
WORK
&CO

DOCKER IRL

Get familiar with Docker key concepts and basic usage.

Miloš Pavlićević
Developer at Work & Co
pavlicevic@work.co

OVERVIEW

- What is Docker?
- Docker architecture
- Why Docker?
- Usage & workflow
- Examples
- Best practices
- More!

WHAT IS DOCKER?

“It works on my machine!”

Docker engine

- Docker Daemon (in charge of building images and running/managing containers)
- Docker CLI (interfaces Docker API)
- Docker Registry

Architecture

- Client - Server (CLI - Daemon)
- Docker HUB (cloud service, library, storage, automation)

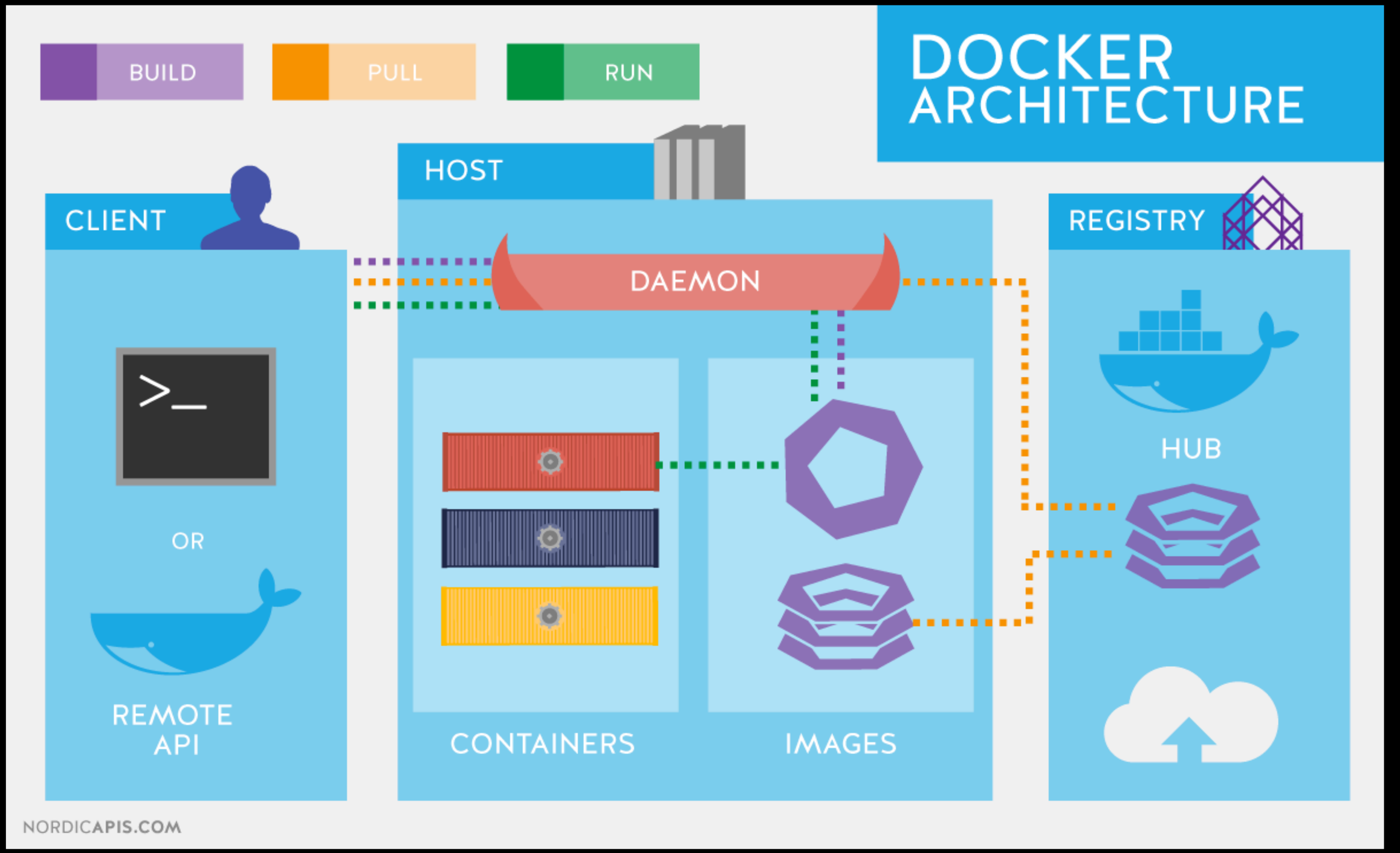
DOCKER ARCHITECTURE

- **Containers**
  Not VMs. The concept of resource allocation/isolation is present, however in a more efficient and less robust manner. (OS virtualization vs HW virtualization)

- **Images**
  A snapshot of container.

- **Container lifecycle**
  Create, start, pause, unpause, stop, kill, destroy.

- **Layers and container states**
  Each change to a container is treated as a separate layer. Each container layer can be converted into an image. Git-like approach.


BUILD
PULL
RUN
DOCKER ARCHITECTURE
HOST
CLIENT
REGISTRY
DAEMON
OR
HUB
REMOTE API
CONTAINERS
IMAGES
NORDICAPIS.COM

WHY DOCKER?

- Platform independence
- Lightweight, portable
- Simplicity - easy to install, operate and automate processes.
- Awesome for micro-services
- Huge community
- It’s free!

“3000+ Community contributors!”

USAGE & WORKFLOW

- **Commands**

```
docker ps [opts]   # lists containers
docker run [opts] <image_name>   # creates container based on image
docker attach <container_name>   # attach to container's process
docker exec [opts] <container_name>   # execute a command within container

# docker stop/start/pause/unpause/kill …

docker rm <container_name/id>   # removes specified (non-running) container
docker rmi <image_name/id>   # removes specified (currently unused) image

docker commit [opts] <container> [<repo>[:tag]]   # create an image
docker build [opts]   # create an image, based on Dockerfile

# docker login/push/pull …
```

- **Automation**

  Automate container control via scripts, webhooks, etc.

Time for example!

USAGE & WORKFLOW

- Dockerfiles

```
FROM httpd

MAINTAINER Milos Pavlicevic <pavlicevic@work.co>   #deprecated, use LABEL

COPY ./app/build/ /usr/local/apache2/htdocs/

CMD ["httpd-foreground"]
```

- Port mapping
- Volumes
- Debugging
- Docker Compose

More examples

Best practices:

Containers should be disposable and have a single responsibility.

Keep your stuff lean, by minimizing dependencies, layers, and don’t forget about .dockerignore.

```
# Bad
COPY file.tar.gz tmp/                       # new layer
RUN tar -xvf file.tar.gz                    # new layer
RUN rm file.tar.gz                          # new layer

RUN apt-get install -y package-one          # new layer
RUN apt-get install -y package-two          # new layer
RUN apt-get install -y package-three        # new layer

# ============================================
# Good
COPY file.tar.gz tmp/                       # new layer
RUN tar -xvf file.tar.gz \
    && rm file.tar.gz \
    && apt-get install \
    package-one \
    package-two \
    package-three                           # new layer
```

But also make your Dockerfiles readable.

```
# Use LABEL
LABEL "com.example.vendor"="ACME Incorporated"
LABEL com.example.label-with-value="foo"
LABEL version="1.0"
LABEL description="This text illustrates \
that label-values can span multiple lines."

# Sort arguments alphanumerically
RUN apt-get install -y \
  a-package \
  b-package \
  m-package \
  z-package

# Use EXPOSE
EXPOSE 27017
```

Make the most out of caching mechanism…

…but also beware of it.

```
# Bad
RUN apt-get update
RUN apt-get install -y my-package

# Good
RUN apt-get update && apt-get install -y \
    My-package-2

#============================================
# Bad
RUN git clone <git_repository>
RUN npm install

# Good
ADD https://api.github.com/repos/<user>/<repo>/git/refs/
heads/<branch> version.json
RUN git clone -b <branch> <git_repository>
```

Take care of logs.

Stay secure.

More:

- Docker Compose
- Swarm mode
- Venice

Thank you! Questions?

pavlicevic@work.co

www.work.co