Vilniaus Gedimino technikos universitetas

Elektronikos fakultetas

Kompiuterijos ir ryšių technologijų katedra

Debesų kompiuterija

Modulis ELKRM17304

Susipažinimas su Kubernetes platforma

Laboratorinio darbo nr. 3 ataskaita

Atliko: TETfm-20 grupės magistrantas
Saulius Krasuckas
Tikrino: lekt. dr. Liudas Duoba

VILNIUS, 2022

Laboratorinis darbas nr. 3

Susipažinimas su Kubernetes platforma

Darbo tikslas

- Išbandyti `Kubernetes` platformą.
- Atlikti pagrindinius veiksmus su `Kubernetes` resursais.

Darbo eiga

Puslapyje **Learn Kubernetes using Interactive Browser-Based Labs | Katacoda (https://www.katacoda.com/courses/kubernetes)** susikūriau prisijungimą pasinaudodamas "*Log In with Github*" metodu ir autorizuodamas savo akademinę GitHub paskyrą.

Užduotis nr. 1: "Launch Single Node Kubernetes Cluster (https://www.katacoda.com/courses/kubernetes/launch-single-node-cluster)"

Įvadas

Minikube — įrankis pradėti naudotis Kubernetes lokaliai, bet produkcinio tinklo. Jis virtualioje mašinoje (VM) startuoja vienamazgį (angl. *single-node*) Kubernetes klasterį ir tinka asmeniniam kompiuteriui. Skirtas naudotojams, siekiantiems išmėginti Kubernetes ar net vykdyti kasdienį sistemų kūrimą Kubernetes pagrindu.

Žingsnis 1.1: `Minikube` startas

- patikrinu versiją:

```
$ minikube version
minikube version: v1.8.1
commit: cbda04cf6bbe65e987ae52bb393c10099ab62014
```

asciinema.org/a/452640 (https://asciinema.org/a/452640?autoplay=1)

- startuoju VM su `Kubernetes` klasteriu:

```
$ minikube start --wait=false
* minikube v1.8.1 on Ubuntu 18.04
* Using the none driver based on user configuration
* Running on localhost (CPUs=2, Memory=2460MB, Disk=145651MB) ...
* OS release is Ubuntu 18.04.4 LTS
* Preparing Kubernetes v1.17.3 on Docker 19.03.6 ...
  - kubelet.resolv-conf=/run/systemd/resolve/resolv.conf
* Launching Kubernetes ...
* Enabling addons: default-storageclass, storage-provisioner
* Configuring local host environment ...
* Done! kubectl is now configured to use "minikube"
```

asciinema.org/a/452643 (https://asciinema.org/a/452643?autoplay=1)

Žingsnis 1.2: *Kubernetes* klasterio informacija

- tikrinu klasterio būseną:

```
$ kubectl cluster-info
Kubernetes master is running at https://172.17.0.86:8443
KubeDNS is running at https://172.17.0.86:8443/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
```

asciinema.org/a/452671 (https://asciinema.org/a/452671?autoplay=1)

- klasterio mazgų sąrašas:

```
$ kubectl get nodes
NAME       STATUS     ROLES    AGE   VERSION
minikube   NotReady   master   16s   v1.17.3

$ kubectl get nodes
NAME       STATUS   ROLES    AGE   VERSION
minikube   Ready    master   18s   v1.17.3
```

asciinema.org/a/452686 (https://asciinema.org/a/452686?autoplay=1)

Žingsnis 1.3: diegiame konteinerį klasteryje

- konteinerio diegimas iš atvaizdo:

```
$ kubectl create deployment first-deployment --image=katacoda/docker-http-server
deployment.apps/first-deployment created
```

asciinema.org/a/452688 (https://asciinema.org/a/452688?autoplay=1)

- tikrinu diegimo būseną:

```
$ kubectl get pods
NAME                               READY   STATUS              RESTARTS   AGE
first-deployment-666c48b44-92c2z   0/1     ContainerCreating   0          3s

$ kubectl get pods
NAME                               READY   STATUS              RESTARTS   AGE
first-deployment-666c48b44-92c2z   0/1     ContainerCreating   0          4s

$ kubectl get pods
NAME                               READY   STATUS    RESTARTS   AGE
first-deployment-666c48b44-92c2z   1/1     Running   0          5s
```

asciinema.org/a/452708 (https://asciinema.org/a/452708?autoplay=1)

- paviešinu konteinerį tinkle:

```
$ kubectl expose deployment first-deployment --port=80 --type=NodePort
service/first-deployment exposed
```

asciinema.org/a/452709 (https://asciinema.org/a/452709?autoplay=1)

- susirandu alokuotą TCP-portą ir vykdome HTTP-užklausą:

```
$ kubectl get svc first-deployment -o go-template='{{range.spec.ports}}{{if .nodePort}}
{{.nodePort}}{{"\n"}}{{end}}{{end}}'
31900

$ export PORT=$(kubectl get svc first-deployment -o go-template='{{range.spec.ports}}{{if
.nodePort}}{{.nodePort}}{{"\n"}}{{end}}{{end}}')

$ echo "Accessing host01:$PORT"
Accessing host01:31900

$ curl host01:$PORT
<h1>This request was processed by host: first-deployment-666c48b44-92c2z</h1>
```

asciinema.org/a/452711 (https://asciinema.org/a/452711?autoplay=1)

Žingsnis 1.4: *Kubernetes Dashboard* sąsaja (web-UI)

- įgalinu *Minicube* priedą *Dashboard*:

```
$ minikube addons enable dashboard
* The 'dashboard' addon is enabled
```

asciinema.org/a/452714 (https://asciinema.org/a/452714?autoplay=1)

- diegiu *Kubernetes Dashboard* pagal duotą YAML šabloną:

```
$ kubectl apply -f /opt/kubernetes-dashboard.yaml
namespace/kubernetes-dashboard configured
service/kubernetes-dashboard-katacoda created
```

asciinema.org/a/452718 (https://asciinema.org/a/452718?autoplay=1)

- patikrinu šablono turinį:

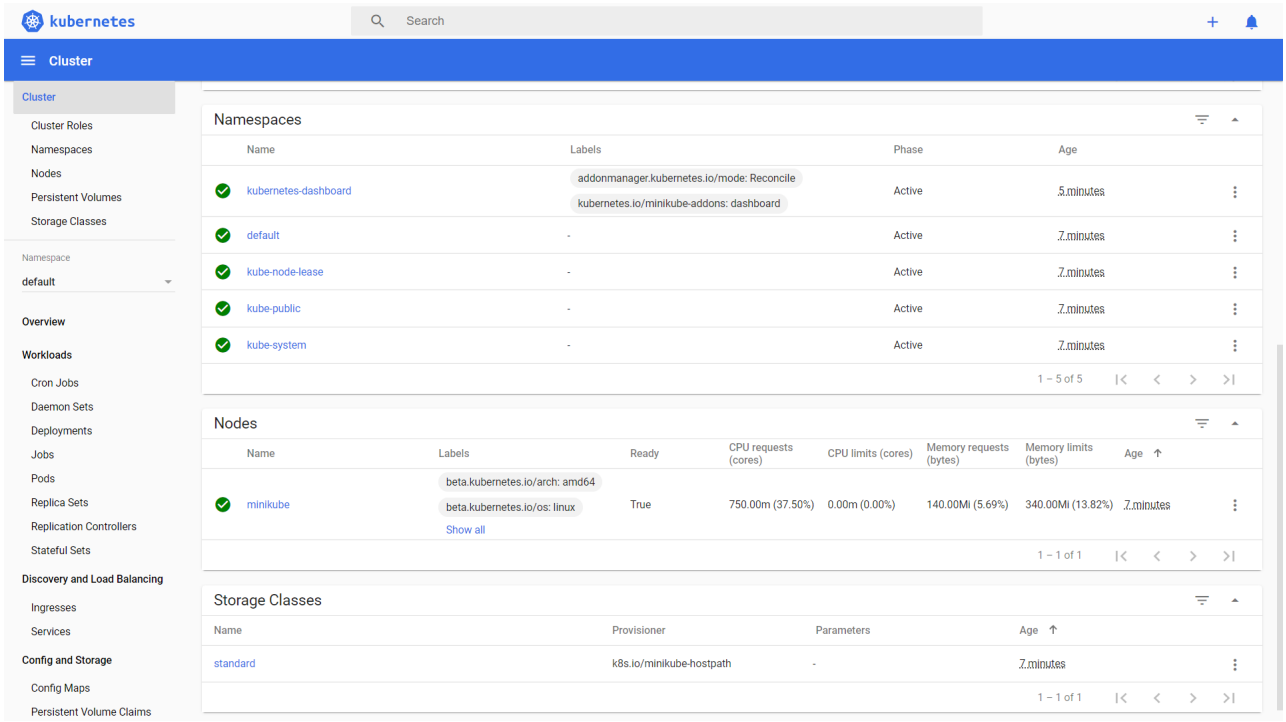
```
$ ls -l /opt/kubernetes-dashboard.yaml
-rw-r--r-- 1 root root 588 Mar  8  2020 /opt/kubernetes-dashboard.yaml

$ cat /opt/kubernetes-dashboard.yaml
apiVersion: v1
kind: Namespace
metadata:
  labels:
    addonmanager.kubernetes.io/mode: Reconcile
    kubernetes.io/minikube-addons: dashboard
  name: kubernetes-dashboard
  selfLink: /api/v1/namespaces/kubernetes-dashboard
spec:
  finalizers:
  - kubernetes
status:
  phase: Active
---
apiVersion: v1
kind: Service
metadata:
  labels:
    app: kubernetes-dashboard
  name: kubernetes-dashboard-katacoda
  namespace: kubernetes-dashboard
spec:
  ports:
  - port: 80
    protocol: TCP
    targetPort: 9090
    nodePort: 30000
  selector:
    k8s-app: kubernetes-dashboard
  type: NodePort
```

- stebiu *Dashboard* konteinerių startą:

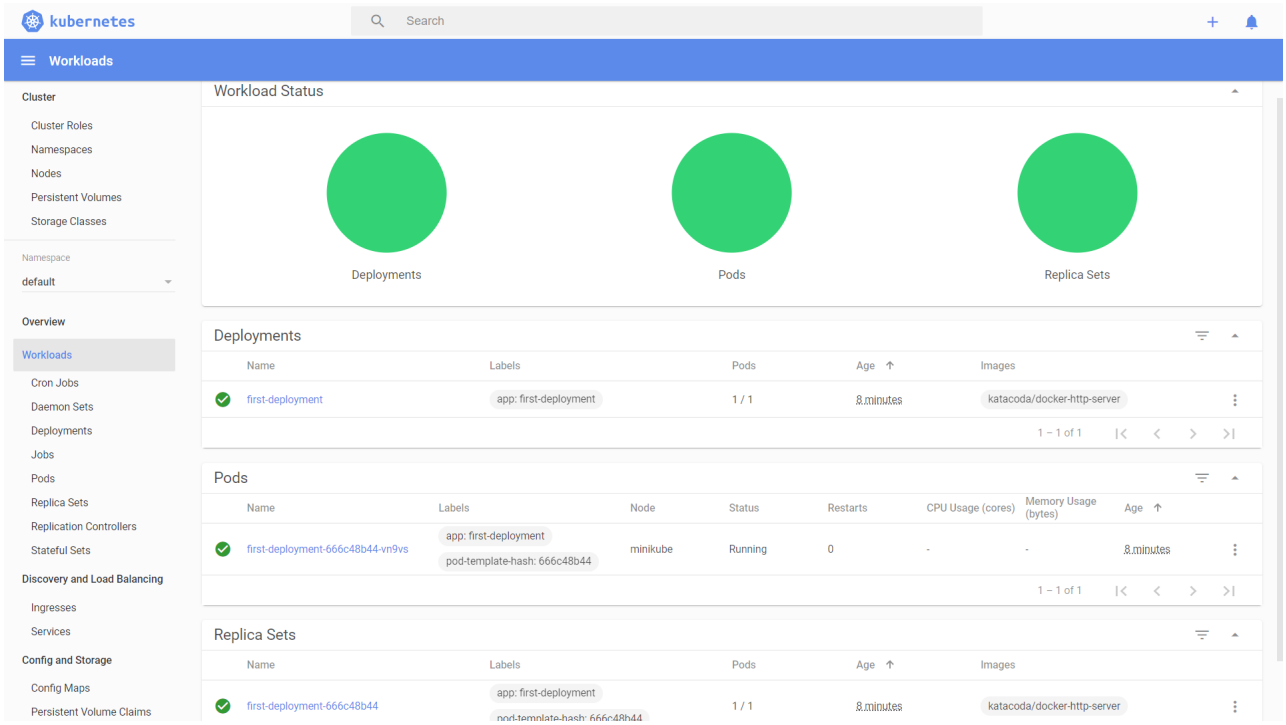
```
$ kubectl get pods -n kubernetes-dashboard -w
NAME                                         READY   STATUS              RESTARTS   AGE
dashboard-metrics-scraper-7b64584c5c-7x46c   0/1     ContainerCreating   0          1s
kubernetes-dashboard-79d9cd965-7f5pb         0/1     ContainerCreating   0          1s
kubernetes-dashboard-79d9cd965-7f5pb         1/1     Running             0          1s
dashboard-metrics-scraper-7b64584c5c-7x46c   1/1     Running             0          2s
^C
$
```

asciinema.org/a/452725 (https://asciinema.org/a/452725?autoplay=1)

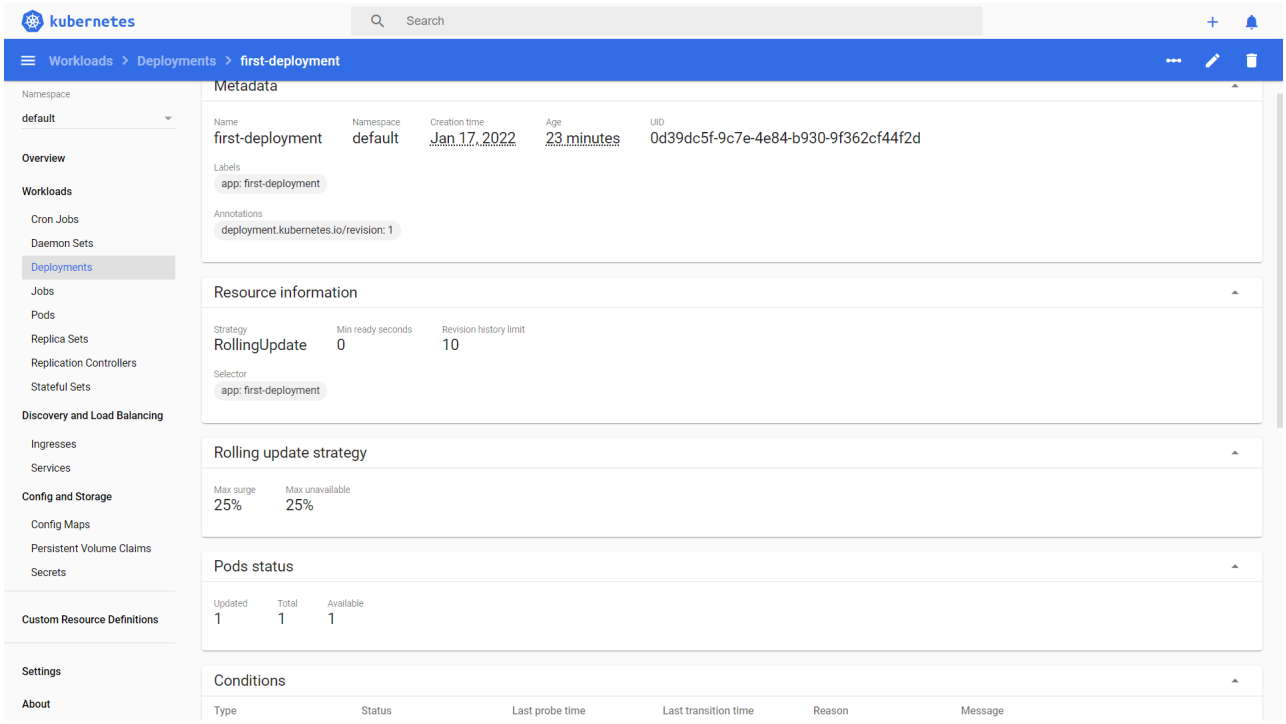
- tikrinu web-UI sąsają tiesiogiai:
  https://2886795274-30000-cykoria04.environments.katacoda.com/
  - klasterio apžvalga:



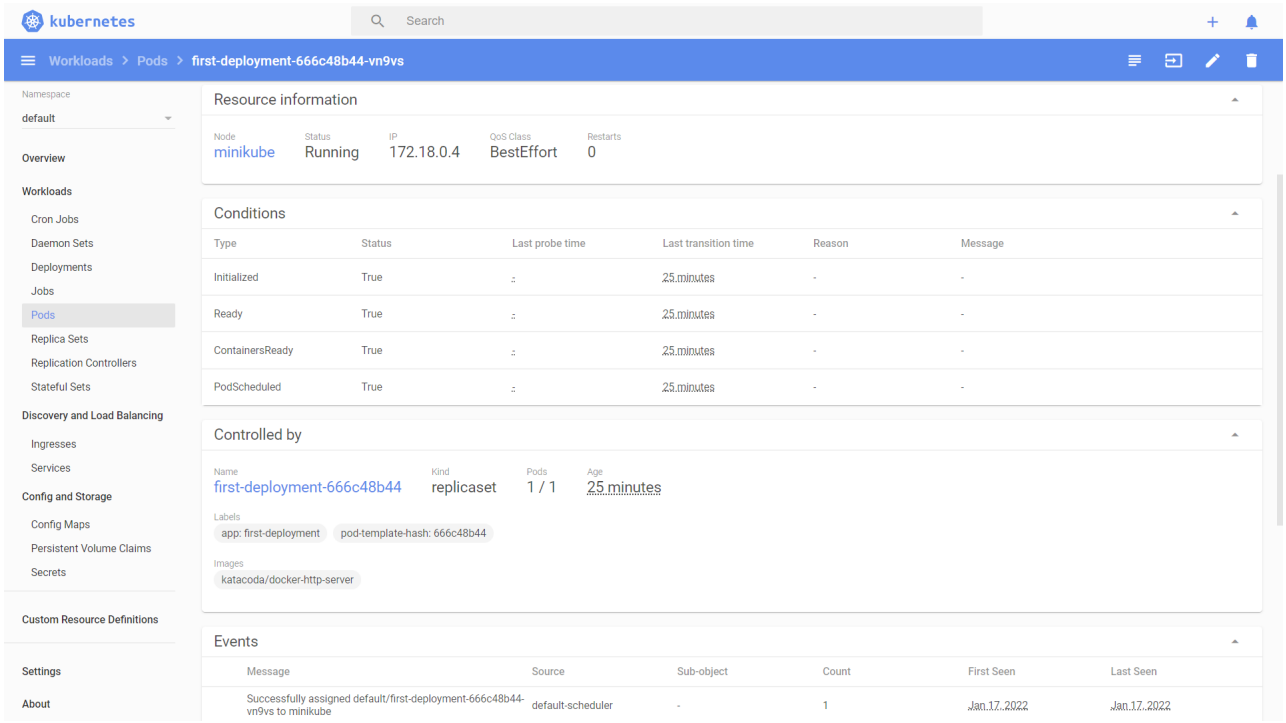
- vardų srities apkrovos apžvalga:



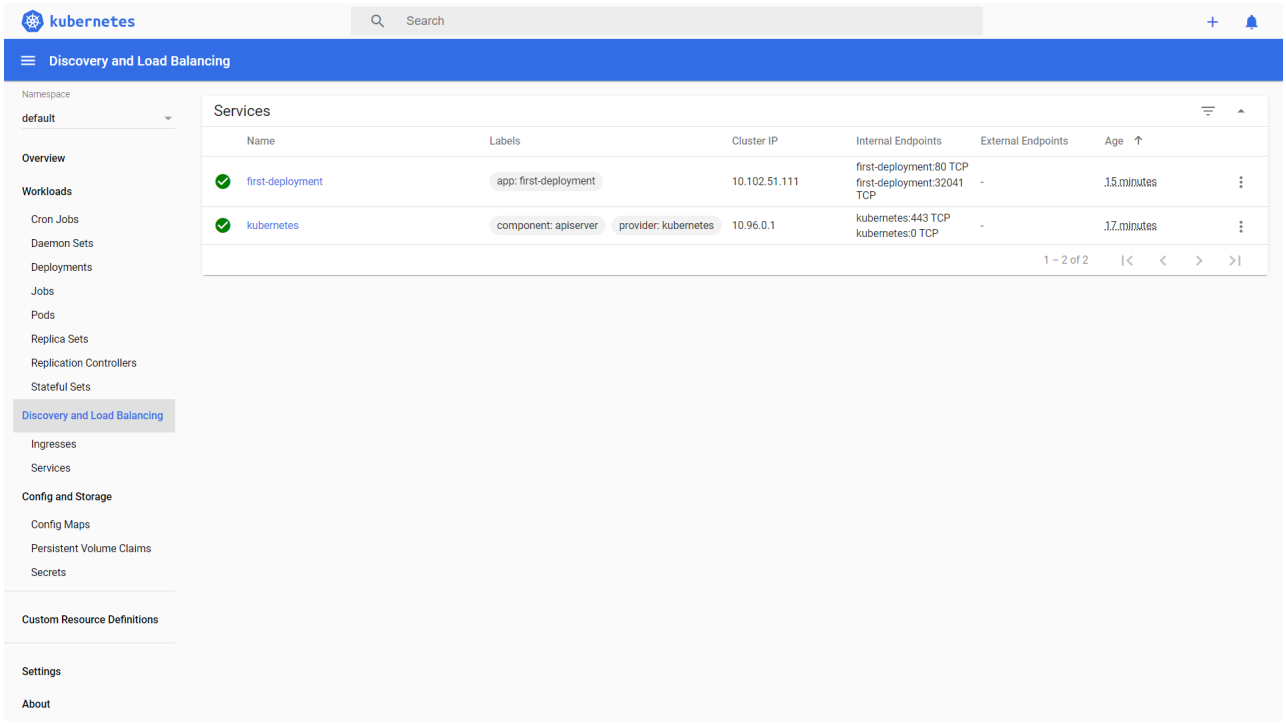
- bandomojo diegimo būsena:

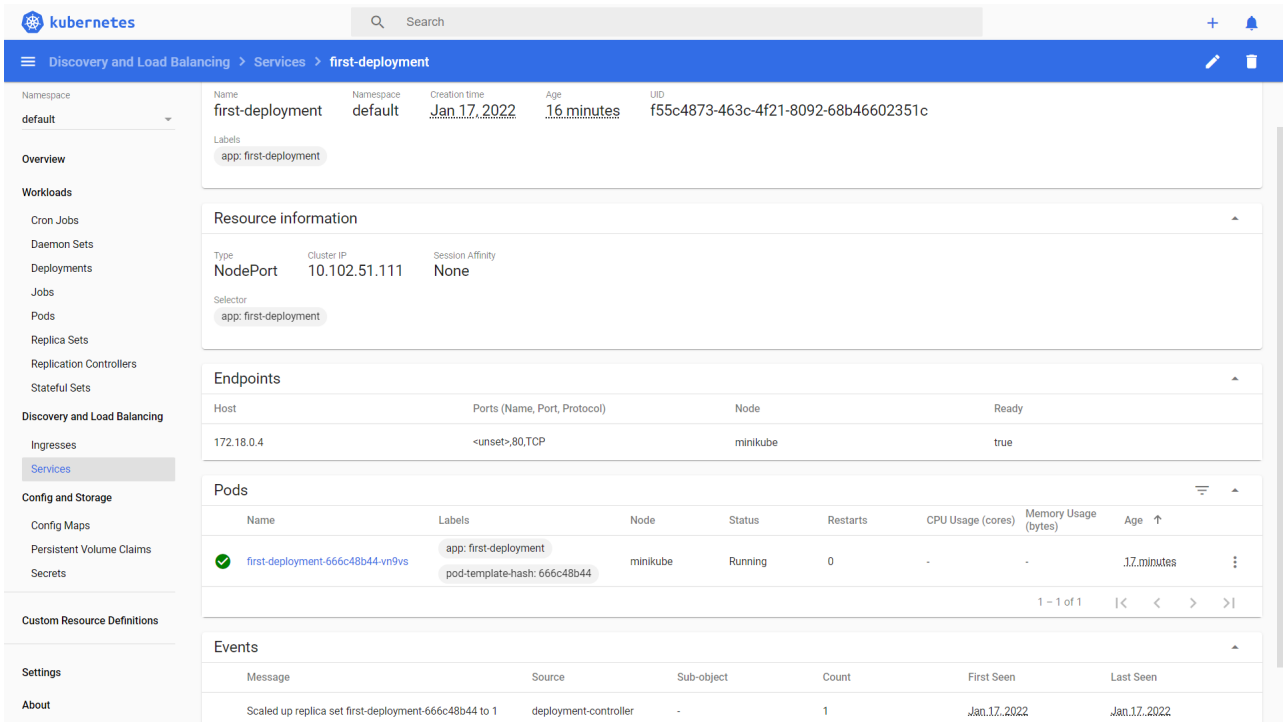


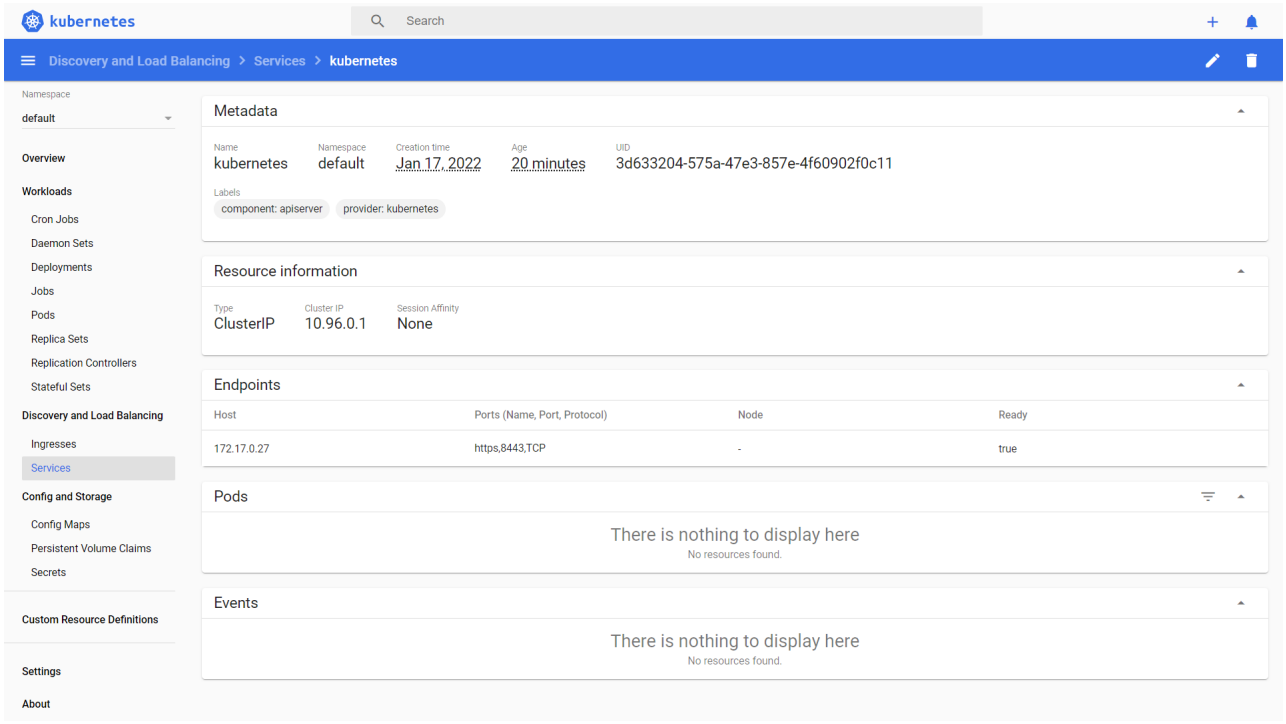
- jo „ankšties“ būsena:



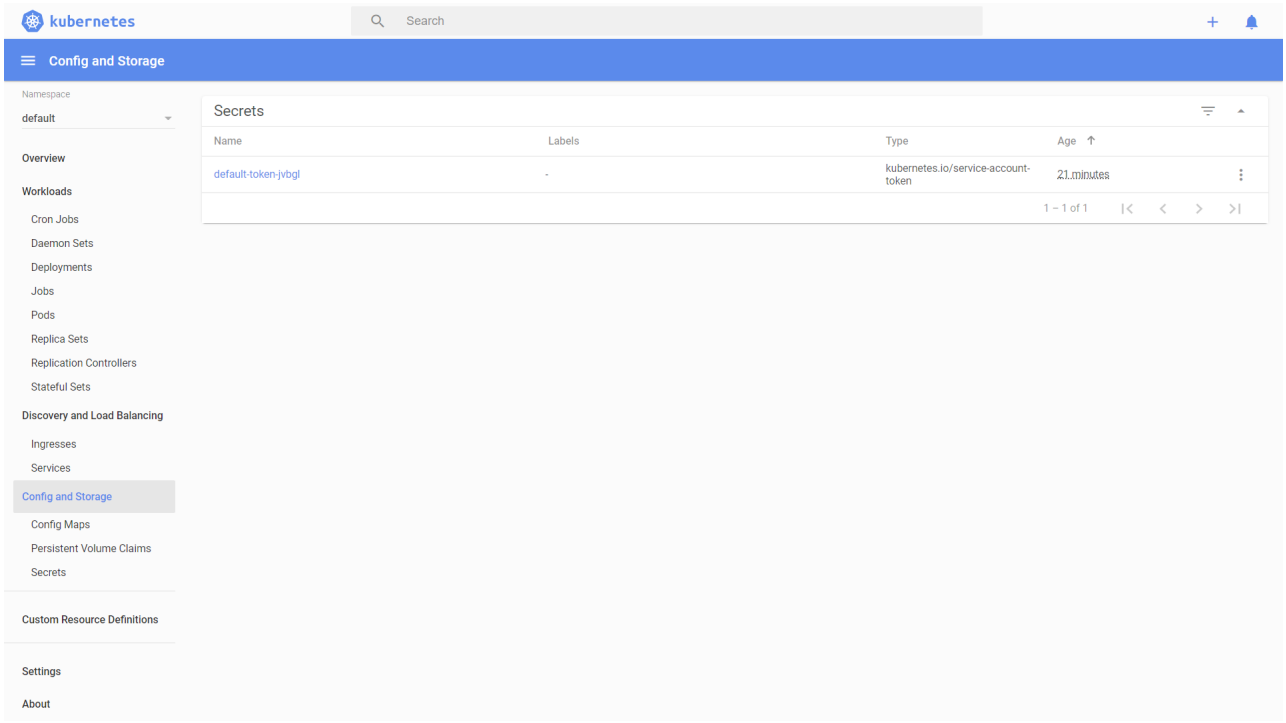
- paslaugų būsena, apkrovos balansavimas:

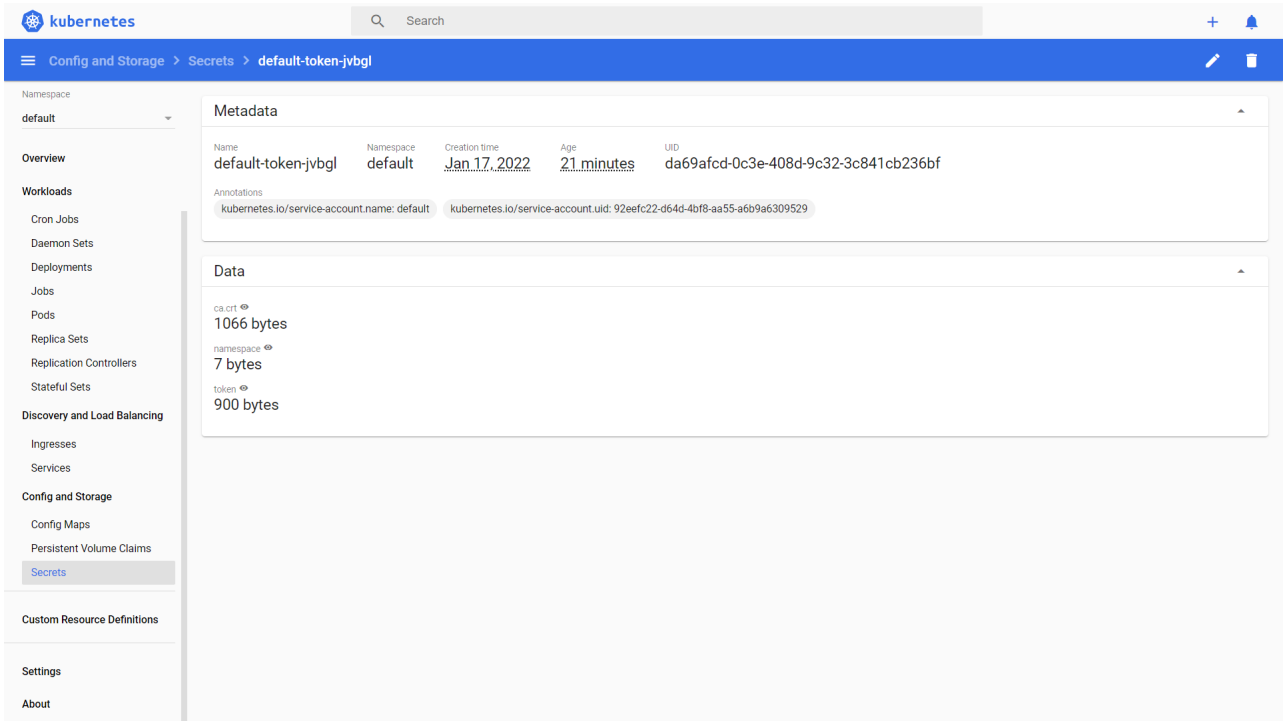

kubernetes
Search
Discovery and Load Balancing
Namespace
default
Overview
Workloads
Cron Jobs
Daemon Sets
Deployments
Jobs
Pods
Replica Sets
Replication Controllers
Stateful Sets
Discovery and Load Balancing
Ingresses
Services
Config and Storage
Config Maps
Persistent Volume Claims
Secrets
Custom Resource Definitions
Settings
About
Services
Name
Labels
Cluster IP
Internal Endpoints
External Endpoints
Age
first-deployment
app: first-deployment
10.102.51.111
first-deployment:80 TCP
first-deployment:32041 TCP
-
15 minutes
kubernetes
component: apiserver
provider: kubernetes
10.96.0.1
kubernetes:443 TCP
kubernetes:0 TCP
-
17 minutes
1 – 2 of 2


kubernetes
Search
Discovery and Load Balancing > Services > first-deployment
Namespace
default
Overview
Workloads
Cron Jobs
Daemon Sets
Deployments
Jobs
Pods
Replica Sets
Replication Controllers
Stateful Sets
Discovery and Load Balancing
Ingresses
Services
Config and Storage
Config Maps
Persistent Volume Claims
Secrets
Custom Resource Definitions
Settings
About
Name
first-deployment
Namespace
default
Creation time
Jan 17, 2022
Age
16 minutes
UID
f55c4873-463c-4f21-8092-68b46602351c
Labels
app: first-deployment
Resource information
Type
NodePort
Cluster IP
10.102.51.111
Session Affinity
None
Selector
app: first-deployment
Endpoints
Host
Ports (Name, Port, Protocol)
Node
Ready
172.18.0.4
<unset>,80,TCP
minikube
true
Pods
Name
Labels
Node
Status
Restarts
CPU Usage (cores)
Memory Usage (bytes)
Age
first-deployment-666c48b44-vn9vs
app: first-deployment
pod-template-hash: 666c48b44
minikube
Running
0
-
-
17 minutes
1 – 1 of 1
Events
Message
Source
Sub-object
Count
First Seen
Last Seen
Scaled up replica set first-deployment-666c48b44 to 1
deployment-controller
-
1
Jan 17, 2022
Jan 17, 2022



- vardų srities konfigūracija ir talpinimas:





Suvestinė nr. 1:

- Panaudojau `minikube` bei `kubectl` komandas (jų subkomandas) ir:
  1. startavau vieno mazgo Kubernetes miniklasterį;
     (atskiroje VM, pasak gido)
  2. patikrinau klasterio būseną: veikiantis;
  3. sukūriau konteinerį pagal `katacoda/docker-http-server` atvaizdą;
     (tik vaizdo įraše padariau klaidą įterpdamas vieną papildomą raidę: `katacon`**`n`**`da`)
  4. patikrinau diegimo „ankštį“: ji susikūrė konteinerį ir veikia;
  5. paviešinau konteinerinę paslaugą tinkle atskiru `31900/TCP` portu;
  6. prisijungiau šiuo portu su `curl` ir patikrinau paslaugos būseną: veikia;
  7. įdiegiau ir startavau *Minicube* priedą — Web sąsają *Dashboard*
  8. bei patikrinau klasterio būseną joje naudodamasis savo naršykle.
     (Nuoroda Web prisijungimui pateikė pats *katacoda.com* gidas)
- *Dashboard* interfeisas *Overview* skiltyje pasirenka `default` vardų sritį (*Namespace*):
  1. joje nematyti savo paties „ankščių“ (*Pods*):
     `kubernetes-dashboard-79d9cd965-7f5pb` ,
     `dashboard-metrics-scraper-7b64584c5c-7x46c`
  2. pastarosios tampa matomos pasirinkus `All namespaces` vardų sritį.

Užduotis nr. 2: "Deploy Containers Using Kubectl (https://www.katacoda.com/courses/kubernetes/kubectl-run-containers)"

Įvadas

Mokinsimės *Kubectl* pagalba kurti ir startuoti įdiegimus, replikavimo valdiklius ir viešinti juos kaip paslaugas. Čia nenaudosime YAML apibrėžčių. Šis būdas klasteryje įgalina sparčiai pradėti konteinerius kūrimą ir jų vykdymą.

Žingsnis 2.1: startuojame Kubernetes klasterį

- startuojame klasterį ir įgaliname Kubectl CLI:

```
$ minikube start --wait=false
* minikube v1.8.1 on Ubuntu 18.04
* Using the none driver based on user configuration

* Running on localhost (CPUs=2, Memory=2460MB, Disk=145651MB) ...
* OS release is Ubuntu 18.04.4 LTS

* Preparing Kubernetes v1.17.3 on Docker 19.03.6 ...
  - kubelet.resolv-conf=/run/systemd/resolve/resolv.conf
* Launching Kubernetes ...

* Enabling addons: default-storageclass, storage-provisioner
* Configuring local host environment ...
* Done! kubectl is now configured to use "minikube"
$
```

asciinema.org/a/462314 (https://asciinema.org/a/462314?autoplay=1)

- patikriname mazgo būseną:

```
$ kubectl get nodes
NAME       STATUS     ROLES    AGE   VERSION
minikube   NotReady   master   15s   v1.17.3
$
$ kubectl get nodes
NAME       STATUS   ROLES    AGE   VERSION
minikube   Ready    master   23s   v1.17.3
$
```

asciinema.org/a/462317 (https://asciinema.org/a/462317?autoplay=1)

Žingsnis 2.2: vykdome `kubectl` su `run`

- sukuriame įdiegimą ir startuojame jo „ankštis“ bei konteinerius:

```
$ kubectl run http --image=katacoda/docker-http-server:latest --replicas=1
kubectl run --generator=deployment/apps.v1 is DEPRECATED and will be removed in a future version.
Use kubectl run --generator=run-pod/v1 or kubectl create instead.
deployment.apps/http created
$
```

asciinema.org/a/462319 (https://asciinema.org/a/462319?autoplay=1)

- tikriname įdiegimų būsenas:

```
$ kubectl get deployments
NAME   READY   UP-TO-DATE   AVAILABLE   AGE
http   0/1     1            0           35s
$
$ kubectl get deployments
NAME   READY   UP-TO-DATE   AVAILABLE   AGE
http   0/1     1            0           39s
$
$ kubectl get deployments
NAME   READY   UP-TO-DATE   AVAILABLE   AGE
http   0/1     1            0           42s
$
$ kubectl get deployments
NAME   READY   UP-TO-DATE   AVAILABLE   AGE
http   0/1     1            0           47s
$
$ kubectl get deployments
NAME   READY   UP-TO-DATE   AVAILABLE   AGE
http   1/1     1            1           55s
$
```

asciinema.org/a/462320 (https://asciinema.org/a/462320?autoplay=1)

- tikriname išsamų įdiegimo aprašą:

```
$ kubectl describe deployment http
Name:                   http
Namespace:              default
CreationTimestamp:      Mon, 17 Jan 2022 18:02:36 +0000
Labels:                 run=http
Annotations:            deployment.kubernetes.io/revision: 1
Selector:               run=http
Replicas:               1 desired | 1 updated | 1 total | 1 available | 0 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
  Labels:  run=http
  Containers:
   http:
    Image:        katacoda/docker-http-server:latest
    Port:         <none>
    Host Port:    <none>
    Environment:  <none>
    Mounts:       <none>
  Volumes:        <none>
Conditions:
  Type           Status  Reason
  ----           ------  ------
  Available      True    MinimumReplicasAvailable
  Progressing    True    NewReplicaSetAvailable
OldReplicaSets:  <none>
NewReplicaSet:   http-774bb756bb (1/1 replicas created)
Events:
  Type    Reason             Age   From                   Message
  ----    ------             ----  ----                   -------
  Normal  ScalingReplicaSet  76s   deployment-controller  Scaled up replica set http-774bb756bb to
1
$
```

asciinema.org/a/462321 (https://asciinema.org/a/462321?autoplay=1)

Žingsnis 2.3: vykdome `kubectl` su `expose`

- sukuriame paslaugą paviešindami konkretų konteinerio portą:

```
$ kubectl expose deployment http --external-ip="172.17.0.11" --port=8000 --target-port=80
service/http exposed
$
```

asciinema.org/a/462325 (https://asciinema.org/a/462325?autoplay=1)

- patikriname paslaugos veikimą:

```
$ curl http://172.17.0.11:8000
<h1>This request was processed by host: http-774bb756bb-bbvm9</h1>
$
```

asciinema.org/a/462326 (https://asciinema.org/a/462326?autoplay=1)

Žingsnis 2.4: vykdome `kubectl` su `run` + `expose` iškart

- sukuriame naują įdiegimą ir paviešiname naują paslaugą kitu portu vienu ypu, kitu būdu:

```
$ kubectl run httpexposed --image=katacoda/docker-http-server:latest --replicas=1 --port=80 --
hostport=8001
kubectl run --generator=deployment/apps.v1 is DEPRECATED and will be removed in a future version.
Use kubectl run --generator=run-pod/v1 or kubectl create instead.
deployment.apps/httpexposed created
$
```

asciinema.org/a/462331 (https://asciinema.org/a/462331?autoplay=1)

- patikriname naujos paslaugos veikimą:

```
$ curl http://172.17.0.11:8001
<h1>This request was processed by host: httpexposed-68cb8c8d4-d9b6w</h1>
$
```

asciinema.org/a/462333 (https://asciinema.org/a/462333?autoplay=1)

- tikriname, ar naujas portas tikrai neatsirado paslaugų sąraše:

```
$ kubectl get svc
NAME         TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE
http         ClusterIP   10.96.205.142   172.17.0.11   8000/TCP   19m
kubernetes   ClusterIP   10.96.0.1       <none>        443/TCP    39m
$
```

asciinema.org/a/462336 (https://asciinema.org/a/462336?autoplay=1)

- tikriname, ar naujas portas atsirado tos pačios „ankšties“ tinkliniame konteineryje `pause`: (per *Docker Port Mapping* mechanizmą)

```
$ docker ps | grep httpexposed
5945f9a4fa9b        katacoda/docker-http-server   "/app"                   10 minutes ago      Up
10 minutes                                 k8s_httpexposed_httpexposed-68cb8c8d4-
d9b6w_default_f2718b05-501c-4158-8d8e-0e4a62e99db9_0
6cc613c77542        k8s.gcr.io/pause:3.1          "/pause"                 10 minutes ago      Up
10 minutes       0.0.0.0:8001->80/tcp   k8s_POD_httpexposed-68cb8c8d4-d9b6w_default_f2718b05-501c-
4158-8d8e-0e4a62e99db9_0
$
$ # OK
$
$ docker ps | wc -l
21
```

asciinema.org/a/462338 (https://asciinema.org/a/462338?autoplay=1)

Žingsnis 2.5: dauginame konteinerius

- pakeliame „ankščių“ skaičių iki 3:

```
$ kubectl scale --replicas=3 deployment http
deployment.apps/http scaled
$
```

asciinema.org/a/462340 (https://asciinema.org/a/462340?autoplay=1)

- tikriname „ankščių“ būsenas:

```
$ kubectl get pods
NAME                         READY   STATUS    RESTARTS   AGE
http-774bb756bb-bbvm9        1/1     Running   0          43m
httpexposed-68cb8c8d4-d9b6w  1/1     Running   0          18m
$
$ kubectl get pods
NAME                         READY   STATUS    RESTARTS   AGE
http-774bb756bb-bbvm9        1/1     Running   0          51m
http-774bb756bb-jcbgf        1/1     Running   0          7m50s
http-774bb756bb-qvqkc        1/1     Running   0          7m50s
httpexposed-68cb8c8d4-d9b6w  1/1     Running   0          26m
```

asciinema.org/a/462341 (https://asciinema.org/a/462341?autoplay=1)

- tikriname, ar „ankštys“ pateko į apkrovos balansavimą šiai paslaugai:

```
$ kubectl describe svc http
Name:              http
Namespace:         default
Labels:            run=http
Annotations:       <none>
Selector:          run=http
Type:              ClusterIP
IP:                10.96.205.142
External IPs:      172.17.0.11
Port:              <unset>  8000/TCP
TargetPort:        80/TCP
Endpoints:         172.18.0.4:80,172.18.0.6:80,172.18.0.7:80
Session Affinity:  None
Events:            <none>
$
```

asciinema.org/a/462343 (https://asciinema.org/a/462343?autoplay=1)

- atliekame kelias tos pačios paslaugos užklausas iš eilės:

```
$ curl http://172.17.0.11:8000
<h1>This request was processed by host: http-774bb756bb-bbvm9</h1>
$
$ curl http://172.17.0.11:8000
<h1>This request was processed by host: http-774bb756bb-qvqkc</h1>
$
$ # OK, kitas hosto ID
$
$ curl http://172.17.0.11:8000
<h1>This request was processed by host: http-774bb756bb-bbvm9</h1>
$
$ curl http://172.17.0.11:8000
<h1>This request was processed by host: http-774bb756bb-jcbgf</h1>
$
$ # dar vienas naujas hosto ID
$
$ curl http://172.17.0.11:8000
<h1>This request was processed by host: http-774bb756bb-qvqkc</h1>
$
```

asciinema.org/a/462344 (https://asciinema.org/a/462344?autoplay=1)

Suvestinė nr. 2:

- Panaudojau `kubectl` komandas (ir subkomandas), ir:
  1. startavau klasterį, įgalinau Kubectl CLI;
  2. patikrinau mazgo būseną: veikia;
  3. sukūriau įdiegimą su viena replika komandos `kubectl run …` pagalba;
  4. patikrinau HTTP paslaugos įdiegimo būseną: pradėjo veikti;
  5. patikrinau išsamų įdiegimo aprašą: atitinka planą;
  6. sukūriau paslaugą paviešindamas HTTP portą kaip `8000/TCP`;
  7. patikrinau paslaugos veikimą: atsiliepia be klaidų;
  8. sukūriau naują HTTP paslaugos diegimą kitu būdu — iškart viešinant paslaugos portą;
  9. šįkart HTTP portas yra `8001/TCP`;
  10. patikrinau paslaugos veikimą: atsiliepia irgi;
  11. patikrinau paslaugų sąrašą: naujojo porto nematyti;
  12. patikrinau konteinerių sąrašą su Docker komanda:
      naujasis portas priklauso "k8s.gcr.io/pause" tipo konteineriui;
  13. pakėliau pirmosios paslaugos „ankšties“ kopijų skaičių nuo 1 iki 3;
  14. tikrinau jų būsenas ir sulaukiau, kol startuos dvi papildonos;
  15. įsitikinau, kad visų trijų paslaugos „ankščių“ HTTP-portai pateko į apkrovos balansavimą;
  16. atlikau šiai paslaugai keletą užklausų iš eilės:
      įsitikinau, kad atsako skirtingas Host ID (iš trijų galimų);
  17. tyrimas baigtas.
- `kubectl run --image= …` komanda pyksta dėl *Deprecated* opcijos `--generator`, nors aš tokios nenaudojau.
  Ir rekomenduoja naudoti vieną iš dviejų kitokių komandų.

  ⇒ Turbūt verta parašyti `katacoda` treniruoklio autoriams, kad atėjo metas atnaujinti instrukcijas. :)
- Tikėtina, kad *Docker Port Mapping* mechanizmas veikia būtent taip minima punkte nr. 12.

Tačiau nežinau, kaip įsitikinti garantuotai, kad jis čia panaudotas.

- Pasigedau veiksmo, kuriame būtume kurę replikavimo valdiklius, kaip žadėta užduoties aprašyme.

Užduotis nr. 3: "Deploy Containers Using YAML (https://www.katacoda.com/courses/kubernetes/creating-kubernetes-yaml-definitions)"

Įvadas

Mokinsimės Kubectl pagalba kurti ir startuoti įdiegimus, replikavimo valdiklius ir viešinti juos kaip paslaugas šįkart *jau* pasinaudojant YAML apibrėžtimis (YAML formatu).

YAML apibrėžtimis aprašomi Kubernetes objektai, paskirti įdiegimams. Taip pat bus ir galimybė keičiantis konfigūracijai šiuos objektus atnaujinti bei perdiegti į klasterį iš naujo.

Žingsnis 3.1: įdiegimo kūrimas

- automatinis klasterio startas su *Shell*:

```
Your Interactive Bash Terminal. A safe place to learn and execute commands.

$ minikube start --wait=false
* minikube v1.8.1 on Ubuntu 18.04
* Using the none driver based on user configuration
* Running on localhost (CPUs=2, Memory=2460MB, Disk=145651MB) ...
* OS release is Ubuntu 18.04.4 LTS
* Preparing Kubernetes v1.17.3 on Docker 19.03.6 ...
  - kubelet.resolv-conf=/run/systemd/resolve/resolv.conf
* Launching Kubernetes ...
* Enabling addons: default-storageclass, storage-provisioner
* Configuring local host environment ...
* Done! kubectl is now configured to use "minikube"
$
```

- įkeliu YAML šabloną `deployment.yaml`:

```
$ ls -l
total 8
-rw-r--r-- 1 root root  335 Jan 17 22:37 deployment.yaml
drwxr-xr-x 2 root root 4096 Mar  1  2020 Desktop

$ cat deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: webapp1
spec:
  replicas: 1
  selector:
    matchLabels:
      app: webapp1
  template:
    metadata:
      labels:
        app: webapp1
    spec:
      containers:
      - name: webapp1
        image: katacoda/docker-http-server:latest
        ports:
        - containerPort: 80
$
```

asciinema.org/a/462387 (https://asciinema.org/a/462387?autoplay=1)

- į klasterį diegiu aplikaciją `webapp1` iš Doker atvaizdo `katacoda/docker-http-server:latest`:

```
$ kubectl create -f deployment.yaml
deployment.apps/webapp1 created
$
```

asciinema.org/a/462388 (https://asciinema.org/a/462388?autoplay=1)

- peržiūriu įdiegimų sąrašą:

```
$ kubectl get deployment
NAME      READY   UP-TO-DATE   AVAILABLE   AGE
webapp1   1/1     1            1           4m37s
$
```

asciinema.org/a/462389 (https://asciinema.org/a/462389?autoplay=1)

- peržiūriu `webapp1` įdiegimo aprašą:

```
$ kubectl describe deployment webapp1
Name:                   webapp1
Namespace:              default
CreationTimestamp:      Mon, 17 Jan 2022 22:45:45 +0000
Labels:                 <none>
Annotations:            deployment.kubernetes.io/revision: 1
Selector:               app=webapp1
Replicas:               1 desired | 1 updated | 1 total | 1 available | 0 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
  Labels:  app=webapp1
  Containers:
   webapp1:
    Image:        katacoda/docker-http-server:latest
    Port:         80/TCP
    Host Port:    0/TCP
    Environment:  <none>
    Mounts:       <none>
  Volumes:        <none>
Conditions:
  Type           Status  Reason
  ----           ------  ------
  Available      True    MinimumReplicasAvailable
  Progressing    True    NewReplicaSetAvailable
OldReplicaSets:  <none>
NewReplicaSet:   webapp1-6b54fb89d9 (1/1 replicas created)
Events:
  Type    Reason             Age    From                   Message
  ----    ------             ----   ----                   -------
  Normal  ScalingReplicaSet  6m55s  deployment-controller  Scaled up replica set webapp1-6b54fb89d9
to 1
$
```

asciinema.org/a/462390 (https://asciinema.org/a/462390?autoplay=1)

Žingsnis 3.2: paslaugos kūrimas

- įkeliu YAML šabloną `service.yaml`:

```
$ ls -l
total 12
-rw-r--r-- 1 root root  335 Jan 17 22:37 deployment.yaml
drwxr-xr-x 2 root root 4096 Mar  1  2020 Desktop
-rw-r--r-- 1 root root  180 Jan 17 22:57 service.yaml

$ cat service.yaml
apiVersion: v1
kind: Service
metadata:
  name: webapp1-svc
  labels:
    app: webapp1
spec:
  type: NodePort
  ports:
  - port: 80
    nodePort: 30080
  selector:
    app: webapp1
$
```

asciinema.org/a/462391 (https://asciinema.org/a/462391?autoplay=1)

- įdiegiu paslaugą:

```
$ kubectl create -f service.yaml
service/webapp1-svc created
$
```

asciinema.org/a/462392 (https://asciinema.org/a/462392?autoplay=1)

- peržiūriu įdiegtų paslaugų sąrašą:

```
$ kubectl get svc
NAME          TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
kubernetes    ClusterIP   10.96.0.1       <none>        443/TCP        28m
webapp1-svc   NodePort    10.105.23.172   <none>        80:30080/TCP   100s
$
```

asciinema.org/a/462393 (https://asciinema.org/a/462393?autoplay=1)

- peržiūriu paslaugos `webapp1-svc` aprašą:

```
$ kubectl describe svc webapp1-svc
Name:                     webapp1-svc
Namespace:                default
Labels:                   app=webapp1
Annotations:              <none>
Selector:                 app=webapp1
Type:                     NodePort
IP:                       10.105.23.172
Port:                     <unset>  80/TCP
TargetPort:               80/TCP
NodePort:                 <unset>  30080/TCP
Endpoints:                172.18.0.4:80
Session Affinity:         None
External Traffic Policy:  Cluster
Events:                   <none>
$
```

asciinema.org/a/462397 (https://asciinema.org/a/462397?autoplay=1)

- tikrinu paslaugos veikimą:

```
$ curl host01:30080
<h1>This request was processed by host: webapp1-6b54fb89d9-qz98l</h1>
$
$ curl host01:30080
<h1>This request was processed by host: webapp1-6b54fb89d9-qz98l</h1>
$
$ curl host01:30080
<h1>This request was processed by host: webapp1-6b54fb89d9-qz98l</h1>
$
$ curl host01:30080
<h1>This request was processed by host: webapp1-6b54fb89d9-qz98l</h1>
$
$ curl host01:30080
<h1>This request was processed by host: webapp1-6b54fb89d9-qz98l</h1>
$
$ curl host01:30080
<h1>This request was processed by host: webapp1-6b54fb89d9-qz98l</h1>
$
$ curl host01:30080
<h1>This request was processed by host: webapp1-6b54fb89d9-qz98l</h1>
$
```

asciinema.org/a/462398 (https://asciinema.org/a/462398?autoplay=1)

Žingsnis 3.3: įdiegimo dauginimas

- replikų (egzempliorių) skaičių YAML šablone `deployment.yaml` pakeliu iki 4:

```
$ cp -v deployment.yaml deployment.yaml.OLD
'deployment.yaml' -> 'deployment.yaml.OLD'

$ vim deployment.yaml

$ ls -l
total 16
-rw-r--r-- 1 root root  335 Jan 17 23:15 deployment.yaml
-rw-r--r-- 1 root root  335 Jan 17 23:14 deployment.yaml.OLD
drwxr-xr-x 2 root root 4096 Mar  1  2020 Desktop
-rw-r--r-- 1 root root  180 Jan 17 22:57 service.yaml

$ diff -u deployment.yaml.OLD deployment.yaml | colordiff
--- deployment.yaml.OLD 2022-01-17 23:14:54.436000000 +0000
+++ deployment.yaml     2022-01-17 23:15:16.648000000 +0000
@@ -3,7 +3,7 @@
 metadata:
   name: webapp1
 spec:
-  replicas: 1
+  replicas: 4
   selector:
     matchLabels:
       app: webapp1
$
```

asciinema.org/a/462399 (https://asciinema.org/a/462399?autoplay=1)

- padauginu veikiančių replikų (egzempliorių) skaičių:

```
$ kubectl apply -f deployment.yaml
Warning: kubectl apply should be used on resource created by either kubectl create --save-config or
kubectl apply
deployment.apps/webapp1 configured
$
```

asciinema.org/a/462400 (https://asciinema.org/a/462400?autoplay=1)

- tikrinu įdiegimo / klasterio būseną:

```
$ kubectl get deployment
NAME      READY   UP-TO-DATE   AVAILABLE   AGE
webapp1   4/4     4            4           36m
$
```

asciinema.org/a/462402 (https://asciinema.org/a/462402?autoplay=1)

- tikrinu naujų „ankščių“ būseną:

```
$ kubectl get pods
NAME                      READY   STATUS    RESTARTS   AGE
webapp1-6b54fb89d9-27g4g   1/1     Running   0          7m50s
webapp1-6b54fb89d9-2v7vh   1/1     Running   0          7m50s
webapp1-6b54fb89d9-p8lck   1/1     Running   0          7m50s
webapp1-6b54fb89d9-qz98l   1/1     Running   0          40m
$
```

asciinema.org/a/462405 (https://asciinema.org/a/462405?autoplay=1)

- tikrinu užklausas į paslaugą:

```
$ curl host01:30080
<h1>This request was processed by host: webapp1-6b54fb89d9-2v7vh</h1>
$
$ curl host01:30080
<h1>This request was processed by host: webapp1-6b54fb89d9-27g4g</h1>
$
$ curl host01:30080
<h1>This request was processed by host: webapp1-6b54fb89d9-2v7vh</h1>
$
$ curl host01:30080
<h1>This request was processed by host: webapp1-6b54fb89d9-qz98l</h1>
$
$ curl host01:30080
<h1>This request was processed by host: webapp1-6b54fb89d9-27g4g</h1>
$
$ curl host01:30080
<h1>This request was processed by host: webapp1-6b54fb89d9-27g4g</h1>
$
$ curl host01:30080
<h1>This request was processed by host: webapp1-6b54fb89d9-p8lck</h1>
$
$ curl host01:30080
<h1>This request was processed by host: webapp1-6b54fb89d9-27g4g</h1>
$
$ curl host01:30080
<h1>This request was processed by host: webapp1-6b54fb89d9-p8lck</h1>
$
```

asciinema.org/a/462404 (https://asciinema.org/a/462404?autoplay=1)

Suvestinė nr. 3:

- Įvykdžiau diegimą pagal YAML šabloną (arba YAML apibrėžtį, angl. *definition*):
  1. gavau Shell su veikiančiu K8s miniklasteriu;
  2. įkėliau `deployment.yaml` šabloną;

3. pagal jį įdiegiau aplikaciją `webapp1` iš Docker atvaizdo `katacoda/docker-http-server`;
4. įsitikinau, kad įdiegimas pavyko;
5. peržiūrėjau jo aprašą, Host Port nenurodytas (`0/TCP`);
6. įkėliau `service.yaml` šabloną;
7. pagal jį įdiegiau HTTP paslaugą `webapp1-svc`;
8. įsitikinau, kad HTTP paslauga sukonfigūruota;
9. peržiūrėjau jos aprašą, `NodePort` prievadui priskirta `30080/TCP` reikšmė;
10. patikrinau paslaugos veikimą: ta pati „ankštis“ atsako į visas užklausas iš eilės;
11. padidinau replikų skaičių šablone `deployment.yaml` iki 4;
12. pritaikiau šabloną klasteriui su `kubectl apply` komanda;
13. patikrinau įdiegimo ir „ankščių“ būseną: skaičius pakilo iki 4;
14. patikrinau paslaugos veikimą: į užklausas atsako jau 4 skirtingos „ankštys“;
15. patikrinau paslaugos aprašą: yra visi 4 *Endpoints* (iš jų vienas neišvestas dėl teksto trumpumo); (šito ataskaitoje neiliustravau)
16. tyrimas baigtas.

- Naudojant YAML failus **tampa neaišku**:
  1. kodėl atsiranda `app` raktažodis ? (Panašu, kad vietoj anksčiau naudoto `run`)
  2. ką aprašo `spec.template` ir kas bus, jei **ne**nurodysiu `spec.template.metadata.labels.app` ? (Kai jau tas pat nurodyta pas `spec.selector.app`)
  3. kodėl įdiegimo apraše vardą `webapp1` reikia nurodyti net 4x:
     - `spec.metadata.name` ?
     - `spec.selector.matchLabels.app` ?
     - `spec.template.metadata.labels.app` ?
     - `spec.template.spec.containers.name` ?
  4. kodėl TCP portą `80` reikia nurodyti tiek įdiegimui (`spec.template.spec.containers.ports`), tiek paslaugai (`spec.ports.port`), kai per CLI pakakdavo nurodyti tik vieną sykį ?
  5. ar paslaugos `spec.selector.app` nurodo įdiegimo konteinerį, ar įdiegimo „ankštį“ (galvojant ne YAML scenarijaus sąvokomis) ?
  6. Išvada: YAML šablonai įneša painavios į anksčiau susidarytą pradinį supratimą apie K8s.
- Komanda `kubectl apply -f …` pyksta:
  `Warning: kubectl apply should be used on resource created by either kubectl create --save-config or kubectl apply`

  ⇒ Galbūt irgi vertėtų pranešti treniruoklio autoriams (dėl instrukcijų patikslinimo).

Užduotis nr. 4: "Deploy Guestbook example on Kubernetes (https://www.katacoda.com/courses/kubernetes/guestbook)"

Įvadas

Čia mokinsimės su Kubernetes ir Docker pagalba startuoti paprastą, bet daugiapakopę Web aplikaciją. Siūlomos „Svečių knygos“ aplikacijos pavyzdys išsaugos puslapio svečių žinutes *Redis* duomenų bazėje (DB) kviesdamas JavaScript API. *Redis* DB susideda iš *masterio* (duomenų talpinimui) ir rinkinio iš replikuotų Redis *tarnų*.

Numatoma aprėpti tokias esmines sąvokas:

- „ankštys“
- replikavimo valdikliai
- paslaugos
- *NodePort* prievadai

Jos sudaro *Kubernetes* pagrindą.

Žingsnis 4.1: klasterio startavimas

- *Shell* ir automatinis vienamazgio klasterio startas:

```
Your Interactive Bash Terminal. A safe place to learn and execute commands.

controlplane $ mkdir -p /root/tutorial; cd /root/tutorial; launch.sh
Waiting for Kubernetes to start...
Kubernetes started
controlplane $
```

- tikrinu klasterio būseną:

```
controlplane $ kubectl cluster-info
Kubernetes master is running at https://172.17.0.35:6443
KubeDNS is running at https://172.17.0.35:6443/api/v1/namespaces/kube-system/services/kube-
dns:dns/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.

controlplane $ kubectl get nodes
NAME           STATUS     ROLES    AGE   VERSION
controlplane   NotReady   master   19s   v1.14.0

controlplane $ kubectl get nodes
NAME           STATUS   ROLES    AGE    VERSION
controlplane   Ready    master   2m6s   v1.14.0
node01         Ready    <none>   79s    v1.14.0
controlplane $
```

Žingsnis 4.2: Redis *master* valdiklis

- peržiūriu YAML aprašą:

```
controlplane $ cat redis-master-controller.yaml
apiVersion: v1
kind: ReplicationController
metadata:
  name: redis-master
  labels:
    name: redis-master
spec:
  replicas: 1
  selector:
    name: redis-master
  template:
    metadata:
      labels:
        name: redis-master
    spec:
      containers:
      - name: master
        image: redis:3.0.7-alpine
        ports:
        - containerPort: 6379
controlplane $
```

- sukuriu ir startuoju Redis *masterio* replikacinį valdiklį:

```
controlplane $ kubectl create -f redis-master-controller.yaml
replicationcontroller/redis-master created
controlplane $
```

- tikrinu replikacinių valdiklių būseną:

```
controlplane $ kubectl get rc
NAME           DESIRED   CURRENT   READY   AGE
redis-master   1         1         0       5s

controlplane $ kubectl get rc
NAME           DESIRED   CURRENT   READY   AGE
redis-master   1         1         1       18s
controlplane $
```

- tikrinu „ankščių“ / konteinerių būseną:

```
controlplane $ kubectl get pods
NAME                 READY   STATUS    RESTARTS   AGE
redis-master-bv75w   1/1     Running   0          4m1s
controlplane $
```

Žingsnis 4.3: Redis *master* paslauga

- peržiūriu YAML aprašą:

```
controlplane $ cat redis-master-service.yaml
apiVersion: v1
kind: Service
metadata:
  name: redis-master
  labels:
    name: redis-master
spec:
  ports:
    # the port that this service should serve on
  - port: 6379
    targetPort: 6379
  selector:
    name: redis-master
controlplane $
```

- sukuriu ir startuoju Redis *masterio* paslaugą:

```
controlplane $ kubectl create -f redis-master-service.yaml
service/redis-master created
controlplane $
```

- tikrinu paslaugos būseną:

```
controlplane $ kubectl get services
NAME           TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)    AGE
kubernetes     ClusterIP   10.96.0.1      <none>        443/TCP    6m2s
redis-master   ClusterIP   10.98.222.56   <none>        6379/TCP   4s
controlplane $
```

- peržiūriu Redis *masterio* paslaugos aprašą:

```
controlplane $ kubectl describe services redis-master
Name:              redis-master
Namespace:         default
Labels:            name=redis-master
Annotations:       <none>
Selector:          name=redis-master
Type:              ClusterIP
IP:                10.98.222.56
Port:              <unset>  6379/TCP
TargetPort:        6379/TCP
Endpoints:         10.88.0.5:6379
Session Affinity:  None
Events:            <none>
controlplane $
```

Žingsnis 4.4: Redis *tarnų* valdiklis

- peržiūriu YAML aprašą:

```
controlplane $ cat redis-slave-controller.yaml
apiVersion: v1
kind: ReplicationController
metadata:
  name: redis-slave
  labels:
    name: redis-slave
spec:
  replicas: 2
  selector:
    name: redis-slave
  template:
    metadata:
      labels:
        name: redis-slave
    spec:
      containers:
      - name: worker
        image: gcr.io/google_samples/gb-redisslave:v1
        env:
        - name: GET_HOSTS_FROM
          value: dns
          # If your cluster config does not include a dns service, then to
          # instead access an environment variable to find the master
          # service's host, comment out the 'value: dns' line above, and
          # uncomment the line below.
          # value: env
        ports:
        - containerPort: 6379
controlplane $
```

- sukuriu ir startuoju Redis *tarnų* replikacinį valdiklį:

```
controlplane $ kubectl create -f redis-slave-controller.yaml
replicationcontroller/redis-slave created
controlplane $
```

- tikrinu replikacinių valdiklių būseną:

```
controlplane $ kubectl get rc
NAME           DESIRED   CURRENT   READY   AGE
redis-master   1         1         1       6m3s
redis-slave    2         2         0       3s

controlplane $ kubectl get rc
NAME           DESIRED   CURRENT   READY   AGE
redis-master   1         1         1       6m5s
redis-slave    2         2         2       5s
controlplane $
```

Žingsnis 4.5: Redis *tarnų* paslauga

- peržiūriu YAML aprašą:

```
controlplane $ cat redis-slave-service.yaml
apiVersion: v1
kind: Service
metadata:
  name: redis-slave
  labels:
    name: redis-slave
spec:
  ports:
    # the port that this service should serve on
  - port: 6379
  selector:
    name: redis-slave
controlplane $
```

- sukuriu ir startuoju Redis *tarnų* paslaugą:

```
controlplane $ kubectl create -f redis-slave-service.yaml
service/redis-slave created
controlplane $
```

- tikrinu paslaugos būseną:

```
controlplane $ kubectl get services
NAME           TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)    AGE
kubernetes     ClusterIP   10.96.0.1      <none>        443/TCP    8m10s
redis-master   ClusterIP   10.98.222.56   <none>        6379/TCP   2m12s
redis-slave    ClusterIP   10.98.98.227   <none>        6379/TCP   1s
controlplane $
```

- patikrinu ir „ankščių“ būsenas:

```
controlplane $ kubectl get pods
NAME                 READY   STATUS    RESTARTS   AGE
redis-master-bv75w   1/1     Running   0          7m23s
redis-slave-bfzm9    1/1     Running   0          83s
redis-slave-f5f9f    1/1     Running   0          83s
controlplane $
```

Žingsnis 4.6: Frontendas — replikacinis valdiklis ir jo „ankštys“

- peržiūriu YAML aprašą:

```
controlplane $ cat frontend-controller.yaml
apiVersion: v1
kind: ReplicationController
metadata:
  name: frontend
  labels:
    name: frontend
spec:
  replicas: 3
  selector:
    name: frontend
  template:
    metadata:
      labels:
        name: frontend
    spec:
      containers:
      - name: php-redis
        image: gcr.io/google_samples/gb-frontend:v3
        env:
        - name: GET_HOSTS_FROM
          value: dns
          # If your cluster config does not include a dns service, then to
          # instead access environment variables to find service host
          # info, comment out the 'value: dns' line above, and uncomment the
          # line below.
          # value: env
        ports:
        - containerPort: 80
controlplane $
```

- sukuriu ir startuoju replikacinį valdiklį pagal `gcr.io/google_samples/gb-frontend` atvaizdą:

```
controlplane $ kubectl create -f frontend-controller.yaml
replicationcontroller/frontend created
controlplane $
```

- tikrinu replikacinių valdiklių būsenas:

```
controlplane $ kubectl get rc
NAME           DESIRED   CURRENT   READY   AGE
frontend       3         3         3       88s
redis-master   1         1         1       12m
redis-slave    2         2         2       6m53s
controlplane $ kubectl get pods
```

- tikrinu „ankščių“ būsenas:

```
controlplane $ kubectl get pods
NAME                 READY   STATUS    RESTARTS   AGE
frontend-ctpql       1/1     Running   0          93s
frontend-dwkqh       1/1     Running   0          93s
frontend-g998c       1/1     Running   0          93s
redis-master-bv75w   1/1     Running   0          12m
redis-slave-bfzm9    1/1     Running   0          6m58s
redis-slave-f5f9f    1/1     Running   0          6m58s
```

Žingsnis 4.7: „Svečių knygos“ frontendinė paslauga

- peržiūriu YAML aprašą:

```
controlplane $ cat frontend-service.yaml
apiVersion: v1
kind: Service
metadata:
  name: frontend
  labels:
    name: frontend
spec:
  # if your cluster supports it, uncomment the following to automatically create
  # an external load-balanced IP for the frontend service.
  # type: LoadBalancer
  type: NodePort
  ports:
    # the port that this service should serve on
    - port: 80
      nodePort: 30080
  selector:
    name: frontend
controlplane $
```

- sukuriu ir startuoju frontedinę paslaugą:

```
controlplane $ kubectl create -f frontend-service.yaml
service/frontend created
controlplane $
```

- tikrinu frontendinės paslaugos būseną:

```
controlplane $ kubectl get services
NAME           TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)        AGE
frontend       NodePort    10.96.81.216   <none>        80:30080/TCP   114s
kubernetes     ClusterIP   10.96.0.1      <none>        443/TCP        45m
redis-master   ClusterIP   10.98.222.56   <none>        6379/TCP       39m
redis-slave    ClusterIP   10.98.98.227   <none>        6379/TCP       37m
controlplane $
```

Žingsnis 4.8: Jungiamės į „Svečių knygos“ frontendą

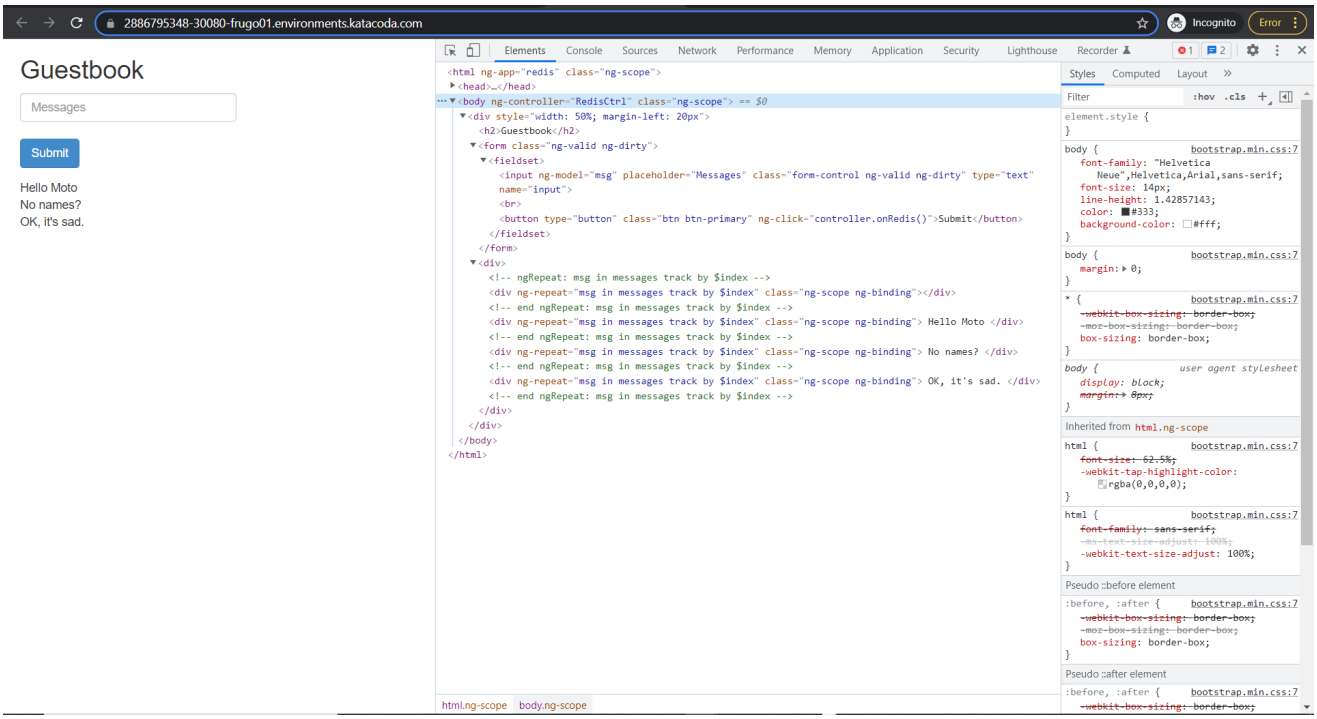
- tikrinu visų „ankščių“ būsenas (įsk. ir frontendines):

```
controlplane $ kubectl get pods
NAME                 READY   STATUS    RESTARTS   AGE
frontend-ctpql       1/1     Running   0          43m
frontend-dwkqh       1/1     Running   0          43m
frontend-g998c       1/1     Running   0          43m
redis-master-bv75w   1/1     Running   0          55m
redis-slave-bfzm9    1/1     Running   0          49m
redis-slave-f5f9f    1/1     Running   0          49m
controlplane $
```

- išsifiltruoju frontendinės paslaugos *NodePort* prievadą:

```
controlplane $ kubectl describe service frontend | grep NodePort:
NodePort:                 <unset>  30080/TCP
controlplane $
```

- tikrinu Web-aplikaciją tiesiogiai (URL gautas iš treniruoklio puslapio): https://2886795348-30080-frugo01.environments.katacoda.com/



Suvestinė nr. 4:

- Įvykdžiau diegimą pagal YAML šabloną (arba YAML apibrėžtį, angl. *definition*):
  1. gavau Shell su veikiančiu *K8s* miniklasteriu;
  2. taip pat gavau šešis YAML šablonus:
     - `redis-master-controller.yaml`
     - `redis-master-service.yaml`
     - `redis-slave-controller.yaml`
     - `redis-slave-service.yaml`
     - `frontend-controller.yaml`
     - `frontend-service.yaml`
  3. pagal juos įdiegiau:
     - `redis-master` replikacinį valdiklį iš Docker atvaizdo `redis:3.0.7-alpine`;
     - `redis-master` paslaugą (1 vnt.);
     - `redis-slave` replikacinį valdiklį iš Docker atvaizdo `gb-redisslave:v1`;
     - `redis-slave` paslaugą (2 vnt.);
     - `frontend` replikacinį valdiklį iš Docker atvaizdo `gb-frontend:v3`;
     - `frontend` paslaugą (3 vnt.);
  4. patikrinau būsenas:
     - replikacinių valdiklių,
     - paslaugų
     - „ankščių“,

       — veikia 3 valdikliai, 3 paslaugos ir 6 „ankštys“.
  5. patikrinau paslaugos veikimą iš išorinio interneto — veikia puikiai (tik primityviai);
  6. tyrimas baigtas.

- Naudojant YAML failus man **tampa išvis neaišku**, kaip konfigūruoti moduliai sąveikauja žemame lygmenyje.
  Suvokiu tik abstraktų vaizdą, ir visiškai neaišku, kaip reikėtų tikrinti srautus / paslaugų strigimus įprastinėmis OS priemonėmis.

Laboratorinio darbo išvados

Minimaliai susipažinta su Kubernetes platforma.