

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.

```
muzyka-michal@michal-virtualbox:~$ minikube start --nodes 4 --cnf='calico' -d docker --memory=1900 --cpus=2
🕒 minikube v1.37.0 on Ubuntu 24.04 (vbox/amd64)
⭐ Using the docker driver based on user configuration
📌 Using Docker driver with root privileges
⚠ Starting "minikube" primary control-plane node in "minikube" cluster
_PULLING base image v0.0.48 ...
🔥 Creating docker container (CPUs=2, Memory=1900MB) ...
⚡ Preparing Kubernetes v1.34.0 on Docker 28.4.0 ...
🌐 Configuring Calico (Container Networking Interface) ...
🌐 Verifying Kubernetes components...
  └─ Using image gcr.io/k8s-minikube/storage-provisioner:v5
★ Enabled addons: storage-provisioner, default-storageclass

⚠ Starting "minikube-m02" worker node in "minikube" cluster
_PULLING base image v0.0.48 ...
🔥 Creating docker container (CPUs=2, Memory=1900MB) ...
🌐 Found network options:
  └─ NO_PROXY=192.168.49.2
⚡ Preparing Kubernetes v1.34.0 on Docker 28.4.0 ...
  └─ env NO_PROXY=192.168.49.2
🌐 Verifying Kubernetes components...

⚠ Starting "minikube-m03" worker node in "minikube" cluster
_PULLING base image v0.0.48 ...
🔥 Creating docker container (CPUs=2, Memory=1900MB) ...
🌐 Found network options:
  └─ NO_PROXY=192.168.49.2,192.168.49.3
⚡ Preparing Kubernetes v1.34.0 on Docker 28.4.0 ...
  └─ env NO_PROXY=192.168.49.2
  └─ env NO_PROXY=192.168.49.2,192.168.49.3
🌐 Verifying Kubernetes components...

⚠ Starting "minikube-m04" worker node in "minikube" cluster
_PULLING base image v0.0.48 ...
🔥 Creating docker container (CPUs=2, Memory=1900MB) ...
🌐 Found network options:
  └─ NO_PROXY=192.168.49.2,192.168.49.3,192.168.49.4
⚡ Preparing Kubernetes v1.34.0 on Docker 28.4.0 ...
  └─ env NO_PROXY=192.168.49.2
  └─ env NO_PROXY=192.168.49.2,192.168.49.3
  └─ env NO_PROXY=192.168.49.2,192.168.49.3,192.168.49.4
🌐 Verifying Kubernetes components...
💡 Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
muzyka-michal@michal-virtualbox:~$
```

Figure 1: alt text

```

muzyka-michal@michal-virtualbox:~$ kubectl get nodes
NAME      STATUS   ROLES    AGE     VERSION
minikube   Ready    control-plane   20m    v1.34.0
minikube-m02 Ready    <none>    19m    v1.34.0
minikube-m03 Ready    <none>    19m    v1.34.0
minikube-m04 Ready    <none>    18m    v1.34.0
muzyka-michal@michal-virtualbox:~$ kubectl label nodes minikube-m02 zone=node-a
node/minikube-m02 labeled
muzyka-michal@michal-virtualbox:~$ kubectl label nodes minikube-m03 zone=node-b
node/minikube-m03 labeled
muzyka-michal@michal-virtualbox:~$ kubectl label nodes minikube-m04 zone=node-c
node/minikube-m04 labeled
muzyka-michal@michal-virtualbox:~$ kubectl get nodes --show-labels
NAME      STATUS   ROLES    AGE     VERSION   LABELS
minikube   Ready    control-plane   22m    v1.34.0   beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,kubernetes.io/arch=amd64,kubernetes.io/hostname=minikube,kubernetes.io/os=linux,minikube.k8s.io/commit=65318f4cff9c12cc87ec9eb8f4cd57b25047f3,minikube.k8s.io/name=minikube,minikube.k8s.io/primary=true,minikube.k8s.io/updated_at=2025_12_07T15_19_58_0700,minikube.k8s.io/version=v1.37.0,node-role.kubernetes.io/control-plane=,node.kubernetes.io/exclude-from-external-load-balancers=
minikube-m02 Ready    <none>    21m    v1.34.0   beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,kubernetes.io/arch=amd64,kubernetes.io/hostname=minikube-m02,kubernetes.io/os=linux,minikube.k8s.io/commit=65318f4cff9c12cc87ec9eb8f4cd57b25047f3,minikube.k8s.io/name=minikube,minikube.k8s.io/primary=false,minikube.k8s.io/updated_at=2025_12_07T15_20_17_0700,minikube.k8s.io/version=v1.37.0|zone=node-a
minikube-m03 Ready    <none>    21m    v1.34.0   beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,kubernetes.io/arch=amd64,kubernetes.io/hostname=minikube-m03,kubernetes.io/os=linux,minikube.k8s.io/commit=65318f4cff9c12cc87ec9eb8f4cd57b25047f3,minikube.k8s.io/name=minikube,minikube.k8s.io/primary=false,minikube.k8s.io/updated_at=2025_12_07T15_20_50_0700,minikube.k8s.io/version=v1.37.0|zone=node-b
minikube-m04 Ready    <none>    20m    v1.34.0   beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,kubernetes.io/arch=amd64,kubernetes.io/hostname=minikube-m04,kubernetes.io/os=linux,minikube.k8s.io/commit=65318f4cff9c12cc87ec9eb8f4cd57b25047f3,minikube.k8s.io/name=minikube,minikube.k8s.io/primary=false,minikube.k8s.io/updated_at=2025_12_07T15_21_25_0700,minikube.k8s.io/version=v1.37.0|zone=node-c
muzyka-michal@michal-virtualbox:~$ █

```

Figure 2: alt text

```

muzyka-michal@michal-virtualbox:~$ kubectl create deploy frontend --image=nginx --replicas=3 --dry-run=client -o yaml > frontend.yaml
muzyka-michal@michal-virtualbox:~$ nano frontend.yaml
muzyka-michal@michal-virtualbox:~$ cat frontend.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: frontend
    name: frontend
spec:
  replicas: 3
  selector:
    matchLabels:
      app: frontend
  strategy: {}
  template:
    metadata:
      labels:
        app: frontend
    spec:
      nodeSelector:
        zone: node-a
      containers:
        - image: nginx
          name: nginx
          resources: {}
status: {}
muzyka-michal@michal-virtualbox:~$ █

```

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" --label
ls=app=mysql" --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 pody tego Deploymentu znalazły 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. Deploymenty i pody dziążają. Pody znajdują się na odpowiednich węzłach.

```
muzyska-michal@michal-virtualbox:~$ kubectl apply -f frontend.yaml
deployment.apps/frontend created
muzyska-michal@michal-virtualbox:~$ kubectl apply -f backend.yaml
deployment.apps/backend created
muzyska-michal@michal-virtualbox:~$ kubectl apply -f mysql.yaml
pod/mysql created
muzyska-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
muzyska-michal@michal-virtualbox:~$ kubectl get pods -o wide
NAME        READY   STATUS    RESTARTS   AGE      IP          NODE
backend-6bb55d7f84-96w75  1/1     Running   0          5m35s   10.244.151.1  minikube-m03
frontend-58fcfd74c-2pjc2  1/1     Running   0          5m41s   10.244.205.193  minikube-m02
frontend-58fcfd74c-5qcqk  1/1     Running   0          5m41s   10.244.205.194  minikube-m02
frontend-58fcfd74c-s9kfb  1/1     Running   0          5m41s   10.244.205.195  minikube-m02
my-sql       1/1     Running   0          5m29s   10.244.59.129  minikube-m04
muzyska-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=client -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=my-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 ustaloną wcześniejszą 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=mysql   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-s9kf8  1/1    Running   0          115m
muzyka-michal@michal-virtualbox:~$ kubectl exec -it frontend-58fcfd74c-2pjc2 -- curl -v --connect-timeout 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-timeout 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.

```
muzyka-michal@michal-virtualbox:~$ kubectl run test-intruder --image=busybox --restart=Never --rm -i
t -- wget -O- --timeout=2 my-sql:3306
All commands and output from this session will be recorded in container logs, including credentials
and sensitive information passed through the command prompt.
If you don't see a command prompt, try pressing enter.
wget: download timed out
pod "test-intruder" deleted from default namespace
pod default/test-intruder terminated (Error)
muzyka-michal@michal-virtualbox:~$ kubectl run test-impostor --image=busybox --labels="app=monitoring" --restart=Never --rm -it -- wget -O- --timeout=2 my-sql:3306
All commands and output from this session will be recorded in container logs, including credentials
and sensitive information passed through the command prompt.
If you don't see a command prompt, try pressing enter.
wget: download timed out
pod "test-impostor" deleted from default namespace
pod default/test-impostor terminated (Error)
muzyka-michal@michal-virtualbox:~$ kubectl run test-authorized --image=busybox --labels="app=backend"
" --restart=Never --rm -it -- wget -O- --timeout=2 my-sql:3306
Connecting to my-sql:3306 (10.102.165.2:3306)
wget: bad header line: 5.7.44
pod "test-authorized" deleted from default namespace
pod default/test-authorized terminated (Error)
muzyka-michal@michal-virtualbox:~$
```

Figure 14: alt text