What’s up docker

By Patrick Louis

What’s a container, does it differ from a Virtual Machine?

VM vs VE

LXC vs Docker

Two container tech

- LXC for full OS
- Docker for single app (microservices)
- Docker uses the dockerengine (libcontainer)
- Docker offers building of image and control of network, storage, logging, etc..

Getting a taste of docker

```
docker pull hello-world
```

```
docker run hello-world
```

We can search online for available images on https://hub.docker.com

Or on the command line directly

```
docker search hello-world
```

The Lingo

- Image
- Container
- Dockerfile
- Volume
- Network

Creating our own Image

Dockerfile

- **FROM**: extend another image, imagename:tagname
- **WORKDIR**: where to be in the image
- **EXPOSE**: mention that a port is running something, a hint
- **RUN**: run a command
- **CMD** and **ENTRYPOINT**, what to run as last, can be string (passed to sh -c) or array, as is
- **ENV**: update environment variable
- **COPY/ADD**: copy from host to container (**ADD** automatically does extraction of compressed files)

A Simple Example

Building & Running

```
docker build <dockerfilename> -t <tagname>
```

```
docker image ls
```

Exposing ports/what's inside the container:

```
docker run --volume "$(pwd):/mydir" <container>
```

```
docker run --publish <hostPort>:<containerPort> imageName
```

(Can put the IP too -> bind to a socket)

```
docker cp "<container>:<filename>" .
```

Management commands

● docker image ls	*docker images*
● docker image rm <image>	*docker rmi*
● docker image pull <image>	*docker pull*
● docker container ls -a	*docker ps -a*
● docker container run <image>	*docker run* can have -d in background or --rm to remove afterwards
● docker container rm <container>	*docker rm*
● docker container stop <container>	*docker stop*
● docker container exec <container>	*docker exec*
● docker tag ubuntu:latest ubuntu:20	
● docker diff <container>	
● docker commit <container> tag	

Use it when running in interactive mode Example:

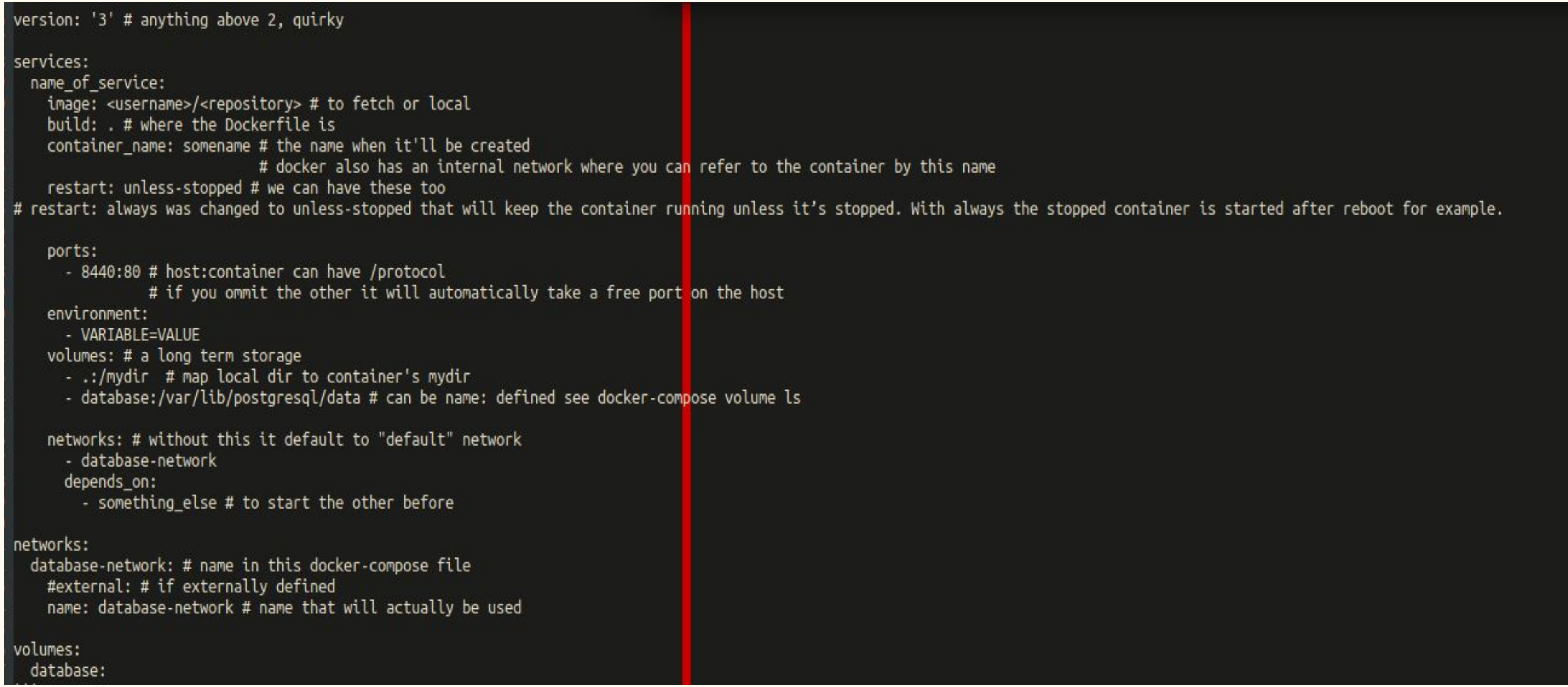
- docker exec -it <container> bash
- docker run -it <image> bash
- docker stop/start <container>

Composing

Running multiple containers

- docker-compose
- kubernetes

docker-compose



```
version: '3' # anything above 2, quirky

services:
  name_of_service:
    image: <username>/<repository> # to fetch or local
    build: . # where the Dockerfile is
    container_name: somename # the name when it'll be created
                             # docker also has an internal network where you can refer to the container by this name
    restart: unless-stopped # we can have these too
# restart: always was changed to unless-stopped that will keep the container running unless it’s stopped. With always the stopped container is started after reboot for example.

    ports:
      - 8440:80 # host:container can have /protocol
                # if you ommit the other it will automatically take a free port on the host
    environment:
      - VARIABLE=VALUE
    volumes: # a long term storage
      - .:/mydir  # map local dir to container's mydir
      - database:/var/lib/postgresql/data # can be name: defined see docker-compose volume ls

    networks: # without this it default to "default" network
      - database-network
      depends_on:
        - something_else # to start the other before

networks:
  database-network: # name in this docker-compose file
    #external: # if externally defined
    name: database-network # name that will actually be used

volumes:
  database:
```

Optimizations & Quirks

- One feature per containers
- User permission, don't run as root (podman: no need for daemon doesn't run as root, drop-in replacement for docker, but can't build container images)
- Fewer layers
- Multi-stage build
- Lighter distro
- The issue of DB upgrades
- Running as PID1
- Finding the host IP, communicating between different containers
- Hosting our own repository

Conclusion

Think of docker as either:

- An easy to deploy/test env
- A pkg manager

More info:

- Our Wiki
- Man pages (1): In the form of `docker-<subcommand>` (ex: `man docker-build`)

Thank You!