Kubernetes Exam Rappport

# Indledning

Kubernetes er et kraftfuldt software, som gøre deployment af services nemt og robust. Det er et vigtigt technologi at have kendskab til, da det er hvad der levere ens kode til kunderne. I dette forløb vil vi kigge på hvad Kubernetes er, hvordan vi kan få det til at virke, og lave et projekt til at teste vores viden.

# Målet / problemformulering

# MoSCoW

Da Kubernetes er advanceret, og har mulighed for så meget mere end dette projekt kan vise, kan vi fravælge områder som vi ikke vil se på, og sætte prioriteringer på produktet. En god måde at gøre dette på, er med MoSCoW modellen. [1]

MoSCoW modellen har primært 4 punkter, hvor man organisere ens ”to-do” liste. Den består af ”Must have” som man kan tænke på som hårde krav. Set på den måde at for at have en MVP (minimal viable product) skal disse punkter være til stede. Så er der ”should have” som er blødere krav. De er ikke nødvendige for MVP’en, men uden dem føles produktet tomt. Dernæst er der ”Could have”, som er bløde krav. Ting som kunne være fedt hvis produktet havde, men som ville være ekstra ting som ikke er noget specifikt nævnt skulle være en del af produktet. Og til sidst er der ”Wont have” som er ens fravalg. De ting der er for store eller uden for scope til at være muligt at inkludere i sit produkt, enden MVP eller det fulde produkt. Så med det grundlag, hvad ser dette projekt ud i den model?

## Must Have

Igennem projektet har kravene ændret sig en del. Til at starte med kunne vi se på casen og tænkte dem som vores hårde krav. Men rettelser eller opdatering til casen har ændret hvad var krav. Da der ikke er et sted længere at se alle de hårde krav, bliver vi nødt til at definere dem selv, og håbe det er nok.

Til det formål, har vi valgt at vi gerne ville have et projekt med en frontend og en backend kubernetes cluster. Med kommunikation mellem dem i form af api kald. Dette er et meget ”bare-bones” version, uden fancy database og CI/CD-pipelines [2]. Tanken bag denne beslutning var at fokusere på hvad kubernetes er, og hvordan det virker.

## Could have

Automatiske CI/CD-pipelines er en måde at gøre kubernetes nemmere at arbejde med. Og i dette tilfælde med at lille projekt, giver det ikke mening at sætte det op. Og database er dog en anden form af pod, men ville tage meget fokus, da det skal fungere fundamentalt anderledes end normale pods, hvis data skal være persistent. Men det er stadig ting som er beskrevet i casen. Dermed kan vi tænke på disse områder, når MVP er færdig, og vi har mulighed for at tilføje mere til programmet. Hvis vi ikke når dette, betyder det dog ikke for meget for om vi kan sige at opgaven er en ”succes” eller ej.

## Should have

Logging og tests i et system er generelt godt at have, da det vil hjælpe med eventuelle fejl. Kubernetes / Docker containers har allerede en log man kan se hvis der er problemer i en pod, og dermed tænkte vi at custom logging var ting vi kun ville implementere, hvis vi havde alt andet på plads, og selv havde lyst til at lave mere på programmet.

Tests for et større program er vigtige, da der er for meget at kigge på hver fil inolveret for at se hvor en fejl sker, og det kan også være svært at vide om sit program gør som forventet, når mange filer / linjer er i gang på samme tid. Men i vores lille projekt til at få kubernetes op at køre, er der kun 2 filer i gang, som begge kun har meget lidt indhold. Dermed ville inkludering af tests være noget som ville blive tilføjet, med det ene formål at vise hvordan det ville være i et større system, men næsten ingen funktion til vores projekt.

## Wont Have

Vores projekt er småt, og er ment til at være en test af kubernetes. Dermed for at se om det virker, holder vi os til en instans af både frontend og backend. Det er kubernetes / dockers styrke at have flere instancer kørende, men for at få noget ud af flere containers / pods, ville det kræve en suite af tests, der loader flere instancer fra flere pods. Det er meget ekstra at lave for at få næsten ingen ekstra information om kubernetes. Dermed har vi valgt at vi kun vil have en container kørende af frontend og backend. Og større opscaling vil vi ikke komme tæt på i dette projekt.

# Fremgang

# Problemer

# Resultat

# Konklusion