Vedlegg A - Systemkonfigurasjon

Innhold:

1 Generelt

Dette vedlegget tar for seg hvordan maskinene ble konfigurert til å gjennomføre testene.

I prosjektet ble tre like maskiner benyttet. Disse ble utdelt av Cyberingeniørskolen og modellen er en ThinkBook 14 G2 ITL. Alle maskinene har samme maskinvare, men det er forekomster av ulike versjoner med tanke på OS og programmer. Maskinvaren til maskinene er:

- 16GB RAM
- 512GB SSD
- 11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.80GHz prosessor

For enkelthetsskyld blir maskinene navngitt hhv. A, B og C. Maskin A ble benyttet til test én, maskin B ble benyttet til test to og maskin C ble benyttet til test tre. Maskinene ble tanket med unix-basert operativsystem. Operativsystemet på maskin A og C er Ubuntu og maskin B er Pop! OS. Tabell 1 viser hvilke versjoner av operativsystemer og programvarer som er installert på maskinene.

Tabell 1: Oversikt maskiner

Maskin	OS-versjon	Docker-versjon	minikube-versjon	kubectl-versjon
A	Ubuntu 20.04.4 LTS	20.10.14	v1.25.2	v1.23.3
B	Pop!_OS 21.10	20.10.12	v1.25.2	v1.23.3
C	Ubuntu 20.04.4 LTS	20.10.14	v1.25.2	v1.23.3

2 GitHub

Git ble benyttet til versjonskontroll av applikasjonen som er laget og et fellesområde hvor filer tilknyttet dette prosjektet er blitt lagret. Alle konfigurasjonsfiler, applikasjoner og vedlegg ligger i GitHub. Alle repoene er samlet i en organisasjon som finnes på denne lenken: https://github.com/CISK-2022-bachelorgruppe

Organisasjonen inneholder tre repoer. Dette er applikasjoner, kubernetes-config og vedlegg.

3 Basiskonfigurasjon

Dette kapitlet tar for seg basiskonfigurasjon på maskinene som ble benyttet i prosjektet. Dette er for å kunne lage et tilnærmet likt labmiljø som er benyttet i dette prosjektet, og for å kunne ha likt utgangspunkt for testing. I basiskonfigurasjonen ligger det hovedsakelig to programmer. Dette er Docker og Minikube. De neste kapitlene vil gjennomgå de prosedyrene som ble utført ved installasjon på maskinene som ble benyttet i prosjektet.

> **MERK:** *Denne installasjonen av programvarene vil installere de nyeste versjonene som eksisterer. For å etterprøve testene i rapporten er det hensiktsmessig og spesifisere de versjonene som ble benyttet i testene! ved installasjon av programvarene!*
> *For å se hvilke versjoner av de ulike programvarene som ble installert, se Tabell 1*

3.1 Docker

I dette prosjektet blir Docker benyttet som en driver for minikube og må derfor installeres før minikube. For installasjon av Docker, ble installasjonsguiden *(https://docs.docker.com/engine/install/ubuntu/)* til Docker Inc fulgt.
Under vil kommandoene som ble benyttet i installasjonen bli listet opp.

Først ble all gammel konfigurasjon av Docker fjernet med følgende kommando:

```
$ sudo apt-get remove docker docker-engine docker.io containerd runc
```

Så ble Docker installert med et repo. Med de neste kommandoene oppdateres og installeres det pakker som gjør det mulig å tillate repo over HTTPS.

```
$ sudo apt-get update

$ sudo apt-get install \
    ca-certificates \
    curl \
    gnupg \
    lsb-release
```

Her legges Docker Inc sin offisielle GPG-nøkkel til maskinen:

```
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --
dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg
```

Så eksekveres en kommando for å sette opp et stabilt repo og gjør det klart til installering:

```
$ echo \
  "deb [arch=$(dpkg --print-architecture) signed-
by=/usr/share/keyrings/docker-archive-keyring.gpg]
https://download.docker.com/linux/ubuntu \
  $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list
> /dev/null
```

Til slutt installeres siste versjon av Docker Engine med følgende kommandoer:

> **MERK:** *Skal testen etterprøves, bør det spesifiseres hvilken versjon som skal lastes ned her, for å få samme versjon som ble benyttet under forsøket i dette prosjektet. Dette er for å skape et likt testmiljø*

```
$ sudo apt-get update

$ sudo apt-get install docker-ce docker-ce-cli containerd.io docker-
compose-plugin
```

Nå er Docker installert på maskinen.

3.2 minikube

For å utføre konseptbevisene som er laget, trengs et K8s *cluster*. På grunn av bacheloroppgavens tidsbegrensning var det ikke tid nok til å sette opp et fullskala K8s *cluster*. Derfor ble det installert minikube på egne maskiner. minikube lager et virtuelt K8s *cluster* som tillater å teste funksjoner som finnes i fullt oppsatte *clustere*.

For installasjon av minikube, ble installasjonsguiden *(https://minikube.sigs.k8s.io/docs/start/)* til "The Kubernetes Authors" fulgt.
Under vil kommandoene som ble benyttet i installasjonen bli listet opp.

For å installere siste versjon av minikube på x86-64 Linux med binær nedlastning:

```
$ curl -LO
https://storage.googleapis.com/minikube/releases/latest/minikube-linux-
amd64
$ sudo install minikube-linux-amd64 /usr/local/bin/minikube
```

Etter dette legges brukeren til gruppen docker, så Docker kan kjøres uten å benytte sudo foran hver kommando.

```
$ sudo usermod -aG docker $USER && newgrp docker
```

For å starte minikube og gjøre installasjonen ferdig ved førstegangsinstallasjon:

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

For å installere kubectl må denne kommandoen skrives:

```
$ minikube kubectl -- get pods -A
```

4 K8s konfigurasjon

For å kjøre opp applikasjonen, beskrevet i bachelorrapporten kapittel 4.1, i K8s er det benyttet yaml-filer. Disse yaml-filene ligger i repoet *kubernetes-config*, se *Vedlegg B - Kildekode*.

4.1 django-applikasjon

Til django-applikasjonen benyttes kun objektene *Deployment* og *Service*. *Deploymenten* henter et Docker image fra Docker Hub som er utviklet ifb. med dette prosjektet. Docker imaget som er benyttet er sjohans1/django-bachelor:6.0 *(https://hub.docker.com/r/sjohans1/django-bachelor/)*. I tillegg er det definert fem miljøvariabler i denne *Deploymenten*. Dette er variabler som setter opp forbindelse med mysql-tjenesten og inneholder databasenavn, databasebruker, databasebrukerens passord, databasens IP/DNS og databasens port. *Servicen* er satt opp som en NodePort og gir derfor tilgang til django-applikasjonen ved

hjelp av en port. Denne gjør slik at *django-entrypoint*, som er navnet på *Servicen*, blir eksponert som på porten 30001. Dette vil si at det går an å nå tjenesten fra utsiden av K8s *clusteret* ved hjelp av ip-adressen til minikube og porten 30001.

Når yaml-filene skal legges inn i K8s blir *kustomization.yaml* benyttet. Dette bidrar til at kun én kommando må skrives for å deployere én mikrotjeneste. Skal mikrotjenesten django-applikasjon deployeres, vil det derfor være nok å ha en terminal åpen i mappen django i *kubernetes-config*-repoet og kjøre kommandoen:

```
$ minikube kubectl -- apply -k .
```

4.2 mysql

Til mysql-databasen benyttes objektene *PersistenVolume*, *Secret*, *ConfigMap*, *StatefulSet* og *Service*.

I *PersistentVolume* defineres hvor i minikube dataen, som linkes mot dette volumet, skal lagres. Stien til denne mappen i minikube er: "*/data/bachelor-mysql*"

I *Secret* defineres passordet til databasebrukeren, mens i *ConfigMap* bestemmes databasens navn. Disse verdiene er hhv. "*cGFzc3dvcmQK*" og "*bachelor_db*".

StatefulSettet henter et Docker image, på samme måte som django-applikasjonen over, fra Docker Hub. Imaget er derimot ikke spesielt utviklet til dette prosjektet og er derfor et offisielt Docker image av mysql. Docker imaget og versjonen som er benyttet er mysql:5.7. Siden dette er et *StatefulSet* er det montert et volum som bidrar til persistent lagring av data. Det er også definert en *PersistentVolumeClaim* i *StatefulSettet* som kobles opp mot *PersistentVolumet*.

I tillegg er det definert to miljøvariabler. Disse variablene ligger ikke i selve *StatefulSettet*, men de ligger i objektene *Secret* og *ConfigMap*. Variablene bestemmer databasepassordet til databasen og databasenavnet til databasen som skal opprettes ved deployering av dette *StatefulSettet*. *Service* er en vanlig ClusterIP K8s *Service*, noe som gjør at databasen kun kan aksesseres fra innsiden av K8s *clusteret*.

4.3 sched-applikasjon

Til sched-applikasjonen benyttes objektet *Deployment*.
Deploymenten henter et Docker image, kalt sched:latest, fra lokal maskin, og kjører dette opp med to miljøvariabler. Den ene variabelen bestemmer intervallet til GET-forespørsler, altså tiden det skal ta mellom hver GET-forespørsel, mens den andre variabelen bestemmer hvilken IP/DNS API-et befinner seg på. Her er django-entrypoint:8000 benyttet, ettersom *django-entrypoint* er *Servicen* til django-applikasjonen.

Ved test én skal sched-applikasjonen benyttes og blir deployert automatisk via scriptet i mikrotjenesten pod-sletting.