AARHUS UNIVERSITET

Microservices and DevOps

DevOps and Container Technology

Docker Swarm

Henrik Bærbak Christensen

AARHUS UNIVERSITET

Docker Swarm

- *”A swarm consists of multiple Docker hosts which run in swarm mode and act as managers (to manage membership and delegation) and workers (which run swarm services).”*
  - Swarm manager:
    - You can execute docker commands
  - Worker:
    - Slaves that can only accept services from the manager
- **Node**: Physical or virtual machine
  - In production, a physical machine makes most sense, but in the cloud, well…

AARHUS UNIVERSITET

Service and Task

- Service = the definition of a task to execute
  - That is, a running container
  - The ‘services:’ section of the docker-compose file
- Replicated service
  - A service can be replicated, that is you define how many instances of it to run
- Task
  - ‘container + commands to run it’
  - assigned to a node; will never leave it

AARHUS UNIVERSITET

Cluster Creation

- On my ESXi hypervisor, I created
  - Headless Ubuntu 18.04 LTS machines
    - Malkia00 and three Nyuki'es (queen and bees ☺)
  - 'docker swarm init' Creates a join token
  - 3 x 'docker swarm join' Using the join token

```
hbc@malkia00:~$ docker swarm init
Swarm initialized: current node (ugwyylb82ayckaq57qgasixvu) is now a manager.

To add a worker to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-2fcaibzb0ycc3evmn49hs6cf5tv8t0e419eackqpnfh5bqjqvj-cqklzt3hpcw5x0139r7tssm8b 10.17.98.60:2377

To add
```

```
hbc@malkia00:~$ docker node ls
ID                            HOSTNAME            STATUS              AVAILABILITY        MANAGER STATUS
ugwyylb82ayckaq57qgasixvu *   malkia00            Ready               Active              Leader
pppyhwee5pp244ncz3elowbpz     nyuki01             Ready               Active
rklajfs25dygs7y9gon4znxyi     nyuki02             Ready               Active
irke0w7ht083kqgzieeevaas7     nyuki03             Ready               Active
hbc@malkia00:~$
```

AARHUS UNIVERSITET

Small Cluster Creation

- In our context, Swarm is very approachable as a swarm may consist of *a single node*.
- Just ‘swarm init’ in Mxx, and you can test run all your work, just as if it was on a multi-node swarm

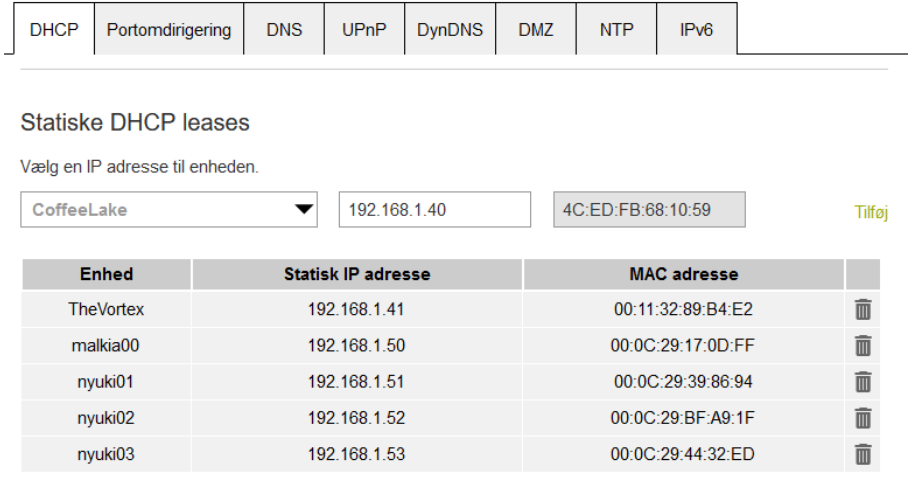
```
hbc@malkia00:~$ docker swarm init
Swarm initialized: current node (ugwyylb82ayckaq57qgasixvu) is now a manager.

To add a worker to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-2fcaibzb0ycc3evmn49hs6cf5tv8t0e419eackqpnfh5bqjqvj-cqklzt3hpcw5x0139r7tssm8b 10.17.98.60:2377

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions.
```

- (Of course, RAM and CPU are limiting factors…)

AARHUS UNIVERSITET

SideNote

- Swarm nodes talk through static IP addresses
  - I.e. they need these to be fixed
- On my home router in my Corona Bubble lab I found that I could assign static DHCP leases to swarm nodes



Stack

- You can manipulate services directly using docker engine commands
  - That is – *manual interaction* ☹
- *Infrastructure-as-code*
  - Automate through scripting

> • *Coding infrastructure logic*: The programming of logic for the deployment of services. Traditionally handled by manual procedures (installing, configuring, and linking services), but in face of large-scale deployments, this too must be coded. Example: Developing scripts that start the application server, inventory service and associated database, initialize them, and connect them correctly—i.e. create a *staging environment*.

- **Stack:** Group of interrelated services that share dependencies, and are *orchestrated and scaled together*.

- **Compose-file**: IaC for defining how containers behave in production. Writting in a specific YAML format.

Or is it a 'stack file'? Shooting a moving target ☹

AARHUS UNIVERSITET

Anatomy of Compose-file

- A typical Compose-file
  - YAML file
  - Hierarchy by **indentation**
    - **(use spaces!!!)**
    - **(be aware of whitespace)**
  - State
    - *Services to deploy*
    - *Networks to create*
    - *(Volumes to create)*
    - *…*

```yaml
version: "3"
services:
  web:
    # replace username/repo:tag with your name and image details
    image: username/repo:tag
    deploy:
      replicas: 5
      restart_policy:
        condition: on-failure
      resources:
        limits:
          cpus: "0.1"
          memory: 50M
    ports:
      - "80:80"
    networks:
      - webnet
  visualizer:
    image: dockersamples/visualizer:stable
    ports:
      - "8080:8080"
    volumes:
      - "/var/run/docker.sock:/var/run/docker.sock"
    deploy:
      placement:
        constraints: [node.role == manager]
    networks:
      - webnet
networks:
  webnet:
```

AARHUS UNIVERSITET

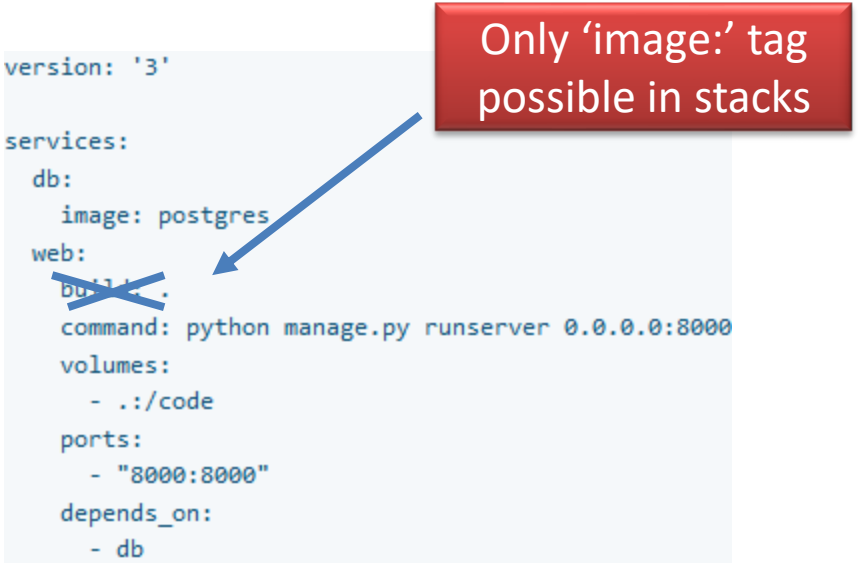
Anatomy of Compose-file

- Service definition
  - Docker image that holds task
    - Only docker hub images (or in local image cache)
  - Properties
    - Ports to expose
    - Network to use
    - (Volume to use)
    - Etc.

```yaml
version: "3"
services:
  web:
    # replace username/repo:tag with your name and image details
    image: username/repo:tag
    deploy:
      replicas: 5
      restart_policy:
        condition: on-failure
      resources:
        limits:
          cpus: "0.1"
          memory: 50M
    ports:
      - "80:80"
    networks:
      - webnet
  visualizer:
    image: dockersamples/visualizer:stable
    ports:
      - "8080:8080"
    volumes:
      - "/var/run/docker.sock:/var/run/docker.sock"
    deploy:
      placement:
        constraints: [node.role == manager]
    networks:
      - webnet
networks:
  webnet:
```

Anatomy of Compose-file

- Instructing the service/task
  - Command:
    - Overrules CMD in image
  - Depends_on:
    - States service dependencies and order of service starts



- *But swarm does not respect ‘is-up’ by services even if there is a depends_on*
  - If the ‘db’ is slow to start, the client service may try to connect before it will accept connections ☹

AARHUS UNIVERSITET

Orchestration

- Services need to communicate with each other...
- ***Each container on the network is assigned the 'service name' as hostname***
  - A swarm has its own DNS
  - There will be a node named 'db' and one named 'web' on this network!

```
version: '3'

services:
  db:
    image: postgres
  web:
    build: .
    command: python manage.py runserver 0.0.0.0:8000
    volumes:
      - .:/code
    ports:
      - "8000:8000"
    depends_on:
      - db
```

- So, you have to configure your 'manage.py' to connect to 'db:5432' (postgres port).

AARHUS UNIVERSITET

Anatomy of Compose-file

- Format 3.x of compose-file allows *deployment to be specified!*

Here:
5 instances,
limit use of CPU and mem
(max 10% cpu/50MB),
and set restart policy

```yaml
version: "3"
services:
  web:
    # replace username/repo:tag with your name and image details
    image: username/repo:tag
    deploy:
      replicas: 5
      restart_policy:
        condition: on-failure
      resources:
        limits:
          cpus: "0.1"
          memory: 50M
    ports:
      - "80:80"
    networks:
      - webnet
  visualizer:
    image: dockersamples/visualizer:stable
    ports:
      - "8080:8080"
    volumes:
      - "/var/run/docker.sock:/var/run/docker.sock"
    deploy:
      placement:
        constraints: [node.role == manager]
    networks:
      - webnet
networks:
  webnet:
```

AARHUS UNIVERSITET

Five Instances???

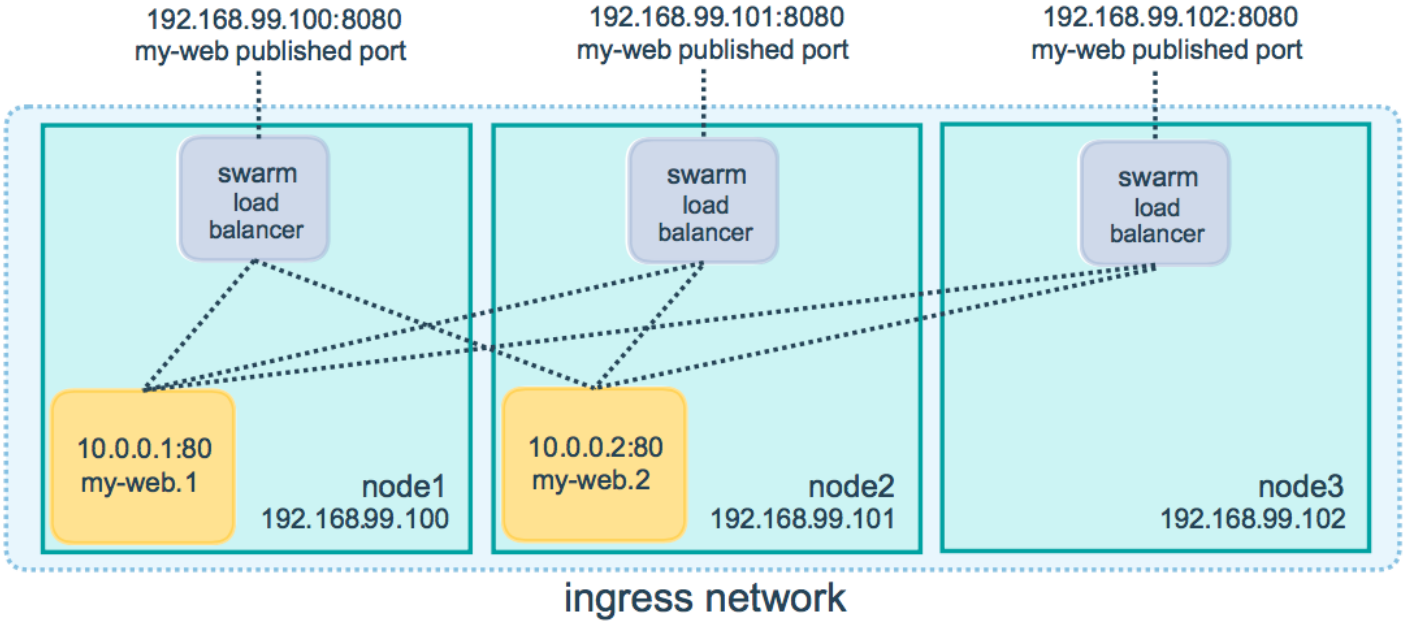
- The default network of swarm is a *ingress routing mesh:*

```
    networks:
      - webnet
networks:
  webnet:
```

AARHUS UNIVERSITET

“Load Balancing”

- *Ingress routing mesh:* Each node in swarm accepts connections on published ports for *any* service in the swarm



AARHUS UNIVERSITET

Example

- My Telemedical application is on my malkia swarm:

```
csdev@m1:~/proj/broker/telemed$ gradle homeHttp -Pid=pid01 -Psys=127 -Pdia=77 -Phost=malkia.st.lab.au.dk
```

- But can upload measurements on any node in swarm

```
csdev@m1:~/proj/broker/telemed$ gradle homeHttp -Pid=pid01 -Psys=127 -Pdia=79 -Phost=nyuki01.st.lab.au.dk
```

SideBar

- We will talk more about horizontal scaling and load balancing later…

Routing

- ”The swarm makes the service accessible at the target port **on every swarm node**. If an external host connects to that port on any swarm node, the routing mesh routes it to a task. The external host does not need to know the IP addresses or internally-used ports of the service tasks to interact with the service. When a user or process connects to a service, *any* worker node running a service task may respond.”

- Note: It states ‘swarm load balancer’ on the previous figure, but – it is not *round-robin* ☺. It “*routes incoming requests to published ports on available nodes.”*
  - Which may be hitting the same node again and again…
    - If it is not working too hard…

AARHUS UNIVERSITET

Peer Bech's Thesis

- Peer's Master Thesis (2020)
  - 10 AWS nodes in swarm, having 40 REST containers, and hitting the cluster with JMeter generated traffic…

- Looks like the swarm does a pretty good job at distributing load…

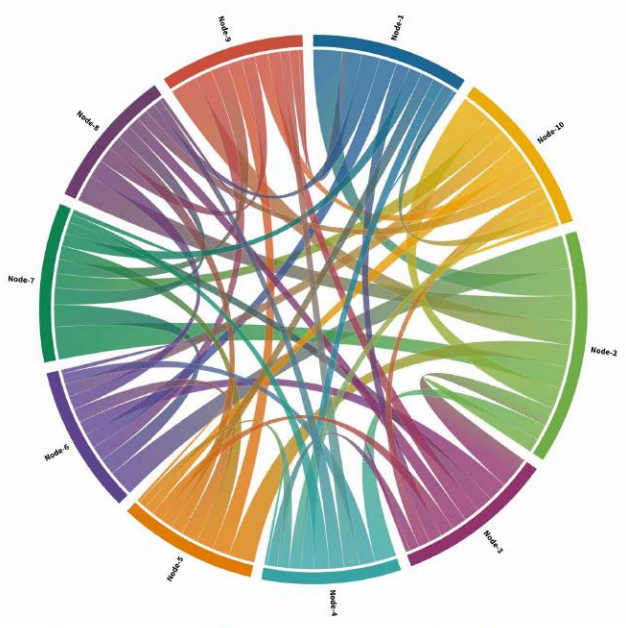


Figure 19: Distribution of REST requests between nodes for the Swarm setup

AARHUS UNIVERSITET

Stack Manipulation

- Docker commands for manipulating a **stack:**
  - ‘docker stack deploy –c (composefile) (stackname)’
  - ‘docker stack ls’ List running stacks
  - ‘docker stack ps (n) List tasks in stack ‘n’
  - ‘docker stack services (n)’ List services in ‘n’
  - ‘docker stack rm (n)’ Remove stack ‘n’
- Only works on a manager node

Tech note: deploy with ‘--with-registry-auth’ if your image is only available in a private repository...

AARHUS UNIVERSITET

Service Manipulation

- ‘stack deploy’ starts *services* in the swarm, so
  - ‘docker service ls’ — Shows running services
  - ‘docker service ps (name)’ — Shows ‘ps’ status of service
  - ‘docker service logs (name)’ — Shows logs for *all* replicas of given name
    - (provide the –f to ‘logs’ to follow/tail the log output)

AARHUS UNIVERSITET

Example

- I made a *full SkyCave system* (Stack ‘dilab’)
  - *Subscription service with associated MongoDB, 2x SkyCave daemons with shared Memcached db and a Docker visualizer*

hbc@malkia00: ~/proj/digiinno19/solution

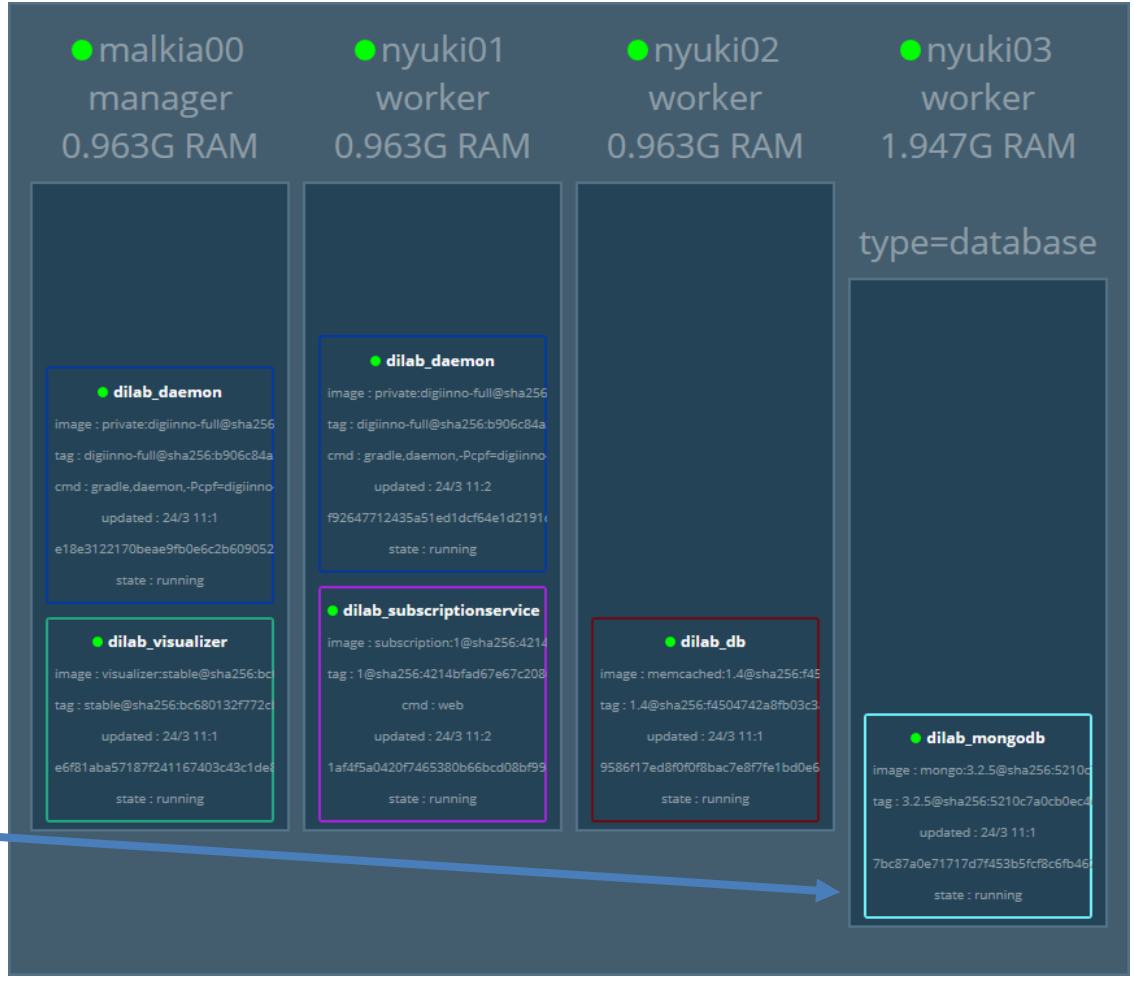
```
hbc@malkia00:~/proj/digiinno19/solution$ docker stack deploy --with-registry-auth -c docker-swarm-full-system.yml dilab
Creating network dilab_frontend
Creating network dilab_backend
Creating service dilab_visualizer
Creating service dilab_subscriptionservice
Creating service dilab_mongodb
Creating service dilab_db
Creating service dilab_daemon
hbc@malkia00:~/proj/digiinno19/solution$ docker stack ps dilab
ID                  NAME                        IMAGE                                NODE                DESIRED STATE       CURRENT STATE
y5zkfm7qvqfl        dilab_daemon.1              henrikbaerbak/private:digiinno-full  nyuki01             Running             Running 29 seconds ago
3ws6afjeelcl        dilab_db.1                  memcached:1.4                        nyuki02             Running             Running about a minute ago
n8fxznxwkdck        dilab_mongodb.1             mongo:3.2.5                          nyuki03             Running             Running about a minute ago
u5nb547tfejb        dilab_subscriptionservice.1 henrikbaerbak/subscription:1          nyuki01             Running             Running about a minute ago
2bvic5bpoogf        dilab_visualizer.1          dockersamples/visualizer:stable      malkia00            Running             Running about a minute ago
nd7jbmpy27ul        dilab_daemon.2              henrikbaerbak/private:digiinno-full  malkia00            Running             Running about a minute ago
hbc@malkia00:~/proj/digiinno19/solution$
```

AARHUS UNIVERSITET

Deployment

- One node has
  - More RAM
  - More Disk
    - ‘type=database’

- I can state that the MongoDB must be deployed on such a node.



AARHUS UNIVERSITET

Constraints

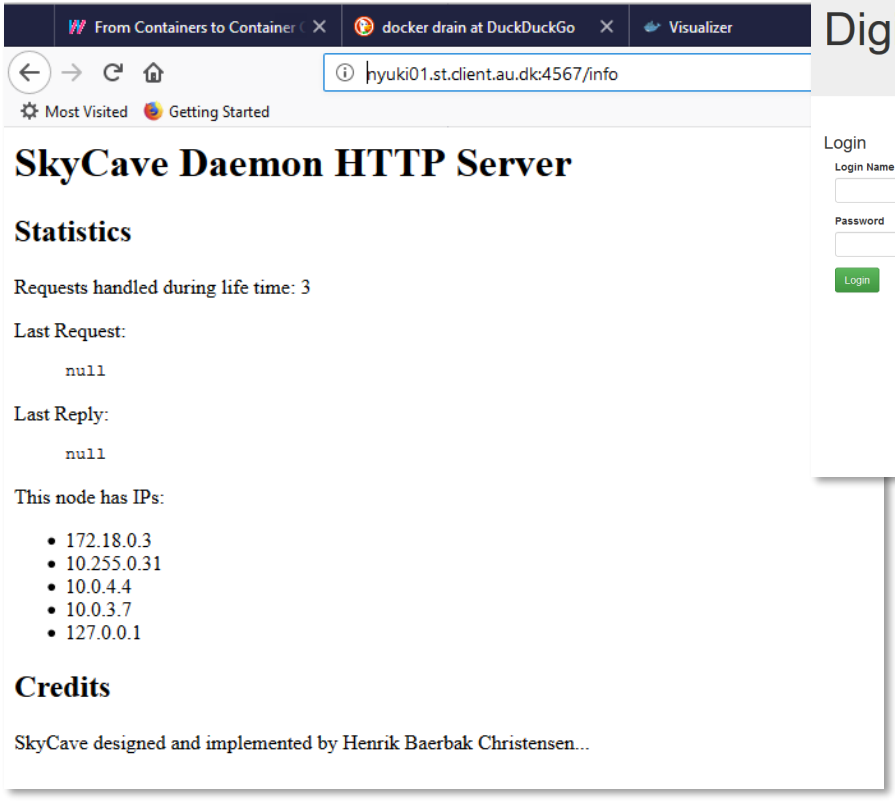
- Wire the DB to the right VM

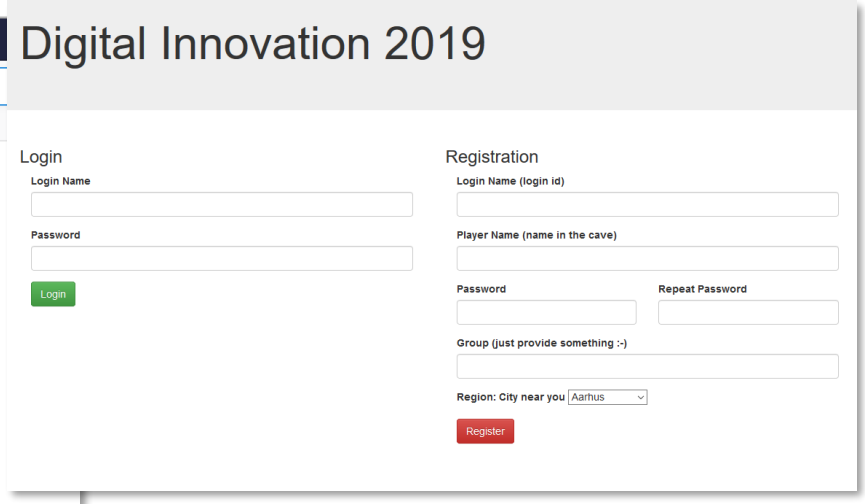
```
# --- MongoDB used by subscription server
mongodb:
  image: mongo:3.2.5

  networks:
    - backend

  # Fix the deployment so data is persisted (disabled to allow easy restart)
  # volumes:
  #   - mongo-data:/data/db

  deploy:
    placement:
      # NOTE: only works if 'docker node update --label-add type=database (node)'
      # has been issued by the manager!
      constraints: [node.labels.type == database]
```

```
deploy:
  placement:
    # NOTE: only works if 'docker node update --label-add type=database (node)'
    # has been issued by the manager!
   constraints:
     - node.labels.type == database
```

AARHUS UNIVERSITET

And Visually





AARHUS UNIVERSITET

Updating the Stack

- Two central operations
  - *I need to adjust the stack's configuration*
    - Edit the compose/stack file
    - Issue the 'docker stack deploy' again
    - **Easy** for *stateless services*
      - *For instance change the replication factor of 'daemon'*
    - **Not so easy** for *statefull services*
      - Do not move a db service away from its volume ☺
        - » *Solution: Use architecture for that – redundancy!*
  - *I need to do maintenance of node X*
    - *Docker node drain X*

AARHUS UNIVERSITET

Drain

- Need to do stuff on nyuki02

```
hbc@malkia00:~/proj/cave$ docker node update --availability drain nyuki02
nyuki02
hbc@malkia00:~/proj/cave$
```

Drain

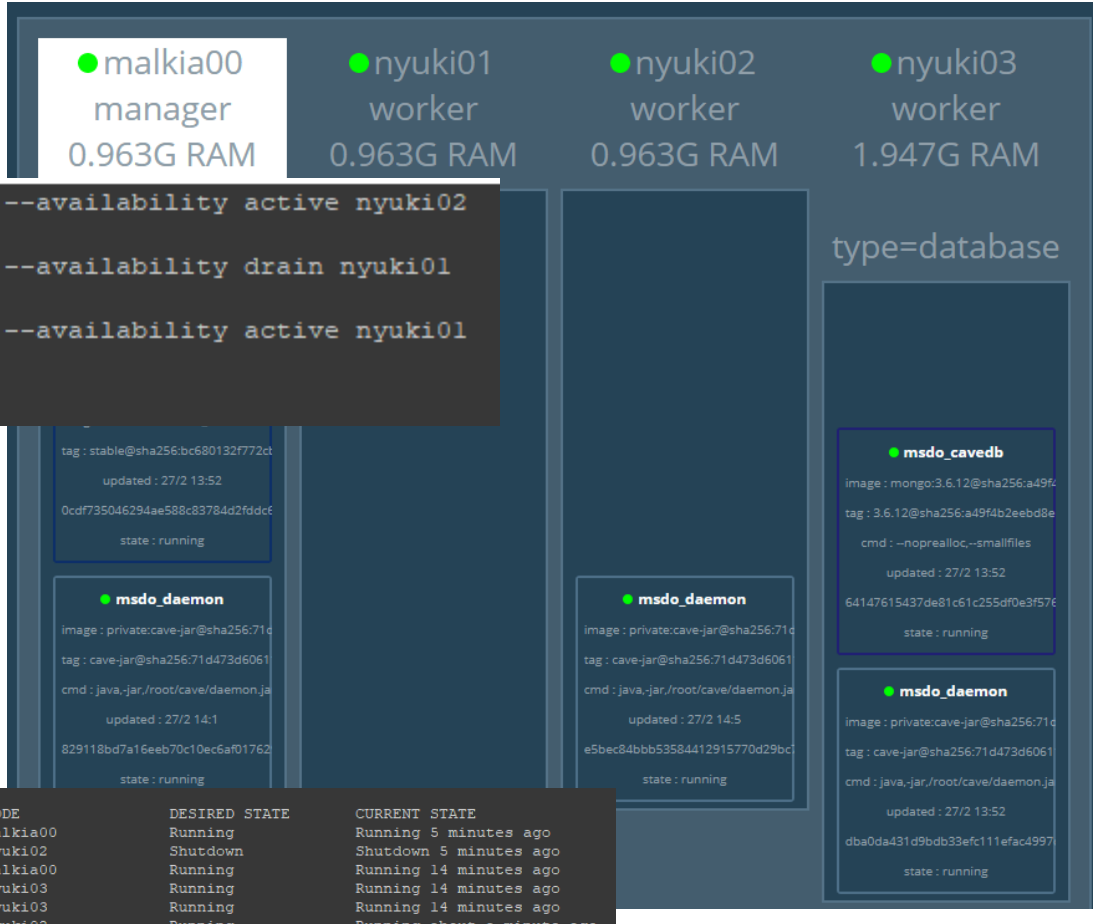
- Service now deployed on malkia00

```
hbc@malkia00:~/proj/cave$ docker node inspect nyuki02 --pretty
ID:                     jx7mh6dxl7pnwnn3se02fohes
Hostname:               nyuki02
Joined at:              2020-02-27 12:50:10.561004278 +0000 utc
Status:
 State:                 Ready
 Availability:          Drain
 Address:               10.24.12.27
Platform:
 Operating System:      linux
 Architecture:          x86_64
Resources:
 CPUs:                  1
 Memory:                985.6MiB
```

AARHUS UNIVERSITET

Back to work

```
hbc@malkia00:~/proj/cave$ docker node update --availability active nyuki02
nyuki02
hbc@malkia00:~/proj/cave$ docker node update --availability drain nyuki01
nyuki01
hbc@malkia00:~/proj/cave$ docker node update --availability active nyuki01
nyuki01
hbc@malkia00:~/proj/cave$
```




```
hbc@malkia00:~/proj/cave$ docker stack ps msdo
ID                  NAME                IMAGE                                NODE                DESIRED STATE       CURRENT STATE
3ugzqg77rf9n        msdo_daemon.1       henrikbaerbak/private:cave-jar       malkia00            Running             Running 5 minutes ago
wp922rhugkw2         \_ msdo_daemon.1   henrikbaerbak/private:cave-jar       nyuki02             Shutdown            Shutdown 5 minutes ago
ioyhcvwqzm8v        msdo_visualizer.1   dockersamples/visualizer:stable      malkia00            Running             Running 14 minutes ago
zwl5e5q0848x        msdo_cavedb.1       mongo:3.6.12                         nyuki03             Running             Running 14 minutes ago
tfdkkqblmmid        msdo_daemon.2       henrikbaerbak/private:cave-jar       nyuki03             Running             Running 14 minutes ago
lw7nnz5gp6s3        msdo_daemon.3       henrikbaerbak/private:cave-jar       nyuki02             Running             Running about a minute ago
ky09u2lq8edj         \_ msdo_daemon.3   henrikbaerbak/private:cave-jar       nyuki01             Shutdown            Shutdown about a minute ago
hbc@malkia00:~/proj/cave$
```

AARHUS UNIVERSITET

Updating the Stack

- Old services actually hang around
  - To allow rollback

```
crunch@crunch3:~/proj/crunch4$ docker stack ps crunch-webui-stack
ID                  NAME                                IMAGE                                NODE                DESIRED STATE       CURRENT STATE            ERROR
PORTS
o5f6ml9duyz6        crunch-webui-stack_crunchui.1       henrikbaerbak/private:crunchwebui    crunch3             Running             Running 2 minutes ago
kn9afbjdkezj        crunch-webui-stack_submissiondb.1   mongo:3.4                            crunch3             Running             Running 2 minutes ago
362ko934vmnt        crunch-webui-stack_crunchui.1       henrikbaerbak/private:crunchwebui    crunch3             Shutdown            Shutdown 18 hours ago
4anijf657vth         \_ crunch-webui-stack_crunchui.1   henrikbaerbak/private:crunchwebui    crunch3             Shutdown            Shutdown 25 hours ago
s75rilf58r74         \_ crunch-webui-stack_crunchui.1   henrikbaerbak/private:crunchwebui    crunch3             Shutdown            Shutdown 2 days ago
80czxcw9ntx7         \_ crunch-webui-stack_crunchui.1   henrikbaerbak/private:crunchwebui    crunch3             Shutdown            Shutdown 3 days ago
9jjsoxiddu9j        crunch-webui-stack_submissiondb.1   mongo:3.4                            crunch3             Shutdown            Complete 2 minutes ago
```

  - (But I am not aware of efficient prune command…)

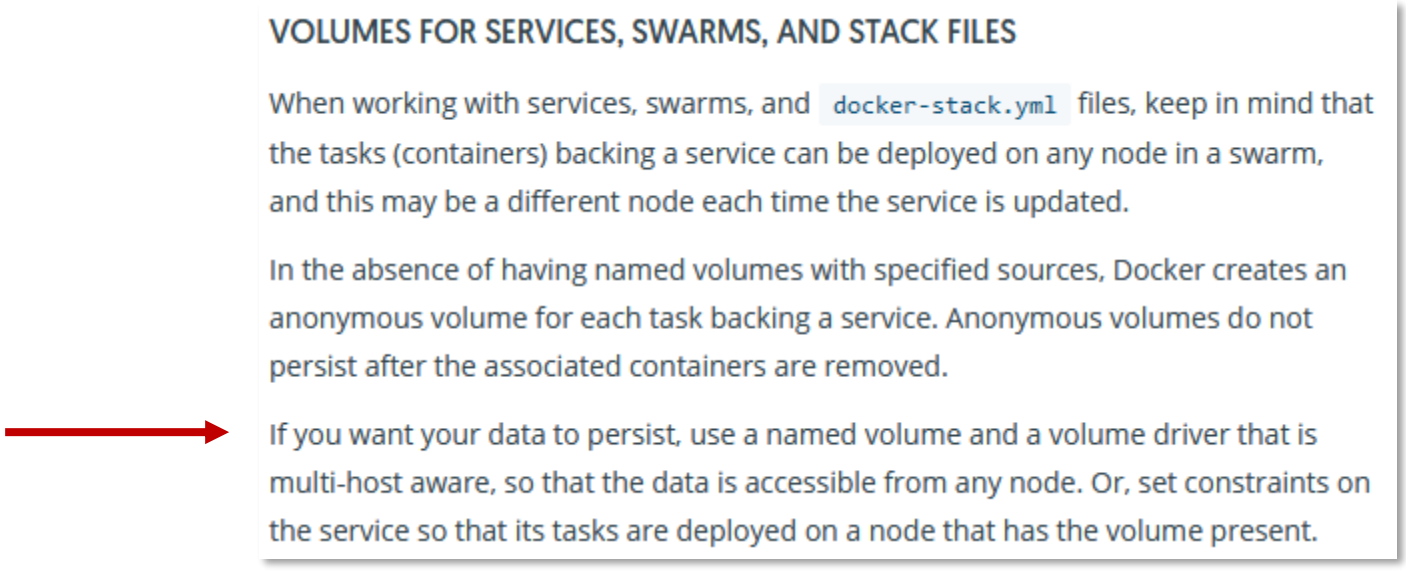
AARHUS UNIVERSITET

Persistence

- An ordinary docker container can mount specific folders on the host's file system
- Ex:
  - Mongo stores data in folder /data/db, so
  - -v ~/my-mongo-folder:/data/db
  - Will ensure that this folder is mounted as ~/my-mongo-folder on the host machine
- But – swarm services may be redeployed?!?

Persistence

- Solution: *Use 'named volumes'*



VOLUMES FOR SERVICES, SWARMS, AND STACK FILES

When working with services, swarms, and `docker-stack.yml` files, keep in mind that the tasks (containers) backing a service can be deployed on any node in a swarm, and this may be a different node each time the service is updated.

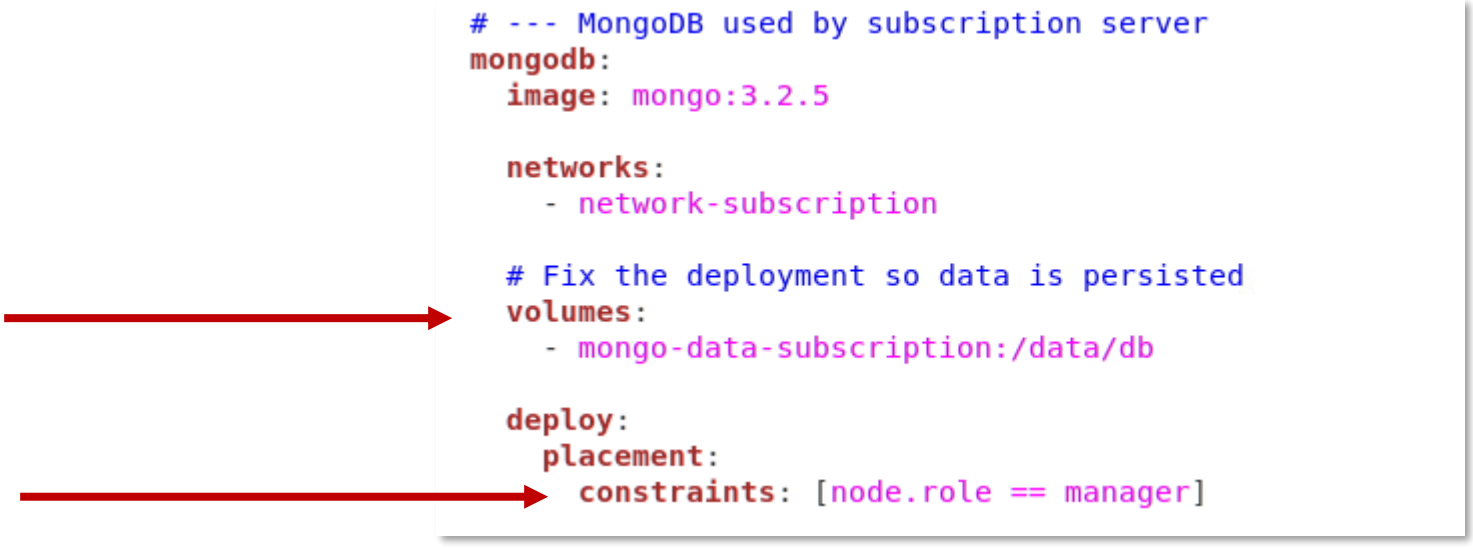
In the absence of having named volumes with specified sources, Docker creates an anonymous volume for each task backing a service. Anonymous volumes do not persist after the associated containers are removed.

If you want your data to persist, use a named volume and a volume driver that is multi-host aware, so that the data is accessible from any node. Or, set constraints on the service so that its tasks are deployed on a node that has the volume present.

AARHUS UNIVERSITET

Example:

- From the compose file of the ‘cavereg.baerbak.com’ subscription service



```
# --- MongoDB used by subscription server
mongodb:
  image: mongo:3.2.5

  networks:
    - network-subscription

  # Fix the deployment so data is persisted
  volumes:
    - mongo-data-subscription:/data/db

  deploy:
    placement:
      constraints: [node.role == manager]
```

Secrets

- Secret = Blob of data
  - Passwords, keystore files, certificates, you name it
  - Ex: ‘docker secret create baerbak.jks /home/secret/baerbak.jks’
- Once created, is distributed using TLS to all nodes
  - i.e. all nodes in swarm have proper access
- A service must be given access to a secret
  - ‘secrets:’ section in compose file

Compose?

- It is swarm and stacks and services
  - Why is it then called a ‘compose file’

- Docker compose was a precursor to swarm
  - And still maintained… You need to install ‘docker-compose’
  - Main difference to swarm
    - Runs ONLY on a single node (all services deployed locally)
    - ‘image: ..’ entry could be replace by ‘build: …’
      - So compose-files could refer to local Dockerfiles and include the build step…

Summary

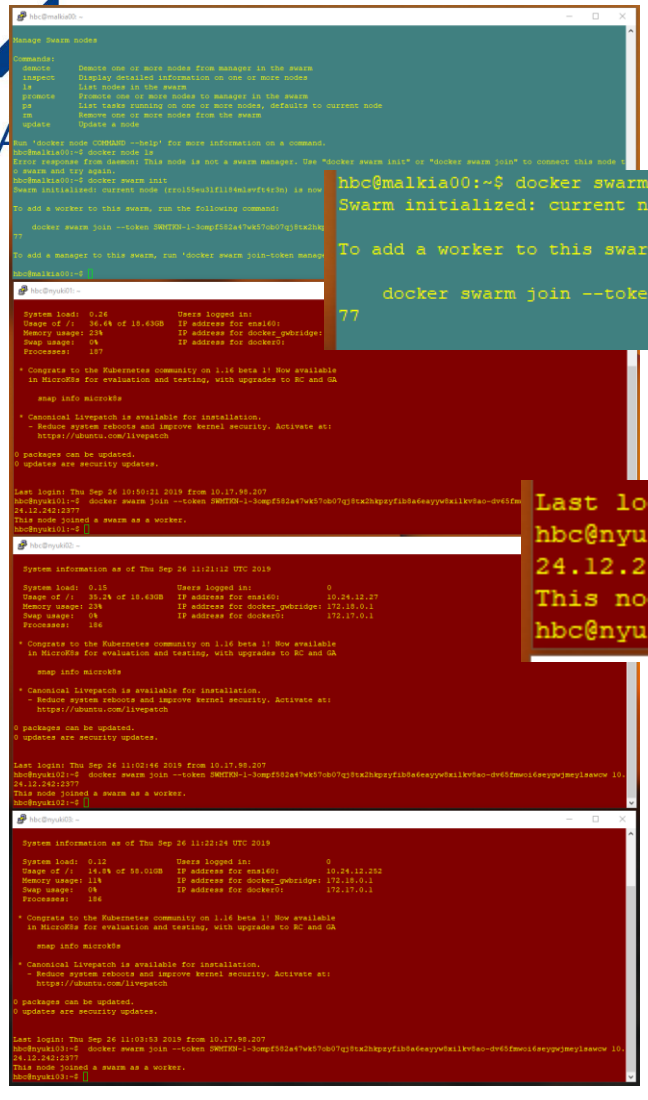
- *Infrastructure-as-code:*
  - Finally, full production architecture is codified!
- The terminology layering
  - Service
    - A single container running in ‘docker-engine’
  - Docker-compose
    - Sets of containers deployed on single docker-engine
  - Swarm
    - A set of docker-engines clustered in a network
  - Stack
    - A set of services deployed on a swarm

AARHUS UNIVERSITET

Appendix

Setting up the Swarm

Init



```
hbc@malkia00:~$ docker swarm init
Swarm initialized: current node (rrol55eu3lfll84mlsvft4r3n) is now a manager.

To add a worker to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-3ompf582a47wk57ob07qj8tx2hkpzyfib8a6eayyw8xilkv8ao-dv65fmwoi6seygwjmeylsawcw 10.24.12.242:2377

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions.
```

```
Last login: Thu Sep 26 10:50:21 2019 from 10.17.98.207
hbc@nyuki01:~$  docker swarm join --token SWMTKN-1-3ompf582a47wk57ob07qj8
24.12.242:2377
This node joined a swarm as a worker.
hbc@nyuki01:~$
```

AARHUS UNIVERSITET

Status

- Overview

```
hbc@malkia00: ~
hbc@malkia00:~$ docker node ls
ID                            HOSTNAME            STATUS              AVAILABILITY        MANAGER STATUS      ENGINE VERSION
rrol55eu3lfll84mlsvft4r3n *   malkia00            Ready               Active              Leader              19.03.2
mgj4eo5q9liijmhebjhg2jtjx     nyuki01             Ready               Active                                  19.03.2
wx0sjvllacxf6wusblofel0nj     nyuki02             Ready               Active                                  19.03.2
xcjv0gpvjzcdm93wl7q9303qg     nyuki03             Ready               Active                                  19.03.2
hbc@malkia00:~$
```

- Setting type

```
hbc@malkia00:~$ docker node update --label-add type=database nyuki03
nyuki03
hbc@malkia00:~$
```

AARHUS UNIVERSITET

TestRun

- Run subscription service with MongoDB on ‘database’ node

```
hbc@malkia00:~/proj/msdo-operation$ docker stack ps subscription
ID                  NAME                                    IMAGE                                   NODE                DESIRED STA
E       CURRENT STATE           ERROR               PORTS
ljz2klg3m6x3        subscription_mongodb.1                  mongo:3.2.5                             nyuki03             Running
        Running 53 seconds ago
0nzby7llodxk        subscription_subscriptionservice.1      henrikbaerbak/private:subscription-v100 nyuki02             Running
        Running 17 seconds ago
hbc@malkia00:~/proj/msdo-operation$
```