STS, KMD I2 og STIL

Hvordan man bygger docker images

**Version**: 1.1.1

**Date**: 26.02.2021

**Author**: BSG

Indhold

[1 Indledning 3](#_Toc65229192)

[1.1 Forudsætning 3](#_Toc65229193)

[2 Placering af docker byggescripts 3](#_Toc65229194)

[2.1 build-image.sh 3](#_Toc65229195)

[2.2 Dockerfile 3](#_Toc65229196)

[3 Eksempel kørsel 3](#_Toc65229197)

# Indledning

Nærværende dokument er kort vejledning til hvordan man bygger Docker images ud fra den kode der findes i koderepositoriet.

Det antages at man er bekendt med Docker og allerede har Docker installeret på sin maskine

## Forudsætning

For at bygge disse docker images, skal man have en Linux maskine med Docker installeret. Det er ikke nødvendigt at have udviklingsværktøjer, da disse hentes som en del af bygget.

# Placering af docker byggescripts

Hvert kode-repository har en docker folder under sig, med en underfolder ved navn ”build”.

I build folderen findes 2 filer

## build-image.sh

build-image.sh er et simpelt shell script, der kopierer den relevante kildekode ind i en byggefolder, og eksekverer en Docker build kommando på denne kildekode.

Den faktiske udførsel af bygget er defineret i Dockerfile filen som beskrevet nedenfor

## Dockerfile

Der findes en Dockerfile til hvert kode-repository, men ud over små forskelle i porte og filnavne, så er strukturen identisk.

Der anvendes en såkaldt docker build-chain, hvor der først startes en Docker container op, der indeholder de relevante byggeværktøjer (fx en Java kompiler), hvor kildekoden så bygges.

Dernæst overføres de binære artifakter til et Docker base-image, der indeholder netop de ting der skal til for at afvikle den kompilerede applikation.

Det endelige docker image ligger lokalt på den maskine hvor den er bygget, og kan enten anvendes der, eller publiceres i et docker repository.

For at publicere docker images i et repository skal man enten opsætte et repository, eller få en adgang til et offentlige sådanne (fx Dockerhub). Dette er udenfor scope af denne vejledning.

# Eksempel kørsel

Nedenfor vises det forventede output (minimalistiske output) ved en kørsel. Man må forvente at første gang man kører jobbet at der vil være betydeligt mere output, da der skal downloades resourcer.

$ ./build-image.sh

Sending build context to Docker daemon 61.95kB

Step 1/11 : FROM maven:3-jdk-11 as build

---> 4fbb58beddc4

Step 2/11 : WORKDIR /workspace/app

---> Using cache

---> ddc0535210bd

Step 3/11 : COPY build/pom.xml pom.xml

---> Using cache

---> bfe5327e10ce

Step 4/11 : RUN mvn dependency:go-offline -B

---> Using cache

---> c41d01ec328b

Step 5/11 : COPY build/src src

---> Using cache

---> 59ae1c9ba974

Step 6/11 : RUN mvn package -DskipTests

---> Using cache

---> 57ba76c2126b

Step 7/11 : FROM openjdk:11

---> 4a438d74f9ed

Step 8/11 : ARG TARGET=/workspace/app/target

---> Using cache

---> ae59e7537f61

Step 9/11 : COPY --from=build ${TARGET}/os2rollekatalog-integration.jar .

---> Using cache

---> fbc0dd1c2c31

Step 10/11 : EXPOSE 8999

---> Using cache

---> fcb6851c4fc1

Step 11/11 : ENTRYPOINT ["java", "-Xmx128m", "-jar", "os2rollekatalog-integration.jar"]

---> Using cache

---> 66d0c373ea25

Successfully built 66d0c373ea25

Successfully tagged os2services:rolesfromos2rollekatalog