
docker

Andrii Zhuk

- 12 years c++ dev
- 2 years Scala dev
  - AWS cloud
  - Docker

A little history

Virtual Machine - isolated system

A little history

Virtual Machine - isolated system

- Do not depend on current system environment

A little history

Virtual Machine - isolated system

- Do not depend on current system environment
- Universal

A little history

Virtual Machine - isolated system

- Do not depend on current system environment
- Universal
- Security

A little history

Virtual Machine - isolated system

- Do not depend on current system environment
- Universal
- Security
- If something goes wrong..


wallpaperdx.com

A little history

Virtual Machine - isolated system

- Do not depend on current system environment
- Universal
- Security
- If something goes wrong.. BOOM :)
- Manual work (OS installation, environment set up)

A little history

Virtual Machine - isolated system

- Do not depend on current system environment
- Universal
- Security
- If something goes wrong.. BOOM :)
- Manual work (OS installation, environment set up)
- Slow (full OS emulation)

A little history

Bitnami - partial environment isolation

A little history

Bitnami - partial environment isolation

- Fast (Uses current system)

A little history

Bitnami - partial environment isolation

- Fast (Uses current system)
- Effective resources usage

A little history

Bitnami - partial environment isolation

- Fast (Uses current system)
- Effective resources usage
- Easy to set up (Copy/Paste)

A little history

Bitnami - partial environment isolation

- Fast (Uses current system)
- Effective resources usage
- Easy to set up (Copy/Paste)
- If something goes wrong.. no BOOM :(, just remove it

A little history

Bitnami - partial environment isolation

- Fast (Uses current system)
- Effective resources usage
- Easy to set up (Copy/Paste)
- If something goes wrong.. no BOOM :(, just remove it
- System environment dependency

A little history

Bitnami - partial environment isolation

- Fast (Uses current system)
- Effective resources usage
- Easy to set up (Copy/Paste)
- If something goes wrong.. no BOOM :(, just remove it
- System environment dependency
- Security

A little history

Bitnami + **Virtual machine** = **Docker**

Docker - full environment isolation

A little history

Bitnami + **Virtual machine** = **Docker**

Docker - full environment isolation

- Fast

A little history

Bitnami + **Virtual machine** = **Docker**

Docker - full environment isolation

- Fast
- Effective resources usage

A little history

Bitnami + **Virtual machine** = **Docker**

Docker - full environment isolation

- Fast
- Effective resources usage
- Easy to set up

A little history

Bitnami + **Virtual machine** = **Docker**

Docker - full environment isolation

- Fast
- Effective resources usage
- Easy to set up
- Security

A little history

Bitnami + **Virtual machine** = **Docker**

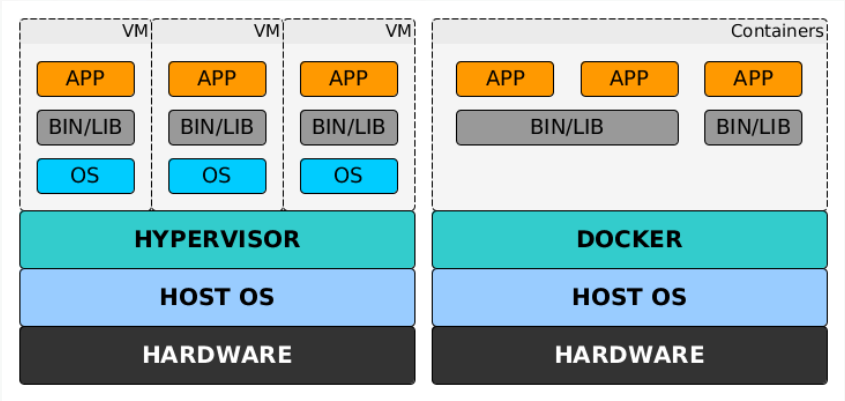
Docker - full environment isolation

- Fast
- Effective resources usage
- Easy to set up
- Security
- If something goes wrong.. BOOM :)

Docker Advantages

Docker Advantages

- **Fast** - no OS installation, using parts of current system

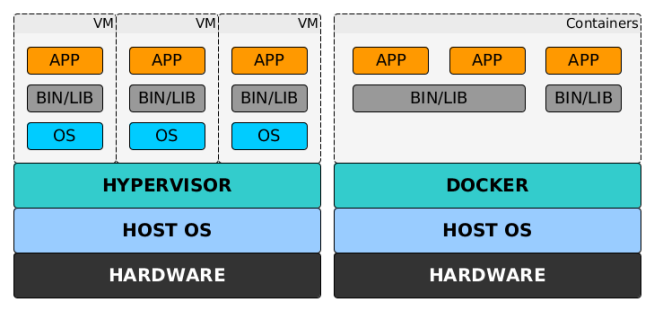


Docker Advantages

- **Fast** - no OS installation, using parts of current system
- **Resources usage efficiency**

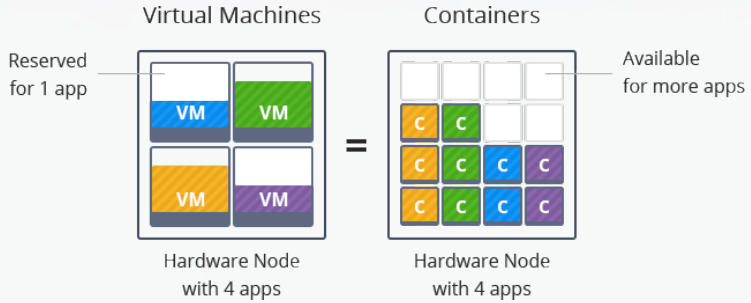
Docker Advantages

Running multiple instances at the same time with no penalty, reusing unchanged data



Docker Advantages

No resources reservation



Docker Advantages

- **Easy to set up** - script base images
  - Automated tests, which may be used for *Build Server*

Docker Advantages

- **Easy to set up** - script base images
  - Automated tests, which may be used for *Build Server*
- **Easy to bind together** - run and test services communication (integration tests)
  - Easy compatibility tests (e.g. **new** *Frontend* with **old** *Backend*)

Docker Advantages

- **Easy to set up** - script base images
  - Automated tests, which may be used for *Build Server*
- **Easy to bind together** - run and test services communication (integration tests)
  - Easy compatibility tests (e.g. **new** *Frontend* with **old** *Backend*)
- **Easy to deploy and migrate** - additional abstract layer. You are sure it will be the same on Production as you tested.

Docker Momentum

450+

Docker EE commercial customers

37B

Container downloads

15K

Job listings on LinkedIn

3.5M

Dockerized apps

200+

Active Docker user groups

Docker Limitations

Docker Limitations

- **No interaction** 
  - Test native UI ? - Go back to VM
  - **Ideal** - web services (backend + frontend), server services

Docker Limitations

- **No interaction** 
  - Test native UI ? - Go back to VM
  - **Ideal** - web services (backend + frontend), server services
- **System type dependency**
  - **Native: Lin** on Lin (docker is Linux product after all)
  - **over VM: Win** on Win, **MacOS** on MacOS, **Lin** on Win, **Win** on Lin
  - **Ideal** - same system containers

Why do we need it ?

- **Easy** test piece of software
- **Easy** run different software versions at the same time
- **Easy** compatibility testing (e.g. new frontend + old backend)
- **Easy** to create automation tests
- **Easy** deploy piece of software
- **Fast**
- **Free**

Success stories

MetLife

- **-70%** VMs
- **-67%** Cores
- **10x** Average CPU Utilization
- **-66%** Cost Reduction

VISA

- **deploying patches in seconds vs days**

Success stories

PayPal

- **18** month
- **700+** apps
- **150000** containers
- **2x** build-test-deploy cycles speed up
- **20%** performance boost
- **0** code change

TASK: Test software without installation

VirtualMachine - isolated system

- Install OS
- Install needed environment
- Copy binary files in proper place
- Copy configuration
- Run if you can
- If something changed - repeat with **manual copy** with possible **human error**

TASK: Test software without installation

Docker - full environment isolation

- Write small script, more like configuration, which does all we need
  - Copy binary files in proper place
  - Copy configuration
  - Install needed environment
- Build image and run if you can
- If something changed - build image and run if you can. It's **automated** already

Installation

Docker Enterprise Edition vs Community Edition

- Based on the same opensource code - no difference in base functionality
- Support
- Images and plugins certifications
- Vulnerability scan

Linux Installation

- No problem, just install it
- Ordinary user should be added to the **docker** group
- **Note:** watch the space, */var/lib/docker*

Windows Installation

- **Docker for windows** needs **HyperV**, which means **All other Hypervisors will be disabled**
  - Nested virtualization for **VmWare** will do the trick (watch the version)
  - **VirtualBox** does not support nested virtualization
  - **Dual Boot** (with HyperV or without)

Windows Installation

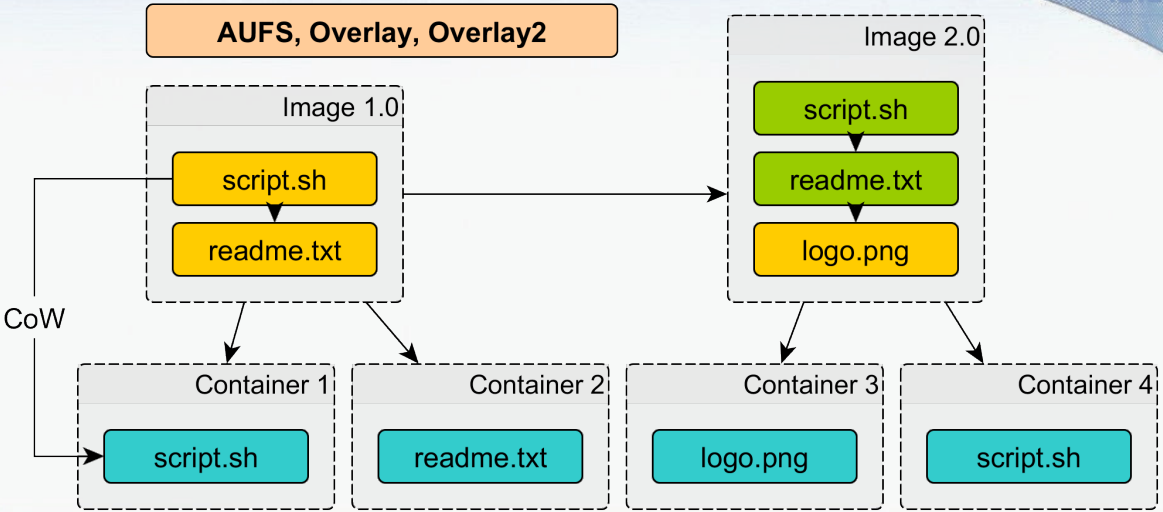
- Ordinary user should be added to **docker-users** group and **restart**
- **Note:** switch to proper containers type Linux or Windows
- **Note:** watch the space, change target directory

Docker fact I

Images vs Containers

- **Image** is buildable and static
- **Container** is runnable.
- We can run any number of **containers** out of the same **image**

Layered filesystem and Copy-on-Write



Docker fact II

Docker HUB

- Central images repository
- There are already **base images** available there
- You can register your **own images** for the **public**

Docker fact III

Docker has GUI

- Kitematic
- Dockstation
- Rancher
- Portainer
- Shipyard
- and many more

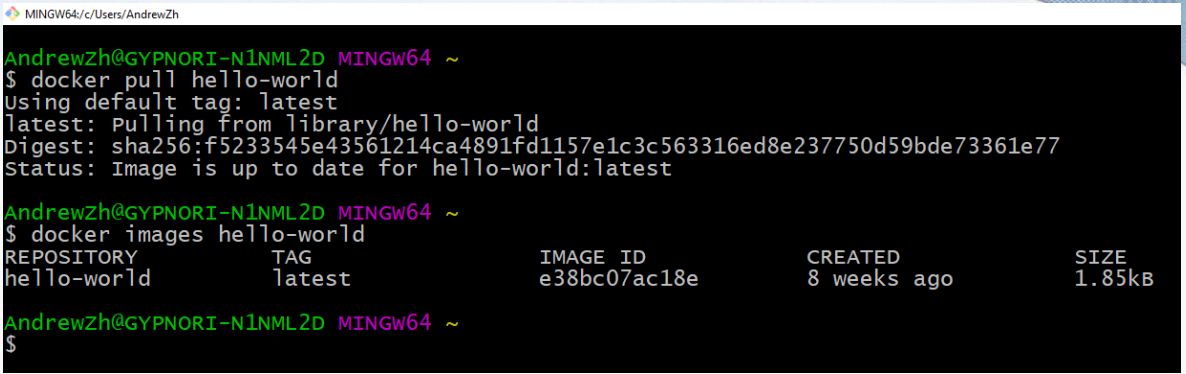
Piece a cake (Hello world!)

- **docker version** - get version, part of the sanity check
- **docker images** - list local images
- **docker pull** - get image from repository
- **docker run** - run container

Piece a cake (Hello world!)

- docker version
- docker images
- docker pull hello-world
- docker run hello-world

Piece a cake (Hello world!)



```
AndrewZh@GYPNORI-N1NML2D MINGW64 ~
$ docker run hello-world

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

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/engine/userguide/
```

```
AndrewZh@GYPNORI-N1NML2D MINGW64 ~
$ docker pull hello-world
Using default tag: latest
latest: Pulling from library/hello-world
bce2fbc256ea: Pulling fs layer
83eec61707e8: Pulling fs layer
eaac79658f14: Pulling fs layer
09da50837088: Pulling fs layer
09da50837088: Waiting
eaac79658f14: Verifying Checksum
eaac79658f14: Download complete
09da50837088: Verifying Checksum
09da50837088: Download complete
83eec61707e8: Verifying Checksum
83eec61707e8: Download complete
bce2fbc256ea: Verifying Checksum
bce2fbc256ea: Download complete
bce2fbc256ea: Pull complete
83eec61707e8: Pull complete
eaac79658f14: Pull complete
09da50837088: Pull complete
Digest: sha256:f5233545e43561214ca4891fd1157e1c3c563316ed8e237750d59bde73361e77
Status: Downloaded newer image for hello-world:latest

AndrewZh@GYPNORI-N1NML2D MINGW64 ~
$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
hello-world         latest              b1bf314a296b        7 weeks ago         1.11GB
```

Let's play (Minecraft)

Filesystem Tree

```
Minecraft
|-- Dockerfile
|-- minecraft_server.1.12.2.jar
```

Let's play (Minecraft)

Dockerfile

```
FROM java:8                              # Base image (mandatory)
ADD minecraft_server.1.12.2.jar /        # Copy file into image
RUN mkdir -p /opt/minecraft              # Create work directory
RUN echo "eula=true" > /opt/minecraft/eula.txt # Accept license
EXPOSE 25565                             # Expose network port
WORKDIR /opt/minecraft                   # Set work directory
# Set default command
CMD java -Xmx1024M -Xms1024M -jar /minecraft_server.1.12.2.jar nogui
```

Let's play (Minecraft)

- **docker build** - build image
- **docker run** - run container
- **docker ps** - get running containers
- **docker stop** - stop running container
- **docker rm** - remove stopped container
- **docker image rm** - remove image

Let's play (Minecraft)

- docker build -t minecraft .
- docker images
- docker run -d -p 25565:25565 --restart=always minecraft:latest
- docker ps
- docker stop <container id>
- docker rm <container id>
- docker image rm <image id>

Let's play (Minecraft)

```
AndrewZh@GYPNORI-N1NML2D MINGW64 /d/Doc/Presentations/Dockerfiles/Minecraft
$ docker build -t mc .
Sending build context to Docker daemon  30.22MB
Step 1/7 : FROM java:8
 ---> d23bdf5b1b1b
Step 2/7 : ADD minecraft_server.1.12.2.jar /
 ---> f53dd2a4190d
Step 3/7 : RUN mkdir -p /opt/minecraft
 ---> Running in 58705318a83a
Removing intermediate container 58705318a83a
 ---> 87cd6821e056
Step 4/7 : RUN echo "eula=true" > /opt/minecraft/eula.txt
 ---> Running in 02c85f81a5de
Removing intermediate container 02c85f81a5de
 ---> 730656b1b743
Step 5/7 : EXPOSE 25565
 ---> Running in 202a57d3c7b8
Removing intermediate container 202a57d3c7b8
 ---> 4f1d98a4ef9b
Step 6/7 : WORKDIR /opt/minecraft
Removing intermediate container faa5742c8da0
 ---> b4c036f54eb9
Step 7/7 : CMD java -Xmx1024M -Xms1024M -jar /minecraft_server.1.12.2.jar nogui
 ---> Running in ff26d4d4c1c6
Removing intermediate container ff26d4d4c1c6
 ---> 16bea97fa834
Successfully built 16bea97fa834
Successfully tagged mc:latest
SECURITY WARNING: You are building a Docker image from Windows against a non-Windows Docker host. All files and
ommended to double check and reset permissions for sensitive files and directories.
```

Do something useful (Chat Server)

Filesystem Tree

```
simplechat
|-- Dockerfile
|-- chat
    |-- icons
    |-- libs
    |-- LICENSE
    |-- README.md
    |-- resources
    |-- run.sh
    |-- sounds
```

Do something useful (Chat Server)

Dockerfile

```
FROM    debian:8                # Base image (mandatory)
RUN     apt-get update && apt-get install -y libboost1.55-all-dev
ENV     chatPath=/opt/chat      # Set environment variable
RUN     mkdir -p $chatPath      # Create work directory
COPY    chat        $chatPath   # Copy directory to work directory
WORKDIR $chatPath               # Set work directory
EXPOSE  8080/tcp                # Expose port
CMD     ./run.sh                # Set default command
```

Do something useful (Chat Server)

- docker build -t chat:0.1 .
- docker images
- docker run --name chat1 -p 8080:8080 --rm chat:0.1
- docker run --name chat2 -p 8081:8080 --rm chat:0.1
- docker ps
- docker stop <container id>
- docker rm <container id>
- docker image rm <image id>

Do something useful (Docker repository)

```
# Run own repository
docker run -d -p 5000:5000 --restart=always \
 --name registry registry:2
```

```
# Tag the image as localhost:5000/my-chat.
# This creates an additional tag for the existing image.
# When the first part of the tag is a hostname and port,
# Docker interprets this as the location of a registry,
# when pushing.
docker tag chat:0.1 localhost:5000/my-chat
# Push the image to the local registry running at localhost:5000
docker push localhost:5000/my-chat
```

Do something useful (Docker repository)

```
# Remove the locally-cached chat:0.1 and
# localhost:5000/my-chat images,
# so that you can test pulling the image from your registry.
# This does not remove the localhost:5000/my-chat image
# from your registry.
docker image remove chat:0.1
docker image remove localhost:5000/my-chat
# Pull the localhost:5000/my-chat image from your local registry.
docker pull localhost:5000/my-chat
```

Summary

So now we know:

- Why do we need it and cases we can't use it
- How to create docker image
- How to put the image into repository
- How to get the image from repository
- How to run the container

Materials

- www.docker.com
- https://github.com/ashlander/docker-lecture