AARHUS UNIVERSITET

Software Engineering and Architecture

Containers - Docker

Lightweight Virtual Machines

AARHUS UNIVERSITET

From Dev to Ops

- We have this wonderful client-server system developed
  - Lots of unit and integrations tests, highly flexible, highly reliable
- *But we only have it on our local laptops*
  - Git clone, gradle something

- **We need to make it run in *production***
  - **Deployment** ***idriftsættelse***
- Requires
  - A) A computer with DNS name on the 'big internet'
  - B) Get our executable system on to that machine

AARHUS UNIVERSITET

The Old Way

- … a manual procedure
- *Buy a server farm* *or* *Rent a VM in the cloud*

- *Ah – we need a database for the server*
  - *Rent one more machine; install linux, install MySQL, copy table init scripts, configure linux for database systems, …*
- *Now, log into the server*
  - *Install linux, git, java, gradle, …*
  - *Git clone the repository*
  - *Run the executable using 'parameters for production'*

AARHUS UNIVERSITET

But it does not scale…

- … Imagine 10.000+ machines to do that on ☹
  - And imagine that 100 persons handles 100 machines each, each doing it in a slightly different way
  - In one year, you have 10.000 machines *configured in about 500 different ways, meaning any update/fix of the software requires different actions for each machine* ☹
- ***The multiple maintenance problem for operations***
  - Not multiple copies of code, but of machine configurations !

Example: Uber

- ~100.000 VMs to run the Uber infrastructure…

58K

Builds / week

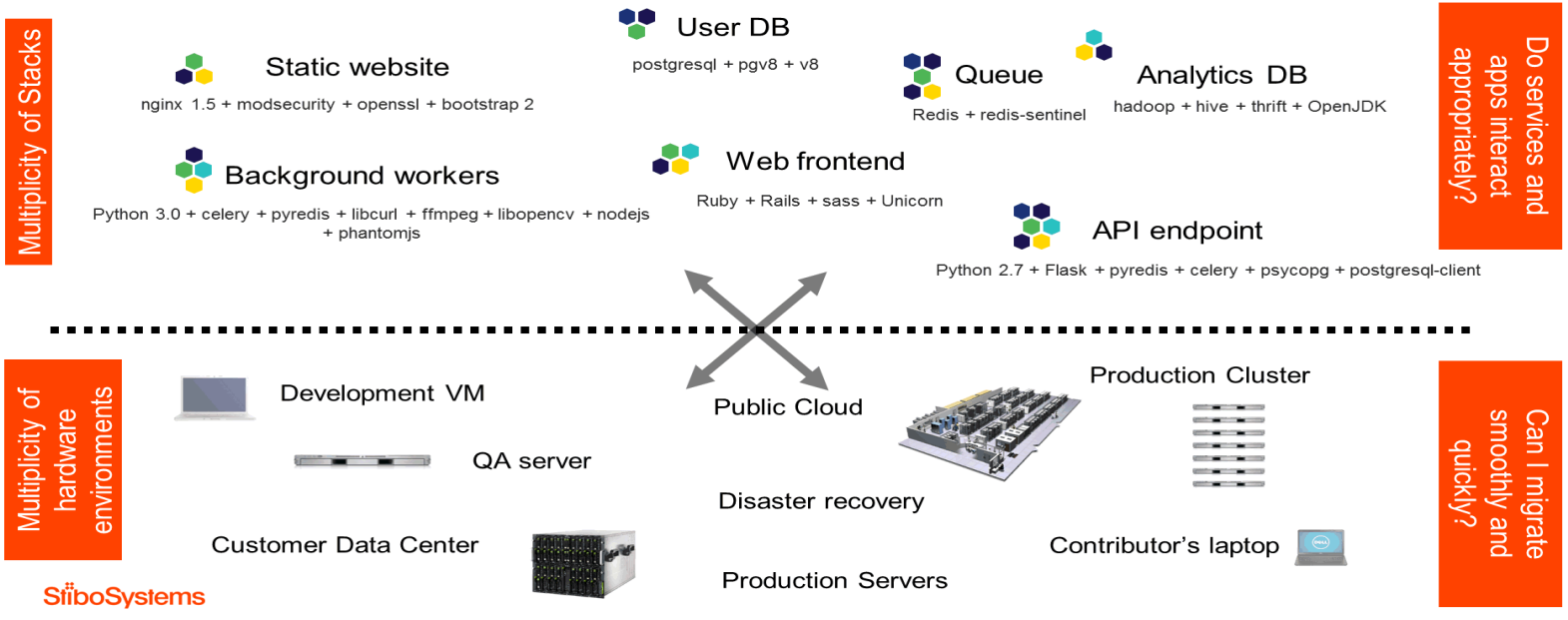
5K

Production deploys / week

AARHUS UNIVERSITET

The *Moving Code* Problem

- Crossing boundaries, that is, *moving code*

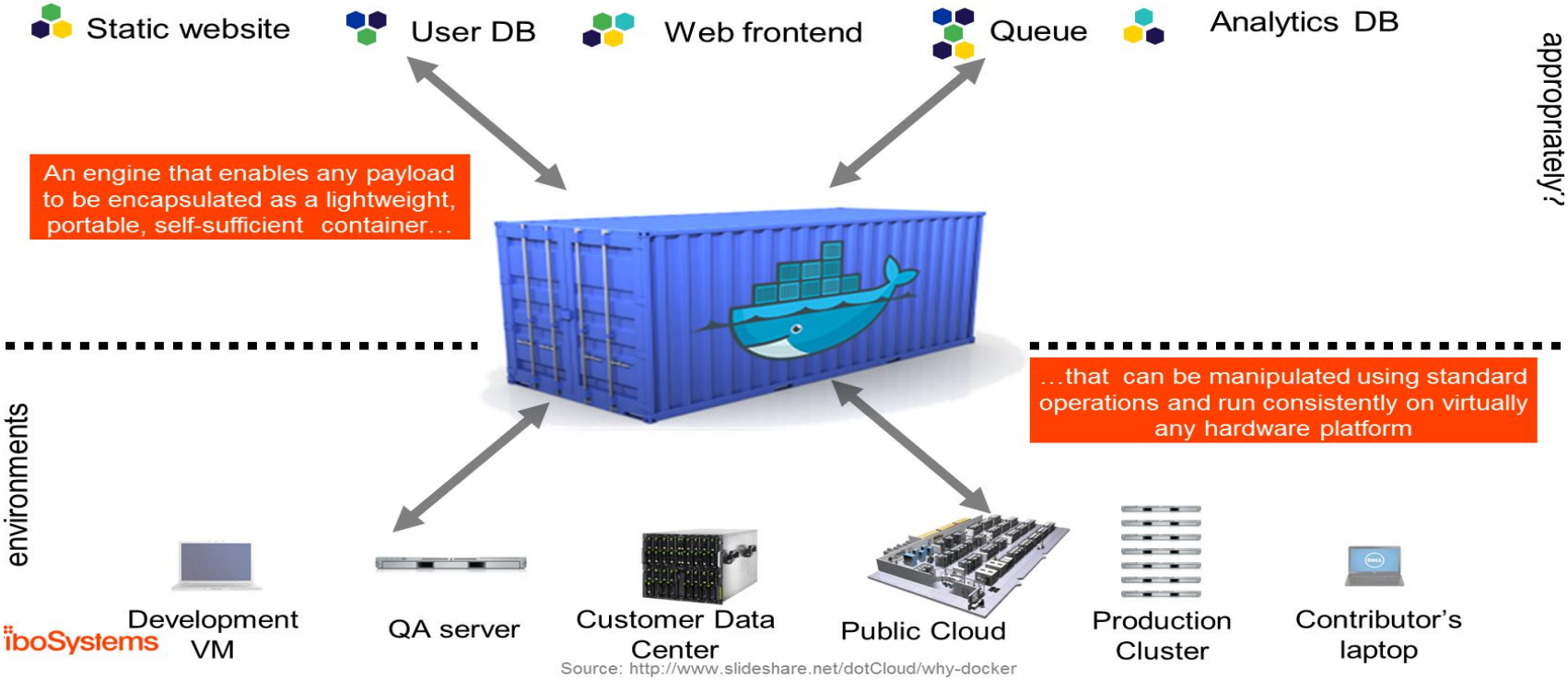


Source: Torben Haagh, StiboSystems

AARHUS UNIVERSITET

Was Solved in 1960'ies

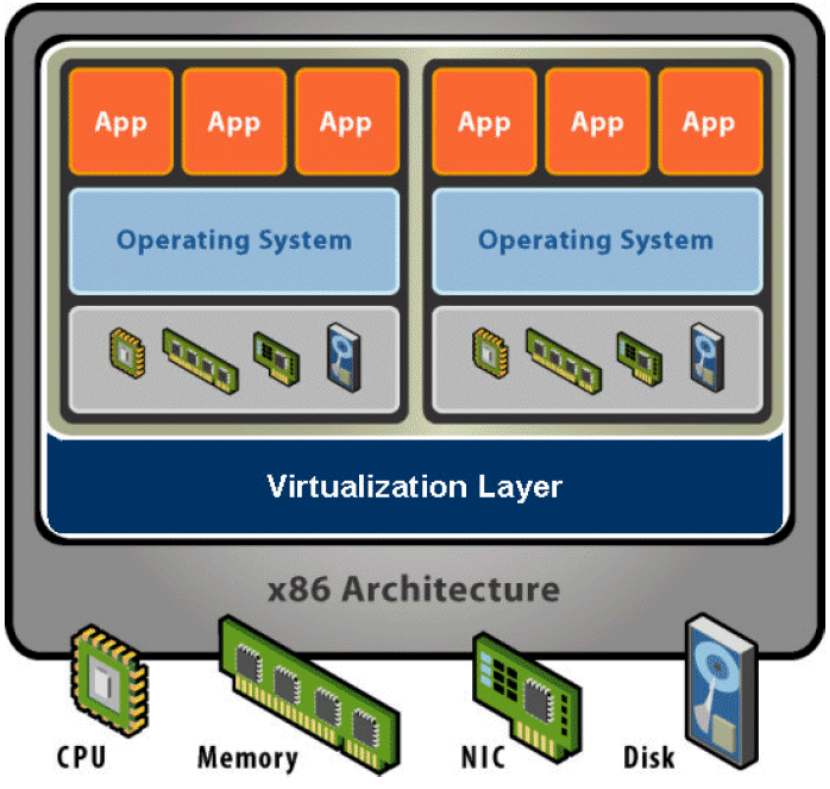
Docker = Container



Not Just Programs - Environments

AARHUS UNIVERSITET

- A Container is a **Virtual Machine**
  - A VM is
    - The operating system
    - All supporting libraries
    - The executable / system
- That is:
  - The code
    + the full environment of the (virtual) machine
  - ***It is self-contained! No external dependencies!***



AARHUS UNIVERSITET

So – What do We Get?

- Instead of your specific labtop which is
  - Lots of programs and your specific configuration of OS
  - *And then HotStone (server) manually fiddled to make it run*
- … we instead build a *container* which
  - Includes the OS + all supporting libraries + HotStone code
- That can be *run on a "bare metal" computer that has no specific configuration*
  - Except – it of course must be able to run containers:
- **Docker Engine**
  - A program to deploy/monitor a set of Docker containers

Solves Scaling Issue

- Problem solved:
  - We rent 10.000 identical machines with no special configuration
    - (that runs docker engine, ok)
  - And then deploy containers with our code in.

AARHUS UNIVERSITET

Building a Container

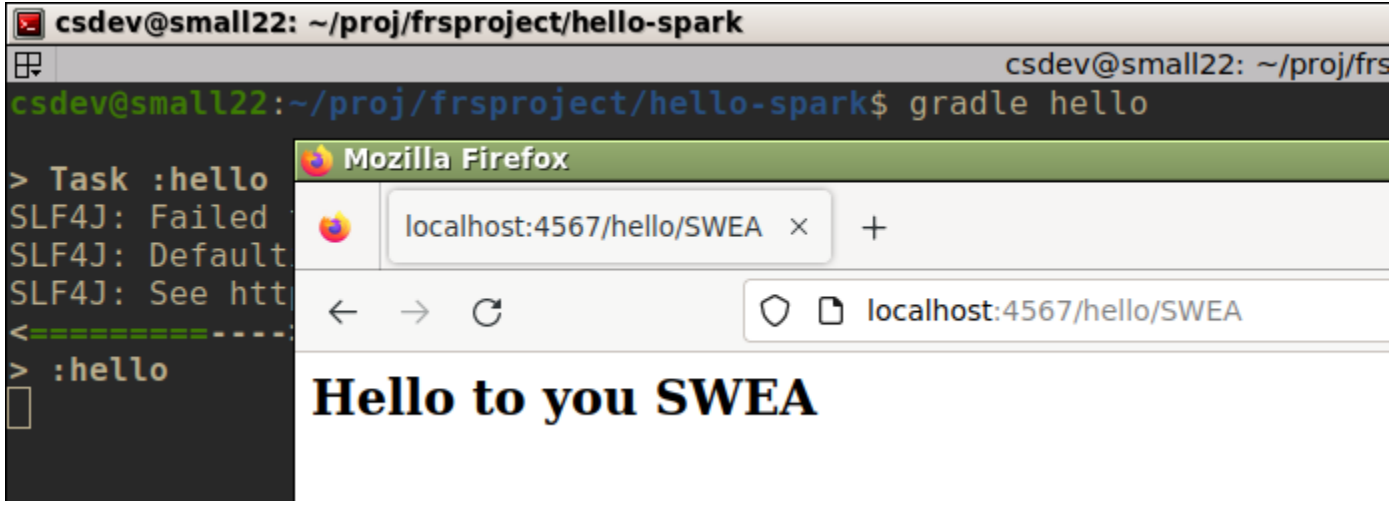
Infrastructure-as-code

“IaC”

AARHUS UNIVERSITET

Example: Hello-Javalin

- The most basic web server system
  - ‘gradle hello’
    - Starts a web server on port 4567
  - Browse to ‘(servername):4567/hello/(a name here)’



AARHUS UNIVERSITET

My Container…

- I want to build a docker container which contains
  - Linux Operating System
  - Java 25 runtime system
  - Gradle v9.2
  - And my Hello-Javalin code
    - build.gradle
    - src/ folder
- Can run on *any machine in the world* if it has docker engine installed *and nothing else required!*

AARHUS UNIVERSITET

Building Containers

- *Infrastructure-as-code, IaC*
- We write *code* to build the container
  - Dockerfile
    - A DSL for building Docker images
- Image = Static unit
- Container = Runtime unit

```
# Hello Javalin Demo docker file

# To build: docker build -t henrikbaerbak/private:hello-javalin .

# To run: docker run -d -p 4567:4567 henrikbaerbak/private:hello-javalin

# Base image - java 25 + gradle 9.2
FROM gradle:9.2-jdk25

# Insert my name and email in resulting container
LABEL maintainer="HenrikBaerbakChristenen_hbc@cs.au.dk"

# Create the /root/hello folder and change to it
WORKDIR /root/hello

# Copy the source code from HOST into GUEST
COPY src/ src/
COPY build.gradle build.gradle

# Expose the port that hello-javalin will use
EXPOSE 4567

# Define the default command to run
CMD ["gradle","hello"]
```

IaC

- Parts:
  - Base
  - Copy
  - Execution

```
# Hello Javalin Demo docker file

# To build: docker build -t henrikbaerbak/private:hello-javalin .

# To run: docker run -d -p 4567:4567 henrikbaerbak/private:hello-javalin

# Base image - java 25 + gradle 9.2
FROM gradle:9.2-jdk25

# Insert my name and email in resulting container
LABEL maintainer="HenrikBaerbakChristenen_hbc@cs.au.dk"

# Create the /root/hello folder and change to it
WORKDIR /root/hello

# Copy the source code from HOST into GUEST
COPY src/ src/
COPY build.gradle build.gradle

# Expose the port that hello-javalin will use
EXPOSE 4567

# Define the default command to run
CMD ["gradle","hello"]
```

AARHUS UNIVERSITET

Building

- Image name: henrikbaerbak/private:hello-javalin

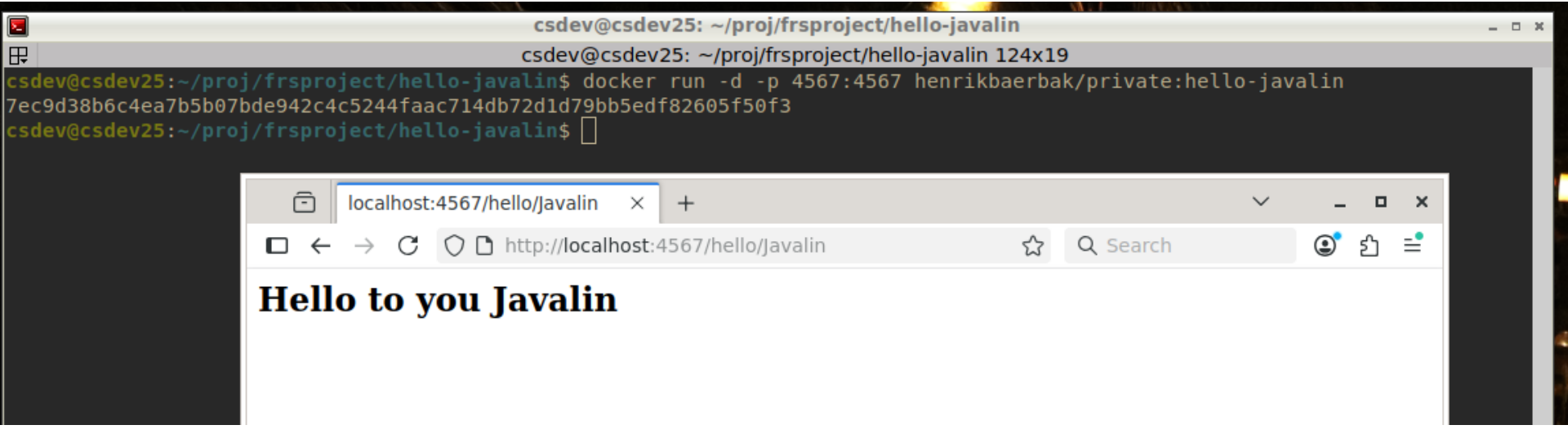
```
^Ccsdev@csdev25:~/proj/frsproject/hello-javalindocker build -t henrikbaerbak/private:hello-javalin ..
[+] Building 58.0s (10/10) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 659B
 => [internal] load metadata for docker.io/library/gradle:9.2-jdk25
 => [auth] library/gradle:pull token for registry-1.docker.io
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [1/4] FROM docker.io/library/gradle:9.2-jdk25@sha256:3dec978fd14b3dc1083dfbeedfab9dd9da3d23bbb1a5d1e8b618b88e1ea3d35d
 => => resolve docker.io/library/gradle:9.2-jdk25@sha256:3dec978fd14b3dc1083dfbeedfab9dd9da3d23bbb1a5d1e8b618b88e1ea3d35d
 => => sha256:20043066d3d5c78b45520c5707319835ac7d1f3d7f0dded0138ea0897d6a3188 29.72MB / 29.72MB
 => => sha256:627c55a201a9095fd9bb5b05a4c058246c463987d178f778ff22db143ca94635 17.46MB / 17.46MB
 => => sha256:378e3a6f165ea02b44863b980aacd0a6ba57f3b96b08da831cc39bb19eb41760 92.17MB / 92.17MB
 => => sha256:0831bfbf48c27218723195229f9cf3f29e3647cbd020e047cad185406b07fbf3 2.88kB / 2.88kB
 => => sha256:d1f42570e4f2a7746a98424c38db09dd41cc03aaaac022003384cb092cd89898 11.05kB / 11.05kB
```

```
 => [internal] load build context
 => => transferring context: 2.40kB
 => [2/4] WORKDIR /root/hello
 => [3/4] COPY src/ src/
 => [4/4] COPY build.gradle build.gradle
 => exporting to image
 => => exporting layers
 => => writing image sha256:e71c1a8533923f465c238c69d2d5fc591cd5773056cb6d7b7b7f70f3a2bd7054
 => => naming to docker.io/henrikbaerbak/private:hello-javalin
```

AARHUS UNIVERSITET

Local Running/Testing

- ‘run’
  - -p 4567:4567 The container’s port is mapped to localhost
  - -d In the background (daemon)



AARHUS UNIVERSITET

Deployment

Via Docker Hub

Hub

- Maven Repository is a cloud service hosting java libraries
- *Docker hub does the same for images!*
  - Push to my ‘henrikbaerbak’ account on docker hub.
- **Now it is globally accessible !**

```
csdev@small22:~/proj/frsproject/hello-spark$ docker push henrikbaerbak/private:hello-spark
The push refers to repository [docker.io/henrikbaerbak/private]
b31fdb6dd8f5: Pushed
6b0cfe77d40c: Pushed
f8c849cc2d38: Pushed
ef3a0dec6a89: Layer already exists
605477091eb2: Layer already exists
07099f814ad2: Layer already exists
03b38add18fe: Layer already exists
cd59091cb1bf: Layer already exists
151ff94a03ea: Layer already exists
a7b7cb7db02e: Layer already exists
793368f2be0c: Layer already exists
01d4e4b4f381: Layer already exists
```

AARHUS UNIVERSITET

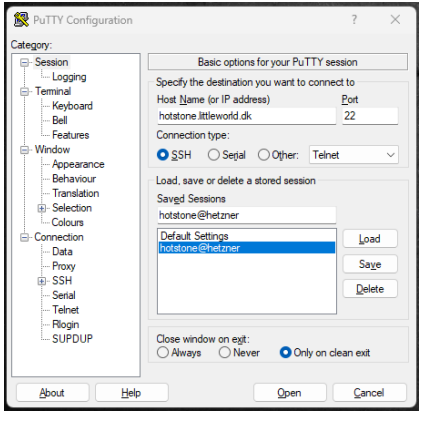
To Helsinki

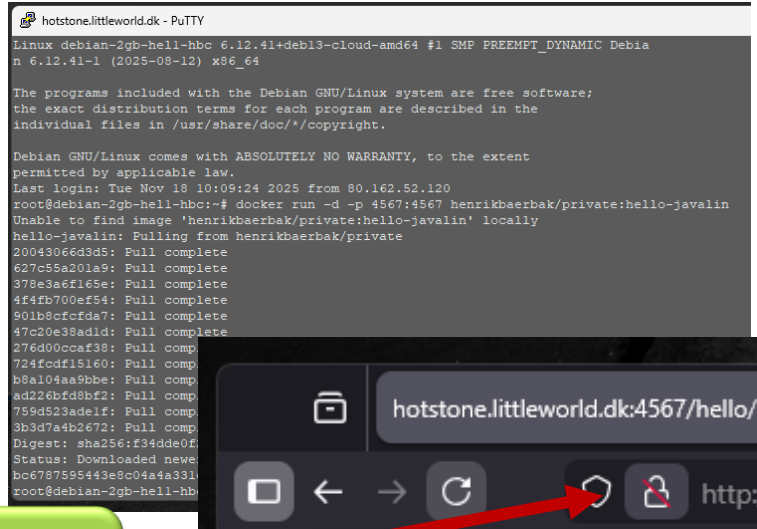
- I want to deploy it on my rented machine in Helsinki
  - Which has DNS ‘hotstone.littleworld.dk’

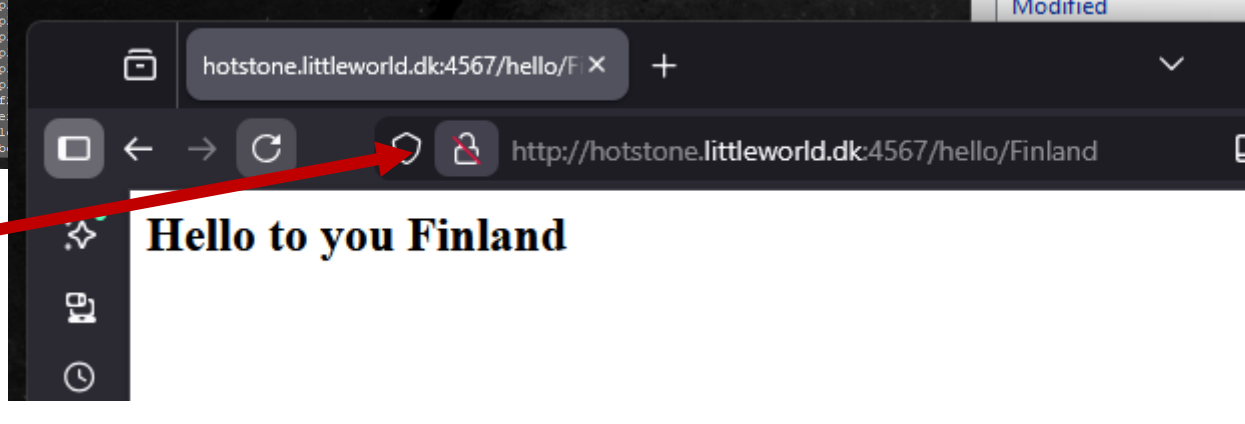
AARHUS UNIVERSITET

Process

- Log into my machine in Helsinki
  - Using my Windows Putty secure shell; and ‘run’ container
    - Ah – also open port 4567 in the firewall ☺







Try it ☺

Bottomline

- It takes about *1 minute!*
- My Helsinki machine does not have neither Java nor Gradle, nor Go, nor Erlang, nor Rust, nor C++… installed
- *And it can run it all, if they are containerized!*

AARHUS UNIVERSITET

Examples

- My ‘cave service’ in GoLang

```
18 # A CaveService written in Go
19
20 # docker build -t henrikbaerbak/caveservice:go .
21 # docker run -d -p 9999:9999 --name cavesrv henrikbaerbak/caveservice:go
22
23 FROM golang:1.19-alpine AS builder
24
25 WORKDIR /app
26
27 COPY go.mod ./
28 COPY go.sum ./
29
30 RUN go mod download
31
32 COPY storage/ storage/
33 COPY position/ position/
34
35 COPY server.go ./
36
37 # Build the executable
38 RUN go build -o cavesrv server.go
39
40 FROM golang:1.19-alpine
41
42 WORKDIR /app
43
44 COPY --from=builder /app/cavesrv ./
45
46 EXPOSE 9999
47
48 # Run it
49 CMD ["/app/cavesrv"]
```

AARHUS UNIVERSITET

Examples

- My ‘cave service’ in Python

```
13 FROM python:3.8-slim-buster
14
15 # FROM python:3.11.2-bullseye
16
17 LABEL maintainer="HenrikBaerbakChristenen_hbc@cs.au.dk"
18
19 WORKDIR /app
20
21 COPY requirements.txt requirements.txt
22
23 RUN pip3 install -r requirements.txt
24
25 COPY app.py app.py
26 COPY storage.py storage.py
27 COPY point3.py point3.py
28
29 EXPOSE 9999
30
31 # Use the cmd below to run the 'werkzeug' development web server
32 # CMD [ "python3", "-m" , "flask", "run", "--host=0.0.0.0", "-p", "9999"]
33
34 # For production usage -using 4 threads
35 # CMD ["gunicorn", "-w", "4", "app:app", "-b", "0.0.0.0:9999"]
36
37 # For debugging
38 #CMD ["gunicorn", "-w", "16", "app:app", "-b", "0.0.0.0:9999", \
39 #                 "--log-level", "info"]
40
41 # Waitress running (defaults to 64 threads)
42 CMD ["waitress-serve", "--threads", "64", "--port", "9999", \
43     "--connection-limit", "400", \
44     "--channel-timeout", "30", \
45     "app:app"]
```

AARHUS UNIVERSITET

Examples

- My ‘cave service’ in Scala

```
 9 FROM openjdk:11 AS builder
10
11 # --- Install Scala and Sbt
12 RUN apt install curl
13 RUN curl -fL https://github.com/coursier/launchers/raw/master/cs-x86_64-pc-linux.gz | gzip -d
   > cs
14 RUN chmod +x cs
15 RUN ./cs setup --yes
16
17 # Copy the scala project files
18 WORKDIR /root/caveservice
19
20 # Copy source files
21 COPY src/ src/
22 COPY project/ project/
23
24 # build files
25 COPY build.sbt build.sbt
26
27 # Create the fat jar containing all code
28 RUN /root/.local/share/coursier/bin/sbt assembly
29
30 # === Execution container
31
32 FROM adoptopenjdk/openjdk11:alpine-jre
33
34 # Create the run folder
35 WORKDIR /root/run
36
37 # Copy the fat jar there
38 COPY --from=builder /root/caveservice/target/scala-2.13/caveservicescala-assembly-1.0.0.jar ./
   caveservice.jar
39
40 # Default port exposed
41 EXPOSE 9999
42
43 # Define the default command to run, defaulting to fake (in-memory) storage
44 CMD ["java","-jar", "caveservice.jar"]
```

AARHUS UNIVERSITET

Examples

- My ‘cave service’ in Java

```
12 FROM henrikbaerbak/jdk11-gradle68 AS builder
13
14 LABEL maintainer="HenrikBaerbakChristenen_hbc@cs.au.dk"
15
16 # Tell Java tools that source files are UTF8
17 ENV JAVA_TOOL_OPTIONS -Dfile.encoding=UTF8
18
19 # Copy all relevant stuff for compilation
20 WORKDIR /root/caveservice
21
22 # Copy source files
23 COPY src/ src/
24
25 # Ensure gradle can produce jar
26 COPY build.gradle build.gradle
27 COPY gradle.properties gradle.properties
28 COPY settings.gradle settings.gradle
29
30 # Create the fat jar containing all code
31 RUN gradle jar
32
33 # === Execution container
34
35 FROM adoptopenjdk/openjdk11:alpine-jre
36
37 # Create the run folder
38 WORKDIR /root/run
39
40 # Copy the fat jar there
41 COPY --from=builder /root/caveservice/build/libs/caveservice.jar ./
42
43 # Default port exposed
44 EXPOSE 9999
45
46 # Define the default command to run, defaulting to fake (in-memory) storage
47 CMD ["java","-jar", "caveservice.jar", "fake", "localhost:7777"]
```

AARHUS UNIVERSITET

And Much More

- My Hetzner machine runs other stuff for me…

```
root@debian-2gb-hell-hbc:~# docker ps -a
CONTAINER ID   IMAGE                                        COMMAND                  CREATED          STATUS
bc6787595443   henrikbaerbak/private:hello-javalin           "/__cacert_entrypoin…"   11 minutes ago   Up 11 minutes
45735b064ca5   henrikbaerbak/private:hotstone-server-0825    "dumb-init java -jar…"   2 months ago     Up 2 months
630b3675a750   mariadb:10.5.8                               "docker-entrypoint.s…"   2 months ago     Up 2 months
root@debian-2gb-hell-hbc:~#
```

- The hot stone game server with its associated MariaDB is is running on the machine...

AARHUS UNIVERSITET

And the World is Big

- All major players of open source have containers for their products.
- Want to have a MongoDB database?
  - Easy! It takes about ½ minute to get it installed and running!

```
csdev@small22:~$ docker run -d --name my-mongo mongo:6
Unable to find image 'mongo:6' locally
6: Pulling from library/mongo
43f89b94cd7d: Already exists
54a7480baa9d: Pull complete
7f9301fbd7df: Pull complete
5e4470f2e90f: Pull complete
40d046ff8fd3: Pull complete
64d7f1f42d4e: Pull complete
c9074c327c8e: Pull complete
aff86e26764a: Pull complete
78bccb6b3ef9: Pull complete
Digest: sha256:9f107d34c284d9be2527341d6d511cd818a569d83b4c52d1aef4ca7a75c487d7
Status: Downloaded newer image for mongo:6
9129d3ab5a64332056db4f030d30b6681a8d2440c9a2b63069a09df54298361c
```

```
docker exec -ti my-mongo mongosh
```

```
my-database> db.course.insertOne({teacher:"hbc", year:2023, name:"swea"})
{
  acknowledged: true,
  insertedId: ObjectId("65537aac86851cf385d40c51")
}
my-database> db.course.insertOne({teacher:"hbc", year:2021, name:"saip"})
{
  acknowledged: true,
  insertedId: ObjectId("65537ab686851cf385d40c52")
}
```

```
my-database> db.course.find()
[
  {
    _id: ObjectId("65537aac86851cf385d40c51"),
    teacher: 'hbc',
    year: 2023,
    name: 'swea'
  },
  {
    _id: ObjectId("65537ab686851cf385d40c52"),
    teacher: 'hbc',
    year: 2021,
    name: 'saip'
  }
]
```

AARHUS UNIVERSITET

Deploying Systems

Outlook to Orchestration Tools

AARHUS UNIVERSITET

Only one service…

- ‘docker run …’ allows me to ‘start one service’
- But systems in practice need many more services
- Ex: HotStone server on hotstone.littleworld.dk

```
d6fe4248ddbc   henrikbaerbak/private:hotstone-server   "dumb-init java -jar…"   4 weeks ago   Up 4 weeks
dc76f411623a   mariadb:10.5.8                         "docker-entrypoint.s…"   4 weeks ago   Up 4 weeks
```

  - That server stores all method calls in a SQL database for “future analysis”

```
public class EventSourcingGameDecorator implements Game, GameObserver {

    @Override
    public Status playCard(Player who, Card card) {
      Status status = game.playCard(who, card);
      database.recordPlayCard(gameId, who, card, status);
      return status;
    }
```

  - Exercise: What pattern is in play here ☺?

AARHUS UNIVERSITET

[SideBar]

- So I can query like “when was a card played by whom?”

```
MariaDB [pldb]> select attime, gameid, eventname, player from events where eventname="PlayCard";
+---------------------+----------------+-----------+-----------+
| attime              | gameid         | eventname | player    |
+---------------------+----------------+-----------+-----------+
| 2023-08-14 10:28:15 | hval54tyr73    | PlayCard  | FINDUS    |
| 2023-08-14 10:28:35 | hval54tyr73    | PlayCard  | PEDDERSEN |
| 2023-08-14 10:28:38 | hval54tyr73    | PlayCard  | PEDDERSEN |
| 2023-08-14 10:28:57 | hval54tyr73    | PlayCard  | FINDUS    |
| 2023-08-14 10:29:24 | hval54tyr73    | PlayCard  | FINDUS    |
| 2023-08-14 10:29:53 | hval54tyr73    | PlayCard  | PEDDERSEN |
| 2023-08-14 10:30:13 | hval54tyr73    | PlayCard  | FINDUS    |
| 2023-08-14 10:30:33 | hval54tyr73    | PlayCard  | PEDDERSEN |
| 2023-08-14 10:31:03 | hval54tyr73    | PlayCard  | FINDUS    |
```

```
| 2023-10-11 13:10:55 | hejre54torsk76 | PlayCard  | PEDDERSEN |
| 2023-10-11 13:10:56 | hejre54torsk76 | PlayCard  | PEDDERSEN |
| 2023-10-31 10:10:52 | abe2lodder10   | PlayCard  | FINDUS    |
| 2023-10-31 10:11:11 | abe2lodder10   | PlayCard  | PEDDERSEN |
| 2023-10-31 10:11:16 | abe2lodder10   | PlayCard  | FINDUS    |
| 2023-10-31 10:12:52 | abe2lodder10   | PlayCard  | PEDDERSEN |
| 2023-10-31 10:13:03 | abe2lodder10   | PlayCard  | FINDUS    |
| 2023-10-31 10:14:16 | abe2lodder10   | PlayCard  | PEDDERSEN |
| 2023-10-31 10:14:35 | abe2lodder10   | PlayCard  | FINDUS    |
| 2023-10-31 10:15:00 | abe2lodder10   | PlayCard  | PEDDERSEN |
| 2023-10-31 10:15:03 | abe2lodder10   | PlayCard  | PEDDERSEN |
| 2023-10-31 10:15:36 | abe2lodder10   | PlayCard  | FINDUS    |
| 2023-10-31 10:16:19 | abe2lodder10   | PlayCard  | PEDDERSEN |
| 2023-10-31 10:17:17 | abe2lodder10   | PlayCard  | FINDUS    |
| 2023-11-07 08:35:25 | grib94hund61   | PlayCard  | FINDUS    |
| 2023-11-07 08:35:35 | grib94hund61   | PlayCard  | PEDDERSEN |
+---------------------+----------------+-----------+-----------+
773 rows in set (0.007 sec)
```

AARHUS UNIVERSITET

The Problem

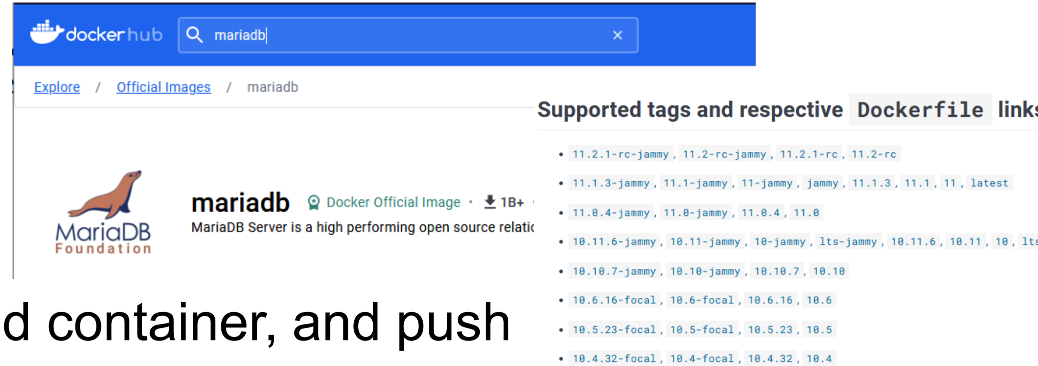
- I need to deploy
  - A MariaDB SQL server container
  - A HotStone server
  - And ensure they talk to each other (DNS, network)
  - Ups – and I need hard disk space for the database itself which survives restarting the system!
    - (By default, a container has its own file system (of course), and thus removing it, will destroy all contents!)

- We need *infrastructure-as-code* to deploy systems

The Containers

- A) MariaDB

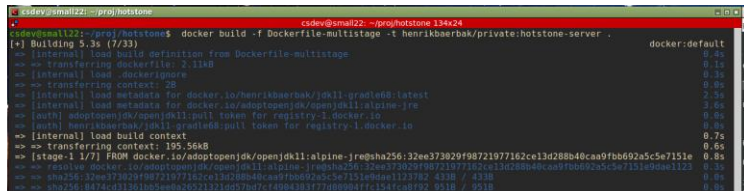
Easy part: Images on Docker Hub



- B) HotStone server
  - I write a Dockerfile, build container, and push

```
FROM henrikbaerbak/jdk11-gradle68 AS builder

LABEL maintainer="HenrikBaerbakChristenen_hbc@cs.au.dk"

# Tell Java tools that source files are UTF8
ENV JAVA_TOOL_OPTIONS -Dfile.encoding=UTF8

# Copy all relevant stuff for compilation
WORKDIR /root/hotstone

# Copy source files (pulling out client is pending)
COPY core/src/ core/src/
COPY domain/src/ domain/src/
COPY ui/src/ ui/src/
COPY uisol/src/ uisol/src/
COPY brokercommon/src/ brokercommon/src/
COPY client/src/ client/src/
COPY solution/src/ solution/src/

# Ensure gradle can produce jar
COPY core/build.gradle core/build.gradle
COPY domain/build.gradle domain/build.gradle
COPY ui/build.gradle ui/build.gradle
COPY uisol/build.gradle uisol/build.gradle
COPY brokercommon/build.gradle brokercommon/build.gradle
COPY client/build.gradle client/build.gradle
COPY solution/build.gradle solution/build.gradle

COPY gradle.properties gradle.properties
COPY settings.gradle settings.gradle

# Create the fat jar containing all code
RUN gradle :solution:jar
```

```
# === Execution container

# --- Moved to a jre with low vulnerability count

FROM adoptopenjdk/openjdk11:alpine-jre

LABEL maintainer="HenrikBaerbakChristenen_hbc@cs.au.dk"

# Install dumb-init
RUN apk add dumb-init

# Create the hotstone folder
RUN mkdir /hotstone
WORKDIR /hotstone

# Run as non-root
RUN addgroup --system javauser && adduser -S -s /bin/false -G javauser javauser

# Copy the fat jar there
COPY --from=builder /root/hotstone/solution/build/libs/hotstoneserver.jar /hotstone/hotstoneserver.jar

# Default port exposed
EXPOSE 5220

# Run as non-root
RUN chown -R javauser:javauser /hotstone
USER javauser

# Define the default command to run:
CMD ["dumb-init", "java","-jar", "hotstoneserver.jar", "null", "chunky"]
```



AARHUS UNIVERSITET

Docker Swarm

- IaC for orchestration

```yaml
version: "3.9"
services:

  # === MariaDB Storage
  mariadbserver:
    image: mariadb:10.5.8

    networks:
      - plnet

    environment:
      MYSQL_ROOT_PASSWORD: secret
      MYSQL_DATABASE: pldb
      MYSQL_USER: pluser
      MYSQL_PASSWORD: secret

    volumes:
      - data-volume:/var/lib/mysql

    deploy:
      replicas: 1
```

```yaml
  # === HotStone server with chunky protocol and MariaDB event storage
  hotstoneserver:
    image: henrikbaerbak/private:hotstone-server

    command: ["dumb-init", "java", "-jar",
      "hotstoneserver.jar", "mariadbserver", "chunky"]

    depends_on:
      - mariadbserver

    ports:
      - "5220:5220"

    networks:
      - plnet

    deploy:
      replicas: 1

networks:
  plnet:
    external: true

volumes:
  data-volume:
```

AARHUS UNIVERSITET

Internal Network and DNS

- Network ‘plnet’ and Named servers

```yaml
version: "3.9"
services:

  # === MariaDB Storage
  mariadbserver:
    image: mariadb:10.5.8

    networks:
      - plnet

    environment:
      MYSQL_ROOT_PASSWORD: secret
      MYSQL_DATABASE: pldb
      MYSQL_USER: pluser
      MYSQL_PASSWORD: secret

    volumes:
      - data-volume:/var/lib/mysql

    deploy:
      replicas: 1
```

```yaml
  # === HotStone server with chunky protocol and MariaDB event storage
  hotstoneserver:
    image: henrikbaerbak/private:hotstone-server

    command: ["dumb-init", "java", "-jar",
      "hotstoneserver.jar", "mariadbserver", "chunky"]

    depends_on:
      - mariadbserver

    ports:
      - "5220:5220"

    networks:
      - plnet

    deploy:
      replicas: 1

networks:
  plnet:
    external: true

volumes:
  data-volume:
```

AARHUS UNIVERSITET

Volume

- Volumes are *stored on the host machine*

```yaml
version: "3.9"
services:

  # === MariaDB Storage
  mariadbserver:
    image: mariadb:10.5.8

    networks:
      - plnet

    environment:
      MYSQL_ROOT_PASSWORD: secret
      MYSQL_DATABASE: pldb
      MYSQL_USER: pluser
      MYSQL_PASSWORD: secret

    volumes:
      - data-volume:/var/lib/mysql

    deploy:
      replicas: 1
```

```yaml
  # === HotStone server with chunky protocol and MariaDB event storage
  hotstoneserver:
    image: henrikbaerbak/private:hotstone-server

    command: ["dumb-init", "java", "-jar",
      "hotstoneserver.jar", "mariadbserver", "chunky"]

    depends_on:
      - mariadbserver

    ports:
      - "5220:5220"

    networks:
      - plnet

    deploy:
      replicas: 1

networks:
  plnet:
    external: true

volumes:
  data-volume:
```

Time to Deploy (Testing)

- On my Staging machine, just ‘docker stack deploy’

Time to Deploy (Production)

- On my cloud machine, the *only contents* is the IaC

```
root@baerbak-e23: ~/proj/hotstone
root@baerbak-e23:~/proj/hotstone# ll
total 16
drwxr-xr-x 2 root root 4096 Aug 11 10:16 ./
drwxr-xr-x 5 root root 4096 Oct 16 12:02 ../
-rwxr-xr-x 1 root root  410 Aug 11 10:16 run-hotstone-server.sh*
-rw-r--r-- 1 root root 1326 Aug 11 10:11 swarm-db.yml
root@baerbak-e23:~/proj/hotstone#
```

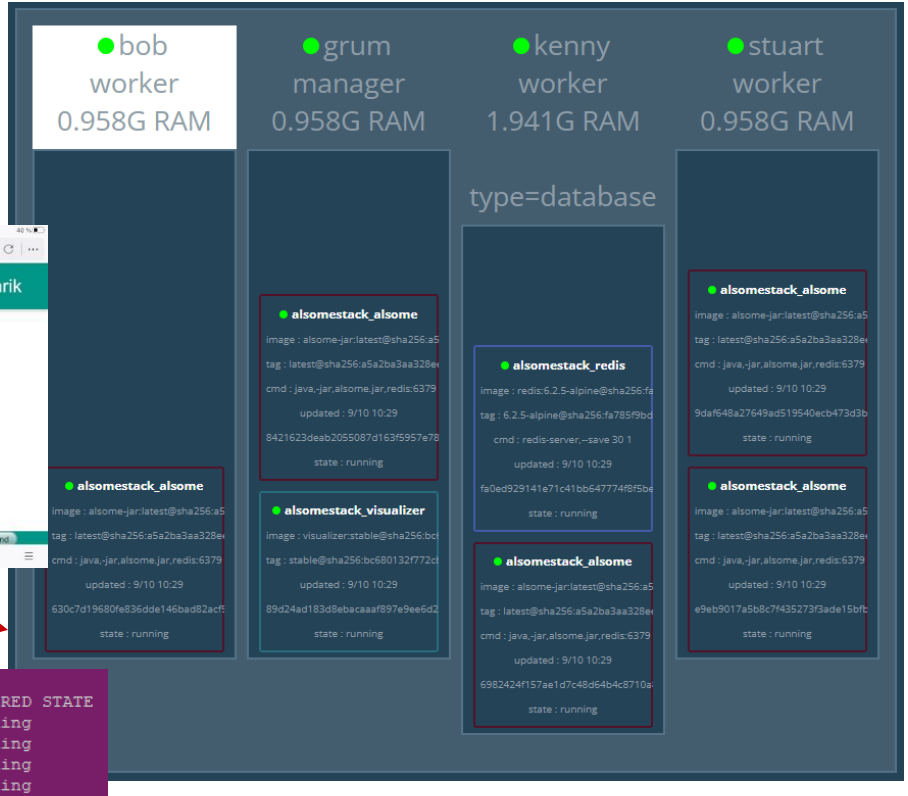
  - The 'run…' script is just the 'docker stack deploy…' command

```
root@baerbak-e23:~/proj/hotstone# docker stack ps hotstack
ID             NAME                          IMAGE                                  NODE          DESIRED STATE
m9wtnvhstkzf   hotstack_hotstoneserver.1     henrikbaerbak/private:hotstone-server  baerbak-e23   Running
w4bt59p7in5z    \_ hotstack_hotstoneserver.1 henrikbaerbak/private:hotstone-server  baerbak-e23   Shutdown
thmdjegrsrw3    \_ hotstack_hotstoneserver.1 henrikbaerbak/private:hotstone-server  baerbak-e23   Shutdown
jokvc03q2mns   hotstack_mariadbserver.1      mariadb:10.5.8                         baerbak-e23   Running
9i0raqbfmmlu    \_ hotstack_mariadbserver.1  mariadb:10.5.8                         baerbak-e23   Shutdown
96yiqpnva6w5    \_ hotstack_mariadbserver.1  mariadb:10.5.8                         baerbak-e23   Shutdown
root@baerbak-e23:~/proj/hotstone# docker stack ps help
```

AARHUS UNIVERSITET

Scaling Out

- Swarm *is* a swarm
  - Four machines in my swarm
    - Grum, bob, stuart, kenny
- Automatically deploys…
  - Redis database
  - Five ‘alsome’ servers
  - One visualizer service




```
root@grum:~/proj# docker stack ps alsomestack
ID             NAME                      IMAGE                                NODE      DESIRED STATE
r964uq1vqlm2   alsomestack_alsome.1      henrikbaerbak/alsome-jar:latest      stuart    Running
nqa55ygildan   alsomestack_alsome.2      henrikbaerbak/alsome-jar:latest      stuart    Running
8ain3gizlwte   alsomestack_alsome.3      henrikbaerbak/alsome-jar:latest      bob       Running
eeq32273q6kj   alsomestack_alsome.4      henrikbaerbak/alsome-jar:latest      kenny     Running
o7ofq83dp7zj   alsomestack_alsome.5      henrikbaerbak/alsome-jar:latest      grum      Running
pp9n4sse2imy   alsomestack_redis.1       redis:6.2.5-alpine                   kenny     Running
fu9qn925tbgf   alsomestack_visualizer.1  dockersamples/visualizer:stable      grum      Running
```

Summary

- This is not SWEA curriculum
  - No exam questions, no questioning at the exam
  - Beyond our ‘software architecture in the small’ focus of SWEA
- But…
  - It is an important conceptual framework and tool stack to master for large scale, industrial, software development and operations.

- Swarm is not widely used (Docker containers are!)
  - The big player is *Kubernetes*
  - From a conceptual point of view, they are the ‘same thing’…
    - An orchestration tool…