



Dockerizing your Application with Docker



About Me

- Syah Dwi Prihatmoko (Moko)
- @sdmoko
- FOSS Enthusiast
- Pejuang GliB (<https://glibogor.or.id>)
- Cloud Gladiator at Btech (<https://btech.id>)
- GNOME-ID (<https://gnome.id>)
- NolSatu (<https://nolsatu.id>)
- Endless Ambassador (<https://endlessos.com>)

References

- Docker Documentation <https://docs.docker.com/>
- Mastering Docker Second Edition - Russ McKendrick, Scott Gallagher – Packt Publishing – July 2017

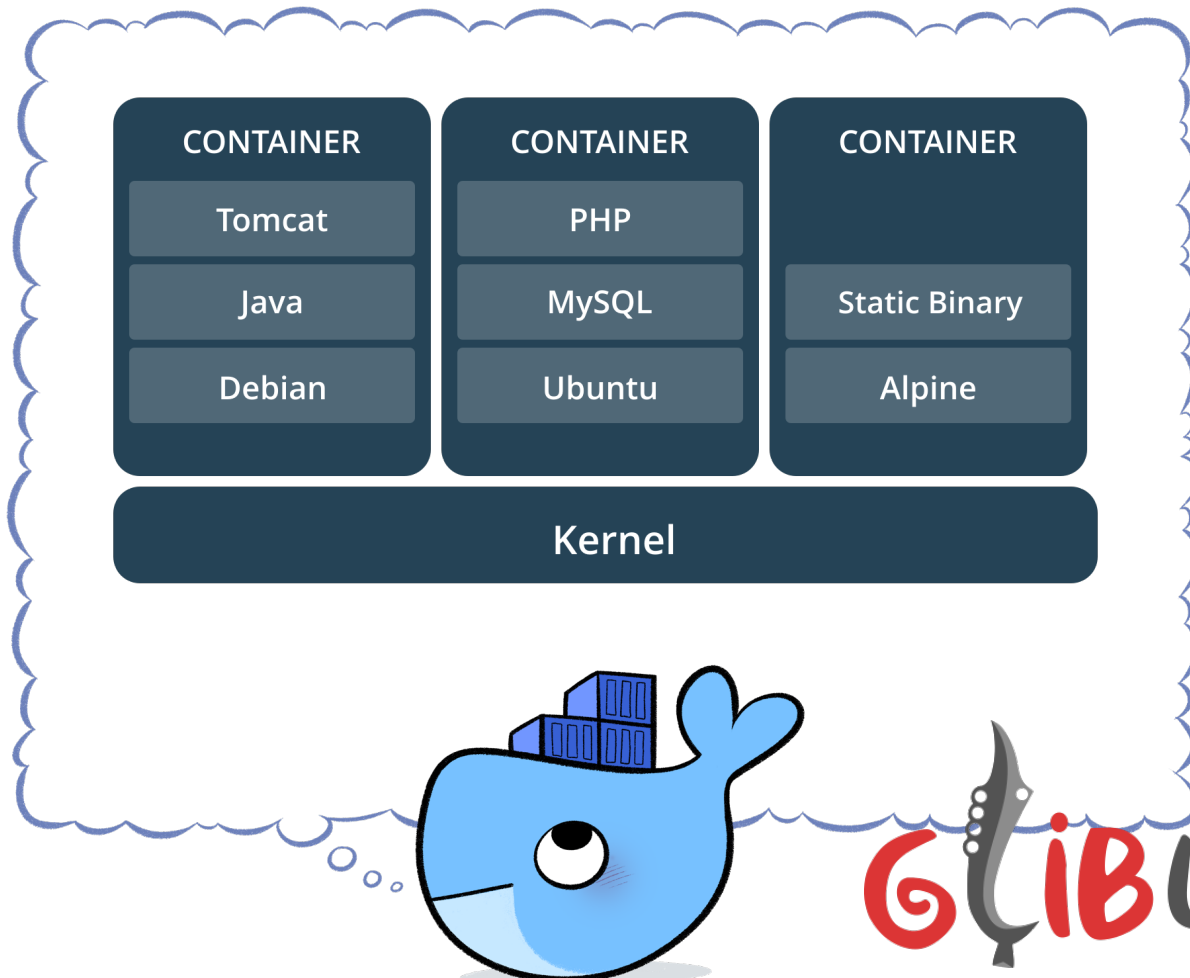




Containers & VMs

Containers

“a lightweight, stand-alone, executable package of a piece of software that includes everything needed to run it: code, runtime, system tools, system libraries, settings.”



Why Containers? Lightweight

- Containers running on a single machine share that machine's operating system kernel; they start instantly and use less compute and RAM.
- Images are constructed from filesystem layers and share common files. This minimizes disk usage and image downloads are much faster.



Why Containers? Standard

- Containers are based on open standards and run on all major Linux distributions, Microsoft Windows, and on any infrastructure including VMs, bare-metal and in the cloud.

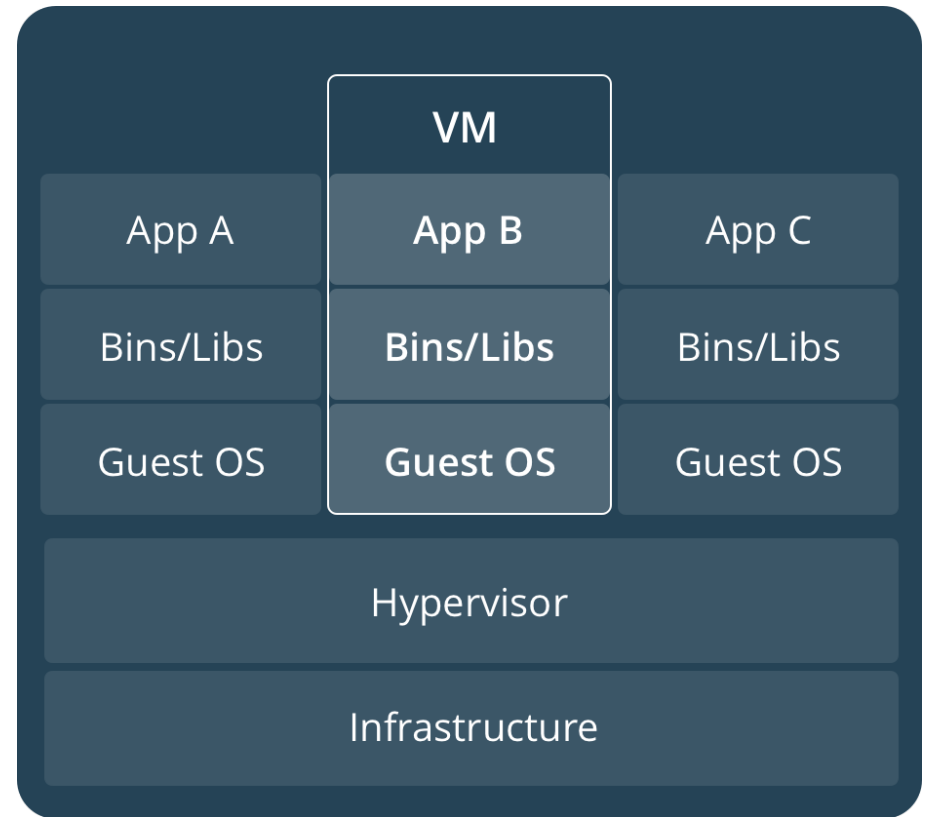
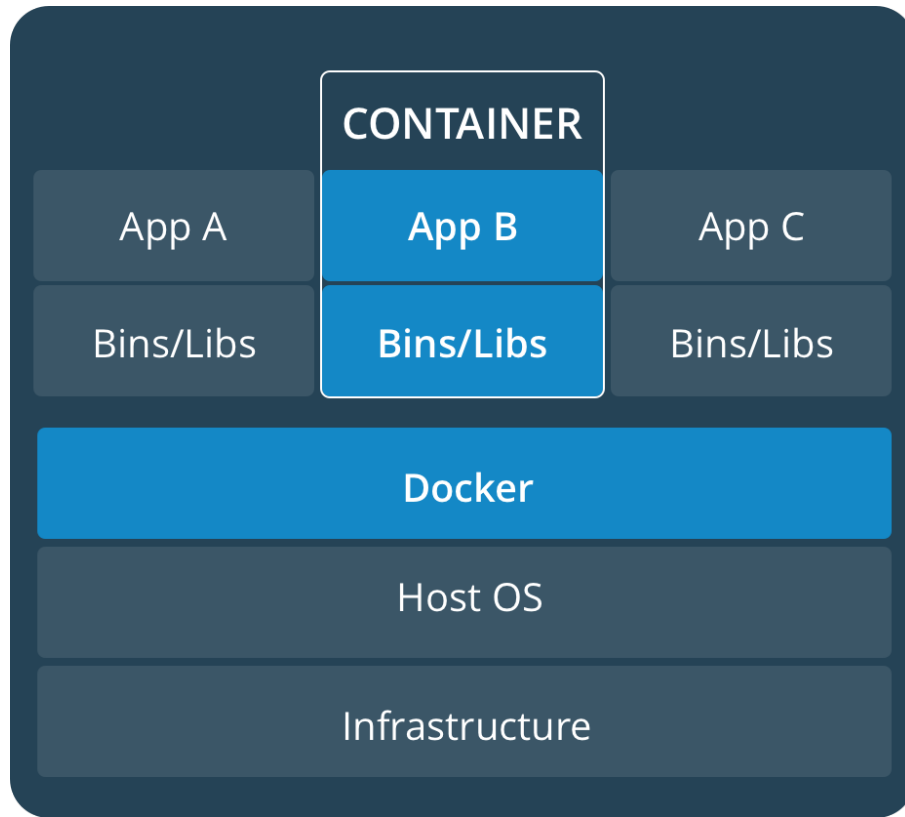


Why Containers? Secure

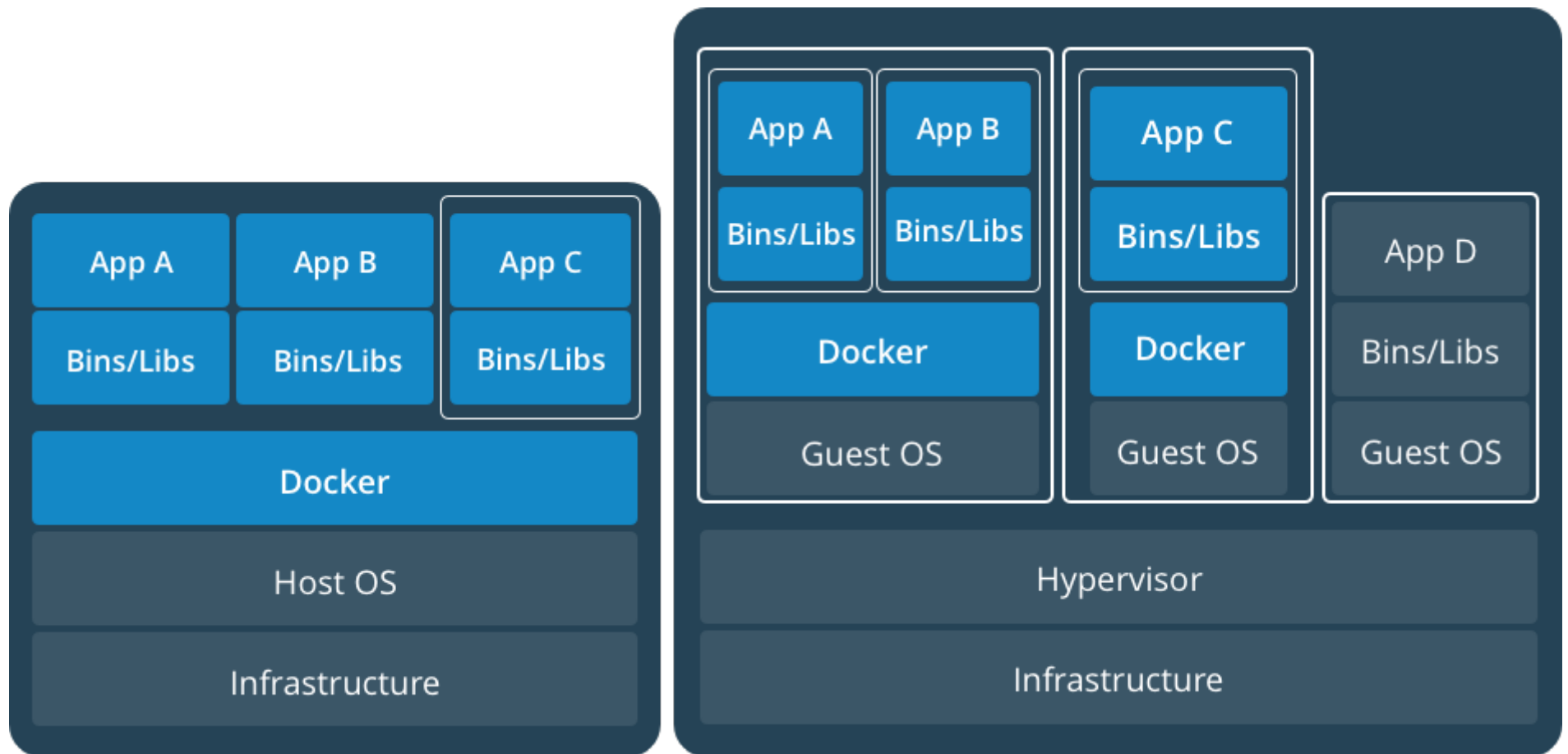
- Docker containers isolate applications from one another and from the underlying infrastructure. Docker provides the strongest default isolation to limit app issues to a single container instead of the entire machine.



Comparing Containers & VMs (1)



Containers & VMs Together





Install Docker



Docker Release Schedule

Starting with Docker 17.03, Docker uses a time-based release schedule.

- Docker CE Stable releases generally happen quarterly, with patch releases as needed.
- Docker EE releases generally happen twice per year, with patch releases as needed.

Updates, and patches

- A given Docker EE release receives patches and updates for at least one year after it is released.
- A given Docker CE Stable release receives patches and updates for one month after the next Docker CE Stable release.

Docker OS

DOCKER CE

Platform	x86_64 / amd64
CentOS	✓
Debian	✓
Fedora	✓
Ubuntu	✓

DOCKER EE

Platform	x86_64 / amd64
CentOS	✓
Oracle Linux	✓
Red Hat Enterprise Linux	✓
SUSE Linux Enterprise Server	✓
Ubuntu	✓
Microsoft Windows Server 2016	✓



Install Docker

- CentOS

`yum -y install docker`

- Ubuntu

`apt -y install docker.io`



First Docker Commands

```
## List Docker CLI commands
```

```
docker
```

```
docker container --help
```

```
## Display Docker version and info
```

```
docker --version
```

```
docker version
```

```
docker info
```

```
## Execute Docker image
```

```
docker run hello-world
```

```
## List Docker images
```

```
docker image ls
```

```
## List Docker containers (running, all, all in quiet mode)
```

```
docker container ls
```


```
docker container ls --all
```

```
docker container ls -aq
```



Images

Docker Hub

[Explore](#) [Help](#)

[Log In](#)

Build, Ship, and Run Any App, Anywhere

Dev-test pipeline automation, 100,000+ free apps, public and private registries

New to Docker Hub?

Create your free account now. No credit card required.

username

email

password

[Sign Up](#)

Join Docker Hub

Automate Build-Test Pipelines

Images with the latest updates, continuously integrated and available.

Collaborate As A Team

Role-based access control for easy sharing.

Assemble Apps

Free Official Repos available as initial building blocks.

Explore Official Repositories

 **redis**

 **ubuntu**

 **mongoDB**

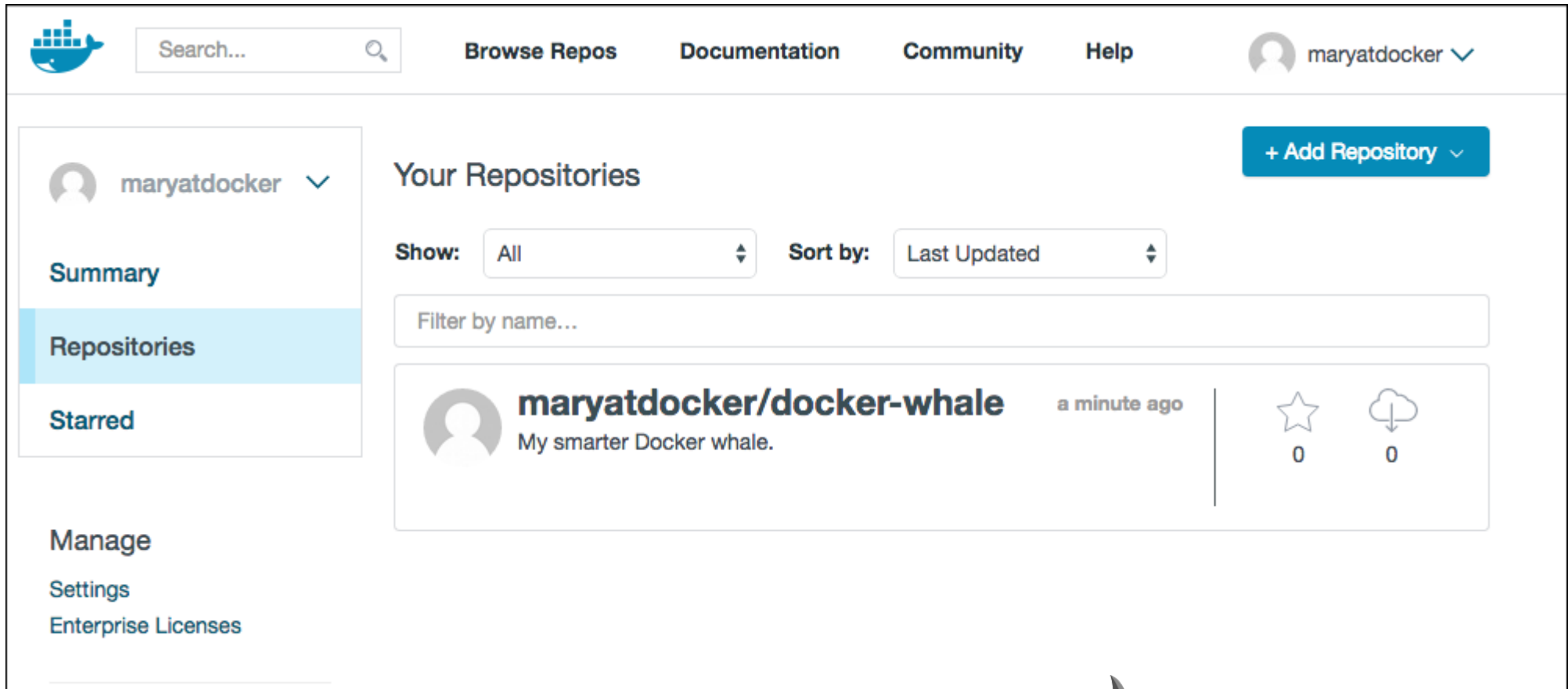
 **node**

 **WordPress**

[See all official repositories](#)

© 2015 Docker Inc.

Image Repositories



The screenshot displays the Docker Hub web interface for a user named 'maryatdocker'. The top navigation bar includes the Docker logo, a search bar, and links for 'Browse Repos', 'Documentation', 'Community', and 'Help'. The user's profile 'maryatdocker' is shown in the top right. On the left sidebar, the 'Repositories' tab is selected, with other options like 'Summary', 'Starred', 'Manage', 'Settings', and 'Enterprise Licenses'. The main content area is titled 'Your Repositories' and features a '+ Add Repository' button. Below this, there are filters for 'Show' (set to 'All') and 'Sort by' (set to 'Last Updated'). A search bar labeled 'Filter by name...' is present. The repository list shows one entry: 'maryatdocker/docker-whale', described as 'My smarter Docker whale.', with a timestamp of 'a minute ago'. To the right of the repository name are icons for stars and downloads, both showing a count of 0.

maryatdocker

Search...

Browse Repos Documentation Community Help

+ Add Repository

Your Repositories

Show: All Sort by: Last Updated

Filter by name...

maryatdocker/docker-whale a minute ago

My smarter Docker whale.

0 0

Summary Repositories Starred

Manage Settings Enterprise Licenses



Dockerfile

A **Dockerfile** is a text document that contains all the commands a user could call on the command line to assemble an image. Using **docker build** users can create an automated build that executes several command-line instructions in succession.



Dockerfile instructions (1)

- **FROM**, initializes a new build stage and sets the Base Image for subsequent instructions.
- **RUN**, execute any commands in a new layer on top of the current image and commit the results.
- **CMD**, provide defaults for an executing container.
- **LABEL**, adds metadata to an image.
- **EXPOSE**, informs Docker that the container listens on the specified network ports at runtime.
- **ENV**, sets the environment variable <key> to the value <value>.





Dockerfile instructions (3)

- **USER**, sets the user name (or UID) and optionally the user group (or GID) to use when running the image and for any RUN, CMD and ENTRYPOINT instructions that follow it in the Dockerfile.
- **WORKDIR**, sets the working directory for any RUN, CMD, ENTRYPOINT, COPY and ADD instructions that follow it in the Dockerfile.
- **ARG**, defines a variable that users can pass at build-time to the builder with the docker build command using the --build-arg <varname>=<value> flag.
- **ONBUILD**, adds to the image a trigger instruction to be executed at a later time, when the image is used as the base for another build.

Dockerfile instructions (4)

- **STOPSIGNAL**. sets the system call signal that will be sent to the container to exit.
- **HEALTHCHECK**, tells Docker how to test a container to check that it is still working.
- **SHELL**, allows the default shell used for the shell form of commands to be overridden.



Dockerfile Example

```
# Firefox over VNC
#
# VERSION                                0.3

FROM ubuntu

# Install vnc, xvfb in order to create a 'fake' display and firefox
RUN apt-get update && apt-get install -y x11vnc xvfb firefox
RUN mkdir ~/.vnc
# Setup a password
RUN x11vnc -storepasswd 1234 ~/.vnc/passwd
# Autostart firefox (might not be the best way, but it does the trick)
RUN bash -c 'echo "firefox" >> ~/.bashrc'

EXPOSE 5900
CMD ["x11vnc", "-forever", "-usepw", "-create"]
```



Dockerizing Application

Clone atau Fork dan Clone:

<https://github.com/sdmoko/2048>

Create Dockerfile

Docker Build

Docker Run





Q&A



Terima Kasih