Vedlegg G - Fremgangsmåte test tre

Innhold:

1 Innledning

Denne testen er basert på HorizontalPodAutoscaler Walkthrough *(https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale-walkthrough/)*, og er en detaljert veiledning i hvordan test tre ble gjennomført!

> **MERK:** *Se Vedlegg B - Kildekode for å se hvor og hvilke versjoner av kildekodene som tilhører test tre*

2 Fremgangsmåte og gjennomføring

Åpne to terminalvinduer, heretter referert til terminal A og terminal B.

1. Start minikube med denne kommandoen i terminal A:

```
$ minikube start --driver docker --extra-config=kubelet.housekeeping-
interval=10s
```

For at metrics-server skal fungere i neste steg, så må `--extra-config=kubelet.housekeeping-interval=10s` være med i oppstart av minikube.

Svar fra kommandoen:

```
😄  minikube v1.25.1 on Ubuntu 20.04
🎉  minikube 1.25.2 is available! Download it:
https://github.com/kubernetes/minikube/releases/tag/v1.25.2
💡  To disable this notice, run: 'minikube config set
WantUpdateNotification false'


✨  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 ...
🐳  Preparing Kubernetes v1.23.1 on Docker 20.10.12 ...
    ▪ kubelet.housekeeping-interval=10s
🔎  Verifying Kubernetes components...
    ▪ Using image k8s.gcr.io/metrics-server/metrics-server:v0.4.2
    ▪ Using image gcr.io/k8s-minikube/storage-provisioner:v5
🌟  Enabled addons: storage-provisioner, default-storageclass, metrics-
server
🏄  Done! kubectl is now configured to use "minikube" cluster and "default"
namespace by default
```

2. Start den innebygde tilleggsfunksjonen 'metrics-server' i minikube:

```
$ minikube addons enable metrics-server
```

Svar fra kommandoen:

```
    ▪ Using image k8s.gcr.io/metrics-server/metrics-server:v0.4.2
🌟  The 'metrics-server' addon is enabled
```

For å se om metrics-server fungerer:

```
$ minikube kubectl -- top pods -n kube-system
```

Svar fra kommandoen vil være noe lik denne:

```
NAME                               CPU(cores)   MEMORY(bytes)
coredns-64897985d-q52bj            3m           13Mi
etcd-minikube                      25m          48Mi
kube-apiserver-minikube            81m          263Mi
kube-controller-manager-minikube   32m          48Mi
kube-proxy-zrn6b                   1m           10Mi
kube-scheduler-minikube            4m           16Mi
metrics-server-6b76bd68b6-g5klg    6m           17Mi
storage-provisioner                2m           9Mi
```

3. Start *Deployment* og eksponer *Serivcen*:

```
$ minikube kubectl -- apply -f php-apache.yaml
```

Svar fra kommandoen:

```
deployment.apps/php-apache created
service/php-apache created
```

4. Lag HPA og sjekk *current* status:

```
$ minikube kubectl -- apply -f hpa-php-apache.yaml
```

Svar fra kommandoen:

```
horizontalpodautoscaler.autoscaling/php-apache created
```

Så sjekkes status til HPA med en gang:

```
$ minikube kubectl -- get hpa
```

Svar fra kommandoen:

```
NAME         REFERENCE                TARGETS         MINPODS   MAXPODS
REPLICAS   AGE
php-apache   Deployment/php-apache   <unknown>/50%   1         10        0
12s
```

Legg merke til `<unknown>`. Dette kan skje hvis statusen sjekkes, før HPA er ferdig deployert. Derfor ble de spurt om status inntil `0%` vises. HPA fungerer selv om `0%` vises.

Svar fra kommandoen etter ett minutt:

```
NAME         REFERENCE                TARGETS    MINPODS   MAXPODS   REPLICAS
AGE
php-apache   Deployment/php-apache   0%/50%     1         10        1
70s
```

5. Generer mer last med BusyBox i Terminal B:

```
$ minikube kubectl -- run -i --tty load-generator --rm --image=busybox:1.28
--restart=Never -- /bin/sh -c "while sleep 0.01; do wget -q -O- http://php-
apache; done"
```

Svar fra kommandoen:

```
If you don't see a command prompt, try pressing enter.
OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!
OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!
OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!
```

6. Overvåk HPA i Terminal A:

```
$ minikube kubectl -- get hpa php-apache --watch
```

Denne kommandoen kjøres på et intervall på 15 sekunder.

Svar fra kommandoen etter noen minutter:

```
NAME         REFERENCE                  TARGETS   MINPODS   MAXPODS   REPLICAS
AGE
php-apache   Deployment/php-apache   0%/50%    1         10        1
58s
php-apache   Deployment/php-apache   132%/50%   1         10        1
75s
php-apache   Deployment/php-apache   171%/50%   1         10        3
90s
php-apache   Deployment/php-apache   157%/50%   1         10        4
105s
php-apache   Deployment/php-apache   61%/50%    1         10        4
2m
php-apache   Deployment/php-apache   65%/50%    1         10        4
2m15s
php-apache   Deployment/php-apache   78%/50%    1         10        4
2m30s
php-apache   Deployment/php-apache   72%/50%    1         10        7
2m45s
php-apache   Deployment/php-apache   57%/50%    1         10        7
3m
php-apache   Deployment/php-apache   43%/50%   1         10        7
3m21s
php-apache   Deployment/php-apache   51%/50%   1         10        7
3m30s
```

Overvåk autoskaleringen frem til prosessorkraften er stabil rundt `targetCPUUtilizationPercentage=50` i 5 minutter.

7. Stopp lasten BusyBox genererer i Terminal B:

```
$ <ctrl> + c
```

Svar fra kommandoen:

```
OK!OK!OK!OK!OK!^Cpod "load-generator" deleted
pod default/load-generator terminated (Error)`
```

8. Overvåk HPA i Terminal A og se den skalere ned:

```
$ minikube kubectl -- get hpa php-apache --watch
```

Når HPA detekterer at CPU=0% skalerer den automatisk ned til 1 replika. Dette kan ta noen minutter.

Svar fra kommandoen:

```
NAME         REFERENCE                  TARGETS   MINPODS   MAXPODS   REPLICAS
AGE
php-apache   Deployment/php-apache   42%/50%   1         10        7
4m46s
php-apache   Deployment/php-apache   5%/50%    1         10        7
5m1s
php-apache   Deployment/php-apache   0%/50%    1         10        7
5m16s
php-apache   Deployment/php-apache   0%/50%    1         10        7
7m
php-apache   Deployment/php-apache   0%/50%    1         10        7
7m52s
php-apache   Deployment/php-apache   0%/50%    1         10        7
9m31s
php-apache   Deployment/php-apache   0%/50%    1         10        6
9m46s
php-apache   Deployment/php-apache   0%/50%    1         10        1
10m
```

3 Forklaring av teknisk innhold:

Her er en mer detaljert beskrivelse av de forskjellige tekniske komponentene i testen. Det vil ikke bli gjort en detaljert linje-for-linje beskrivelse av .yaml-filene. Kapittel 1 forklarer hvor scriptene kan finnes.

3.1 php-apache.yaml

Denne filen lager en *Deployment* og eksponerer den med en *Service*. *Deploymenten* henter et ferdiglaget php-apache image fra Docker Hub. Imaget er `k8s.gcr.io/hpa-example` og har php-image versjon `php:5-apache` og er bygget opp ved hjelp av en Dockerfile som inneholder:

```
FROM php:5-apache
COPY index.php /var/www/html/index.php
RUN chmod a+rx index.php
```

Koden over refererer til en `index.php`-side som utfører komplekse regnestykker som krever mye prosessorkraft. Dette er for å simulere lasten i *clusteret* vårt og er som følger:

```
<?php
  $x ) 0.0001;
  for ($i = 0; $i <= 1000000; $i++) {
    4x += sqrt($x);
  }
  echo "OK!";
?>
```

3.2 hpa-php-apache.yaml

Denne filen konstruerer en HPA som inneholder beskrivelser om at *Deploymenten* php-apache skal ha minst én pod og maks ti podder. Denne HPA-en vil enten øke eller minke antall replikeringer innenfor dette intervallet, for å opprettholde ønsket gjennomsnittlig prosessorutnyttelse på tvers av alle podder på cirka 50%. Algoritmen som HPA benytter for å bestemme antall replikeringer baserer seg på forholdet mellom ønsket metrics-verdi og gjeldende metrics-verdi hentet fra `metrics-server`. Her vises algoritmen, som er forenklet for å lett kunne forstå hva den gjør:

```
ønsketreplikeringer = ceil[GjeldendeReplikas * ( GjeldendeMetricVerdi /
ØnsketMetricVerdi )]
```

Funksjonen `ceil[]` runder opp til neste heltall.

3.3 Lastgenerering med BusyBox

For å generere en økt last mot php-apache servicen, eksekveres en lastgenerator som heter BusyBox. Den henter et image busybox:1.28 og oppretter en pod som sender forespørsler til `http://php-apache` helt til podden termineres manuelt.

> **MERK:** *For videre informasjon om scriptene som er benyttet i dette prosjektet, les kildekoden som finnes i Vedlegg B - Kildekode*