Vedlegg E - Fremgangsmåte test to

Innhold:

1. Introduksjon

Dette vedlegget tar for seg en detaljert beskrivelse av både kode og gjennomføring til test 2. Testen ble gjennomført på maskin B.

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

2. Programvare

Versjoner for programvarer benyttet i denne testen utover det som er beskrevet i oppgaven er:

Programvare	Versjon
Python	3.9.7
Requests (python pakke)	2.25.1
pip3	20.3.4

3. Oppsett

Som i test én slettes eventuelle tidligere minikube *clustere* for å sikre at det ikke ligger tidligere konfigurasjon som kan påvirke testingen. Deretter lages et nytt med docker som driver.

```
$ minikube delete
$ minikube start --driver=docker
```

Deretter ble django og mysql fra den egenutviklede applikasjonen satt opp i det nye *clusteret*. Oppsettet av testen fortsatte ikke før siste kommando i kodeboksen under viste at begge podder hadde *"Running"* status.

```
$ cd $HOME/git/kubernetes-config/k8s-config
$ minikube kubectl -- apply -k ./django/ && minikube kubectl -- apply -k
./mysql/
$ minikube kubectl -- get pods --watch
```

Deretter ble NodePort til `django-entrypoint` testet med kommandoen

```
$ curl $(minikube ip):30001
```

Når kommandoen over returnerte en HTML-side er oppsettet ferdig og testen er klar for å utføres.

Da ble kommandoene under eksekvert for å starte selve testen:

```
$ cd $HOME/git/CISK-2022-bachelorgruppe/applikasjoner/python-script-get
$ ./test-gjennomføring $(minikube ip) 30001 200 10 $HOME/git sj $HOME/git
```

3.1 Forklaring av teknisk innhold:

Scriptene som forklares under er lagd ifb. med mikrotjenesten python-script-get. Kapittel 1 forklarer hvor scriptene kan finnes.

3.1.1 test-gjennomføring.sh

Dette er et script som vil gjøre hele testen selv. Det har tre `for`-løkker inne i hverandre som hører til antall podder, antall tråder og antall gjennomføringer. Nedenfor er et utsnitt av de viktigste linjene fra scriptet.

```
for pod in {1..10}
    do
        minikube kubectl -- scale --replicas=$pod -f $path/k8s-
bachelor/k8s-config/django/django-deployment.yaml
        sleep 12
        for z in {1..10}
        do
            fil="$path/bachelor-applikasjon/python-script-
get/resultater/$tid/$pod-podder.$z-trader-$tid.txt"
            for i in $(seq 1 $gjennomforinger)
            do
                /usr/bin/time -a -o "$fil" -f "%E" nice bash -c "python3
$path/bachelor-applikasjon/python-script-get/req.py --host $host --port
$port --antall $foresporsler --thr $z"

                if [ $i != 10 ]; then
                    sleep 12
                else
                    echo "Test med $z tråd(er) ferdig!"
                fi
            done
        done
    done
```

De viktigste linjene å legge merke til er linjen `minikube kubectl -- scale --replicas=$pod -f $path/k8s-bachelor/k8s-config/django/django-deployment.yaml` og linjen `/usr/bin/time -a -o "$fil" -f "%E" nice bash -c "python3 $path/bachelor-applikasjon/python-script-get/req.py --host $host --port $port --antall $foresporsler --thr $z"`. Den første linjen skalerer opp antall podder mellom hver hundrede gjennomføring av req.py scriptet, mens den andre linjen gjør selve testen ved å ta tiden på hvor lenge det tar å gjennomføre 200 forespørsler med `$z` tråder mot `$pod` podder.

3.1.2 req.py

Nedenfor vises et utsnitt av python scriptet. Som vist vil alle trådene samarbeide om å nå 200 forespørsler. Forespørslene fordeles ikke nødvendigvis likt utover trådene, men neste forespørsel blir delt ut til neste tråd som er ledig. De siste forespørslene er det `master`tråden som utfører selv for å sikre at det blir sendt akkurat 200 forespørsler.

```
def foresporsel(host, port, antall, thr):
    global value
    while value < antall-thr:
        r = requests.get(f"http://{host}:{port}")
        if r.status_code == 200:
            with threadLock:
                value += 1
            if value % 50 == 0:
                print(f"Det er gjennomført {value}/{antall} tester.")

value = 0
threads = []
for i in range(args.thr):
    t = threading.Thread(target=foresporsel, args=(args.host, args.port,
args.antall, args.thr))
    threads.append(t)
    t.start()
```

[...]

```
if args.thr != 0:
    while value < args.antall:
            r = requests.get(f"http://{args.host}:{args.port}")
            if r.status_code == 200:
                with threadLock:
                    value += 1
                if value % 50 == 0:
                    print(f"Det er gjennomført {value}/{args.antall}
tester.")
```

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