Docker illustrated

https://training.play-with-docker.com/

Docker (and it's Swarm)

- Supports Infrastructure as service (IaS)
  - Declaratively specify apps in containers and their service connections over their own network
  - Isolation, increased security, simplified maintenance, supporting Agile software development
- Elastic run time
  - Increase/decrease nodes as needed for demand or maintenance
    - More automation with Kubernetes than Swarm
- Such container technologies behind Amazon, Databricks, etc.
  - Yarn / HDFS a specialized variant for data processing
  - It's possible to run Spark in Docker swarm

Online Docker

- https://training.play-with-docker.com/ops-s1-hello/
  - Training labs
  - Simpler interface
- https://labs.play-with-docker.com/
  - Better interface

Hello World

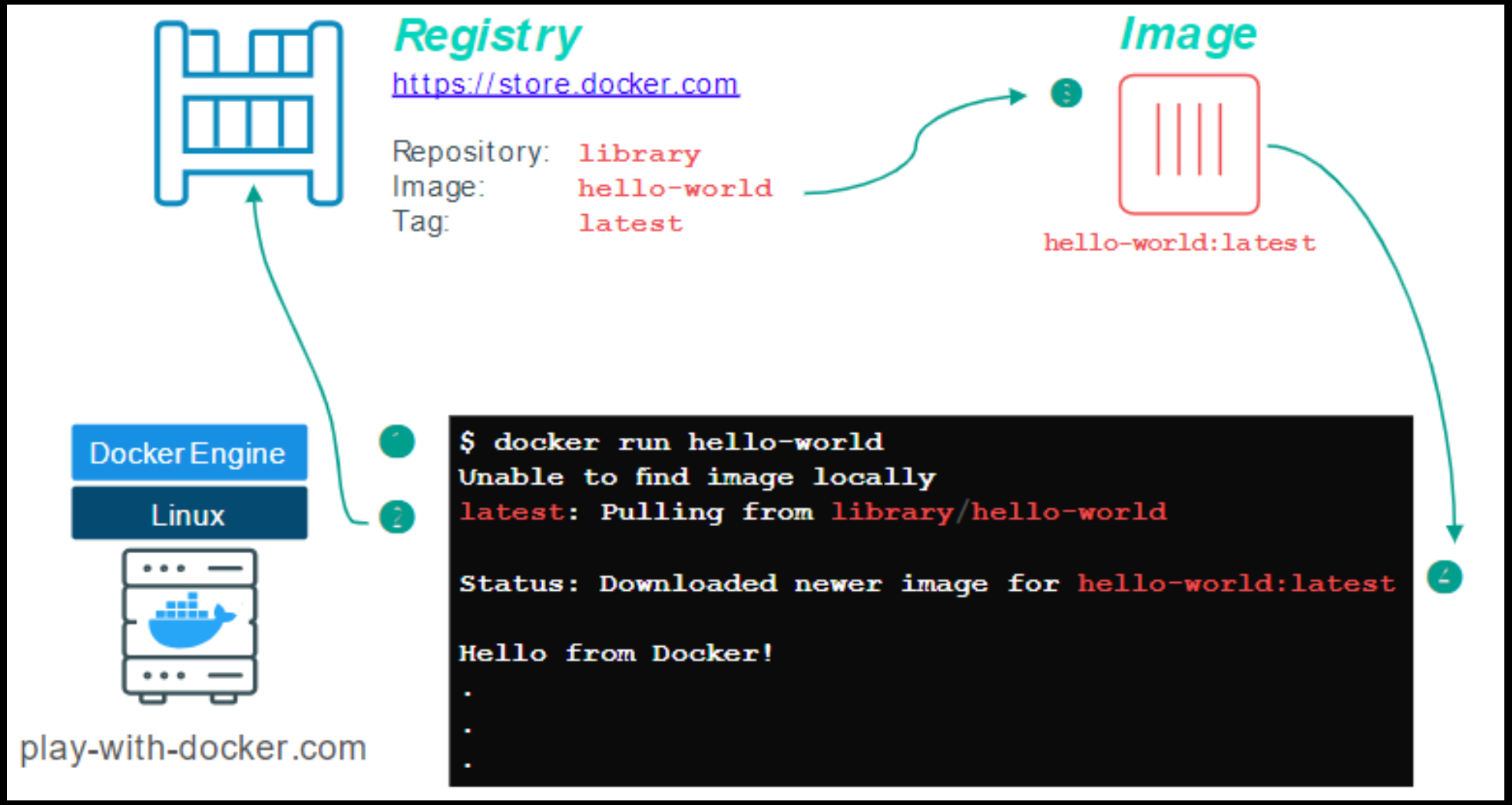
- First run
  - Must pull image from store.docker.com

```
$ docker container run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
1b930d010525: Pull complete
Digest: sha256:92695bc579f31df7a63da6922075d0666e565ceccad16b59c3374d2cf4e8e50e
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
```

- Second run
  - Image is local
  - Run it

```
$ docker container run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.
```

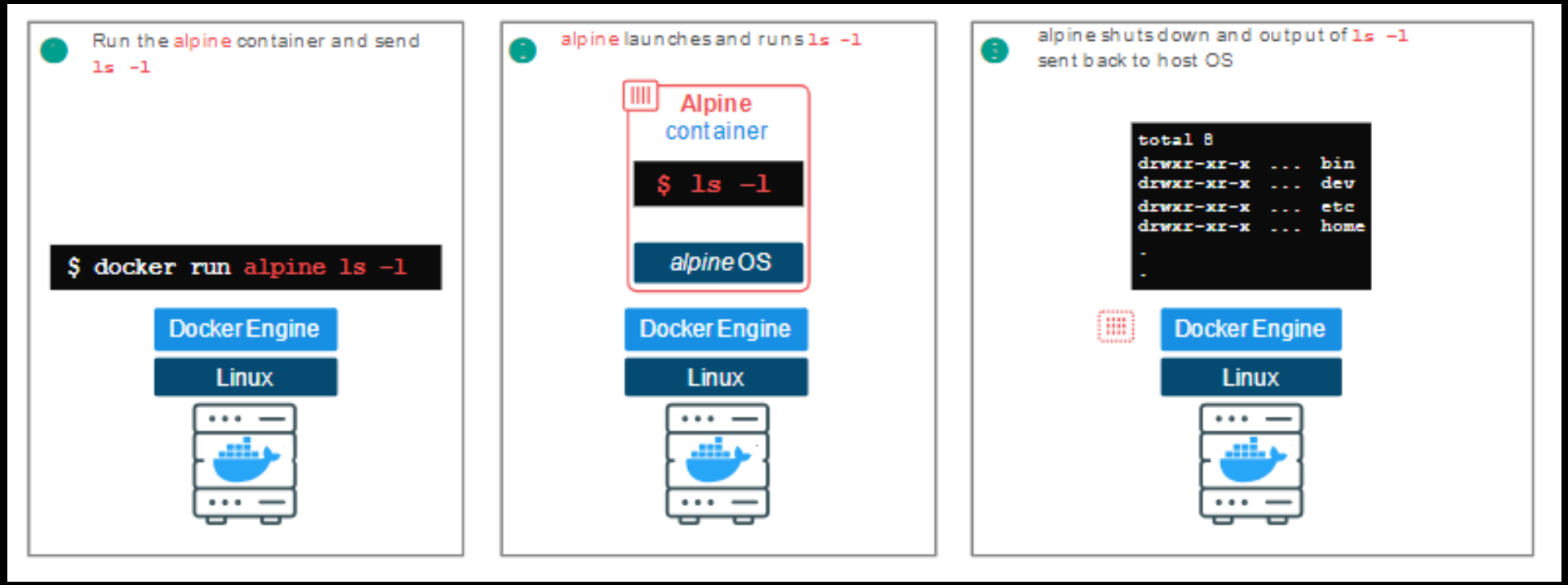
Hello World illustrated



run alpine ls -l

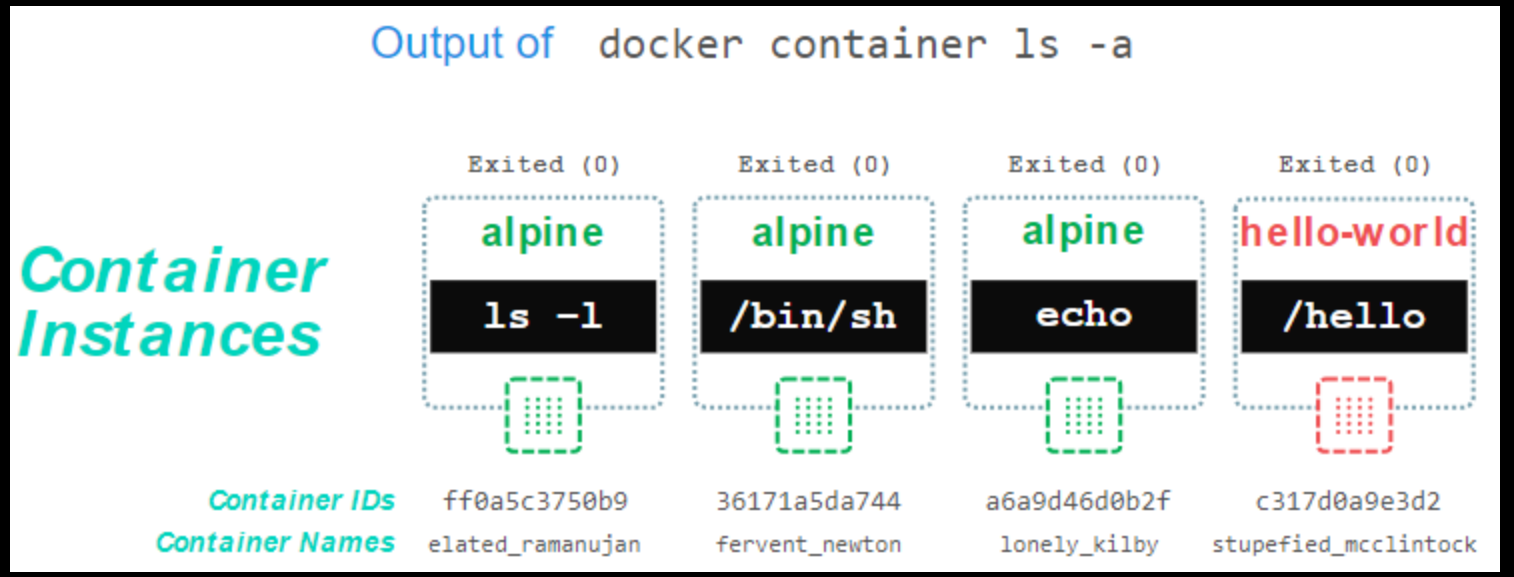
```
$ docker container run alpine ls -l
total 8
drwxr-xr-x    2 root     root          4096 Apr  8 20:30 bin
drwxr-xr-x    5 root     root           340 Apr 16 14:49 dev
drwxr-xr-x    1 root     root            66 Apr 16 14:49 etc
drwxr-xr-x    2 root     root             6 Apr  8 20:30 home
drwxr-xr-x    5 root     root           185 Apr  8 20:30 lib
drwxr-xr-x    5 root     root            44 Apr  8 20:30 media
```

- Start container
- Send "ls –l" to container



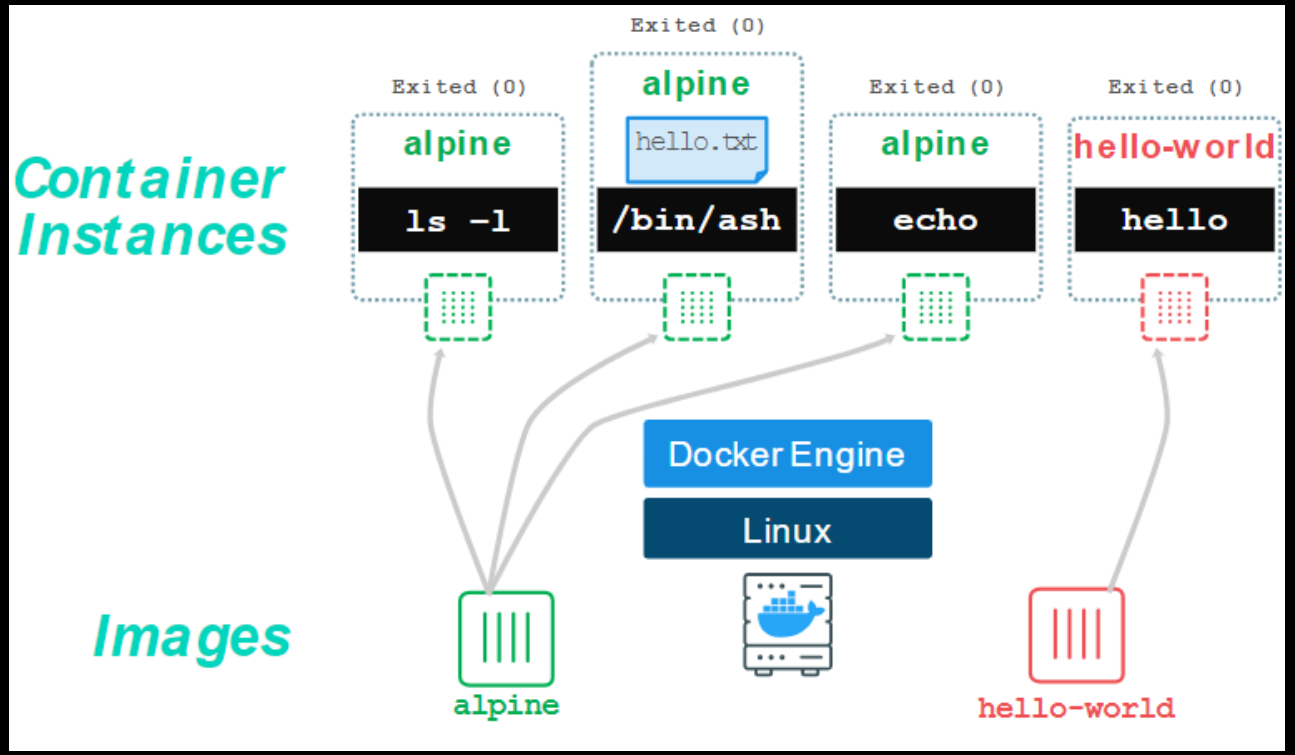
Container history

- **Running containers:** `docker ps`
- **All containers:** `docker ps -a`



Containers are isolated from each other

- Run creates a new container
- Typically Linux
  - Linux and Microsoft containers
- Custom software loaded



Docker run vs start

1. Run: create a new container of an image, and execute the container. You can create N clones of the same image. The command is: `docker run IMAGE_ID` **and not** `docker run CONTAINER_ID`

```
PS C:\Users\Daniele> docker run --help

Usage:  docker run [OPTIONS] IMAGE [COMMAND] [ARG...]

Run a command in a new container
```

2. Start: Launch a container previously stopped. For example, if you had stopped a database with the command `docker stop CONTAINER_ID`, you can relaunch the same container with the command `docker start CONTAINER_ID`, and the data and settings will be the same.

```
PS C:\Users\Daniele> docker start --help

Usage:  docker start [OPTIONS] CONTAINER [CONTAINER...]

Start one or more stopped containers
```

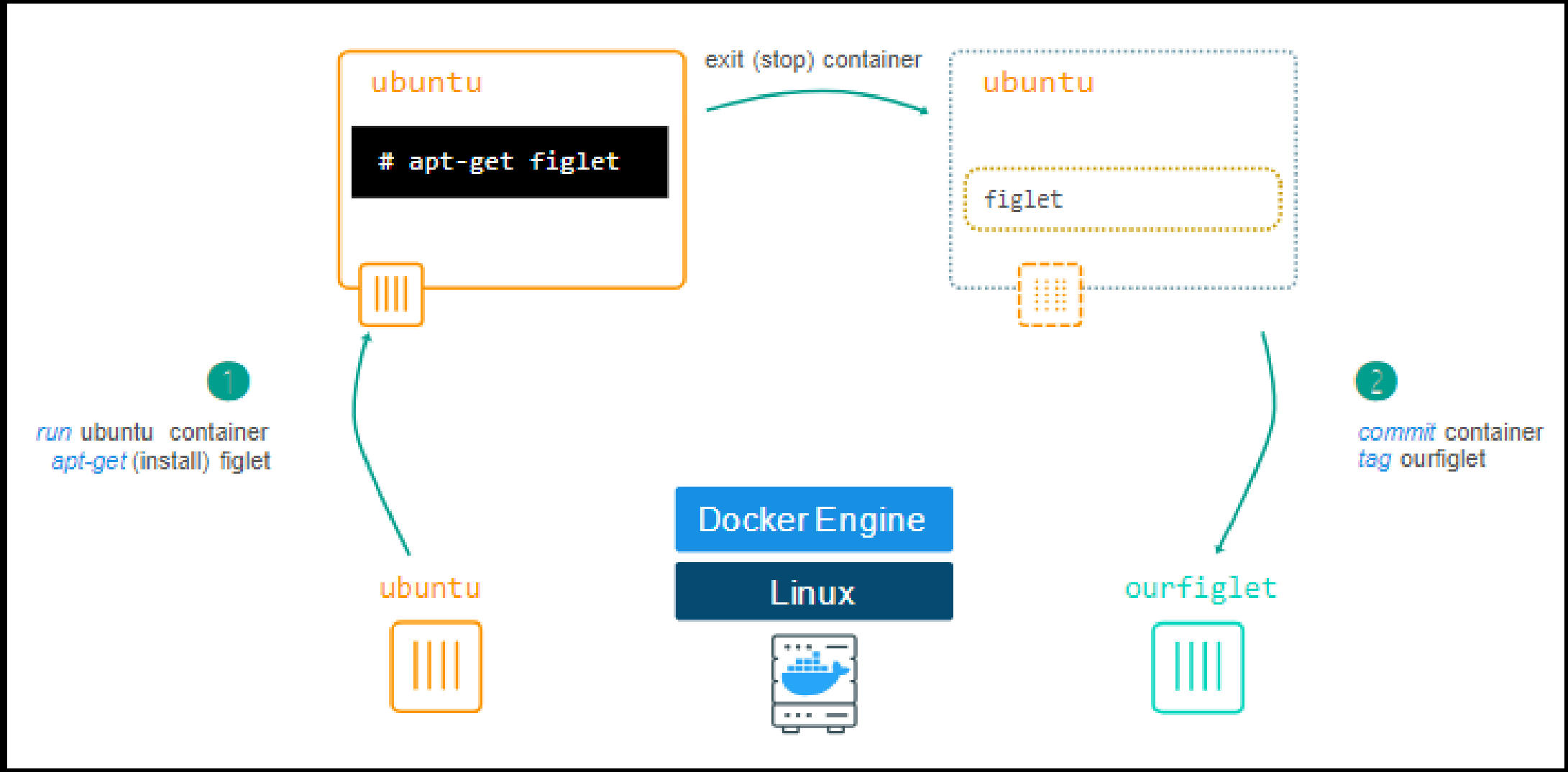
https://stackoverflow.com/questions/34782678/difference-between-running-and-starting-a-docker-container

Docker image

```
$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
ubuntu              latest              94e814e2efa8        5 weeks ago         88.9MB
```

- A **Docker image** is a file, comprised of multiple layers, used to execute code in a **Docker** container.
- An **image** is essentially built from the instructions for a complete and executable version of an application, which relies on the host OS kernel.

Create an image

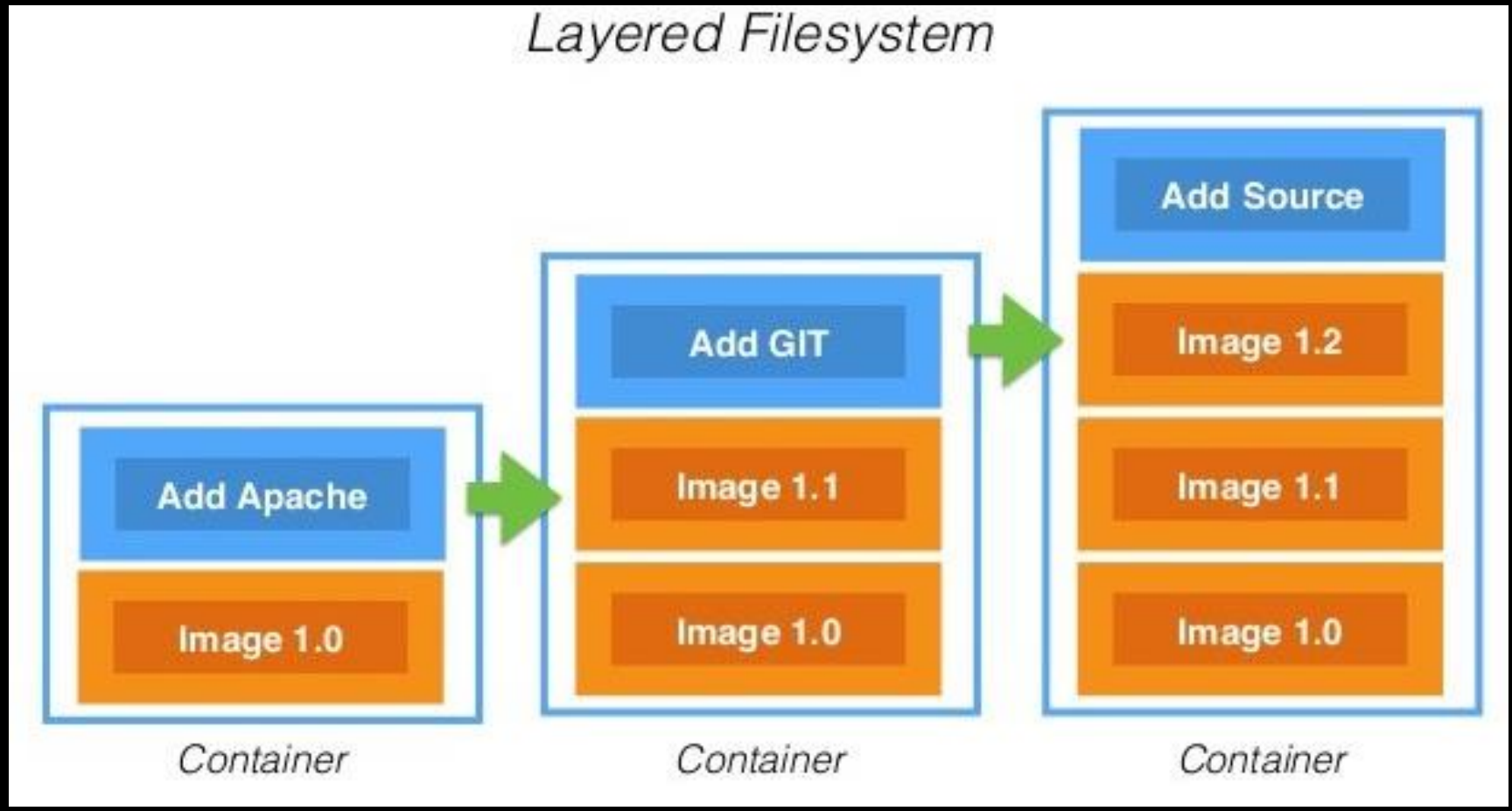


https://training.play-with-docker.com/ops-s1-images/

Running our new image in a container

```
$ docker image tag fb ourfiglet
[node1] (local) root@192.168.0.13 ~
$ docker container run ourfiglet figlet hello

 _                _ _
| |__    ___    | | |  ___
| '_ \  / _ \   | | | / _ \
| | | |  __/   | | | (_) |
|_| |_|\___|   |_|_|\___/
```

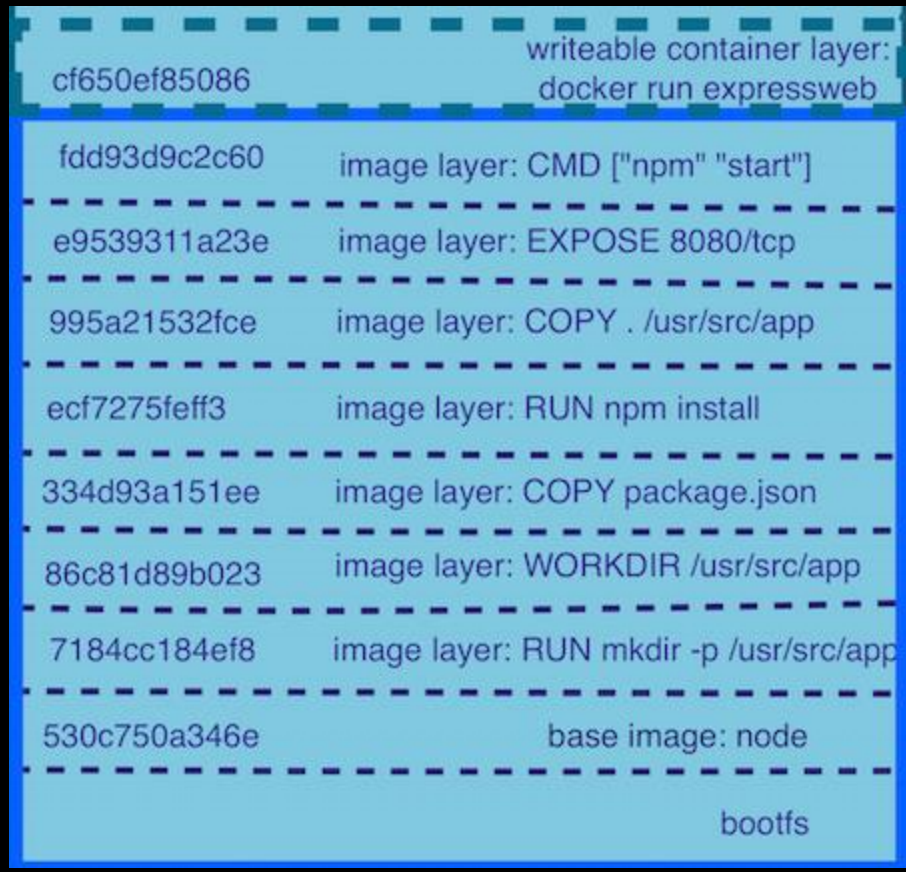
Docker images defined with layers



Docker image layers

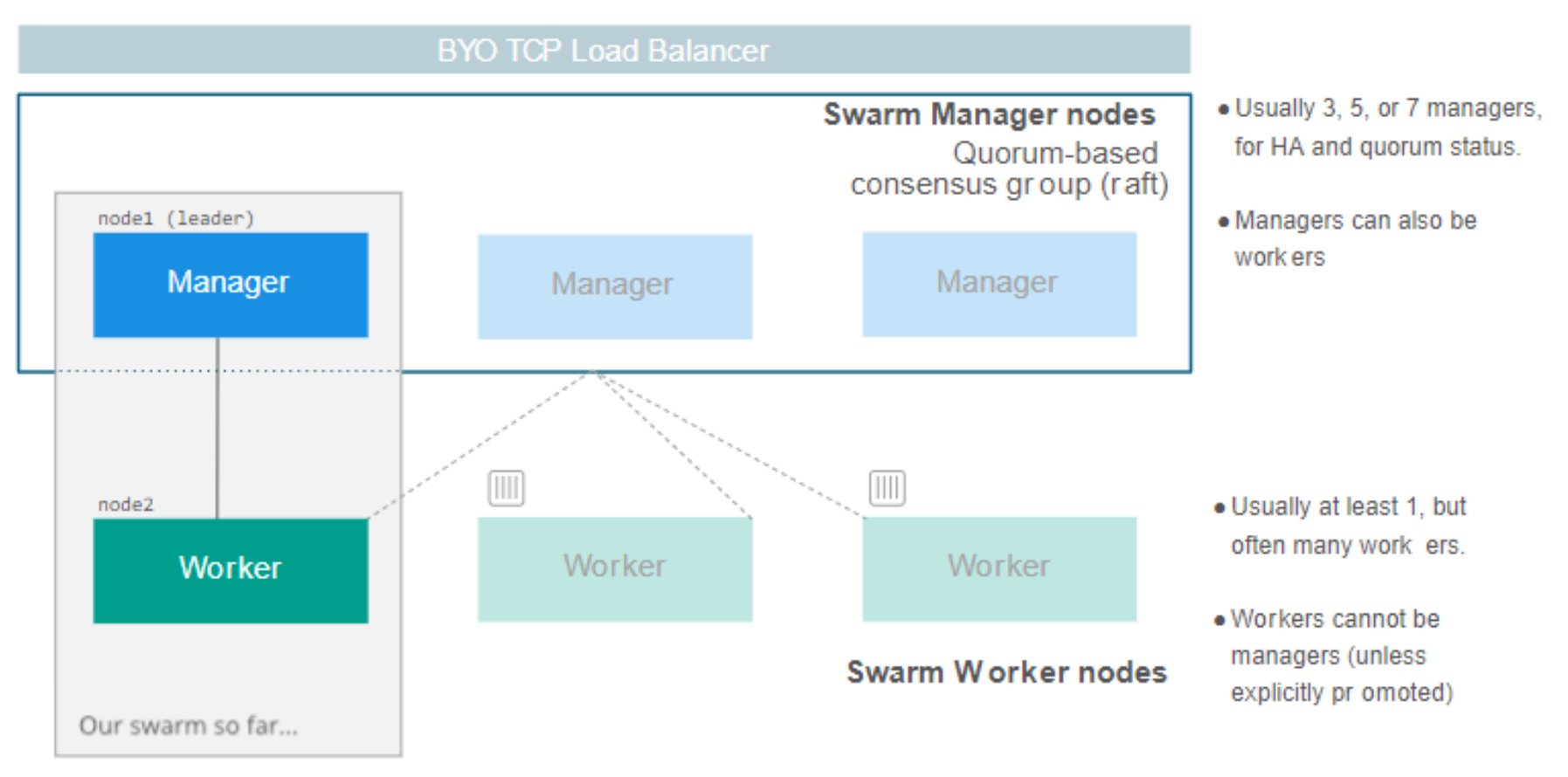
- Layers (also called intermediate images) are generated when the commands in the Dockerfile are executed during the Docker image build.

- Updating or customizing an image is done by adding layers
  - Can be very fast for simple updates



https://medium.com/@jessgreb01/digging-into-docker-layers-c22f948ed612

Docker Swarm (networked containers)



https://training.play-with-docker.com/ops-s1-swarm-intro/

Example of running a Swarm

- git clone https://github.com/docker/example-voting-app
  - Downloads sources
  - Contains docker-compose file, which has instructions on composing containers (pull and deploy)

```
git clone https://github.com/docker/example-voting-app
cd example-voting-app
```

- Deploy the stack
  - A stack is a group of services that are deployed together
  - Each individual service is made up of one or more containers, called tasks
  - The tasks & services together make up a stack
  - Defined in a YAML file

```
docker stack deploy --compose-file=docker-stack.yml voting_stack

docker stack ls
```

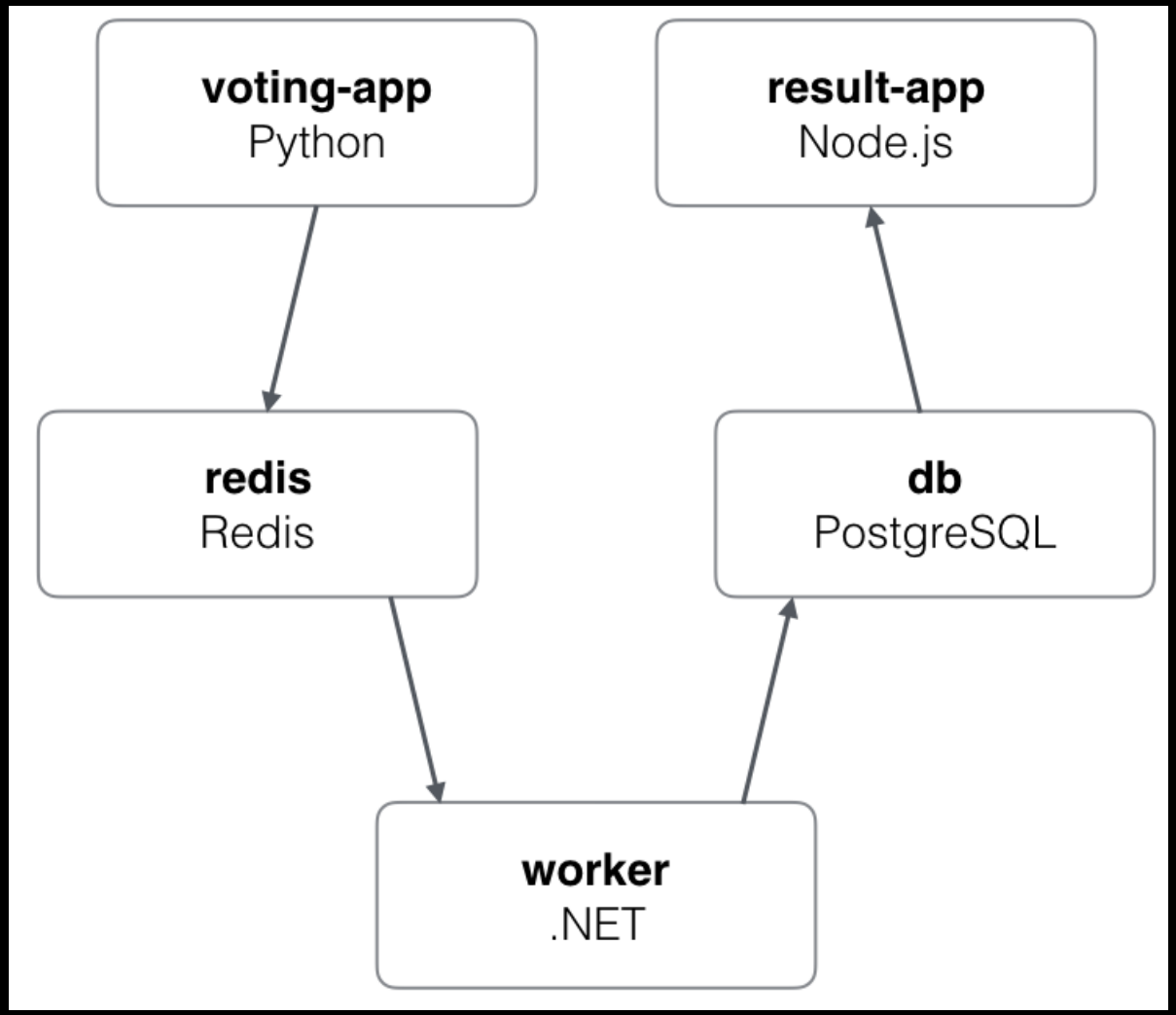
https://training.play-with-docker.com/ops-s1-swarm-intro/

Voting stack YAML file

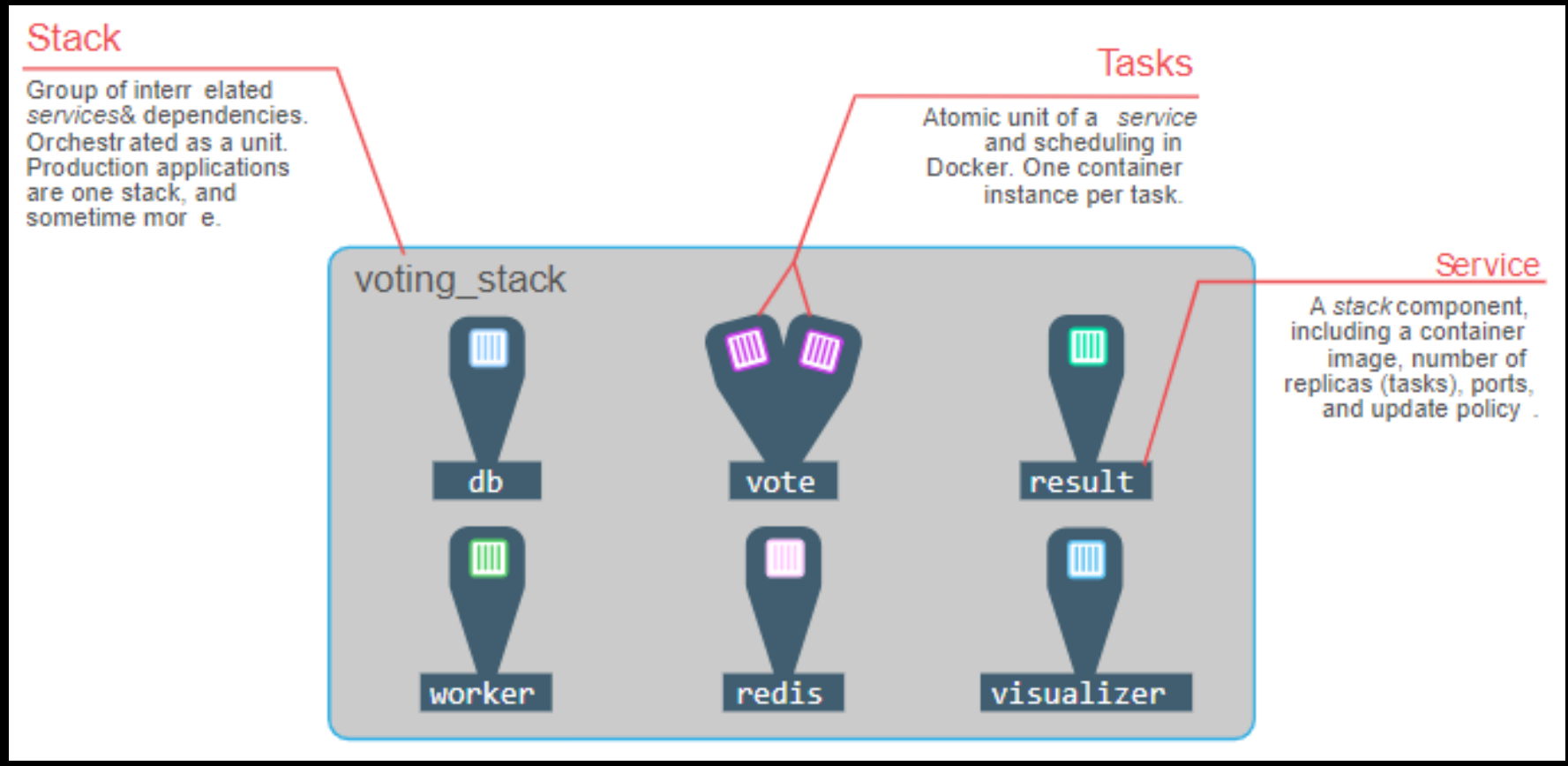
```yaml
services:
  vote:
    build: ./vote
    command: python app.py
    volumes:
     - ./vote:/app
    ports:
      - "5000:80"
    networks:
      - front-tier
      - back-tier

  result:
    build: ./result
    command: nodemon server.js
    volumes:
      - ./result:/app
    ports:
      - "5001:80"
      - "5858:5858"
    networks:
      - front-tier
      - back-tier

  worker:
    build:
      context: ./worker
    depends_on:
      - "redis"
    networks:
      - back-tier

  redis:
    image: redis:alpine
    container_name: redis
    ports: ["6379"]
    networks:
      - back-tier

  db:
    image: postgres:9.4
    container_name: db
    volumes:
      - "db-data:/var/lib/postgresql/data"
    networks:
      - back-tier

volumes:
  db-data:

networks:
  front-tier:
  back-tier:
```

https://github.com/dockersamples/example-voting-app/blob/master/docker-compose.yml

App Container architecture



Swarm: Stacks, Services, Tasks



Docker stack services

- View the services of a running stack

```
docker stack services voting_stack
```

```
$ docker stack services voting_stack
ID                  NAME                    MODE                REPLICAS            IMAGE
                  PORTS
h00br1j1eewu        voting_stack_vote       replicated          2/2                 dockersamples/examplevotingap
p_vote:before     *:5000->80/tcp
nwu8tihc7t1l        voting_stack_redis      replicated          1/1                 redis:alpine

r9ms1sg92fuq        voting_stack_visualizer replicated          1/1                 dockersamples/visualizer:stab
le                *:8080->8080/tcp
sqmkg3xgt7xp        voting_stack_db         replicated          1/1                 postgres:9.4

v8ufgmz3f8rv        voting_stack_result     replicated          1/1                 dockersamples/examplevotingap
p_result:before   *:5001->80/tcp
z6az706cz16f        voting_stack_worker     replicated          1/1                 dockersamples/examplevotingap
p_worker:latest
```

- If there are 0 replicas just wait a few seconds and enter the command again

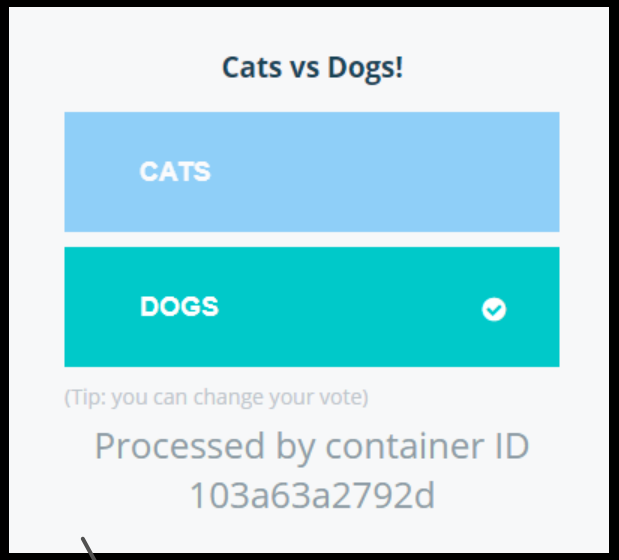
Docker tasks from running services

- View the tasks (replicas) of a running service
  - service ps only shows the running tasks of the service
  - Here, just the voting stack

```
docker service ps voting_stack_vote
```

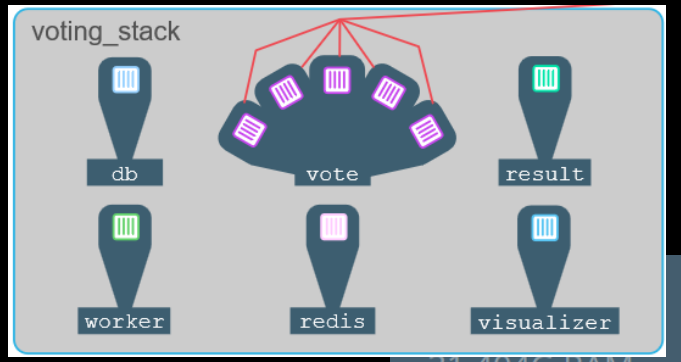
```
$ docker service ps voting_stack_vote
ID                  NAME                  IMAGE                                   NODE                DESIRED STATE       CURRENT STA
TE           ERROR               PORTS
ec0er9ap3f4a        voting_stack_vote.1   dockersamples/examplevotingapp_vote:before   node2               Running             Running 7 m
inutes ago
5wiq4fosmyeb        voting_stack_vote.2   dockersamples/examplevotingapp_vote:before   node1               Running             Running 7 m
inutes ago
```

Swarm visualizer: nodes and their containers



Select built-in SWARM VISUALIZER, front-end web UI, and the result page on play with docker

Scale the application

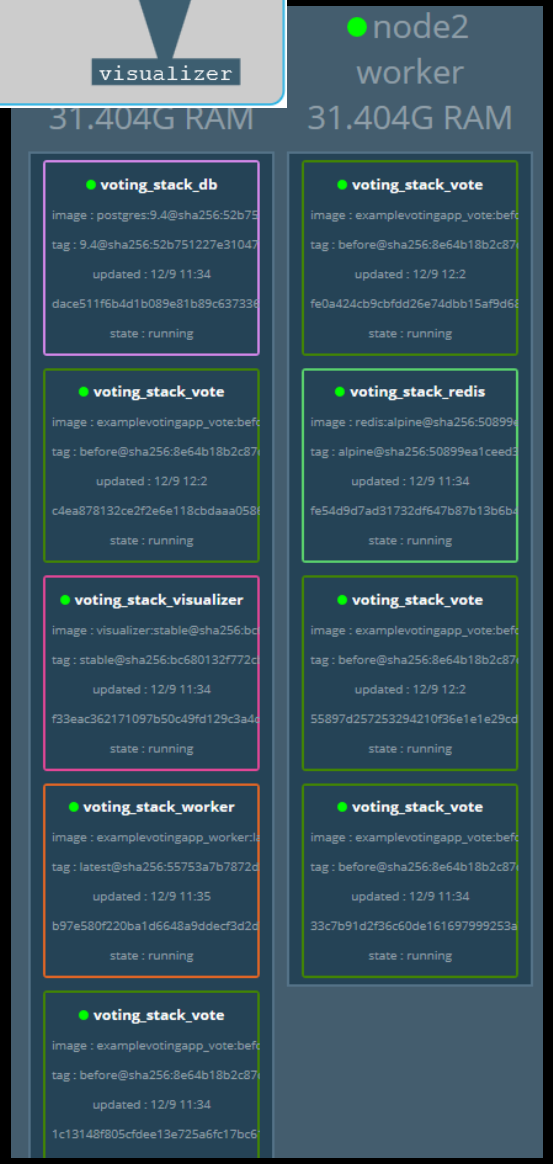


- Assume that you need more computing for the voting service
- To increase the replicas of a

```
docker service scale voting_stack_vote=5
```

- N
r

```
$ docker stack services voting_stack
ID                  NAME                    MODE                REPLICAS
h00br1j1eewu        voting_stack_vote       replicated          5/5
0->80/tcp
nwu8tihc7t1l        voting_stack_redis      replicated          1/1
r9ms1sg92fuq        voting_stack_visualizer replicated          1/1
0->8080/tcp
sqmkg3xgt7xp        voting_stack_db         replicated          1/1
v8ufgmz3f8rv        voting_stack_result     replicated          1/1
1->80/tcp
z6az706cz16f        voting_stack_worker     replicated          1/1
```



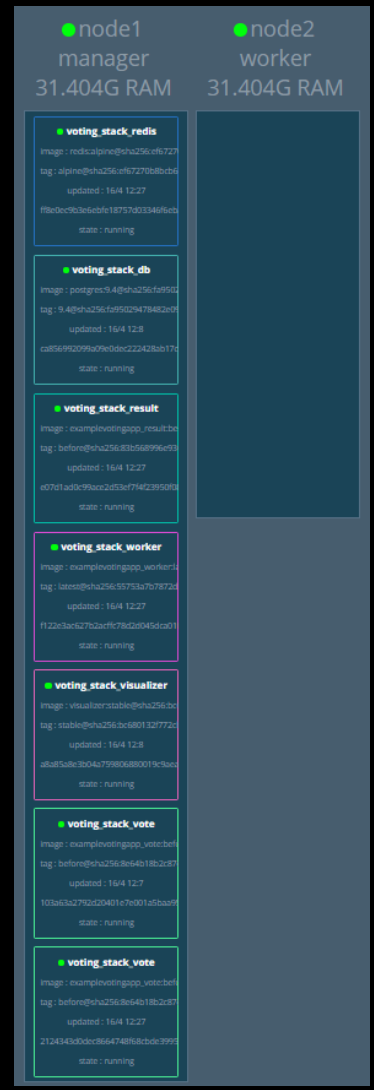
Reduce scale & Drain a node

- Scale down to 2 replicas

```
$ docker service scale voting_stack_vote=2
voting_stack_vote scaled to 2
overall progress: 2 out of 2 tasks
1/2: running
2/2: running
```

- Bring a node down for maintenance
  - docker node update --availability drain node2
- Bring it back up
  - docker node update --availability active <NODE-ID>
- Ensure services migrate to updated node

```
$ docker service update --force voting_stack_vote
voting_stack_vote
overall progress: 2 out of 2 tasks
1/2: running   [==================================================>]
2/2: running   [==================================================>]
verify: Service converged
[node1] (local) root@192.168.0.12 ~/example-voting-app
$ docker service update --force voting_stack_redis
voting_stack_redis
overall progress: 1 out of 1 tasks
1/1: running   [==================================================>]
verify: Service converged
[node1] (local) root@192.168.0.12 ~/example-voting-app
$ docker service update --force voting_stack_result
```



Common Docker swarm commands

- docker service --help
  - logs
  - ls
  - ps
  - Scale
- docker stack --help
  - deploy
  - ls
  - ps
  - rm
  - services

Do it yourself: Swarm lab

- On play with docker
  - https://training.play-with-docker.com/ops-s1-swarm-intro/

Docker Hub Image Demos

- https://labs.play-with-docker.com/

- Docker images
  - https://hub.docker.com/r/jupyter/pyspark-notebook
    - Too big for PWD, but on your own notebook
    - docker run -p 8888:8888 jupyter/pyspark-notebook
  - https://hub.docker.com/_/wordpress/
  - https://hub.docker.com/r/bign8/games

```
import pyspark
if not 'sc' in globals():
    sc = pyspark.SparkContext()

rdd = sc.parallelize(range(100))
rdd.count()
```

Docker (and it's Swarm)

- Supports Infrastructure as service (IaS)
  - Declaratively specify apps in containers and their service connections over their own network
  - Isolation, increased security, simplified maintaince, supporting Agile software development
- Elastic run time
  - Increase/decrease nodes as needed for demand or maintenance
    - More automation with Kubernetes than Swarm
- Such container technologies behind Amazon, Databricks, etc.
  - Yarn / HDFS a specialized variant for data processing
  - It's possible to run Spark in Docker swarm