Michał Muzyka 2.3 – Sprawozdanie Lab 8

Uruchamiam klaster z węzłem głównym i 3 węzłami roboczymi z pluginem CNI i sterownikiem Docker. Dla każdego węzła przydzielana jest pamięć RAM 1900 MB i 2 CPU.

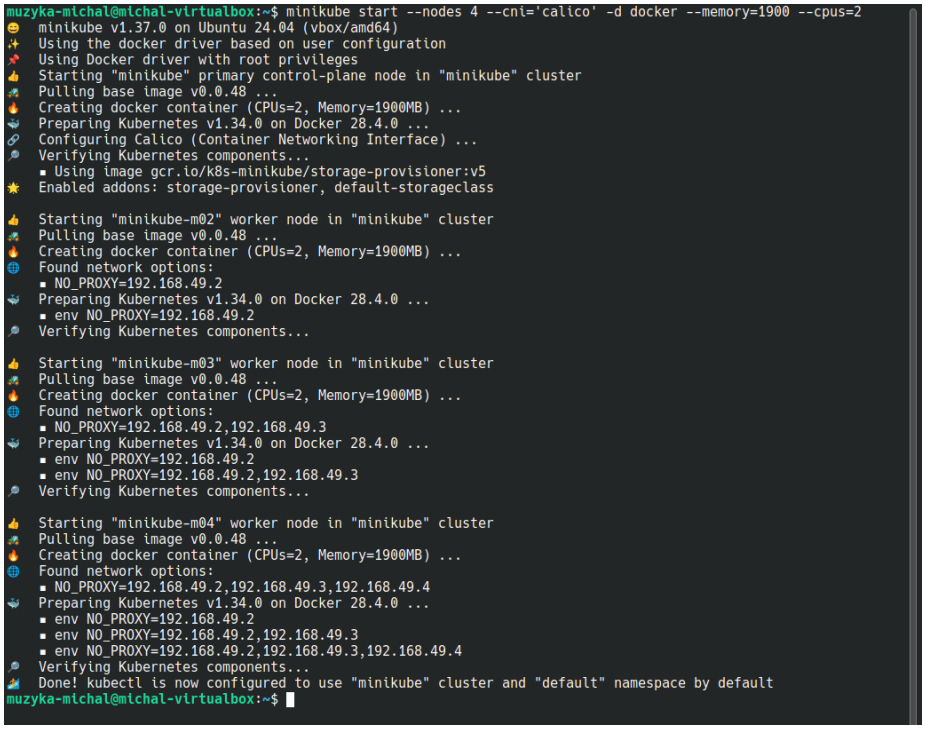

Figure 1: alt text

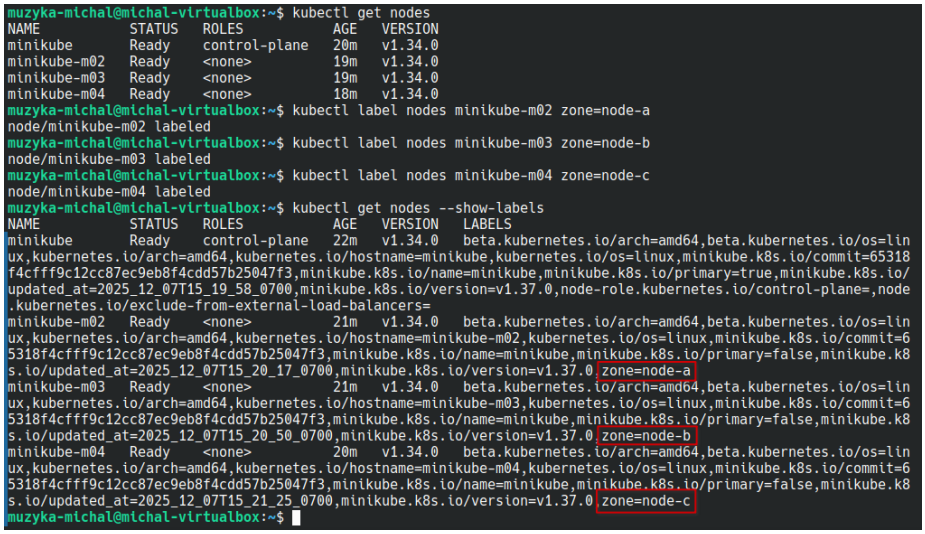

Figure 2: alt text

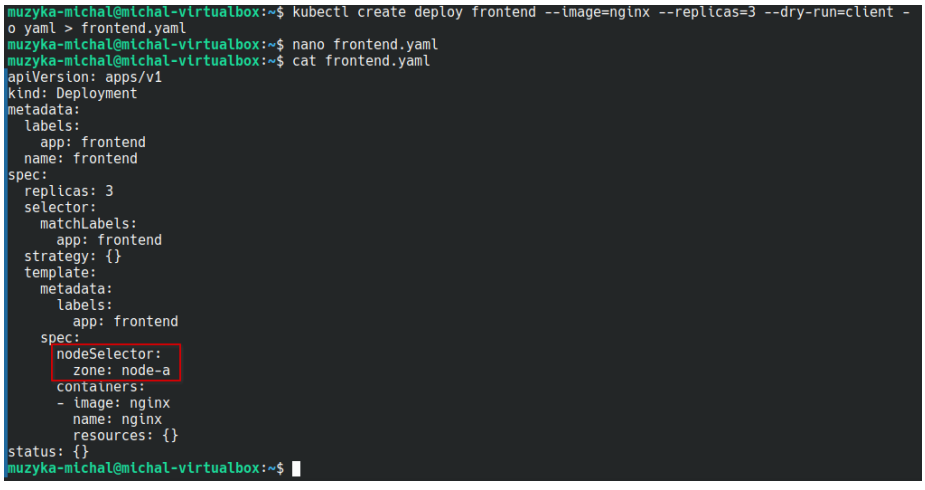

Figure 3: alt text

```
muzyka-michal@michal-virtualbox:~$ kubectl create deploy backend --image=nginx --replicas=1 --dry-run=client -o
 yaml > backend.yaml
muzyka-michal@michal-virtualbox:~$ nano backend.yaml
muzyka-michal@michal-virtualbox:~$ cat backend.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: backend
  name: backend
spec:
  replicas: 1
  selector:
    matchLabels:
      app: backend
  strategy: {}
  template:
    metadata:
      labels:
        app: backend
    spec:
      nodeSelector:
        zone: node-b
      containers:
      - image: nginx
        name: nginx
        resources: {}
status: {}
muzyka-michal@michal-virtualbox:~$
```

Figure 4: alt text

```
muzyka-michal@michal-virtualbox:~$ kubectl run my-sql --image=mysql:5.7 --env="MYSQL_ROOT_PASSWORD=root" --labe
ls="app=my-sql" --dry-run=client -o yaml > mysql.yaml
muzyka-michal@michal-virtualbox:~$ nano mysql.yaml
muzyka-michal@michal-virtualbox:~$ cat mysql.yaml
apiVersion: v1
kind: Pod
metadata:
  labels:
    app: my-sql
  name: my-sql
spec:
  nodeSelector:
    zone: node-c
  containers:
  - env:
    - name: MYSQL_ROOT_PASSWORD
      value: root
    image: mysql:5.7
    name: my-sql
    resources: {}
  dnsPolicy: ClusterFirst
  restartPolicy: Always
status: {}
muzyka-michal@michal-virtualbox:~$
```

Figure 5: alt text

Wyświetlam utworzone węzły. Oznaczam węzły etykietami kolejno A, B i C.

Generuję plik Deploymentu o nazwie frontend na bazie obrazu nginx i 3 replikami. Dodaję do pliku sekcję nodeSelector z etykietą zone=node-a , aby pody tego Deploymentu znalazły się na węźle A.

Generuję plik Deploymentu o nazwie backend na bazie obrazu nginx i 1 repliką. Dodaję do pliku sekcję nodeSelector z etykietą zone=node-b , aby pod tego Deploymentu znalazł się na węźle B.

Generuję plik poda o nazwie my-sql na bazie obrazu mysql w wersji 5.7, ze zmienną środowiskową MYSQL_ROOT_PASSWORD=root z hasłem do bazy danych oraz etykietą app=my-sql, która przyda się później.

Tworzę obiekty i weryfikuję ich działanie. Deployments i pody działają. Pody znajdują się na odpowiednich węzłach.

```
muzyka-michal@michal-virtualbox:~$ kubectl apply -f frontend.yaml
deployment.apps/frontend created
muzyka-michal@michal-virtualbox:~$ kubectl apply -f backend.yaml
deployment.apps/backend created
muzyka-michal@michal-virtualbox:~$ kubectl apply -f mysql.yaml
pod/my-sql created
muzyka-michal@michal-virtualbox:~$ kubectl get deploy
NAME       READY   UP-TO-DATE   AVAILABLE   AGE
backend    1/1     1            1           2m57s
frontend   3/3     3            3           3m3s
muzyka-michal@michal-virtualbox:~$ kubectl get pods -o wide
NAME                        READY   STATUS    RESTARTS   AGE     IP             NODE           NOMINATED NODE   READINESS GATES
backend-6b55d7f584-96w75    1/1     Running   0          5m35s   10.244.151.1   minikube-m03   <none>           <none>
frontend-58fcfd74c-2pjc2    1/1     Running   0          5m41s   10.244.205.193   minikube-m02   <none>           <none>
frontend-58fcfd74c-5qcqk    1/1     Running   0          5m41s   10.244.205.194   minikube-m02   <none>           <none>
frontend-58fcfd74c-s9kfb    1/1     Running   0          5m41s   10.244.205.195   minikube-m02   <none>           <none>
my-sql                      1/1     Running   0          5m29s   10.244.59.129    minikube-m04   <none>           <none>
muzyka-michal@michal-virtualbox:~$
```

Figure 6: alt text

```
muzyka-michal@michal-virtualbox:~$ kubectl create service nodeport frontend --tcp=80:80 --node-port=
30080 --dry-run=client -o yaml > svc-frontend.yaml
muzyka-michal@michal-virtualbox:~$ cat svc-frontend.yaml
apiVersion: v1
kind: Service
metadata:
  labels:
    app: frontend
  name: frontend
spec:
  ports:
  - name: 80-80
    nodePort: 30080
    port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: frontend
  type: NodePort
status:
  loadBalancer: {}
muzyka-michal@michal-virtualbox:~$ kubectl apply -f svc-frontend.yaml
service/frontend created
muzyka-michal@michal-virtualbox:~$
```

Figure 7: alt text

```
muzyka-michal@michal-virtualbox:~$ kubectl create service clusterip backend --tcp=80:80 --dry-run=cl
ient -o yaml > svc-backend.yaml
muzyka-michal@michal-virtualbox:~$ cat svc-backend.yaml
apiVersion: v1
kind: Service
metadata:
  labels:
    app: backend
  name: backend
spec:
  ports:
  - name: 80-80
    port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: backend
  type: ClusterIP
status:
  loadBalancer: {}
muzyka-michal@michal-virtualbox:~$
```

Figure 8: alt text

```
muzyka-michal@michal-virtualbox:~$ kubectl expose pod my-sql --port=3306 --target-port=3306 --name=m
y-sql --type=ClusterIP --dry-run=client -o yaml > svc-mysql.yaml
muzyka-michal@michal-virtualbox:~$ cat svc-mysql.yaml
apiVersion: v1
kind: Service
metadata:
  labels:
    app: my-sql
  name: my-sql
spec:
  ports:
  - port: 3306
    protocol: TCP
    targetPort: 3306
  selector:
    app: my-sql
  type: ClusterIP
status:
  loadBalancer: {}
muzyka-michal@michal-virtualbox:~$
```

Figure 9: alt text

```
muzyka-michal@michal-virtualbox:~$ kubectl apply -f svc-backend.yaml
service/backend created
muzyka-michal@michal-virtualbox:~$ kubectl apply -f svc-mysql.yaml
service/my-sql created
muzyka-michal@michal-virtualbox:~$ kubectl get svc
NAME         TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
backend      ClusterIP   10.111.53.145   <none>        80/TCP         57s
frontend     NodePort    10.103.229.99   <none>        80:30080/TCP   18m
kubernetes   ClusterIP   10.96.0.1       <none>        443/TCP        3h31m
my-sql       ClusterIP   10.102.165.2    <none>        3306/TCP       50s
muzyka-michal@michal-virtualbox:~$
```

Figure 10: alt text

Generuję plik obiektu Service typu NodePort dla Deploymentu frontend z portem usługi 80, target portem 80 i zewnętrznym portem NodePort równym 30080. Tworzę obiekt Service.

Generuję plik obiektu Service typu ClusterIP dla Deploymentu backend z portem usługi 80 i target portem 80.

Generuję plik obiektu Service typu ClusterIP dla poda my-sql z portem usługi 3306 i target portem 3306.

Tworzę obiekty Service dla Deploymentu backend i poda my-sql. Sprawdzam poprawność.

Tworzę plik obiektu NetworkPolicy. W nim ustawiam, że polityka sieciowa odnosi się do poda my-sql o ustawionej wcześniej etykiecie app=my-sql. Ustawiam kontrolowany typ ruchu - wejściowy Ingress, gdzie pody tylko z etykietą app=backend, czyli backend, mogą łączyć się z ustalonym wcześniej podem, ale tylko na porcie 3306 przez TCP. Każdy inny ruch (w tym od app=frontend) zostanie automatycznie odrzucony przez plugin CNI Calico.

```
muzyka-michal@michal-virtualbox:~$ nano netpol.yaml
muzyka-michal@michal-virtualbox:~$ cat netpol.yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: access-to-mysql
spec:
  podSelector:
    matchLabels:
      app: my-sql
  policyTypes:
  - Ingress
  ingress:
  - from:
    - podSelector:
        matchLabels:
          app: backend
    ports:
    - protocol: TCP
      port: 3306
muzyka-michal@michal-virtualbox:~$
```

Figure 11: alt text

```
muzyka-michal@michal-virtualbox:~$ kubectl apply -f netpol.yaml
networkpolicy.networking.k8s.io/access-to-mysql created
muzyka-michal@michal-virtualbox:~$ kubectl get netpol
NAME              POD-SELECTOR   AGE
access-to-mysql   app=my-sql     6m3s
muzyka-michal@michal-virtualbox:~$
```

Figure 12: alt text

Tworzę obiekt NetworkPolicy i weryfikuję czy został utworzony.

Testuję poprawność utworzonej polityki. Sprawdzam adres usługi my-sql. Sprawdzam nazwy podów backendu i frontendu. Uruchmiam polecenie curl na wybranych podach frontendu i backendu. Udało nazwiązać połączenie z my-sql tylko przez pod backendu.

```
muzyka-michal@michal-virtualbox:~$ kubectl get svc my-sql
NAME     TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE
my-sql   ClusterIP   10.102.165.2    <none>        3306/TCP   80m
muzyka-michal@michal-virtualbox:~$ kubectl get pod -l app=backend
NAME                       READY   STATUS    RESTARTS   AGE
backend-6b55d7f584-96w75   1/1     Running   0          115m
muzyka-michal@michal-virtualbox:~$ kubectl get pod -l app=frontend
NAME                        READY   STATUS    RESTARTS   AGE
frontend-58fcfd74c-2pjc2    1/1     Running   0          115m
frontend-58fcfd74c-5qcqk    1/1     Running   0          115m
frontend-58fcfd74c-s9kfb    1/1     Running   0          115m
muzyka-michal@michal-virtualbox:~$ kubectl exec -it frontend-58fcfd74c-2pjc2 -- curl -v --connect-ti
meout 2 my-sql:3306
* Host my-sql:3306 was resolved.
* IPv6: (none)
* IPv4: 10.102.165.2
*   Trying 10.102.165.2:3306...
* Connection timed out after 2002 milliseconds
* closing connection #0
curl: (28) Connection timed out after 2002 milliseconds
command terminated with exit code 28
muzyka-michal@michal-virtualbox:~$ kubectl exec -it backend-6b55d7f584-96w75 -- curl -v --connect-ti
meout 2 my-sql:3306
* Host my-sql:3306 was resolved.
* IPv6: (none)
* IPv4: 10.102.165.2
*   Trying 10.102.165.2:3306...
* Connected to my-sql (10.102.165.2) port 3306
* using HTTP/1.x
> GET / HTTP/1.1
> Host: my-sql:3306
> User-Agent: curl/8.14.1
> Accept: */*
>
* Received HTTP/0.9 when not allowed
* closing connection #0
curl: (1) Received HTTP/0.9 when not allowed
command terminated with exit code 1
muzyka-michal@michal-virtualbox:~$
```

Figure 13: alt text

Dodatkowo testuję poprawność utworzonej polityki przez utworzenie tymczasowych podów busybox - jeden bez etykiety, drugi z inną etykietą, a trzeci z etykietą app=backend. Tylko trzeci zdołał się połączyć z my-sql. Polityka sieciowa działa poprawnie.

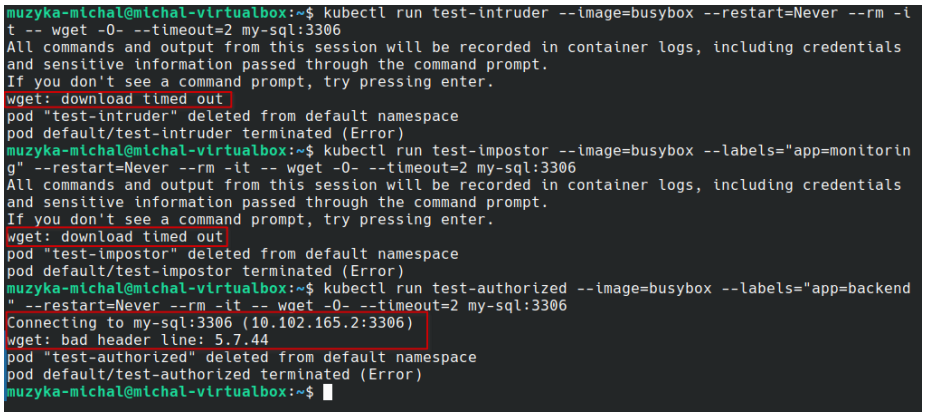

Figure 14: alt text