**Environment DOFFINder**

**(Martin/Tommy prosjekt)**

**(** Kubernetes docker desktop)

-Kubernetes (Docker-desktop) lære deg hvordan du kan distribuere, skalere og administrere containerbaserte applikasjoner.

Et bilde som inneholder tekst, elektronikk, skjermbilde, programvare

Automatisk generert beskrivelse

Container Orchestration Engine (COE)

Et bilde som inneholder tekst, skjermbilde, Font, design

Automatisk generert beskrivelse

**1. Aktiver Kubernetes i Docker Desktop**

1. Åpne Docker Desktop.
2. Gå til **Settings** (Tannhjulikonet).
3. Klikk på **Kubernetes** i menyen.
4. Aktiver **Enable Kubernetes** og klikk **Apply & Restart**.
5. Vent til Kubernetes-klusteret er opprettet. Du kan sjekke statusen ved å se etter en grønn markering ved siden av "Kubernetes is running".

**2. Konfigurer kubectl for å bruke Docker Desktop sitt Kubernetes-kluster**

* Docker Desktop kommer med en ferdigkonfigurert kubectl som peker til Kubernetes-klusteret. For å bekrefte at kubectl fungerer:

bash

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Kubectl get nodes

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Du skal se en liste med Kubernetes-node(r).

**3. Push Docker-image til Docker Desktop sitt lokale registry**

Du trenger ikke laste opp Docker-imaget til en ekstern container registry (som Docker Hub). Du kan bruke det lokale Docker Desktop-registryet.

1. Bygg Docker-imaget for prosjektet:

bash

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

docker build -t doffinder-backend:latest .

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Tag imaget slik at det kan brukes av Kubernetes lokalt:

Bash

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

docker tag doffinder-backend:latest localhost:5000/doffinder-backend:latest

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Push imaget til Docker Desktop sitt innebygde registry:

Bash

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

docker push localhost:5000/doffinder-backend:latest

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Error

Etter å kjøre dette 'docker push localhost:5000/doffinder-backend:latest' fikk jeg denne resultat 'C:\Test\DOFFINder>docker tag doffinder-backend:latest localhost:5000/doffinder-backend:latest C:\Test\DOFFINder>docker push localhost:5000/doffinder-backend:latest The push refers to repository [localhost:5000/doffinder-backend] af302e5c37e9: Waiting b20e3a1a6b0d: Waiting e81d26d6c5a7: Waiting c70af1ce6c86: Waiting 647a4343accf: Waiting 5e70b13a7bd5: Waiting 31fc40b7245b: Waiting 183dda4bfc33: Waiting failed to do request: Head "https://localhost:5000/v2/doffinder-backend/blobs/sha256:31fc40b7245b1b5a9c87e53d8a533c3f8bcd41cc4510e92a14a0f41828e77d45": dial tcp [::1]:5000: connect: connection refused' er dette et feil? og hvis ja, hva anbaler du?

**ChatGPT sa:**



Ja, dette er en feil. Den oppstår fordi Docker prøver å pushe bildet til et containerregister som ikke er tilgjengelig på localhost:5000. Dette skjer fordi det lokale registeret ikke kjører, eller det er ikke konfigurert til å akseptere push-forespørsler.

**Årsak til feilen**

1. **Ingen lokal registry kjører**:
   * Når du bruker localhost:5000, forutsetter Docker at det kjører en container-registry på localhost som lytter på port 5000.
   * Hvis det ikke er konfigurert et registry, vil Docker ikke kunne pushe.
2. **Koble til feil adresse eller port**:
   * Hvis det er et register, men det kjører på en annen port eller er feil konfigurert, vil du få en lignende feil.

**Løsning: Konfigurer og kjør et lokalt Docker-register**

For å bruke localhost:5000, må du først starte et lokalt Docker-registry.

**Steg 1: Start et lokalt Docker-registry**

1. Kjør følgende kommando for å starte et registry-container:

bash

Kopier kode

docker run -d -p 5000:5000 --name registry registry:2

1. Verifiser at registry kjører:

bash

Kopier kode

docker ps

Du bør se en container med navnet registry.

**Steg 2: Pushe Docker-image til lokal registry**

1. Tag bildet på nytt (hvis nødvendig):

bash

Kopier kode

docker tag doffinder-backend:latest localhost:5000/doffinder-backend:latest

1. Pushe bildet til det lokale registeret:

bash

Kopier kode

docker push localhost:5000/doffinder-backend:latest

1. Sjekk at bildet er tilgjengelig i registeret:

bash

Kopier kode

curl http://localhost:5000/v2/\_catalog

**Alternativ: Bruke Docker Desktop sitt innebygde registry**

Hvis du bruker Docker Desktop og ønsker å unngå å sette opp et eget lokalt registry, kan du bruke Docker Desktop sin Kubernetes-integrasjon og bygge bildet direkte i klusteret.

1. Bytt til Kubernetes-konfigurasjon:

bash

Kopier kode

eval $(minikube docker-env)

1. Bygg bildet direkte:

bash

Kopier kode

docker build -t doffinder-backend:latest .

1. Kubernetes vil da bruke bildet direkte uten behov for å pushe til et registry.

**Oppsummering**

* **Feil**: Registeret på localhost:5000 er ikke tilgjengelig.
* **Løsning**:
  1. Start et lokalt Docker-registry på localhost:5000.
  2. Pushe bildet til dette registeret.

Hvis du trenger mer hjelp med noen av trinnene, bare gi beskjed! 😊

**Du sa:**

Nå fungere dette 'curl http://localhost:5000/v2/\_catalog' med svar '{"repositories":["doffinder-backend"]}' skal jeg forsetter på '4. Lag Kubernetes-manifester for DOFFINder Opprett Kubernetes-manifester for prosjektet. 4.1 Deployment (backend) Opprett en fil backend-deployment.yaml:' fra din dokumentasjon?

**ChatGPT sa:**



Ja, siden ditt lokale Docker-register fungerer nå, og du har bekreftet at bildet er tilgjengelig, kan du fortsette med oppgaven **"4. Lag Kubernetes-manifester for DOFFINder"**. Dette er neste logiske trinn for å distribuere applikasjonen din i Kubernetes.

**Fremgangsmåte**

**1. Opprett Kubernetes Deployment**

Opprett en fil kalt backend-deployment.yaml i prosjektmappen din, og legg inn følgende innhold:

yaml

Kopier kode

apiVersion: apps/v1

kind: Deployment

metadata:

name: doffinder-backend

spec:

replicas: 2

selector:

matchLabels:

app: doffinder-backend

template:

metadata:

labels:

app: doffinder-backend

spec:

containers:

- name: doffinder-backend

image: localhost:5000/doffinder-backend:latest

ports:

- containerPort: 8000

env:

- name: OPENAI\_API\_KEY

value: "your-openai-api-key"

- name: GOOGLE\_API\_KEY

value: "your-google-api-key"

* **Forklaring**:
  + **replicas**: Definerer hvor mange kopier av applikasjonen du vil kjøre.
  + **image**: Bruker Docker-imaget fra ditt lokale register.
  + **env**: Setter miljøvariabler som kreves av applikasjonen din.

**2. Opprett Kubernetes Service**

Opprett en fil kalt backend-service.yaml for å eksponere backend-tjenesten:

yaml

Kopier kode

apiVersion: v1

kind: Service

metadata:

name: doffinder-backend-service

spec:

selector:

app: doffinder-backend

ports:

- protocol: TCP

port: 80

targetPort: 8000

type: LoadBalancer

* **Forklaring**:
  + **selector**: Knyttes til pods som har etiketten app: doffinder-backend.
  + **type**: LoadBalancer eksponerer tjenesten utenfor klusteret.

**3. Deploy applikasjonen til Kubernetes**

Bruk kubectl apply for å distribuere ressursene til Kubernetes:

1. Distribuer deployment:

bash

Kopier kode

kubectl apply -f backend-deployment.yaml

1. Distribuer tjenesten:

bash

kubectl apply -f backend-service.yaml

**4. Verifiser distribusjonen**

1. **Sjekk status for pods**:

bash

kubectl get pods

Sørg for at alle pods kjører.

1. **Sjekk tjenesten**:

bash

kubectl get services

Du vil se en ekstern IP-adresse hvis LoadBalancer-typen fungerer som forventet.

1. **Test applikasjonen**:
   * Åpne nettleseren og gå til tjenestens IP-adresse for å bekrefte at backend kjører.
   * For eksempel: http://<external-ip>/docs for Swagger-dokumentasjonen.

**5. Feilsøking (hvis nødvendig)**

* **Pods ikke starter?**
  + Sjekk pod-logger:

bash

kubectl logs <pod-name>

* **Ingen ekstern IP-adresse?**
  + Hvis du bruker et lokalt Kubernetes-kluster, som Minikube eller Docker Desktop, må du bruke NodePort eller Ingress i stedet for LoadBalancer.

**6. Neste steg**

Når backend kjører som forventet, kan du fortsette med følgende:

* Legg til klienten (frontend) som en egen deployment og service.
* Utforsk avanserte Kubernetes-funksjoner som ConfigMaps, Secrets, og Autoscaling.

Lykke til! 🚀 Hvis du støter på problemer, gi beskjed, så hjelper jeg deg videre. 😊

**Du sa:**

Hvilken <external-ip>i ' For eksempel: http://<external-ip>/docs for Swagger-dokumentasjonen.'?

**ChatGPT sa:**



For å finne verdien av <external-ip> i http://<external-ip>/docs, gjør følgende:

**1. Finn ekstern IP for LoadBalancer-tjenesten**

Etter at du har brukt kubectl apply for å opprette tjenesten (backend-service.yaml), kan du finne IP-adressen ved å kjøre denne kommandoen:

bash

kubectl get services

Du vil se en utdata som ser omtrent slik ut:

plaintext

Kopier kode

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE

doffinder-backend-service LoadBalancer 10.96.32.5 <pending> 80:31234/TCP 2m

* **CLUSTER-IP**: Intern IP for Kubernetes-nettverket.
* **EXTERNAL-IP**: IP-en eller domenet som eksponerer tjenesten eksternt.

**2. Scenarier for EXTERNAL-IP**

**a) EXTERNAL-IP viser en faktisk IP-adresse eller domene**

Hvis EXTERNAL-IP viser en gyldig IP-adresse eller et domene, bruker du denne verdien i stedet for <external-ip>. For eksempel:

plaintext

Kopier kode

EXTERNAL-IP = 192.168.49.2

Åpne nettleseren og skriv inn:

plaintext

Kopier kode

http://192.168.49.2/docs

**b) EXTERNAL-IP står som <pending>**

Hvis EXTERNAL-IP fortsatt står som <pending>, avhenger løsningen av hvilken Kubernetes-distribusjon du bruker:

1. **Lokalt kluster (Minikube eller Docker Desktop)**:
   * LoadBalancer fungerer ikke direkte lokalt. Du må bruke minikube service eller NodePort for å få tilgang.
   * For Minikube:

bash

Kopier kode

minikube service doffinder-backend-service

Denne kommandoen åpner en nettleser med riktig IP og port.

1. **Cloud Provider (GKE, EKS, AKS, osv.)**:
   * Sjekk om LoadBalancer-ressursen opprettes i skyen. Det kan ta noen minutter før en ekstern IP tildeles.
   * Når IP-en vises, bruk den til å få tilgang til applikasjonen.

**3. Alternativ: Bruk NodePort**

Hvis du ikke bruker en LoadBalancer, kan du eksponere tjenesten med NodePort. Endre tjenesten til:

yaml

Kopier kode

apiVersion: v1

kind: Service

metadata:

name: doffinder-backend-service

spec:

selector:

app: doffinder-backend

ports:

- protocol: TCP

port: 80

targetPort: 8000

nodePort: 30007

type: NodePort

Finn IP-en til Kubernetes-noden din (ofte localhost for lokale miljøer):

bash

Kopier kode

kubectl get nodes -o wide

Åpne nettleseren og naviger til:

plaintext

Kopier kode

http://<node-ip>:30007/docs

**4. Test og bekreft**

Når du har funnet riktig IP og port, test applikasjonen ved å åpne Swagger-dokumentasjonen i nettleseren:

plaintext

Kopier kode

http://<external-ip>/docs

\*ja det fungerer:

Et bilde som inneholder tekst, skjermbilde, programvare, nummer

Automatisk generert beskrivelse

**Du sa:**

Som du ser på bildet ved external-id vises localhost. Kan jeg bruke den?

**ChatGPT sa:**



Ja, du kan bruke localhost som din <external-ip> i dette tilfellet fordi Kubernetes-tjenesten er konfigurert med LoadBalancer-typen, men det peker lokalt på maskinen din via localhost. Dette er normalt når du kjører Kubernetes i et lokalt miljø som Docker Desktop.

For å teste applikasjonen din, gjør følgende:

**1. Tilgang til Swagger-dokumentasjonen**

1. Kombiner localhost med porten som vises under **PORT(S)**.
   * I dette tilfellet er porten **32419**.
2. Gå til følgende URL i nettleseren din:

plaintext

Kopier kode

http://localhost:32419/docs

1. Dette skal vise Swagger UI for FastAPI-backenden din.

**2. Feilsøking hvis det ikke fungerer**

* Hvis du ikke kan nå tjenesten via http://localhost:32419/docs:
  + Sjekk pod-statusene dine:

bash

Kopier kode

kubectl get pods

Alle pods bør ha status Running.

* + Se på loggene for backend-poden:

bash

Kopier kode

kubectl logs <pod-name>

* Sjekk om porten 32419 er blokkert av brannmuren eller annen programvare på maskinen din.

**3. For fremtidig produksjonsmiljø**

I produksjonsmiljøer vil EXTERNAL-IP være en faktisk IP-adresse eller DNS-navn (hvis du kjører på en skyplattform som GCP, AWS, eller Azure). For nå kan du trygt bruke localhost i ditt lokale miljø.

**Start Klient**

Nå som backend (http://localhost:32419/docs) fungerer, kan du starte klientdelen av applikasjonen for å koble til backend. Siden klienten er bygget med Vue.js, følger du disse trinnene for å sette den opp og koble til backend:

**1. Sjekk client-mappen**

1. Sørg for at klientkoden ligger i client-mappen i prosjektet ditt.
2. Pass på at package.json og package-lock.json finnes i denne mappen, da de brukes til å installere nødvendige avhengigheter.

**2. Konfigurer backend-URL for klienten**

For at klienten skal kunne kommunisere med backenden, må du konfigurere klienten til å bruke backend-URL-en (http://localhost:32419). Dette gjøres vanligvis i en konfigurasjonsfil eller miljøfil.

1. Sjekk etter en fil som .env eller config.js i client-mappen. Hvis en slik fil eksisterer, se etter en linje som definerer backend-URL, som for eksempel:

plaintext

Kopier kode

VITE\_API\_URL=http://localhost:32419

1. Hvis en slik fil ikke finnes, legg til backend-URL-en i riktig del av klientens kode (vanligvis i en axios-oppsettfil eller lignende).

**3. Installer klientavhengigheter**

Gå til client-mappen i terminalen og kjør:

bash

Kopier kode

npm install

Dette laster ned alle nødvendige avhengigheter.

**4. Start klienten i utviklingsmodus**

Start klienten ved å kjøre følgende kommando i client-mappen:

bash

Kopier kode

npm run dev

Dette starter en utviklingsserver for klienten, vanligvis på http://localhost:5173 (eller en annen port som vises i terminalen).

**5. Test klienten**

1. Åpne nettleseren og gå til den oppgitte URL-en (for eksempel http://localhost:5173).
2. Interager med klienten for å verifisere at den kan kommunisere med backend.
   * Hvis du støter på problemer, sjekk nettverksforespørslene i nettleserens utviklerverktøy (F12 > Network) for å bekrefte at forespørslene går til riktig backend-URL.

**6. Bygg klienten for produksjon (valgfritt)**

Hvis du ønsker å bygge klienten for produksjon (slik at den kan serveres sammen med backenden):

1. Kjør følgende kommando i client-mappen:

bash

Kopier kode

npm run build

1. Dette vil generere en dist-mappe i client/, som inneholder ferdige statiske filer.
2. Kopier filene fra dist til backenden for å tjene klienten via FastAPI:
   * Oppdater FastAPI-koden til å tjene statiske filer fra client/dist.

**Info om prosjekt DOFFINder:**

1. Dokumentasjon/kode for DOFFINder:

<https://github.com/ItemConsulting/DOFFINder>

1. Installasjon ved GitHub:

<https://github.com/ItemConsulting/DOFFINder>

Ignacio Tejera 15.01.2025

redigert