



AARHUS UNIVERSITET

# Software Engineering and Architecture

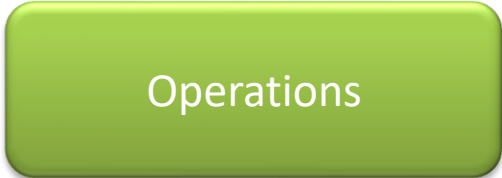
Containers - Docker

*Lightweight Virtual Machines*



# From Dev to Ops

- We have this wonderful client-server system developed
  - Lots of unit and systems tests, highly flexible, highly reliable
- *But we only have it on our local laptops*
  - Git clone, gradle something
- **We need to make it run on the internet**
  - **Deployment**      *idriftsættelse*
- Requires
  - A) A computer with DNS name on the ‘big internet’
  - B) Get our executable system on to that machine





# The Old Way

- ... a manual procedure
- *Build 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’*



# 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

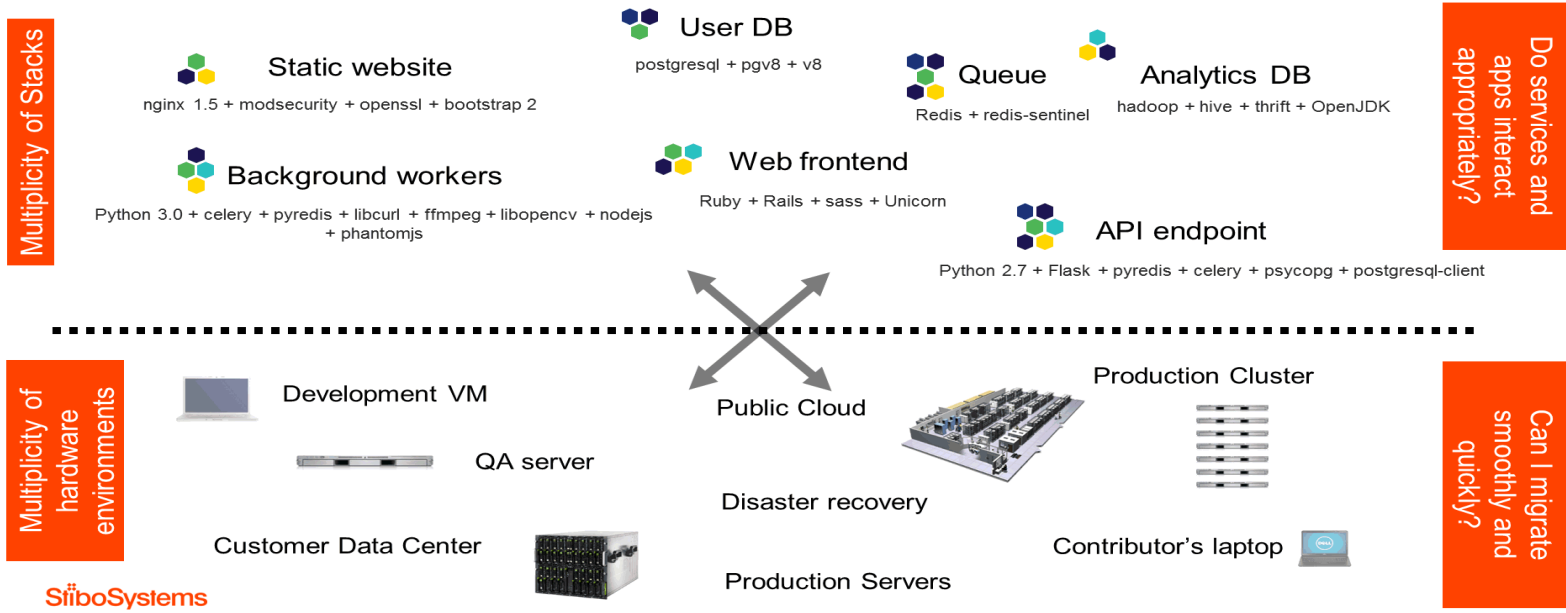




# The *Moving Code* Problem

AARHUS UNIVERSITET

- Crossing boundaries, that is, *moving code*

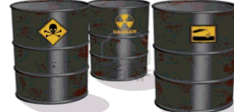


Source: Torben Haag, StiboSystems



# Was Solved in 1960'ies

AARHUS UNIVERSITET

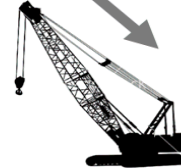


A standard container that is loaded with virtually any goods, and stays sealed until it reaches final delivery.



...in between, can be loaded and unloaded, stacked, transported efficiently over long distances, and transferred from one mode of transport to another

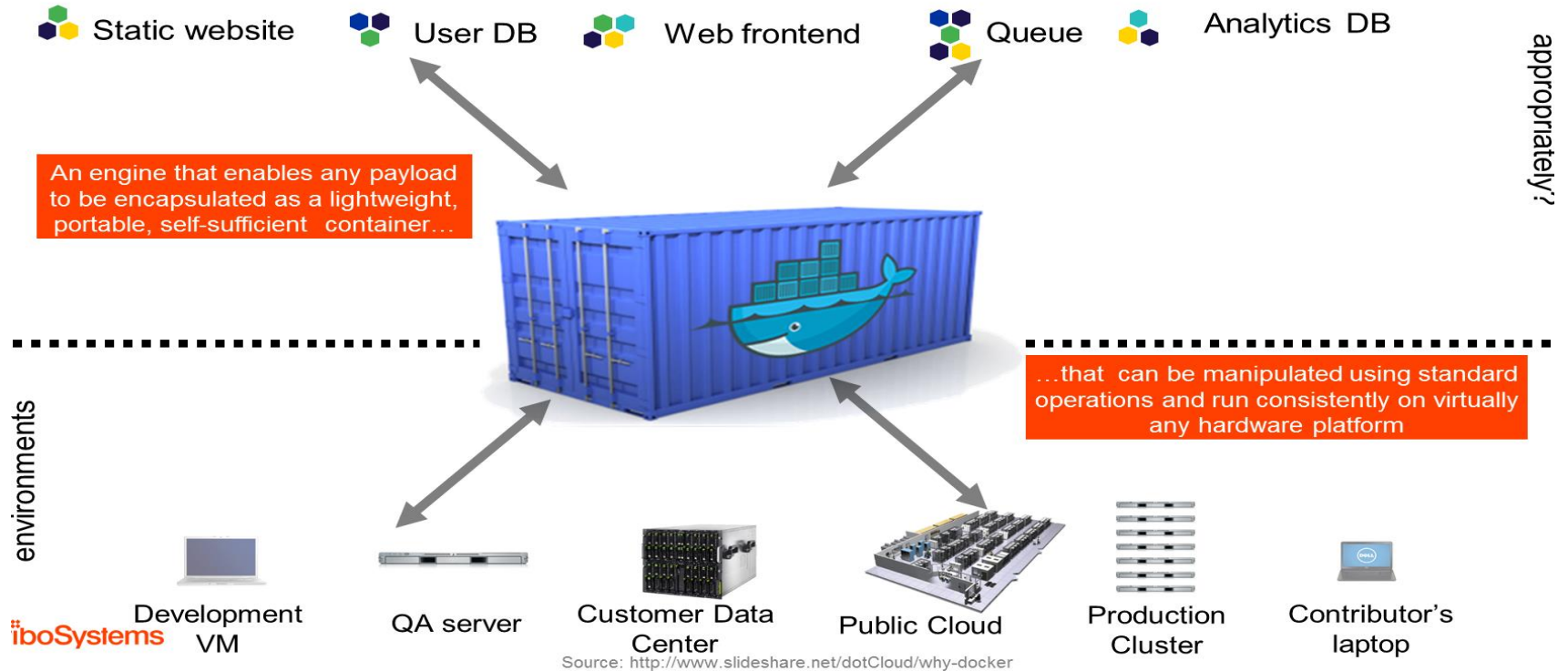
Transportation





# Docker = Container

## Docker is a shipping container system for code

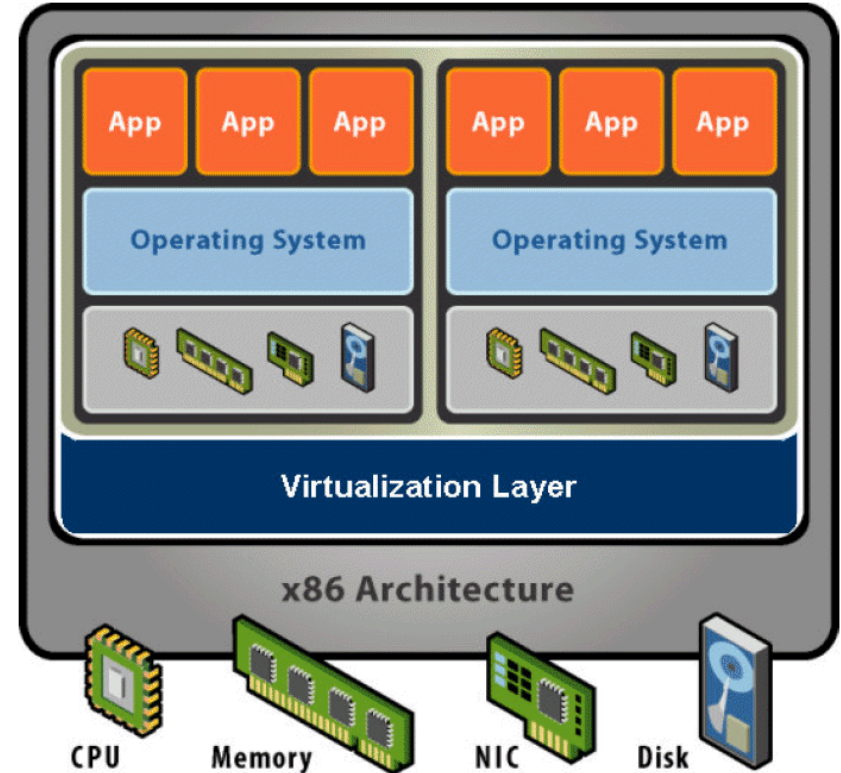






# Not Just Programs - Environments

- 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!***



# 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 – I needs to 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

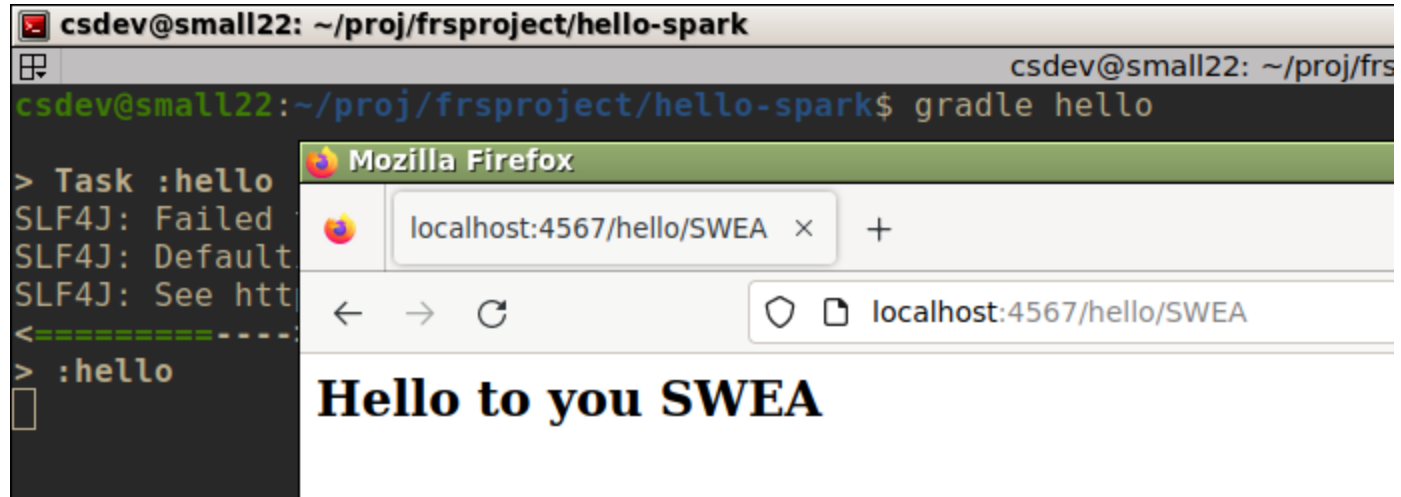
# Building a Container

*Infrastructure-as-code*

*“IaC”*

# Example: Hello-Spark

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



The screenshot shows a terminal window and a Mozilla Firefox browser window. The terminal window displays the following output:

```
csdev@small22: ~/proj/frsproject/hello-spark
csdev@small22:~/proj/frsproject/hello-spark$ gradle hello
> Task :hello
SLF4J: Failed
SLF4J: Default
SLF4J: See htt
<=====
> :hello
█
```

The browser window shows the URL `localhost:4567/hello/SWEA` and the response **Hello to you SWEA**.



# My Container...

- I want to build a docker container which contains
  - Linux Operating System
  - Java 17 runtime system
  - Gradle v8.3
  - And my Hello-spark code
    - build.gradle
    - src/ folder
- Can run on *any machine in the world* if it has docker engine installed *and nothing else required!*



# 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 Spark Demo docker file

# To build: docker build -t henrikbaerbak/private:hello-spark .
# To run: docker run -d -p 4567:4567 henrikbaerbak/private:hello-spark

# Base image - java 17 + gradle 8.3
FROM gradle:8.3-jdk17

# 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-spark will use
EXPOSE 4567

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

- Parts:

- Base

- Copy

- Execution

```
# Hello Spark Demo docker file
# To build: docker build -t henrikbaerbak/private:hello-spark .
# To run: docker run -d -p 4567:4567 henrikbaerbak/private:hello-spark
# Base image - java 17 + gradle 8.3
FROM gradle:8.3-jdk17
# 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-spark will use
EXPOSE 4567
# Define the default command to run
CMD ["gradle","hello"]
```





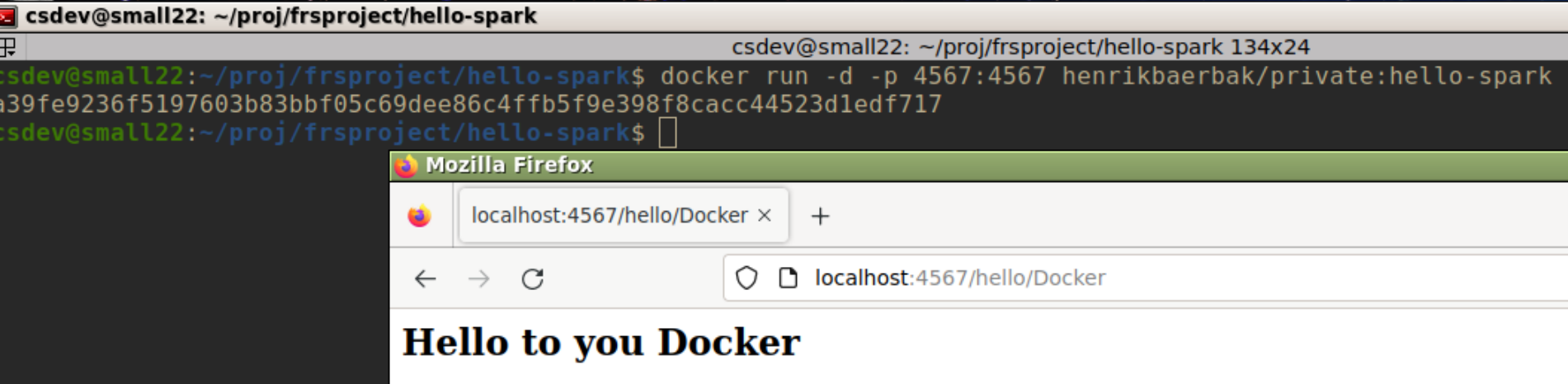
- Image name: henrikbaerbak/private:hello-spark

```
csdev@small22:~/proj/frsproject/hello-spark$ docker build -t henrikbaerbak/private:hello-spark .
[+] Building 2.3s (10/10) FINISHED                                docker:default
=> [internal] load build definition from Dockerfile              0.1s
=> => transferring dockerfile: 651B                             0.0s
=> [internal] load .dockerignore                                0.1s
=> => transferring context: 2B                                    0.0s
=> [internal] load metadata for docker.io/library/gradle:8.3-jdk17 1.9s
=> [auth] library/gradle:pull token for registry-1.docker.io    0.0s
=> CACHED [1/4] FROM docker.io/library/gradle:8.3-jdk17@sha256:461018d754af5e85ca08b8f38e5f583d65ebadf83be8ad01f3bff03740ccecf4 0.0s
=> [internal] load build context                                0.0s
=> => transferring context: 945B                                  0.0s
=> [2/4] WORKDIR /root/hello                                   0.1s
=> [3/4] COPY src/ src/                                        0.0s
=> [4/4] COPY build.gradle build.gradle                       0.0s
=> exporting to image                                          0.1s
=> => exporting layers                                           0.1s
=> => writing image sha256:596d8ffb37b6d946b768fc48cd3949468c28f3cc2bf4cb5c6e363f5a0485d97b 0.0s
=> => naming to docker.io/henrikbaerbak/private:hello-spark    0.0s
csdev@small22:~/proj/frsproject/hello-spark$
```



# 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

- 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 !**

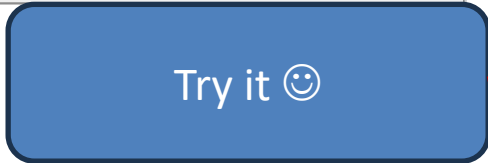
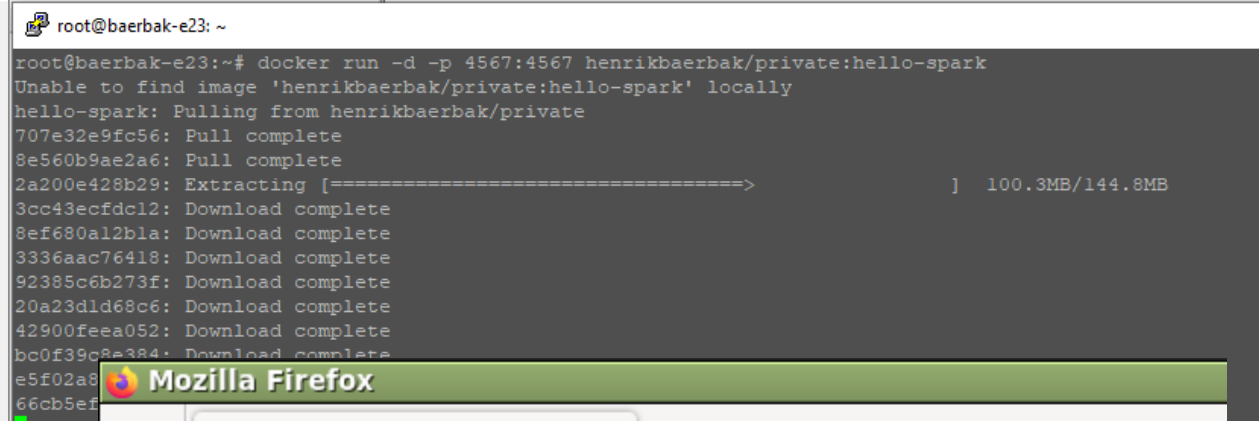
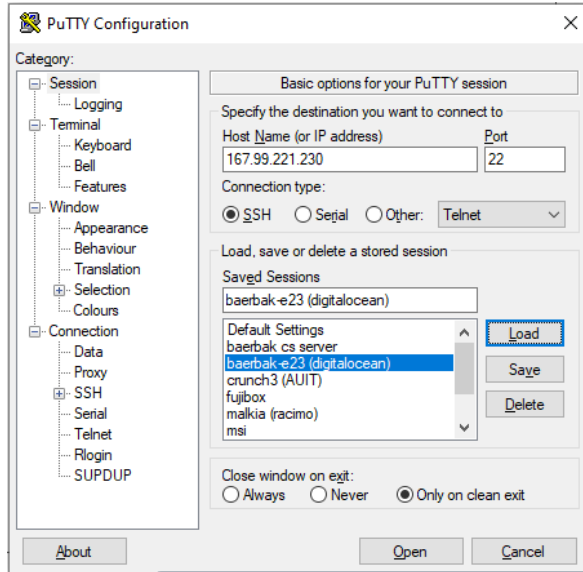
```
csdev@small22:~/proj/frsproject/hello-spark$ docker push henrikbaerbak/private:hello-spark
The push refers to repository [docker.io/henrikbaerbak/private]
b31fdb6dd8f5: Pushed
6b0cfe77d40c: Pushed
f8c849cc2d38: Pushed
ef3a0dec6a89: Layer already exists
605477091eb2: Layer already exists
07099f814ad2: 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
```

# To Amsterdam

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



- Log into my machine in Amsterdam
  - Using my Windows Putty secure shell; and 'run' container



**Hello to you Amsterdam**



# Bottomline

- It takes about *1 minute!*
- My Amsterdam 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!*



- My 'cave service' in GoLang

# Examples

```
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"]
```





# Examples

AARHUS UNIVERSITET

- 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" ]
```



- My 'cave service' in Scala

# Examples

```
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"]
```



# Examples

AARHUS UNIVERSITET

- 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"]
```

# And Much More


- My DigitalOcean machine runs a lot of stuff for me...

```
root@baerbak-e23:~# docker ps
CONTAINER ID   IMAGE                                COMMAND                                  CREATED        STATUS
92136598ea18   henrikbaerbak/private:hello-spark   "/__cacert_entrypoin..."             6 minutes ago Up 6 minutes
00c8cb4ece70   henrikbaerbak/private:subscription2310 "/app/entry-point-ja..."             3 weeks ago   Up 3 weeks
e7238f69d011   mongo:3.6.22-xenial                "docker-entrypoint.s..."             3 weeks ago   Up 3 weeks
bada9b84c323   henrikbaerbak/quote:msdo_2_3        "java -jar /root/quo..."             3 weeks ago   Up 3 weeks
78dafd0b8572   henrikbaerbak/private:e23-cave-jar  "/__cacert_entrypoin..."             4 weeks ago   Up 4 weeks
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
```

- The hot stone game server with its associated MariaDB is one of them...
  - A 'quote' service, used in my MicroService fagpakke
  - A 'subscription'/single sign-on service, in same fagpakke



# And the World is Big

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

```
csdev@small122:~$ 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: sha256:9f107d34c284d9be2527341d6d511cd818a569d83b4c52d1aef4ca7a5c48/d/
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

# 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 😊?

- 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
2023-08-14 10:31:19	hval54tyr73	PlayCard	PEDDERSEN

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)
```





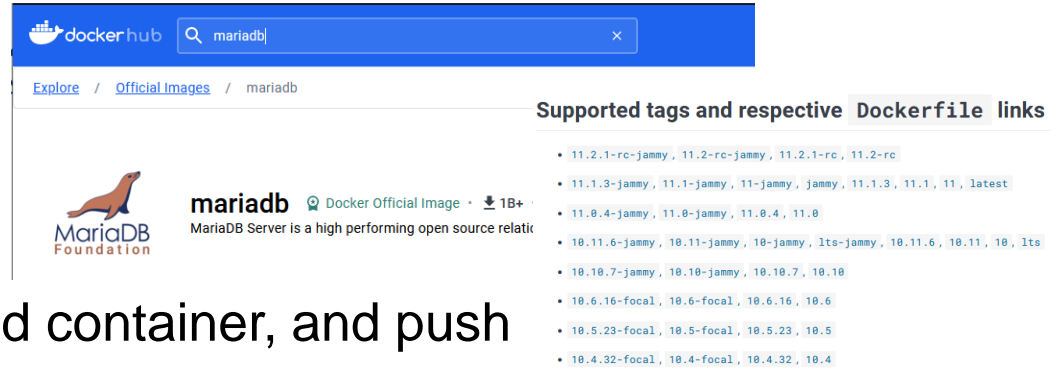
# The Problem

- 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="HenrikBaerbakChristensen_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 uisrc/ uisrc/
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 uisrc/build.gradle uisrc/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="HenrikBaerbakChristensen_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", "chucky"]
```

```
cddev@small22:~/prg/hotstone$ docker build -f Dockerfile-multistage -t henrikbaerbak/private:hotstone-server .
[+] Building 5.3s (7/23)
=> [internal] load build definition from Dockerfile-multistage
=> [internal] load dockerfile
=> [internal] load metadata for docker.io/henrikbaerbak/jdk11-gradle68:latest
=> [internal] load metadata for docker.io/adoptopenjdk/openjdk11:alpine-jre
=> [internal] load metadata for docker.io/adoptopenjdk/openjdk11:pull token for registry-1.docker.io
=> [internal] load context: 2B
=> [internal] load metadata for docker.io/henrikbaerbak/jdk11-gradle68:latest
=> [internal] load metadata for docker.io/adoptopenjdk/openjdk11:alpine-jre
=> [internal] load metadata for docker.io/adoptopenjdk/openjdk11:pull token for registry-1.docker.io
=> [internal] load build context
=> [stage-1 1/7] FROM docker.io/adoptopenjdk/openjdk11:alpine-jre@sha256:32ee373029f9872107162ce13d288b4cae9fb692a5c5e7151e
=> [internal] load metadata for docker.io/adoptopenjdk/openjdk11:pull token for registry-1.docker.io
```



- IaC for orchestration

```
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
```

```
# === 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:
```



# Internal Network and DNS

AARHUS UNIVERSITET

- Network 'plnet' and Named servers

```
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
```

```
# === 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:
```

- Volumes are stored on the host machine

```
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
```

```
# === 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

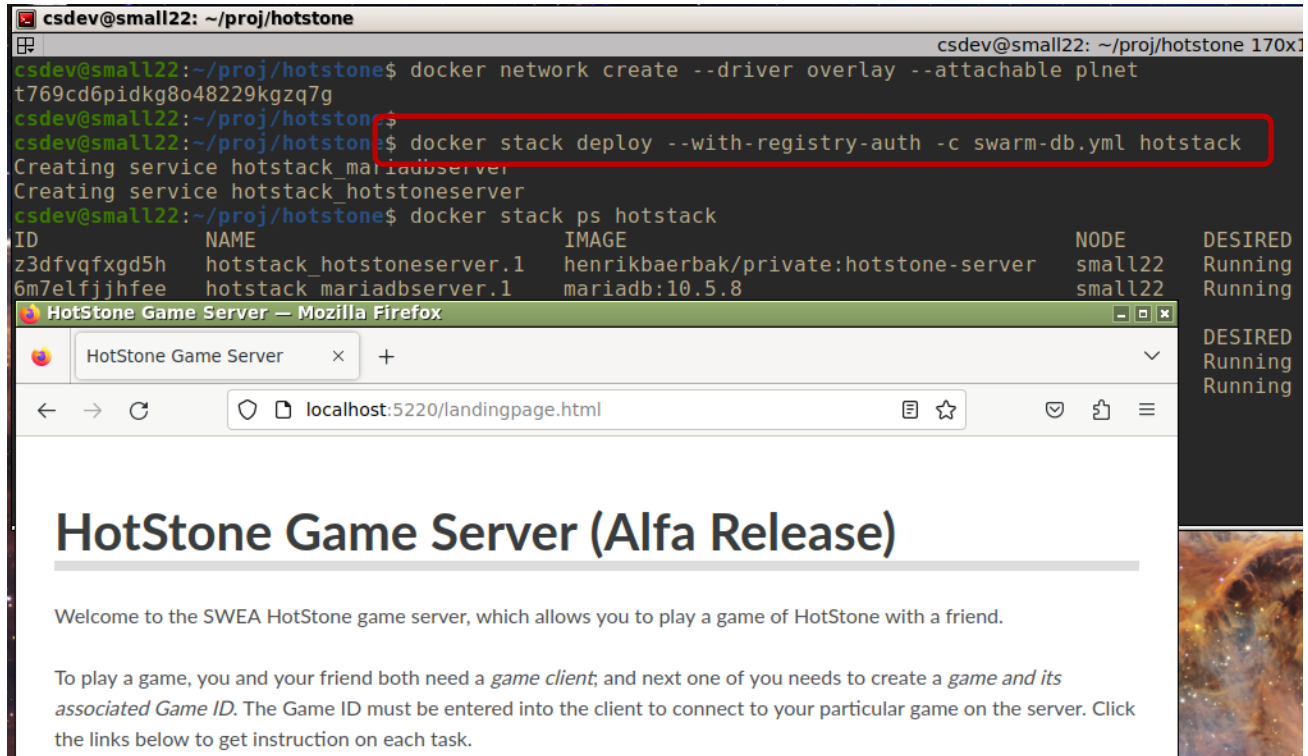
  deploy:
    replicas: 1

networks:
  plnet:
    external: true

volumes:
  data-volume:
```

# Time to Deploy (Testing)

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



```
csdev@small22: ~/proj/hotstone
csdev@small22:~/proj/hotstone$ docker network create --driver overlay --attachable plnet
t769cd6pidkg8o48229kgzq7g
csdev@small22:~/proj/hotstone$ docker stack deploy --with-registry-auth -c swarm-db.yml hotstack
Creating service hotstack_mariadbserver
Creating service hotstack_hotstoneserver
csdev@small22:~/proj/hotstone$ docker stack ps hotstack
```

ID	NAME	IMAGE	NODE	DESIRED
z3dfvqfxgd5h	hotstack_hotstoneserver.1	henrikbaerbak/private:hotstone-server	small22	Running
6m7elfjjhfee	hotstack_mariadbserver.1	mariadb:10.5.8	small22	Running

HotStone Game Server — Mozilla Firefox

HotStone Game Server x +

localhost:5220/landingpage.html

## HotStone Game Server (Alfa Release)

Welcome to the SWEA HotStone game server, which allows you to play a game of HotStone with a friend.

To play a game, you and your friend both need a *game client*; and next one of you needs to create a *game* and its *associated Game ID*. The Game ID must be entered into the client to connect to your particular game on the server. Click the links below to get instruction on each task.

# Time to Deploy (Production)

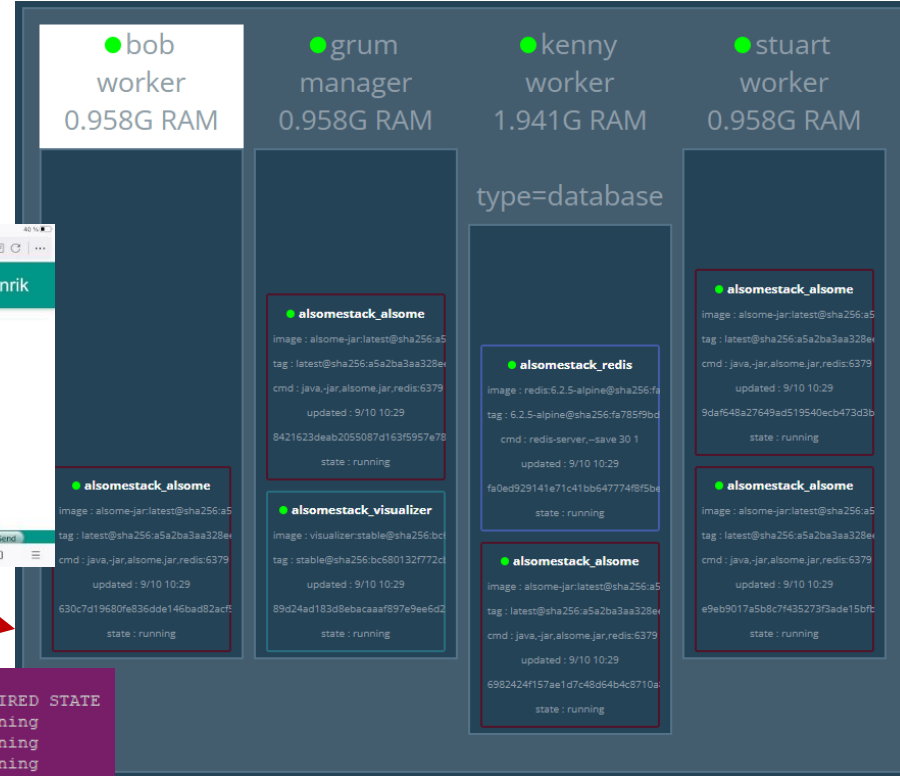
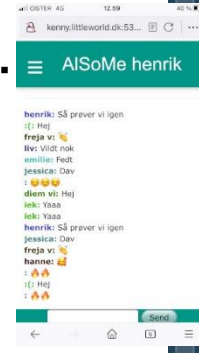
- On my Amsterdam machine, the *only contents* is the IaC

```
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
m9wtvnhstkzf 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
96yigpnva6w5  \_ hotstack_mariadbserver.1      mariadb:10.5.8                        baerbak-e23 Shutdown
root@baerbak-e23:~/proj/hotstone# docker stack ps help
```

- 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
r964uqlvqlm2    alsomestack_alsome.1  henrikbaerbak/alsome-jar:latest  stuart  Running
nga55ygildan    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
  
```





- 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...