaml x mysql-deployment.yaml x wordpress-de
1  apiVersion: v1
2  kind: Service
3  metadata:
4    name: wordpress-mysql
5    labels:
6      app: wordpress
7  spec:
8    ports:
9      - port: 3306
10   selector:
11     app: wordpress
12     tier: mysql
13   clusterIP: None
14   ---
15   apiVersion: v1
16   kind: PersistentVolumeClaim
17   metadata:
18     name: mysql-pv-claim
19     labels:
20       app: wordpress
21   spec:
22     accessModes:
23       - ReadWriteOnce
24     resources:
25       requests:
26         storage: 20Gi
27   ---
28   apiVersion: apps/v1
29   kind: Deployment
30   metadata:
31     name: wordpress-mysql
32     labels:
33       app: wordpress
34   spec:
35     selector:
36       matchLabels:
37         app: wordpress
38         tier: mysql
39     strategy:
40       type: Recreate
41     template:
42       metadata:
43         labels:
44           app: wordpress
45           tier: mysql
46       spec:
47         containers:
48           - image: mysql:5.6
49             name: mysql
50             env:
51               - name: MYSQL_ROOT_PASSWORD
52                 valueFrom:
53                   secretKeyRef:
54                     name: mysql-pass
55                     key: password
56             ports:
57               - containerPort: 3306
58               name: mysql
59             volumeMounts:
60               - name: mysql-persistent-storage
61                 mountPath: /var/lib/mysql
62             volumes:
63               - name: mysql-persistent-storage
64                 persistentVolumeClaim:
65                   claimName: mysql-pv-claim
66
```

Teraz tworzę wordpress-deployment.yaml.

```
kustomization.yaml x mysql-deployment.yaml x wordpress-deplo
1  apiVersion: v1
2  kind: Service
3  metadata:
4    name: wordpress
5    labels:
6      app: wordpress
7  spec:
8    ports:
9      - port: 80
10   selector:
11     app: wordpress
12     tier: frontend
13   type: LoadBalancer
14   ---
15   apiVersion: v1
16   kind: PersistentVolumeClaim
17   metadata:
18     name: wp-pv-claim
19     labels:
20       app: wordpress
21   spec:
22     accessModes:
23       - ReadWriteOnce
24     resources:
25       requests:
26         storage: 20Gi
27   ---
28   apiVersion: apps/v1
29   kind: Deployment
30   metadata:
31     name: wordpress
32     labels:
33       app: wordpress
34   spec:
35     selector:
36       matchLabels:
37         app: wordpress
38         tier: frontend
39     strategy:
40       type: Recreate
41     template:
42       metadata:
43         labels:
44           app: wordpress
45           tier: frontend
46       spec:
47         containers:
48           - image: wordpress:4.8-apache
49             name: wordpress
50             env:
51               - name: WORDPRESS_DB_HOST
52                 value: wordpress-mysql
53               - name: WORDPRESS_DB_PASSWORD
54                 valueFrom:
55                   secretKeyRef:
56                     name: mysql-pass
57                     key: password
58             ports:
59               - containerPort: 80
60                 name: wordpress
61             volumeMounts:
62               - name: wordpress-persistent-storage
63                 mountPath: /var/www/html
64             volumes:
65               - name: wordpress-persistent-storage
66                 persistentVolumeClaim:
67                   claimName: wp-pv-claim
68
```

Robię deploy Wordpressa z mysql:

```
Windows PowerShell
PS H:\Programy studenckie\Semestr 7\Podstawy Kubernetes\wp-db> kubectl apply -f ./ --validate=false
secret/mysql-pass created
service/wordpress-mysql unchanged
persistentvolumeclaim/mysql-pv-claim unchanged
deployment.apps/wordpress-mysql unchanged
service/wordpress unchanged
persistentvolumeclaim/wp-pv-claim unchanged
deployment.apps/wordpress unchanged
PS H:\Programy studenckie\Semestr 7\Podstawy Kubernetes\wp-db>
```

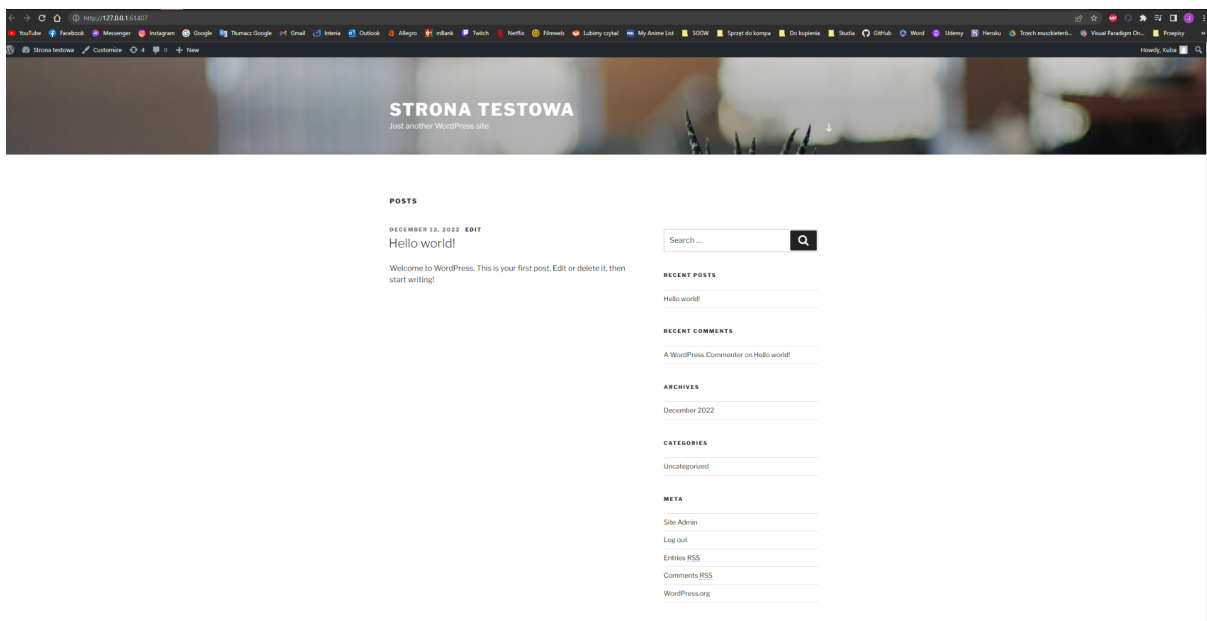
Jak widzimy nasz Wordpress razem z bazą danych działają:

```
PS H:\Programy studenckie\Semestr 7\Podstawy Kubernetes\wp-db> kubectl get services
NAME                TYPE          CLUSTER-IP      EXTERNAL-IP      PORT(S)          AGE
kubernetes           ClusterIP     10.96.0.1       <none>           443/TCP          74m
wordpress            LoadBalancer 10.101.79.73    <pending>        80:31078/TCP     15m
wordpress-mysql      ClusterIP     None            <none>           3306/TCP         15m
PS H:\Programy studenckie\Semestr 7\Podstawy Kubernetes\wp-db> |
```

Używając minikube generujemy adres url:

```
PS H:\Programy studenckie\Semestr 7\Podstawy Kubernetes\wp-db> minikube service wordpress --url
http://127.0.0.1:61407
```

Adres url przekopiuujemy do przeglądarki, strona z Wordpressem działa i możemy ją konfigurować:



Sprawdzam teraz czy wszystko jest dobrze skonfigurowane:

```
Windows PowerShell
PS H:\Programy studenckie\Semestr 7\Podstawy Kubernetes\wp-db> kubectl get secrets
NAME                                TYPE                                DATA  AGE
default-token-mwm5l                 kubernetes.io/service-account-token 3      88m
mysql-pass                          Opaque                             1      18m
PS H:\Programy studenckie\Semestr 7\Podstawy Kubernetes\wp-db> kubectl get pvc
NAME                                STATUS  VOLUME                                     CAPACITY  ACCESS MODES  STORAGECLASS  AGE
mysql-pv-claim                     Bound   pvc-e6214048-60e3-4395-98df-90a450707080 20Gi       RWO            standard      29m
wp-pv-claim                         Bound   pvc-257c1330-423c-4975-b861-4f40a6909524 20Gi       RWO            standard      29m
PS H:\Programy studenckie\Semestr 7\Podstawy Kubernetes\wp-db> kubectl get pods
NAME                                READY  STATUS   RESTARTS  AGE
wordpress-687f77b58d-pcbw8         1/1    Running   0          30m
wordpress-mysql-db6648954-bpmbd    1/1    Running   0          30m
PS H:\Programy studenckie\Semestr 7\Podstawy Kubernetes\wp-db>
```

Jak widać hasło do bazy danych jest przekazywane, pamięć jest przydzielana i pody są zdeployowane i działają.

## Podsumowanie:

Aby uruchomić Wordpresa z bazą danych potrzebujemy zainstalować dockera i minikube (można użyć także rozwiązań z internetu takich jak Google Clouds z Kubernetess Engine). Najpierw tworzymy konterner minicube, później konfigurujemy pliki. Ustawiamy typ (Service), nazwę naszej aplikacji, przydzielamy miejsce, wskazujemy na obraz dostępny online i ustawiamy port. Później robimy deploy i sprawdzamy czy nasza aplikacja została dobrze skonfigurowana i czy działa poprawnie.