Docker

Opgavebeskrivelse

Note: Hvis du ikke nåede at implementere unit-tests i din spilopgave fra sidste uge, så forsøg at komme i mål med det, inden du går starter på nedenstående opgave.

I denne uge skal I arbejde med containerization og Docker.
Docker er et værktøj der pakker din applikation sammen med alt den har brug for – programmeringssprog, biblioteker, konfigurationsfiler og andre afhængigheder – i en såkaldt "container". En container er en letvægts, selvstændig pakke, der kan køre ensartet på enhver computer, uanset om det er Windows, Mac eller Linux. Man undgår derved det berømte problem: "*Jamen det virker på min computer!*"

Docker er industristandard i moderne softwareudvikling, og bruges af virksomheder verden over til udvikling, test og deployment. Ved at lære Docker får du erfaring med et værktøj der er direkte anvendelig både i resten af akademiforløbet og i din fremtidige karriere, da det gør det markant nemmere at dele projekter, samarbejde i teams, og deploye til produktionsmiljøer.

Docker kan synes svært og forvirrende i starten, og vi forventer ikke at I bliver eksperter på området i løbet af én uge. Formålet er at give jer en introduktion og grundlæggende kendskab til Docker, og det er samtidig et redskab vi ville inddrage i nogle af opgaverne senere på akademiet.

Har man tidligere erfaring med Docker og containerisering, anbefaler vi at man starter ved delopgave 4.

Afleveringsformat

- GitHub-repository med mapper for hver delopgave der indeholder jeres Dockerfiler

- Dokumentér dine valg i dit GitHub repository's README. README.md skal indeholde:
    - For hver delopgave:
        - Instruktioner til at bygge og køre din løsning
        - Beskrivelse af tekniske valg (base images, networking, optimering)
        - Kort refleksion over udfordringer og læring
    - Specifikt for delopgave 4:
        - Image størrelse før og efter optimering
        - Liste over implementerede optimeringer
        - Vurdering af hvilke optimeringer havde størst effekt
    - Se README template på Teams for struktur-inspiration

Delopgaver og opmærksomhedspunkter

Delopgave 1: Kom i gang med Docker

- Hvis man ikke har erfaring med Docker, anbefaler vi at man kigger nærmere på den officielle dokumentation for at komme i gang: https://docs.docker.com/get-started/
- Installer Docker Desktop på din computer
- Eksperimenter med basic Docker-kommandoer:

```
docker run hello-world
docker run -it ubuntu bash
docker ps
docker images
docker rm / docker rmi
```

- Læs om forskellen mellem images og containere
- Forstå hvad et Dockerfile er
- Lær om Docker Hub og base images
- Eksperimenter med at køre eksisterende images (f.eks. docker run -it python:3.11)

Delopgave 2: Containeriser en simpel webserver

- Du får en færdig simpel webserver (download mappen "Docker-startkode" på Teams og åben "Delopgave 2 – simpel webserver")
- Skriv en Dockerfile der kan bygge og køre serveren
- Byg image og kør container
- Verificer at serveren virker på localhost:5000

Spørgsmål til refleksion:

- Hvorfor skal vi bruge `-p 5000:5000`?
- Hvad betyder `host='0.0.0.0'` i app.py?
- Hvorfor nulstilles besøgstælleren når containeren genstarter?
- Hvordan kunne vi gemme besøgstælleren permanent? (Hint: Vi ser på det i delopgave 3!)

Delopgave 3: Multi-container setup med Docker Compose

- Udvid webserveren til at bruge en PostgreSQL database (findes i mappen "Delopgave 3 - Multi-container setup")
- Skriv docker-compose.yml der kører både webserver og database
    - OBS: Denne version kan IKKE køre alene uden en database. I skal bruge Docker Compose til at starte både webserver og database sammen
- Forstå networking mellem containers
- Arbejd med volumes for datapersistens

Spørgsmål til reflektion:

- Hvorfor bevares data når vi bruger volumes?
- Hvad er forskellen på `docker-compose down` og `docker-compose down -v`?
- Hvorfor skal web service have `depends_on: db`?
- Hvad sker der hvis du sletter volumes og starter igen?
- Hvordan kommunikerer web og db containerne? (Hint: Docker network)

Delopgave 4: Optimering

- Optimer den ikke-optimerede Dockeropsætning i mappen “Delopgave 4 – optimering"
- Byg Dockerfile i webserver/ mappen og noter image-størrelsen
- Identificer problemer (se README.md i mappen for områder at undersøge)
- Optimer Dockerfile ud fra Docker best practices
- Sammenlign image-størrelse før og efter
- Dokumentér dine valg i din README til ugens opgave

Valgfrie ekstra udfordringer (hvis du bliver tidligt færdig)

- GitHub Actions CI/CD: Opsæt automatisk Docker image building ved push til GitHub
- Docker Hub: Push dit image til Docker Hub så andre kan bruge det
- Arbejd videre på dit Spil fra sidste uge – prøv at sætte det op i en Docker-container og få en medkursist til at spille det på deres PC
- Prøv kræfter med de lidt mere avancerede Docker-tutorials til Docker under “Ressourcer”

Ressourcer

- Containerization.pdf
- Docker – workshop: https://docs.docker.com/get-started/workshop/
- Videoressourcer (søg på YouTube):
  - "Docker Tutorial for Beginners"
  - "Learn Docker in 12 Minutes"
  - "Docker Crash Course"
- https://docker-curriculum.com/
- https://gist.github.com/arrested-developer/0e950b2b74b50e89b305cf6d9a7f28b5
