Dokumentation av examinationsuppgift tre för kursen 1dv032

Översikt

Du kan nå min applikation på: http://194.47.176.240/

Instanserna

För att vi ska kunna sätta upp ett minireddit med hjälp av Kubernetes behövs följande instanser:

- Masterinstans[1]
- Tre olika nodinstaner[2]
- En NFS-instans[3]
- En registry-instans[4]

De instanser jag använder är:

```
Master - 172.168.0.12 (Floating IP: 194.47.176.243)
node-1 - 172.168.0.6
node-2 - 172.168.0.7
node-3 - 172.168.0.9
nfs - 172.168.0.14
registry - 172.168.0.8
```

Samtliga instancer har SSH som security group och Master har även http.

[1] **Masterinstansen** kommer att vara den instansen där samtliga vitala filer för initiering och de tjänster som kommer att driftsättas kommer att placeras. Masterinstansen kommer att innehålla både Kubernetes och Docker. Masterinstansen kommer att agera master efter att man kör script nummer ett i kategorin "Kubernetes på master node" varpå man sedan fäster övriga noder till denna. Det är även masterinstansen som kommer att ta ha Calico samt mappar för Kubernetes konfiguration.

[2] De **tre olika nodinstanerna** existerar för att ge vår applikation redudans. I takt med att vi utvecklar vår applikation kommer vi att skapa *deployments* och *poddar*. Dessa *poddar* kommer att placeras ut på dessa tre noder.

[3] En **NFS-Server** behövs för att vi ska kunna dela kataloger mellan våra instanser på nätverket. Detta är för att vi snabbt och smidigt ska kunna välja var vi vill lagra information från databaserna.

[4] **Registry** används som en egen Dockerhub. Istället för att gå igenom krångligheterna med att skapa en användare och att publicera våra images så används denna. Den kan tänkas agera databas åt våra images. Oavsett så är det mycket smidigare att använda sig av ett registry än att ladda ned från Dockerhub men det kommer som sagt med fördelen att man enkelt kan skapa sina egna, specialgjorda images genom `Docker build .` och lagra dessa på vårt registry.

Kubernetesklustret

- Det talas om *poddar* ovan. En *Pod* är det minsta som kubernetes hanterar. Det represenenterar en uppgift som ska göras. En pod är en eller från containers som du har skapat. Podden kommer att leva för att göra en uppgift. Som vi snart kommer att upptäcka så kommer det att skapas en podd per service (multiplicerat med antalet replikor).
- Vi kommer att hantera dessa poddar med *controllers* som skapar och hanterar dessa pods. Controllern definierar önskat tillstånd som podden ska ha varpå ett deploymoent kommer att styra om utrullningen av podden.
- YAML är ett deklaritivt sätt att bestämma vilka inställningar och kommandon som skickas till Kubernetes. Jag kommer hela tiden att referera till YAML-filer under instruktioner för installationen
- Dockerfile är et fil som skapas när det krävs att vi själva bygger ihop docker images. Detta kommer behövas göra på nästan alla services eftersom det ofta är något vi själva måste modifiera. Antingen skickar vi en med SQL-dump som läses in eller så vill vi köra ett script.
- Kommandon kommer att förklaras i löpande text.

Master node:

1. Kör cloud-init-scriptet:

```
#!/bin/bash
# Uppdatera alla paket
sudo apt update
#apt upgrade -y
# Installera paket som är nödvändiga eller bra att ha
sudo apt-get install -qq apt-transport-https ca-certificates curl software-
properties-common jq
# Hämta och lägg till nycklar för docker och k8s
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg | apt-key add
-
sudo curl -fsSL https://packages.cloud.google.com/apt/doc/apt-key.gpg |
apt-key add -
# Lägg till docker och k8s repos och uppdatera
sudo add-apt-repository -yu "deb [arch=amd64]
https://download.docker.com/linux/ubuntu focal stable"
sudo add-apt-repository -yu "deb https://apt.kubernetes.io/ kubernetes-
xenial main"
# Installera nödvändiga paket
sudo apt-get install -qq docker-ce kubelet kubeadm kubectl
# Ändra så att docker använder systemd
cat <<EOF | sudo tee /etc/docker/daemon.json
cat <<EOF | sudo tee /etc/docker/daemon.json
{
  "exec-opts": ["native.cgroupdriver=systemd"],
  "log-driver": "json-file",
  "log-opts": {
```

```
    "max-size": "100m"
  },
  "storage-driver": "overlay2",
  "insecure-registries": ["<IP TILL REGISTRY SOM SKAPAS NEDAN>:5000"]
}
EOF
sudo mkdir -p /etc/systemd/system/docker.service.d
# Starta om och autostata docker
sudo systemctl daemon-reload
sudo systemctl restart docker
sudo systemctl enable docker
# Här kan du lägga till en kubeadm join, t.ex.
```

2. Ge mastern ett flytande IP
3. Kopiera över SSH-nyckel för att kunna ansluta till övriga instanser

Kubernetes på master node

1. I och med att scriptet ovan har körts så kan vi gå vidare med att initiera kubernetes-klustret.:
    - `sudo kubeadm init --pod-network-cidr=172.168.0.0/16`

NOTERA ATT MITT NÄTVERK GÅR PÅ 172. OM DU HAR DET ANNORLUNDA VÄNLIGEN MODIFIERA DETTA

2. Av detta kommer det output. Spara kubeadm-kommandot för att kunna ansluta noderna till klustret:

```
kubeadm join 172.168.0.12:6443 --token s79d9q.po5f4725es5h1fzz \
    --discovery-token-ca-cert-hash
sha256:50c2ad4003f33d1310ca44397fc08e10909156c217807c8b38779bef88a7185f
```

3. Kör även dessa kommandon föär att skapa mappen där konfigurationsfilerna för kubernetesklustret:

```
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

4. Ladda ned Calico som kommer att köras som ett overlaynätverk:

- `curl https://docs.projectcalico.org/manifests/calico.yaml -O`

5. Installera Calico

- `kubectl apply -f calico.yaml`

Övrigt på master node

6. Installera nfs-common för att kunna prata med databasen som kommer att installeras nedan

- `sudo apt install nfs-common`

7. Installer mongodb för att kunna se så att allt är OK när mongodb körs

- `sudo apt install mongodb`

8. Installera mysql för att kunna se så att allt är OK när mariaDB körs

- `sudo apt install mysql-server`

Slave nodes

1. Likt masternoden så kör vi cloud-init.

```
#!/bin/bash
# Uppdatera alla paket
sudo apt update
#apt upgrade -y
# Installera paket som är nödvändiga eller bra att ha
sudo apt-get install -qq apt-transport-https ca-certificates curl software-
properties-common jq
# Hämta och lägg till nycklar för docker och k8s
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg | apt-key add
-
sudo curl -fsSL https://packages.cloud.google.com/apt/doc/apt-key.gpg |
apt-key add -
# Lägg till docker och k8s repos och uppdatera
sudo add-apt-repository -yu "deb [arch=amd64]
https://download.docker.com/linux/ubuntu focal stable"
sudo add-apt-repository -yu "deb https://apt.kubernetes.io/ kubernetes-
xenial main"
# Installera nödvändiga paket
sudo apt-get install -qq docker-ce kubelet kubeadm kubectl
# Ändra så att docker använder systemd
cat <<EOF | sudo tee /etc/docker/daemon.json
{
  "exec-opts": ["native.cgroupdriver=systemd"],
  "log-driver": "json-file",
  "log-opts": {
    "max-size": "100m"
  },
  "storage-driver": "overlay2",
  "insecure-registries": ["<IP TILL REGISTRY SOM SKAPAS NEDAN>:5000"]
}
EOF
sudo mkdir -p /etc/systemd/system/docker.service.d
# Starta om och autostata docker
sudo systemctl daemon-reload
sudo systemctl restart docker
sudo systemctl enable docker
# Här kan du lägga till en kubeadm join, t.ex.
```

2. Koppla ihop slavnoderna med mastern med hjälp av raden som vi sparade ovan:

```
kubeadm join 172.168.0.12:6443 --token s79d9q.po5f4725es5h1fzz \
  --discovery-token-ca-cert-hash
sha256:50c2ad4003f33d1310ca44397fc08e10909156c217807c8b38779bef88a7185f
```

Registry

1. Kör dessa kommando för att:

- Uppdatera
- Installera docker
- Starta och aktivera docker
- Kolla så att docker installerades
- Skapa en mapp där vi kommer lägga våra images
- Starta container som låter oss spara images

```
sudo apt update
sudo apt install docker.io -y
sudo systemctl start docker
sudo systemctl enable docker
docker --version
mkdir -p ~/docker/repository
sudo docker container run -d -p 5000:5000 --name registry -v
~/docker/registry:/var/lib/registry registry
```

Fotnot: Dessa går givetvis att stoppa in i ett script för att automatisera

Registry på övriga noder (Detta görs redan vid initieringen av master och slave noder men bra att veta)

1. Ändra i filen för att: - Kunna lita på vår registry

`sudo nano /etc/docker/daemon.json`:

```
{
  "exec-opts": ["native.cgroupdriver=systemd"],
  "log-driver": "json-file",
  "log-opts": {
    "max-size": "100m"
  },
  "storage-driver": "overlay2",
  "insecure-registries": ["<IP TILL REGISTRY>:5000"]
}
```

2. Starta om Docker `sudo systemctl restart docker`

Valfritt: Kolla så att allt fungerar

- Dra ner valfri image, här används ubuntu: `sudo docker pull ubuntu`

4. Tagga den:

- `sudo docker tag ubuntu <ip till registry>:5000/ubuntu`

5. Push imagen till registry:

- `sudo docker image push <ip till registry>:5000/ubuntu`

6. Kör kommandot `sudo docker images`, kopiera ID till imagen och kör sedan `sudo docker rmi <ID>`

7. Kör `sudo dock image pull <iptillregistry>:5000/ubuntu`

NFS

1. Installera NFS-kernel:

- `sudo apt install nfs-kernel-server -y`

2. Sakap directories som kommer att användas för att lagra data från MariaDB och mongoDB:

```
sudo mkdir -p /export/kubedata
sudo mkdir -p /mongo/kubedata
sudo chown nobody:nogroup /export/kubedata
sudo chown nobody:nogroup /mongo/kubedata
sudo chmod 777 /export/kubedata
sudo chmod 777 /mongo/kubedata
```

3. Skapa en exportmapp som tillåter bland annat läs/skriv från alla i miljön:

- `sudo nano /etc/exports`

```
/export/kubedata *(rw,no_root_squash,no_subtree_check)
/mongo/kubedata *(rw,no_root_squash,no_subtree_check)
```

4. Starta NFS-tjänsten:

- `sudo exportfs -a`
- `sudo systemctl restart nfs-kernel-server`

Uppsättning av applikationen med hjälp av Kubernetes

Enklast och mest effekt är att börja med att sätta upp databaserna. Detta gör vi först för att vi snabbt ska kännas oss färdiga när de är uppe och rullar. Dessutom bygger applikationerna som ska användas på dessa databaser.

Databaser

Vi kommer att använda två olika databaser. MongoDB och MariaDB

MariaDB

1. Börja med att skapa en mapp på din masterinstans och döp den till "MariaDB" eller liknande.

- `cd MariaDB`
- `sudo nano mariadb.yaml`
- Kopiera innehållet från mariadb.yaml, spara
- `sudo nano Dockerfile`
- Kopiera innehållet från Dockerfile, spara (Notera versal i Dockerfile)
- `sudo nano init.sql`
- Kopiera innehållet från Init.sql

2. Kör sudo docker images. Hitta den tomma imagen
3. Tagga den:

- `sudo docker tag <imageID> <ip till registry>:5000/mariadb`

4. Push imagen till registry:

- `sudo docker image push <ip till registry>:5000/mariadb`

5. "Applya" yaml-filen med `kubectl apply -f mariadb.yaml`
6. Starta Service, PV, PVC, Deployment och poddar med hjälp av `kubectl apply -f mariadb.yaml`

- Man kan för nöjes skull kolla så att allt är uppe och snurrar genom att köra följande kommandon:
- `kubectl get svc`, `kubectl get pv`, `kubectl get pvc`, `kubectl get deploy`, `kubectl get pods`
- Om något mot förmodan inte är igång så kan man köra `kubectl logs <podname>` och/eller `kubectl describe deploy <deployment>`

7. För att se så att all information från MariaDB har kommit in:

- Hämta poddarna: `kubectl get pods`
- Öppna en pod som tillhör MariaDB genom `kubectl exec -it <dittpodnamn> /bin/bash`
- Kör `mysql -uuser -p` skriv sedan in ditt lösenord som du valde i YAML-filen ovan. Ange lösenorder "password" utan citationstecken. Dessa går givetvis att ändra i mariadb.yaml
- Kör kommando `show databases;`, `use user_db;`, `show tables;`. Finns `loggedin` och `users` där så är allt OK.

MongoDB

1. Börja med att skapa en mapp på din masterinstans och döp den till "MongoDB" eller liknande.

- `cd MongoDB`
- `sudo nano mongodb.yaml`
- Kopiera innehållet från mongodb.yaml, , spara
- "Applya" yaml-filen med `kubectl apply -f mongodb.yaml`
- Hämta poddarna: `kubectl get pods`
- Öppna en pod som tillhör MongoDB genom `kubectl exec -it <dittpodnamn> /bin/bash`
- Logga in genom att använda `mongo --username user` sedan lösenorder "password" utan citationstecken, sedan `show dbs;`. Finns det något där så funkar det. Dessa går givetvis att ändra i mongodb.yaml

Services

Vi har ett antal pythonscripts som kommer att köras för att populera databaserna. Enklast är att skapa upp tre mappar för dessa direkt. Dessa mappar är:

- loginsvc
- postsvc
- frontend
- proxy

Detta görs genom `mkdir <servicemapp>`

loginsvc

KÖR kubectl get svc notera alla IP-nummer. Dessa är ClusterIP och kommer att behövas sättas i Yaml-filerna.

1. Gå in i loginsvc-mappen

- `cd loginsvc`
- `sudo nano loginsvc.yaml`
- Kopiera innehållet från loginsvc.yaml, modifera innehållet där det krävs (allt inom <>) spara
- `sudo nano login_svc.py`
- Kopiera innehållet från login_svc.py
- `sudo nano Dockerfile`
- Kopiera innehållet från Dockerfile, spara (Notera versal i Dockerfile)
- sudo nano `requirements.txt`
- Kopiera innehållet från requirements.txt, spara

2. Kör sudo docker images. Hitta den tomma imagen
3. Tagga den:

- `sudo docker tag <imageID> <ip till registry>:5000/loginsvc`

4. Push imagen till registry:

- `sudo docker image push <ip till registry>:5000/loginsvc`

5. "Applya" yaml-filen med `kubectl apply -f loginsvc.yaml`

postsvc

KÖR kubectl get svc notera alla IP-nummer. Dessa är ClusterIP och kommer att behövas sättas i Yaml-filerna.

1. Gå in i postsvc-mappen

- `cd postsvc`
- `sudo nano postsvc.yaml`
- Kopiera innehållet från postsvc.yaml, modifera innehållet där det krävs (allt inom <>) spara
- `sudo nano post_svc.py`
- Kopiera innehållet från post_svc.py
- `sudo nano Dockerfile`
- Kopiera innehållet från Dockerfile, spara (Notera versal i Dockerfile)
- sudo nano `requirements.txt`
- Kopiera innehållet från requirements.txt, spara

2. Kör sudo docker images. Hitta den tomma imagen
3. Tagga den:

- `sudo docker tag <imageID> <ip till registry>:5000/postsvc`

4. Push imagen till registry:

- `sudo docker image push <ip till registry>:5000/postsvc`

5. "Applya" yaml-filen med `kubectl apply -f postsvc.yaml`

frontend

KÖR kubectl get svc notera alla IP-nummer. Dessa är ClusterIP och kommer att behövas sättas i Yaml-filerna.

1. Gå in i frontend-mappen

- `cd frontend`
- `sudo nano frontend.yaml`
- Kopiera innehållet från frontend.yaml, modifera innehållet där det krävs (allt inom <>) spara
- `sudo nano web_svc.py`
- Kopiera innehållet från web_svc.py
- `sudo nano Dockerfile`
- Kopiera innehållet från Dockerfile, spara (Notera versal i Dockerfile)
- sudo nano `requirements.txt`
- Kopiera innehållet från requirements.txt, spara

2. Skapa mappen `templates`

- `mkdir templates`

- Skapa en fil per fil i templates i templates genom `sudo nano filnamn.html` och kopiera in innehållet.

3. Kör sudo docker images. Hitta den tomma imagen
4. Tagga den:

- `sudo docker tag <imageID> <ip till registry>:5000/frontend`

5. Push imagen till registry:

- `sudo docker image push <ip till registry>:5000/frontend`

6. "Applya" yaml-filen med `kubectl apply -f frontend.yaml`
7. Hämta ClusterIP genom `kubectl get svc`, notera detta och sätt det i miljövariabeln för ClusterIP och kör `kubectl apply -f frontend.yaml`igen. Ingen mening egentligen. Här kommer vi sätta nginx-proxy.

Nu ska alla services vara uppe. Vi kan testa dessa genom att cURL:a dem.

1. `curl -X POST -H "Content-Type: application/json" -d '{"login": "namn", "password": "password"}' <clusterIPTilllogginsvc:7070/createuser`. Detta bör returnera {"success": true}. loginsvc och MariaDB fungerar.
2. `curl -X POST -H "Content-Type: application/json" -d '{"subreddit": "subreddit"}' <ClusterIPtillpostsvc>:7075/subreddit` Detta bör returnera: {"success":true}. postsvc fungerar.

proxy

KÖR kubectl get svc notera alla IP-nummer. Dessa är ClusterIP och kommer att behövas sättas i Yaml-filerna.

1. Gå in i proxy-mappen

- `cd proxy`
- `sudo nano proxy.yaml`
- Kopiera innehållet från proxy.yaml, modifera innehållet där det krävs (allt inom <>) spara

1. `curl <ClusterIPtillfrontend>:7080`. Detta bör returnera HTML-kod. Görs detta, så funkar frontend.
2. Du kan också köra `curl <clusterIPtillfrontend>:7080/r/<valfri subreddit>`. Detta bör returnera mer HTML och information från subredditen.

- Nu ska du kunna köra `curl <ClusterIPtillProxy>` för att få samma innehållet som när du curlade frontend.

Experiment

Detta är ett script som kommer att populera databaserna med användare och annan information. Det finns tre json-filer bifogade. Dessa finns dock inte sparade här, utan jag vill hänvisa till uppgiftens github-repo.

Detta eftersom filerna är väldigt stora och tar tid att ladda in.

1. Skapa en mapp vid namn experiment

- `mkdir experiment`
- Skapa filen runexp.py genom `sudo nano runexp.py`
- Kopiera allt innehåll från runexp.py
- Spara json-filerna lokalt på din dator och använd dig av `scp -i <dinNyckel> <dinFil> <dinanvänadre:ip till master>`. Flytta sedan dessa till mappen experiment genom att använda `mv <fil>.json`
- Öppna filen runexp.py och modifiera följande:

```
login_host = '<ClusterIP till loginsvc>'
login_port = 7070

poster_host = '<ClusterIP till postsvc>'
poster_port = 7075
```

Längre ner i koden finns det mer att ändra (vi måste bestämma vilken fil vi vill köra):

```
with open('experiment_small.json') as f:
    jo = json.load(f)
```

Notera 'experiment_small.json'. Denna ändras beroende på vilken fil du vill läsa in.

Lastbalanserare

Lastbalanserare går att sätta upp i Openstack. Dock sker detta genom deras GUI och då vill jag hänvisa till Morgans tutorial. Länk.

Avslut

Notera att samtliga cluserIPn går att ändra till motsvarande DNS.
`<service>.default.svc.cluster.local:7080`

Jag har gjort ett fåtal modifieringar i scripten. Dels ändringarna i runexp.py:

```
login_host = '<ClusterIP till loginsvc>'
login_port = 7070

poster_host = '<ClusterIP till postsvc>'
poster_port = 7075
```

och

```
with open('experiment_small.json') as f:
    jo = json.load(f)
```

Men även så castade jag själv portarna till ints i servicescriptsen (gjorde detta innan den reviderade versionen släpptes).