



# 台北 docker 一日入門篇

Philipz(鄭淳尹)

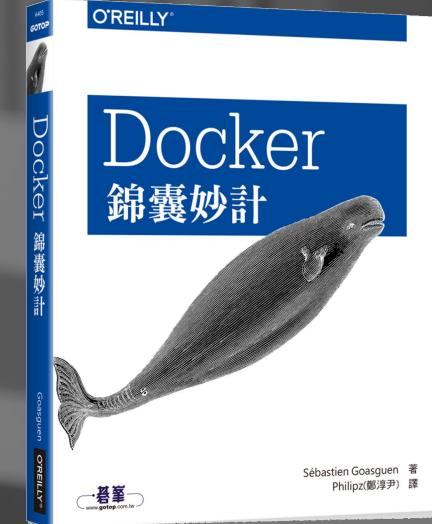
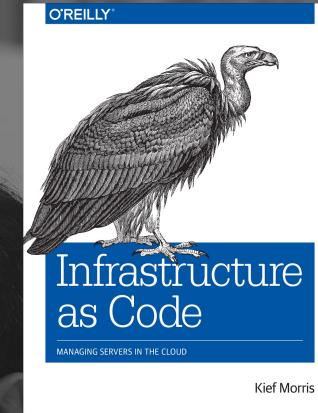
2017-05-21

東吳大學數學系

# Philipz (鄭淳尹)



Docker.Taipei 共同發起人



歐萊禮《Docker 錦囊妙計》譯者

碁峰《Docker入門與實戰》、

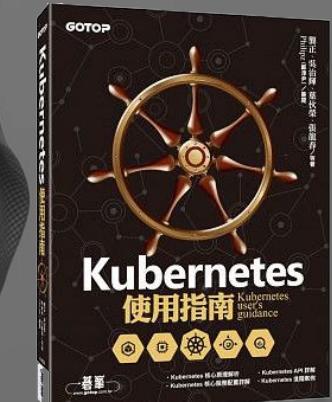
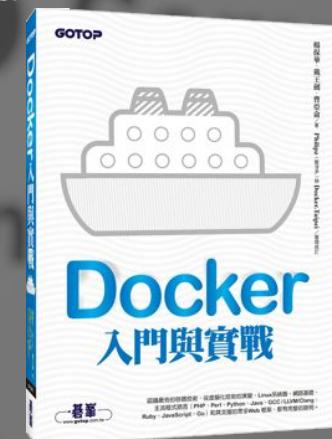
《Kubernetes使用指南》審譯者

2014 COSCUP/iThome Summit 講者

2015 Microsoft Azure 開發者大會 講者

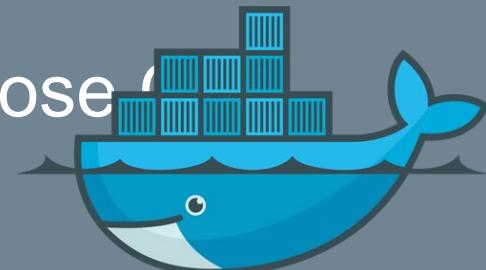
2016 COSCUP Docker 進階工作坊

2016 義守大學資工系 Docker 研習營

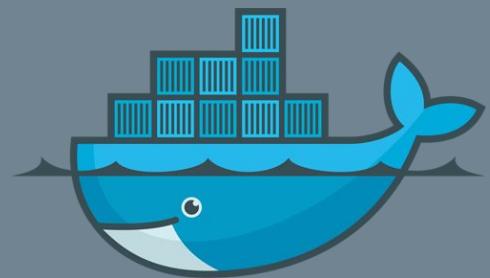


# Today Topics

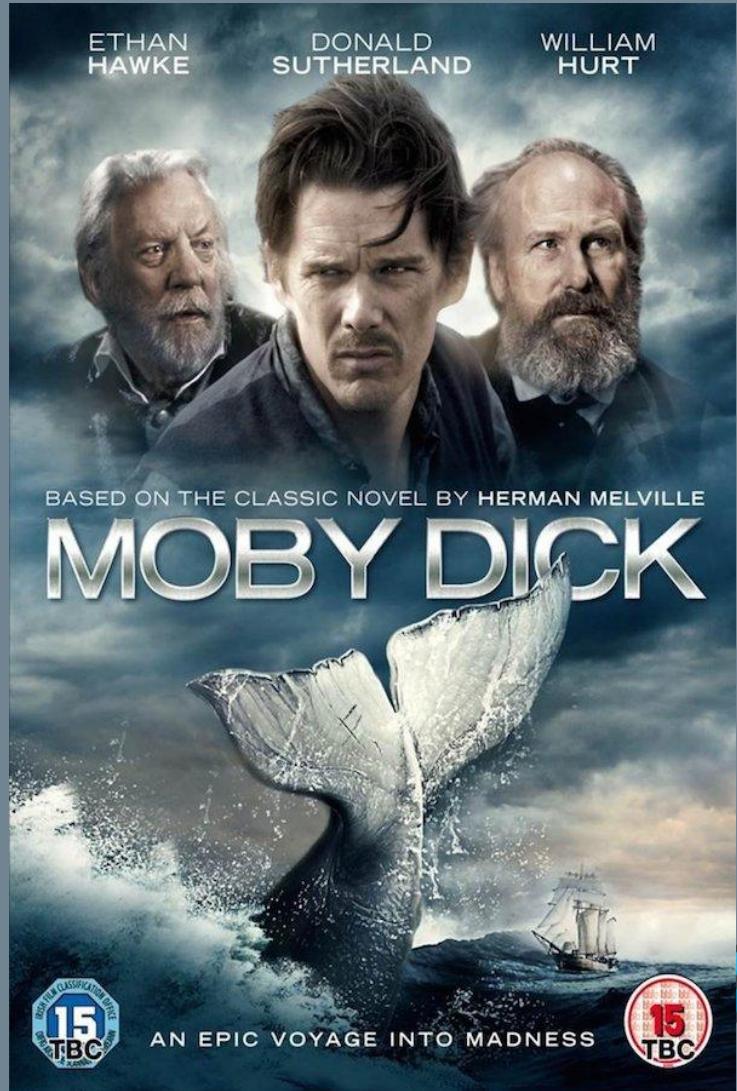
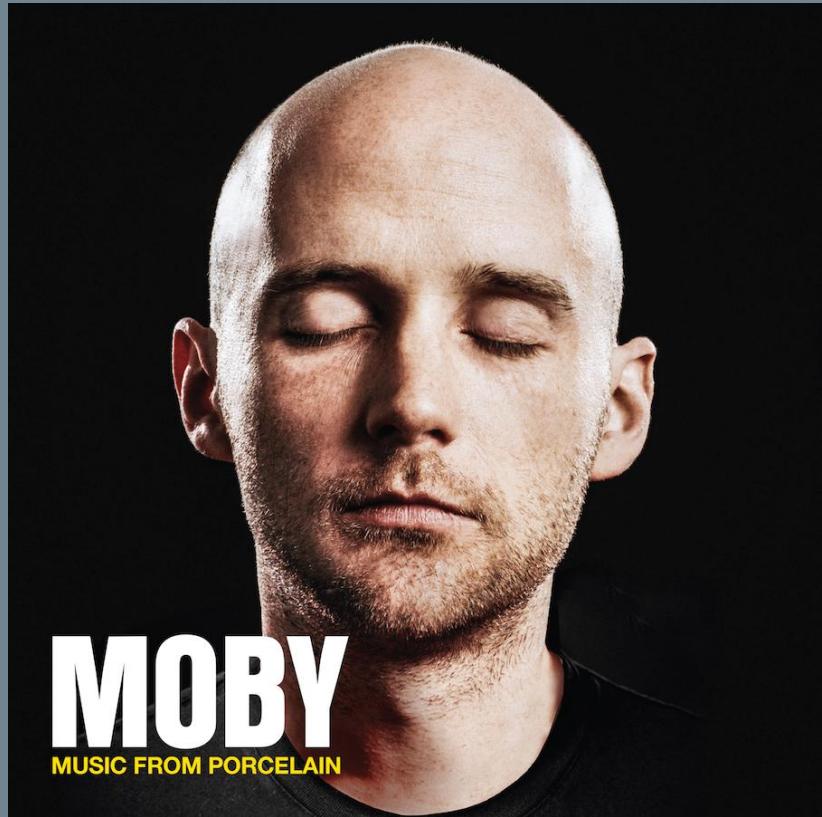
1. The differents between VMs and Container,  
Container lifecycle.
2. Docker ecosystem tools
3. Linux CLI, Docker CLI
4. Using Docker Engine
5. Docker Hub intoduction
6. Docker image & Docker hub autobuild
7. Deploy Docker image to **Azure PaaS**
8. Docker Network CLI & Docker Compose
9. Using Docker Compose



0. Docker now is called  
Moby



# There are a lot of Moby



[Blog](#) & [Home](#) description

**containerd**

**runc**

**swarmKIT**

**LINUXKIT**

**infrakit**

**Notary**

**Registry**

**LibNetwork**

**VPNKit**

**DataKit**

**HyperKit**

**Compose**

**GRPC**



Component Library

Orchestration

Image mgmt

Secret mgmt

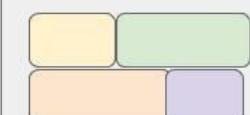
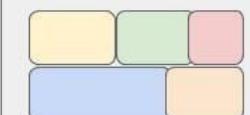
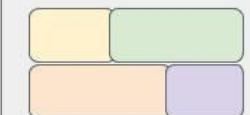
Config mgmt

Networking

Provisioning

Your Component Here

Assemblies



**moby tools**



Upstream *projects*



Your Product Here :)



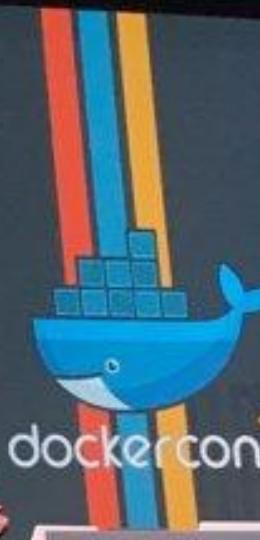
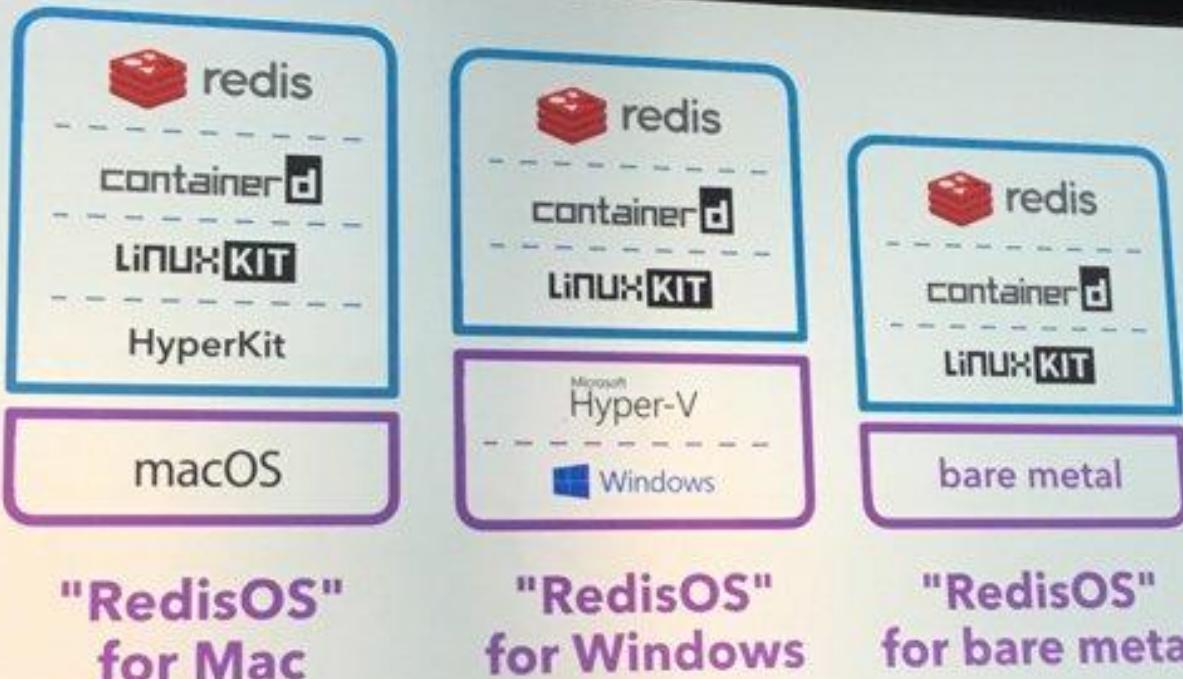
Docker CE



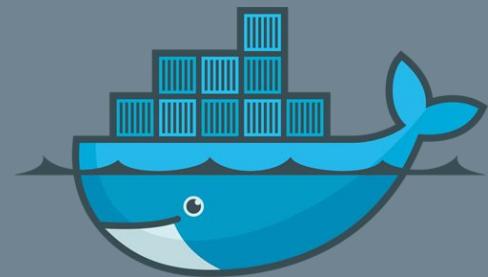
Docker EE



Downstream *products*

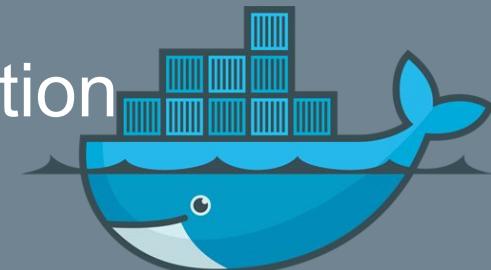


# 1. Compare VM with Container



# Virtualization History

- IBM zOS
- Virtual Hardware - VMware, KVM, Xen, VirtualBox
- Hardware-assisted virtualization
- Paravirtualization
- OS-level virtualization
  - a. OpenVZ
  - b. LXC
  - c. Docker
- IaaS, PaaS, SaaS - Snapshot, Migration



# The Martix of Hell

		?	?	?	?	?	?
		?	?	?	?	?	?
		?	?	?	?	?	?
		?	?	?	?	?	?
		?	?	?	?	?	?
		?	?	?	?	?	?
							

# A Brief History of Containers

1979: Unix V7

2000: FreeBSD Jails

2005: OpenVZ

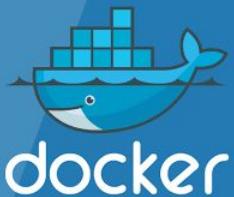
2008: LXC

2013: LMCTFY

2013: Docker

2016: Windows Container

From: [A Brief History of Containers: From 1970s chroot to Docker 2016](#)



# Build, Ship, Run, Any App Anywhere

From Dev



To Ops



Any App



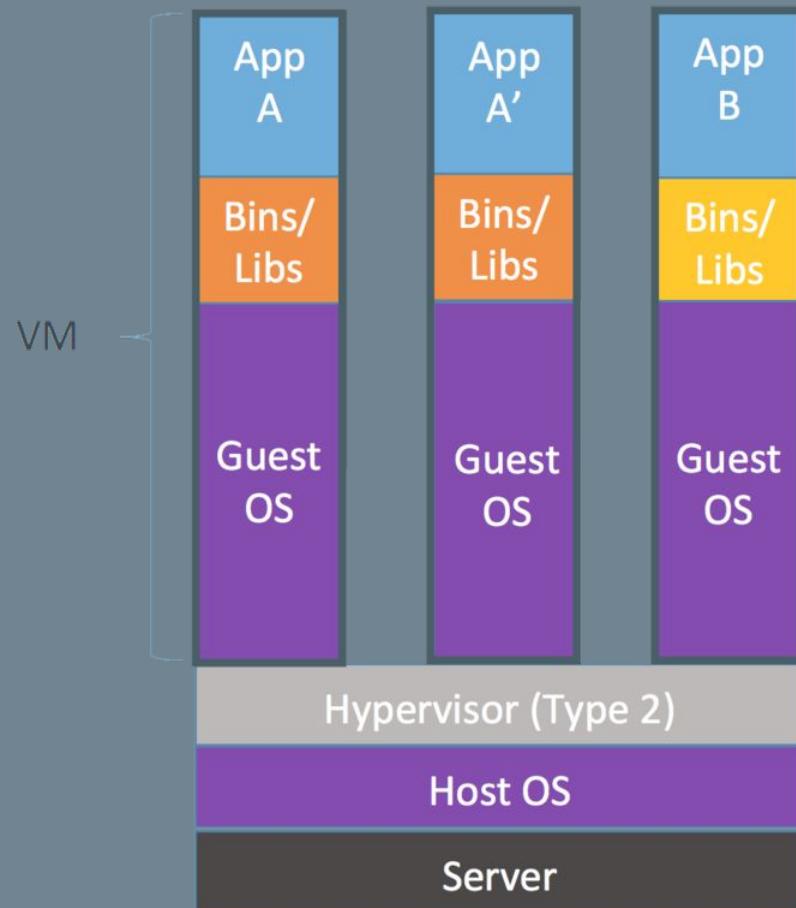
Any OS



Anywhere



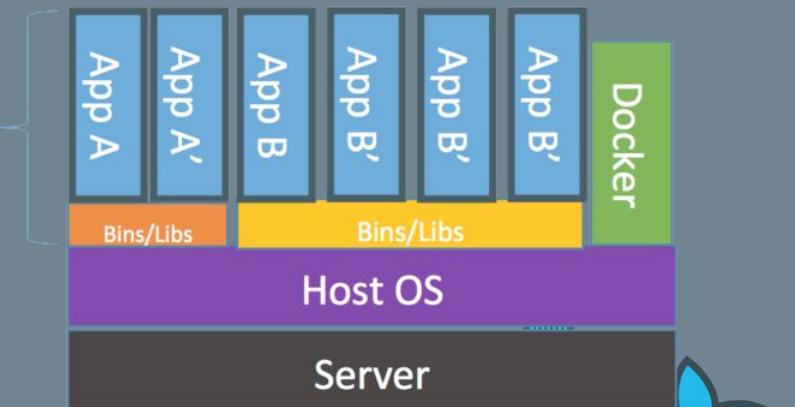
# Containers vs. VMs



Containers are isolated,  
but share OS and, where  
appropriate, bins/libraries

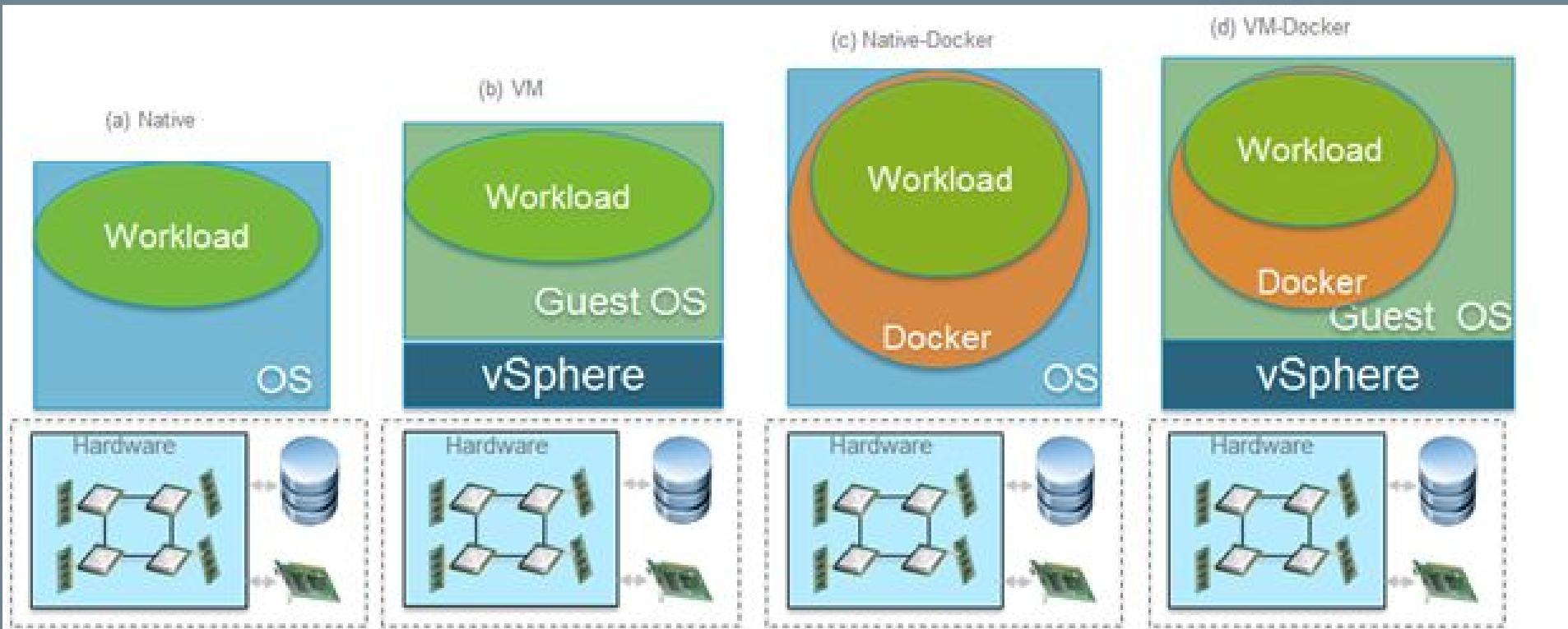
...result is significantly faster deployment,  
much less overhead, easier migration,  
faster restart

Container

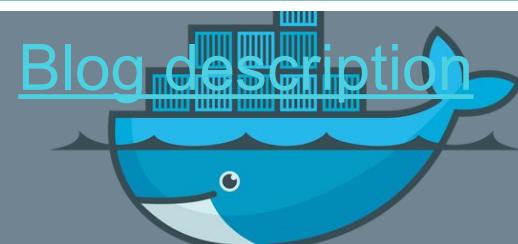
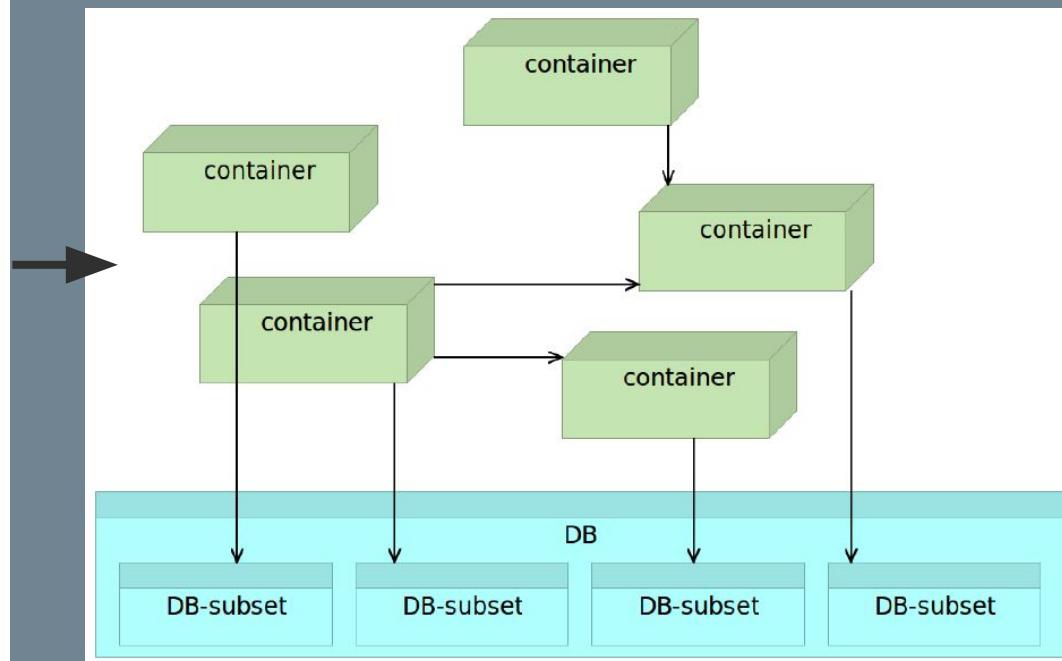
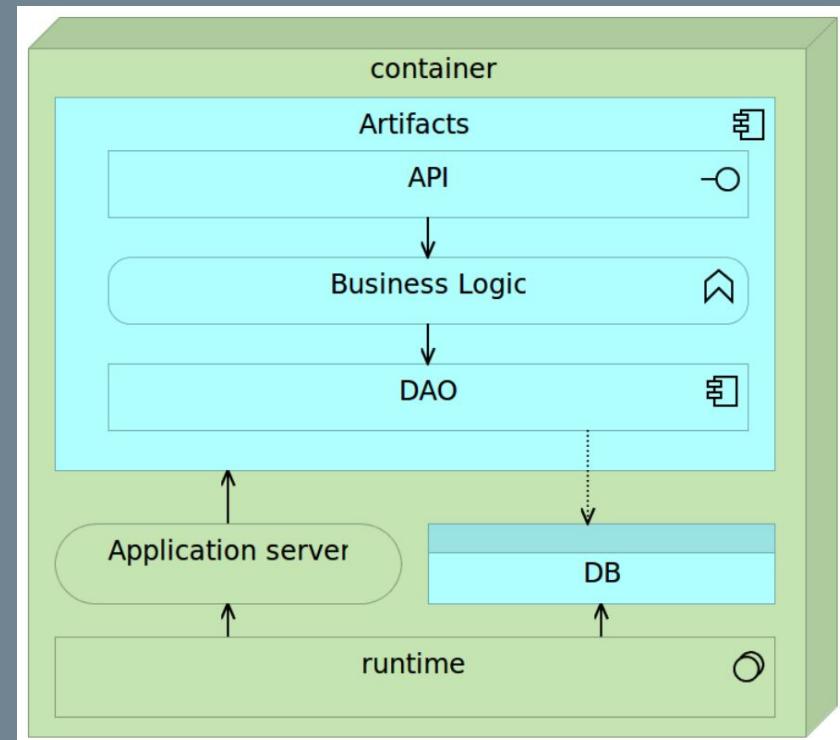


[Blog description](#)

# Containers vs. VMs



# Containers are not VMs



# Container Principle

Real Container

One Container

One Customer

One Commodity

Software Container

One Container

One Process

## The Box

How the Shipping Container Made the World Smaller and the World Economy Bigger



箱子 貨櫃造就的全球貿易與現代經濟生活

沒有它，就沒有全球化。沒有沃爾瑪，甚至沒有高科技。  
貨櫃船的運費降低後，意外生產變成最大贏家。  
它改變了我們的生活，也改變了世界。  
從世界各地為我們帶來各種無法想像的低價商品。

英國《金融時報》年度最佳選書

陳思寬 台大國企系教授 陳國棟 中央研究院人文社會科學研究中心

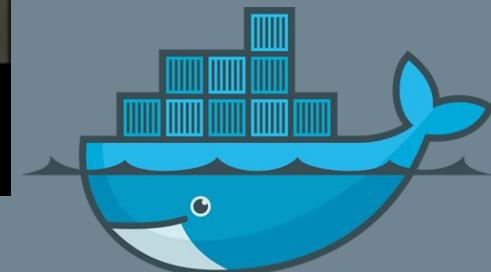
盧峯海 陽明海運董事長 陳柏廷 萬海航運董事長

專文導讀  
聯合推薦

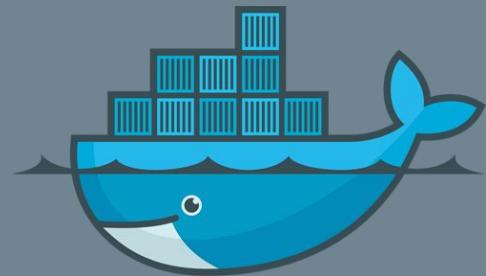


Malcom McLean

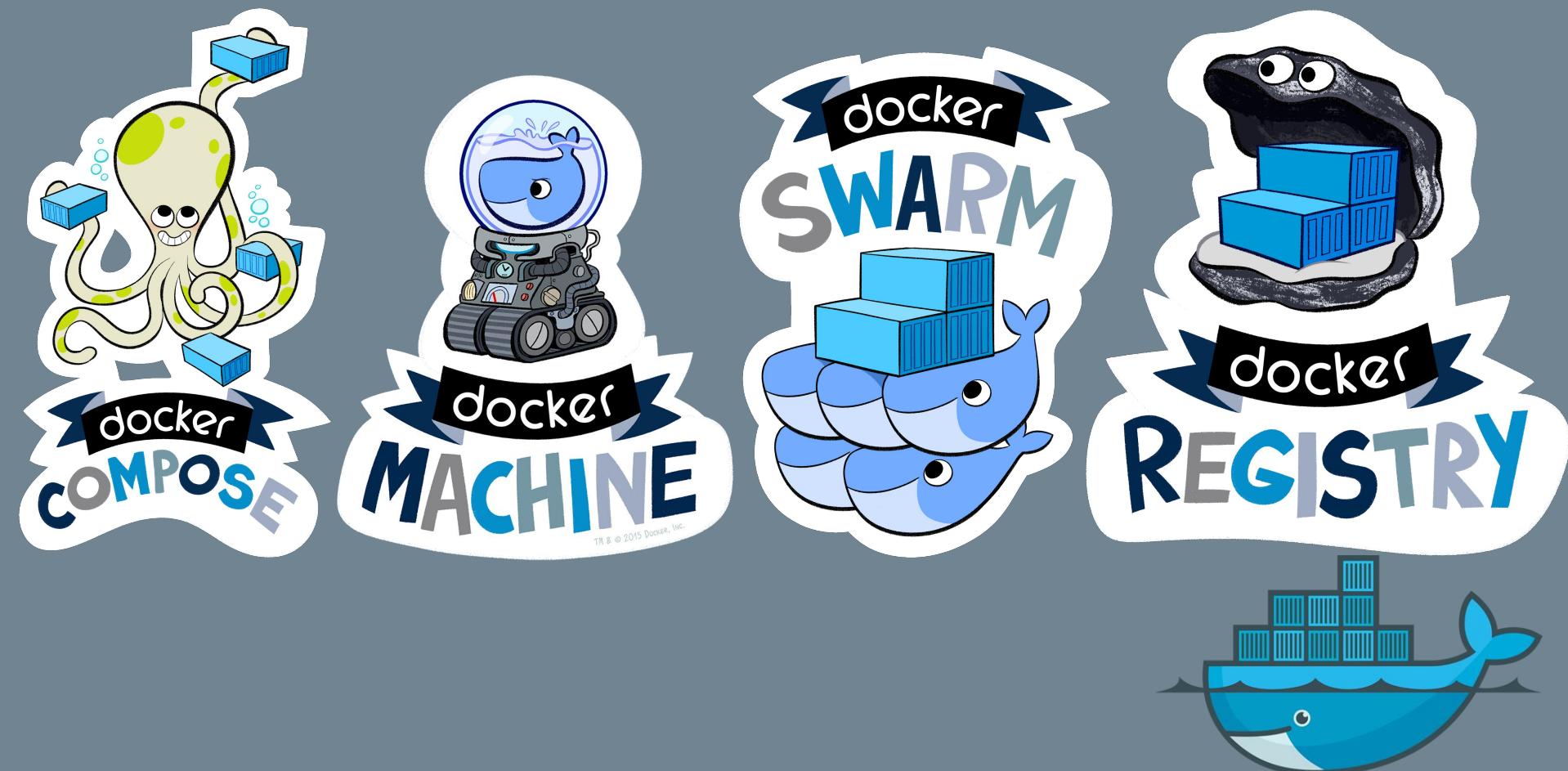
# Adolf Hitler & Docker



## 2. Docker ecosystem tools



# Docker Tools



# Still No Silver Bullet

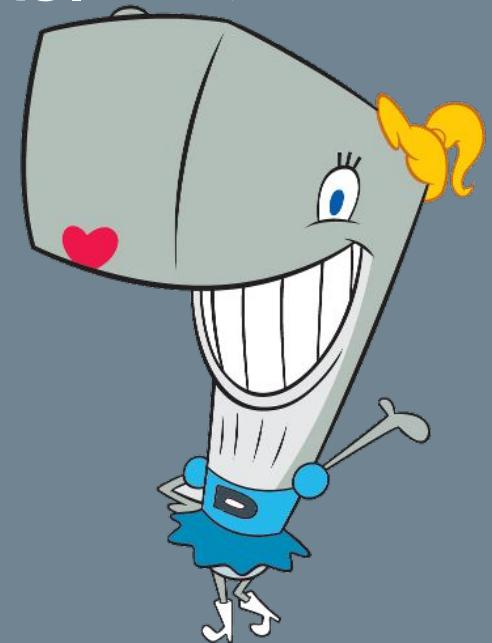
Container is **one** key element, not all.

DevOps pipeline process

**Microservices**, or other service stacks.

Infrastructure as Code

Business model



# The Docker Stack

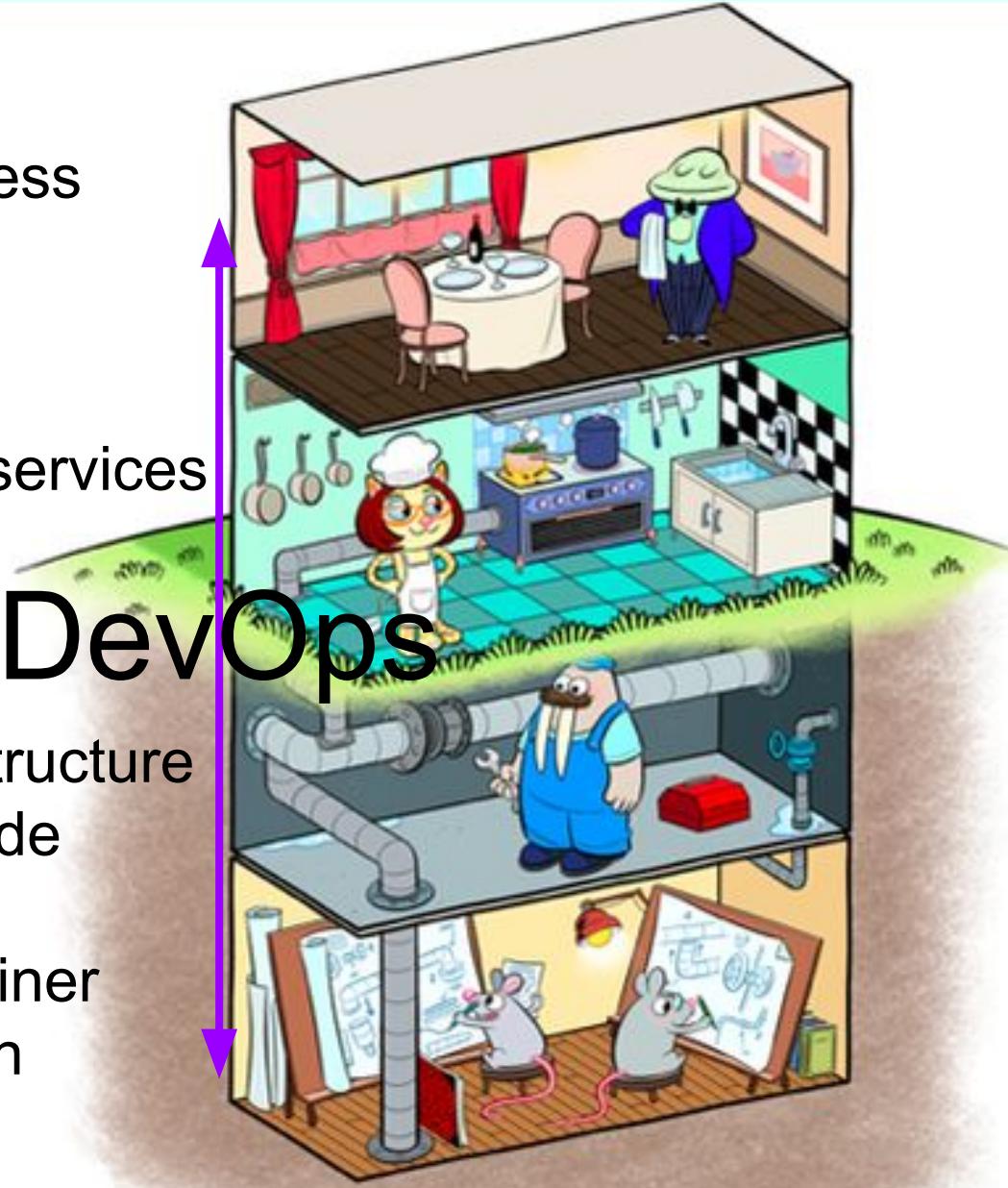
Business  
model

Microservices

DevOps

Infrastructure  
as Code

Container  
Design



# The Docker Stack

基礎架構  
即程式碼

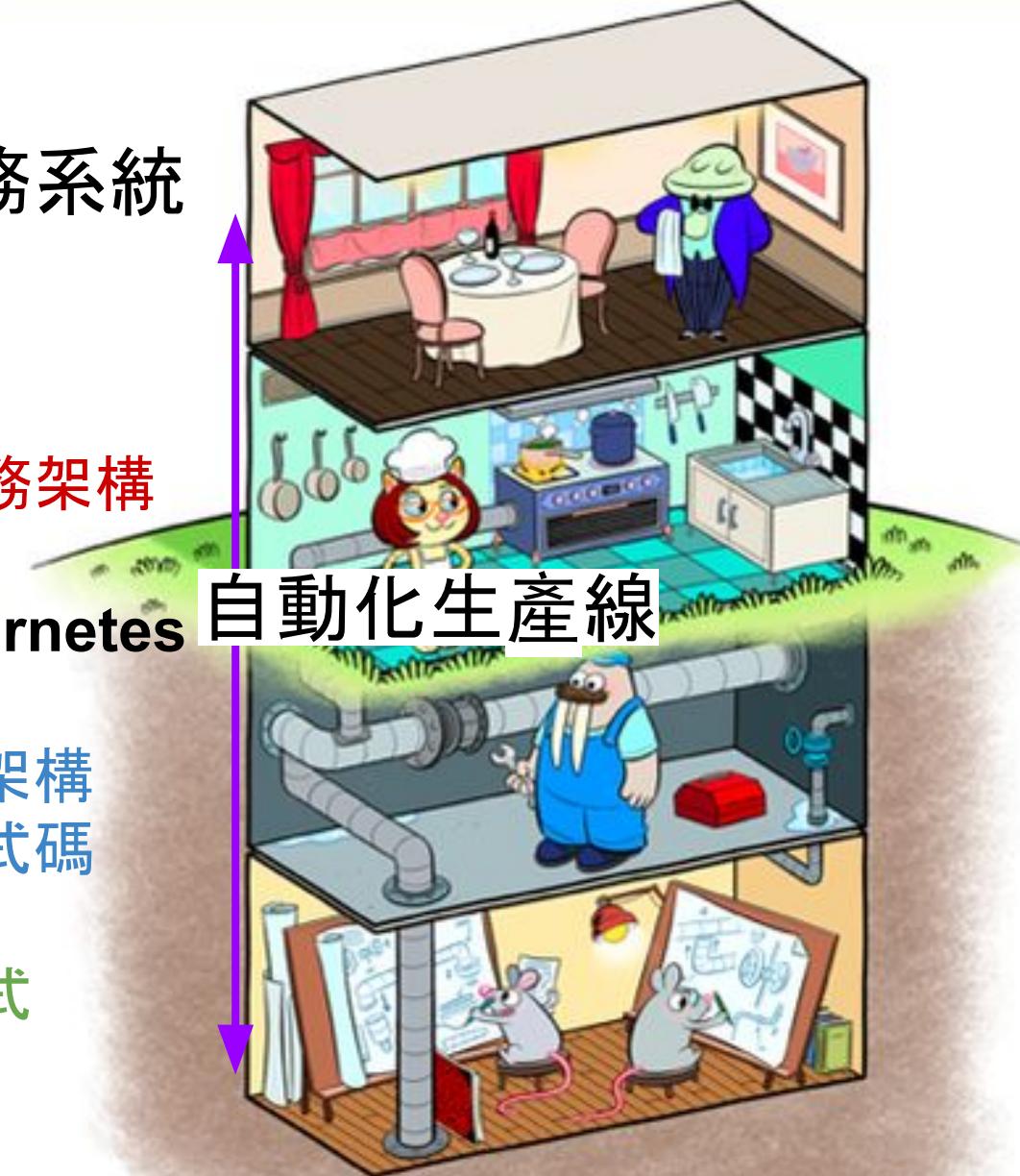
容器式  
設計

微服務架構

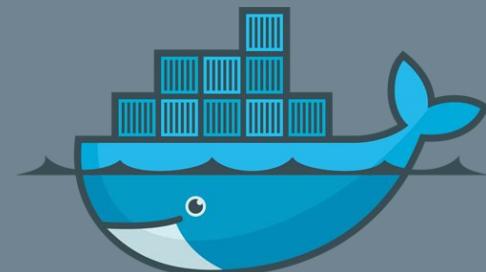
Kubernetes

\*業務系統

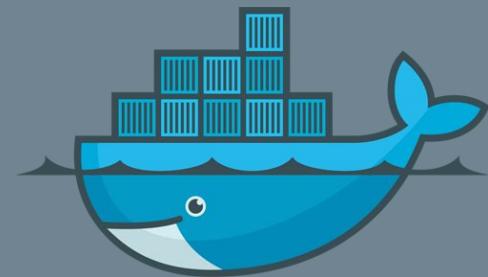
自動化生產線



# Docker Datacenter



## 3.1 Linux command-line

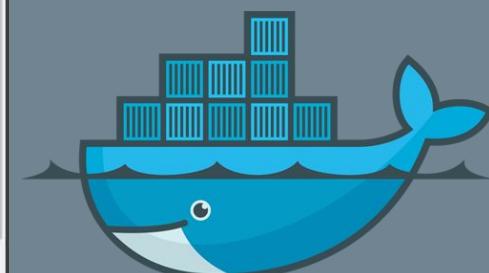


# Microsoft Azure

<https://portal.azure.com/>

The screenshot shows the Microsoft Azure portal interface. On the left, there's a sidebar with various service icons and a search bar labeled "搜尋 Marketplace". Below the search bar, the "MARKETPLACE" section is expanded, showing categories like "計算", "網路", "Storage", "Web + 行動", "Databases", "Intelligence + analytics", "物聯網", "Enterprise Integration", "安全性 + 識別", "Developer tools", and "Monitoring + management". To the right, the main area is titled "計算" and displays a list of selected applications from the marketplace. Each item includes a thumbnail, the application name, and a brief description. The first three items are: "Windows Server 2012 R2 Datacenter", "Windows Server 2016 Datacenter", and "Red Hat Enterprise Linux 7.2". The fourth item, "Ubuntu Server 16.04 LTS", is partially visible at the bottom.

精選應用程式	查看全部
Windows Server 2012 R2 Datacenter Enterprise-class solutions that are simple to deploy, cost-effective,	<a href="#">查看全部</a>
Windows Server 2016 Datacenter Enterprise-class solutions that are simple to deploy, cost-effective,	<a href="#">查看全部</a>
Red Hat Enterprise Linux 7.2 Red Hat Enterprise Linux 7 is the world's leading enterprise Linux platform built to meet the needs	<a href="#">查看全部</a>
Ubuntu Server 16.04 LTS Ubuntu Server delivers the best value scale-out performance available.	<a href="#">查看全部</a>





## FILE COMMANDS

- ⑥ ls - directory listing
- ⑥ ls -al - formatted listing with hidden files
- ⑥ cd dir - change directory to dir
- ⑥ cd - change to home
- ⑥ pwd - show current directory
- ⑥ mkdir dir - create a directory dir
- ⑥ rm file - delete file
- ⑥ rm -r dir - delete directory dir
- ⑥ rm -f file - force remove file
- ⑥ rm -rf dir - force remove directory dir \*
- ⑥ cp file1 file2 - copy file1 to file2
- ⑥ cp -r dir1 dir2 - copy dir1 to dir2; create dir2 if it doesn't exist
- ⑥ mv file1 file2 - rename or move file1 to file2  
if file2 is an existing directory, moves file1 into directory file2
- ⑥ ln -s file link - create symbolic link to file
- ⑥ touch file - create or update file
- ⑥ cat > file - places standard input into file
- ⑥ more file - output the contents of file
- ⑥ head file - output the first 10 lines of file
- ⑥ tail file - output the last 10 lines of file
- ⑥ tail -f file - output the contents of file as it grows, starting with the last 10 lines



## SHORTCUTS

- ⑥ Ctrl+C
- ⑥ Ctrl+Z
- fg in t
- ⑥ Ctrl+D
- ⑥ Ctrl+W
- ⑥ Ctrl+U
- ⑥ Ctrl+R
- ⑥ !! - re
- ⑥ exit -



## SYSTEM

- ⑥ date
- ⑥ cal -
- ⑥ uptime
- ⑥ w - d
- ⑥ whoami
- ⑥ finger
- ⑥ uname
- ⑥ cat /
- ⑥ cat /
- ⑥ man c
- ⑥ df -
- ⑥ du -
- ⑥ free

which



## SEARCHING

- ⑥ grep pattern files - search for pattern in files
- ⑥ grep -r pattern dir - search recursively for pattern in dir
- ⑥ command | grep pattern - search for pattern in the output of command



COMP



## PROCESS MANAGEMENT

- ⑥ ps - display your currently active processes
- ⑥ top - display all running processes
- ⑥ kill pid - kill process id pid
- ⑥ killall proc - kill all processes named proc  
(use with extreme caution)
- ⑥ bg - lists stopped or background jobs; resume a stopped job in the background
- ⑥ fg - brings the most recent job to foreground
- ⑥ fg n - brings job n to the foreground

⑥ tar  
file  
⑥ tar  
⑥ tar  
⑥ Gzip  
⑥ tar  
⑥ tar  
comp  
⑥ tar  
⑥ gzip  
file  
⑥ gzip  
file

- ⌚ !! - repeats the last command
- ⌚ exit - log out of current session

it



## SYSTEM INFO

- ⌚ date - show the current date and time
- ⌚ cal - show this month's calendar
- ⌚ uptime - show current uptime
- ⌚ w - display who is online
- ⌚ whoami - who you are logged in as
- ⌚ finger user - display information about user
- ⌚ uname -a - show kernel information
- ⌚ cat /proc/cpuinfo - cpu information
- ⌚ cat /proc/meminfo - memory information
- ⌚ man command - show the manual for command
- ⌚ df - show disk usage
- ⌚ du - show directory space usage
- ⌚ free - show memory and swap usage
- ⌚ whereis app - show possible locations of app
- ⌚ which app - show which app will be run by default



INSTA



Insta  
./con  
make  
make



dpkg  
rpm -

Cli

① which app - show which app will be run by default

make  
make

① dpkg  
① rpm -

# cli commands



① chmod  
to change  
the file  
permissions

group  
● Example

Example

chmod

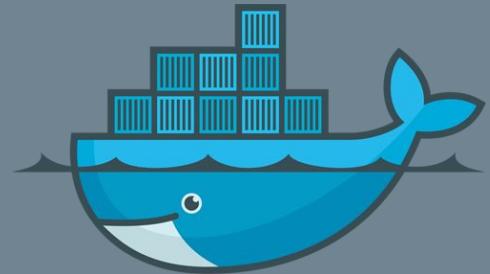
chmod

For more

## ② COMPRESSION

- ① tar cf file.tar files - create a tar named file.tar containing files
- ① tar xf file.tar - extract the files from file.tar
- ① tar czf file.tar.gz files - create a tar with Gzip compression
- ① tar xzf file.tar.gz - extract a tar using Gzip
- ① tar cjf file.tar.bz2 - create a tar with Bzip2 compression
- ① tar xjf file.tar.bz2 - extract a tar using Bzip2
- ① gzip file - compresses file and renames it to file.gz
- ① gzip -d file.gz - decompresses file.gz back to file

## 3.2 Docker command-line



# Install Docker

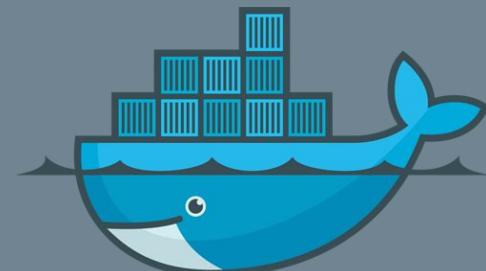
## Install Docker on Ubuntu

or

```
curl -sSL https://get.docker.com/ | sh
```

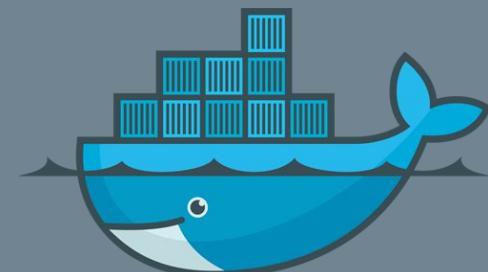
and

```
docker run hello-world
```



# Docker Management commands

Command	Description
<code>dockerd</code>	Launch the Docker daemon
<code>info</code>	Display system-wide information
<code>inspect</code>	Return low-level information on a container or image
<code>version</code>	Show the Docker version information

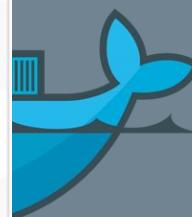


# Docker image commands

Command	Description
build	Build an image from a Dockerfile
commit	Create a new image from a container's changes
history	Show the history of an image
images	List images
import	Import the contents from a tarball to create a filesystem image
load	Load an image from a tar archive or STDIN
rmi	Remove one or more images
save	Save images to a tar archive
tag	Tag an image into a repository

# Docker container commands (1/2)

Command	Description
attach	Attach to a running container
cp	Copy files/folders from a container to a HOSTDIR or to STDOUT
create	Create a new container
diff	Inspect changes on a container's filesystem
events	Get real time events from the server
exec	Run a command in a running container
export	Export a container's filesystem as a tar archive
kill	Kill a running container
logs	Fetch the logs of a container
pause	Pause all processes within a container
port	List port mappings or a specific mapping for the container
ps	List containers

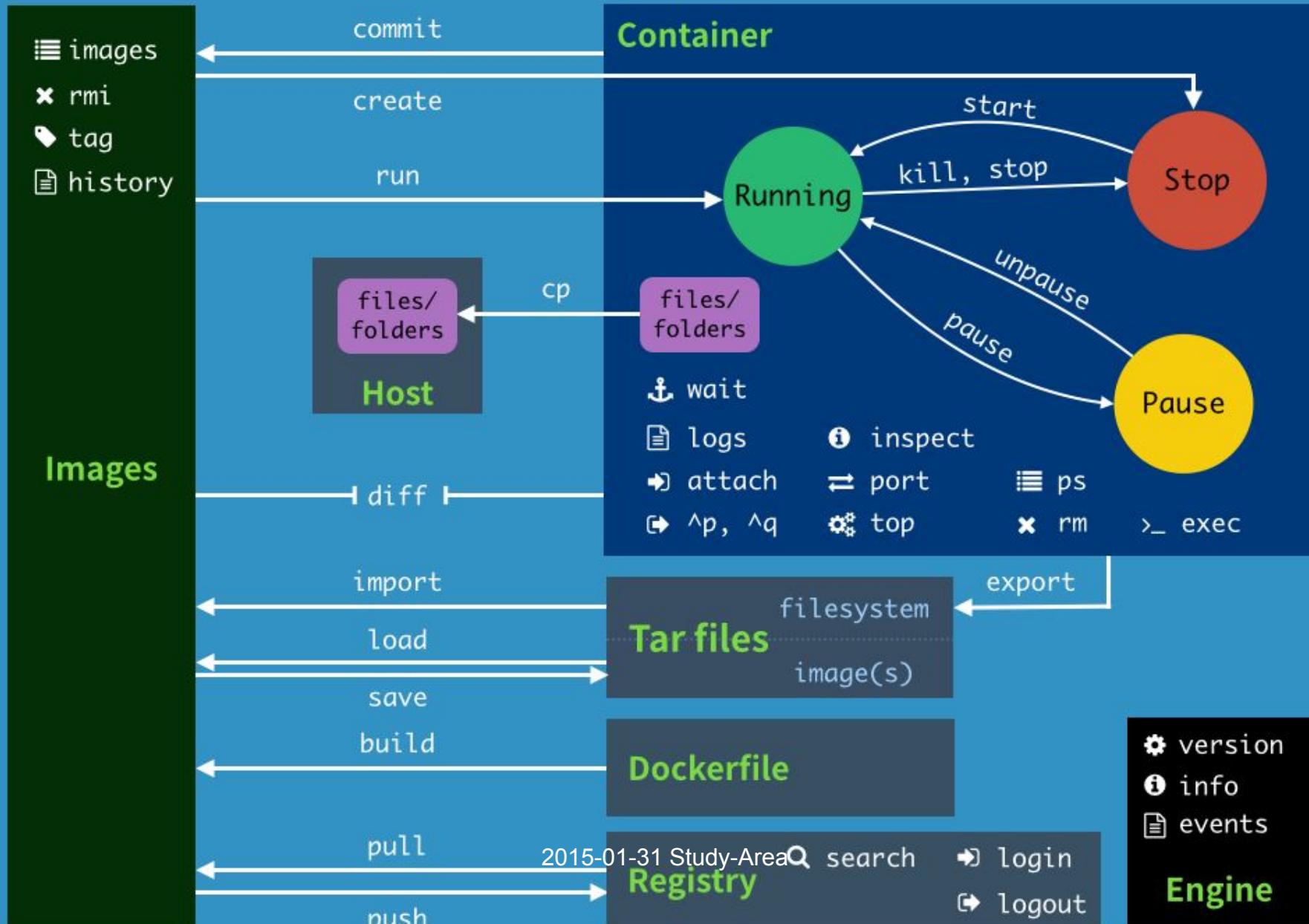


# Docker container commands (2/2)

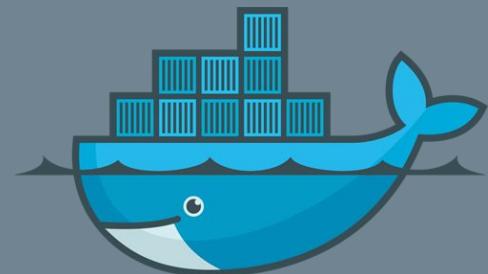
Command	Description
rename	Rename a container
restart	Restart a running container
rm	Remove one or more containers
run	Run a command in a new container
start	Start one or more stopped containers
stats	Display a live stream of container(s) resource usage statistics
stop	Stop a running container
top	Display the running processes of a container
unpause	Unpause all processes within a container
update	Update configuration of one or more containers
wait	Block until a container stops, then print its exit code



# Docker Commands Diagram



# 4. Docker Engine Playground



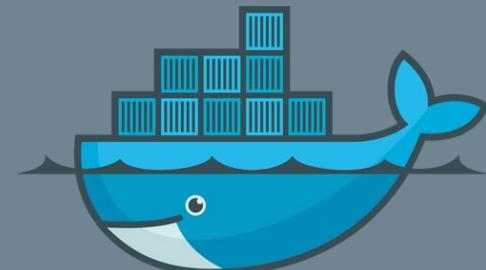
# Azure Firewall

docker run -d -p 80:80 nginx

優先順序	名稱	來源	目的地	服務	動作
1000	default-allow-ssh	Any	Any	SSH (TCP/22)	Allow
1010	web	Any	Any	HTTP (TCP/80)	Allow

docker run -ti --rm -p 80:80 nginx

docker run -ti --rm -p 80:80 nginx bash



# Azure DNS Setting

Microsoft Azure 資源群組 > workshop > docker0001-ip - 組態

重新整理 移動

訂用帳戶 ID: f54fb833-8281-4160-855e-aef64e241aa1  
位置: 東亞

類型	位置	操作
虛擬機器	東亞	...
網路介面	東亞	...
網路安全性群組	東亞	...
公用 IP 位址	東亞	...
虛擬網路	東亞	...
儲存體帳戶	東亞	...
儲存體帳戶	東亞	...

**docker0001-ip - 組態**  
公用 IP 位址

儲存 捨棄

指派: 動態 靜態

IP 位址: 13.75.113.166

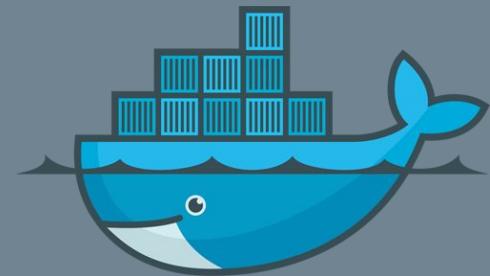
閒置逾時 (分鐘): 4

DNS 名稱標籤 (選用): docker0001  
.eastasia.cloudapp.azure.com

組態 屬性 鎖定 自動化範本

支援與疑難排解 新增支援要求

# 5. Docker Hub introduction



# Docker Hub = App Store

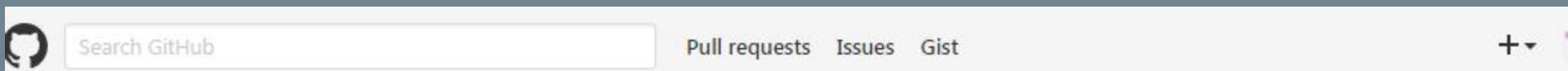
- Public Docker Registry
- One free private repo.
- Auto-build & Webhook
- Security Scanning is not free.

Build, Ship, & Run  
Any App, Anywhere

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



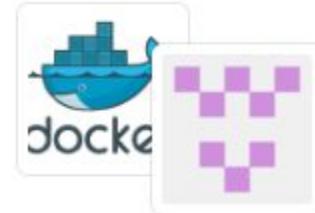
# GitHub & Docker Hub



Search GitHub Pull requests Issues Gist +

## Authorize application

Docker Hub Registry by @docker would like permission to access your account



### Review permissions



Repositories

Public and private



**Authorize application**

Docker Hub Registry

Docker Hub Registry

[Visit application's website](#)

[!\[\]\(01dc96a7e051561aac10d319040279fb\_img.jpg\) Learn more about OAuth](#)

# Vulnerability Analysis

CoreOS Clair

Anchore



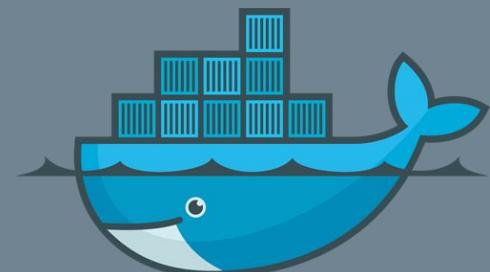
clair

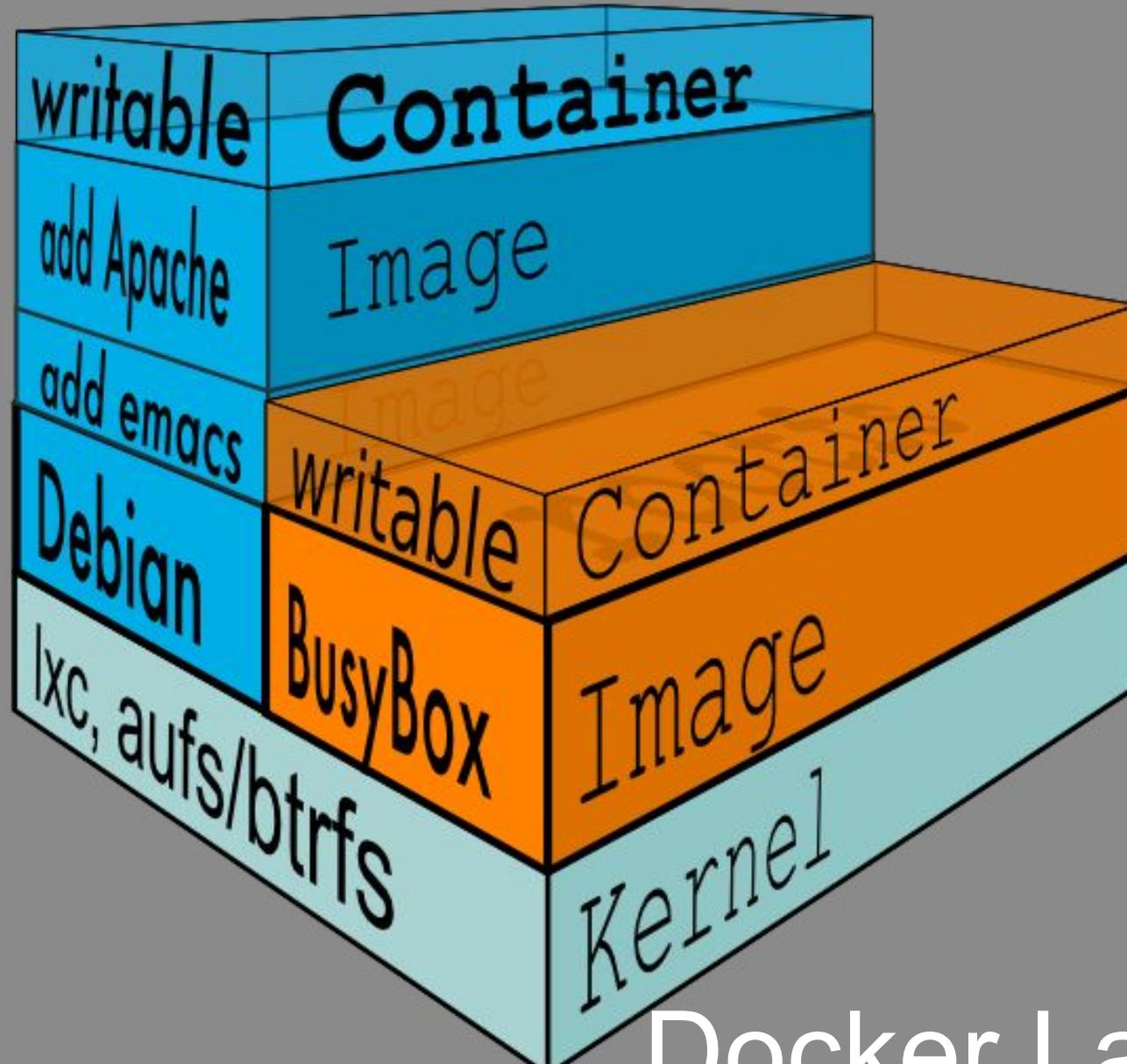
sha256:204fff67067677bbe3db68ba5ab36eb0749cc7e1cb4ac0f35f5a0d07383e1635

linux 3.16.7-ckt20-1+deb8u2 - A

- o **CVE-2016-3134**  
The netfilter subsystem in the Linux kernel through 4.5.2 does not validate certain offset fields, which allows local users to gain privileges or cause a denial of service (heap memory corruption) via an IPT\_SO\_SET\_REPLACE setsockopt call.  
[Link](#)
- o **CVE-2015-8830**  
Integer overflow in the aio\_setup\_single\_vector function in fs/aio.c in the Linux kernel 4.0 allows local users to cause a denial of service or possibly have unspecified other impact via a large AIO iovec. NOTE: this vulnerability exists because of a CVE-2012-6701 regression.  
[Link](#)
- o **CVE-2015-8816**  
The hub\_activate function in drivers/usb/core/hub.c in the Linux kernel before 4.3.5 does not properly maintain a hub-interface data structure, which allows physically proximate attackers to cause a denial of service (invalid memory access and system crash) or possibly have unspecified other impact by unplugging a USB hub device.  
[Link](#)
- o **CVE-2013-7445**  
The Direct Rendering Manager (DRM) subsystem in the Linux kernel through 4.x mishandles requests for Graphics Execution Manager (GEM) objects, which allows context-dependent attackers to cause a denial of service (memory consumption) via an application that processes graphics data, as demonstrated by JavaScript code that creates many CANVAS elements for rendering by Chrome or Firefox.  
[Link](#)
- o **CVE-2016-0758**  
Integer overflow in lib/asn1\_decoder.c in the Linux kernel before 4.6 allows local users to gain privileges via crafted ASN.1 data.  
[Link](#)

# 6.1 Docker image & Dockerfile





Docker Layers

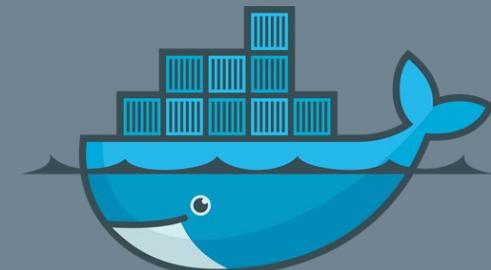
# Create Docker image

1. Docker commit
2. Dockerfile - docker build
3. Docker Hub auto-build
4. FROM scratch
5. Based on others, ubuntu, alpine...

Example:

<https://github.com/docker/labs/tree/master/beginner/static-site>

```
docker save busybox > busybox.tar  
docker load < busybox.tar
```



# Dockerfile Reference

Same folder, docker build .

docker build -f /other/folder/file .

Add tag, docker build -t TAG\_NAME .

Sample:

```
FROM debian:jessie
```

```
MAINTAINER docker "docker@nginx.com"
```

```
RUN apt-get update && apt-get install -y nginx
```

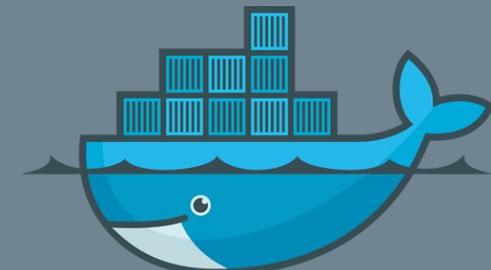
```
CMD ["nginx", "-g", "daemon off;"]
```

Healthcheck from 1.12

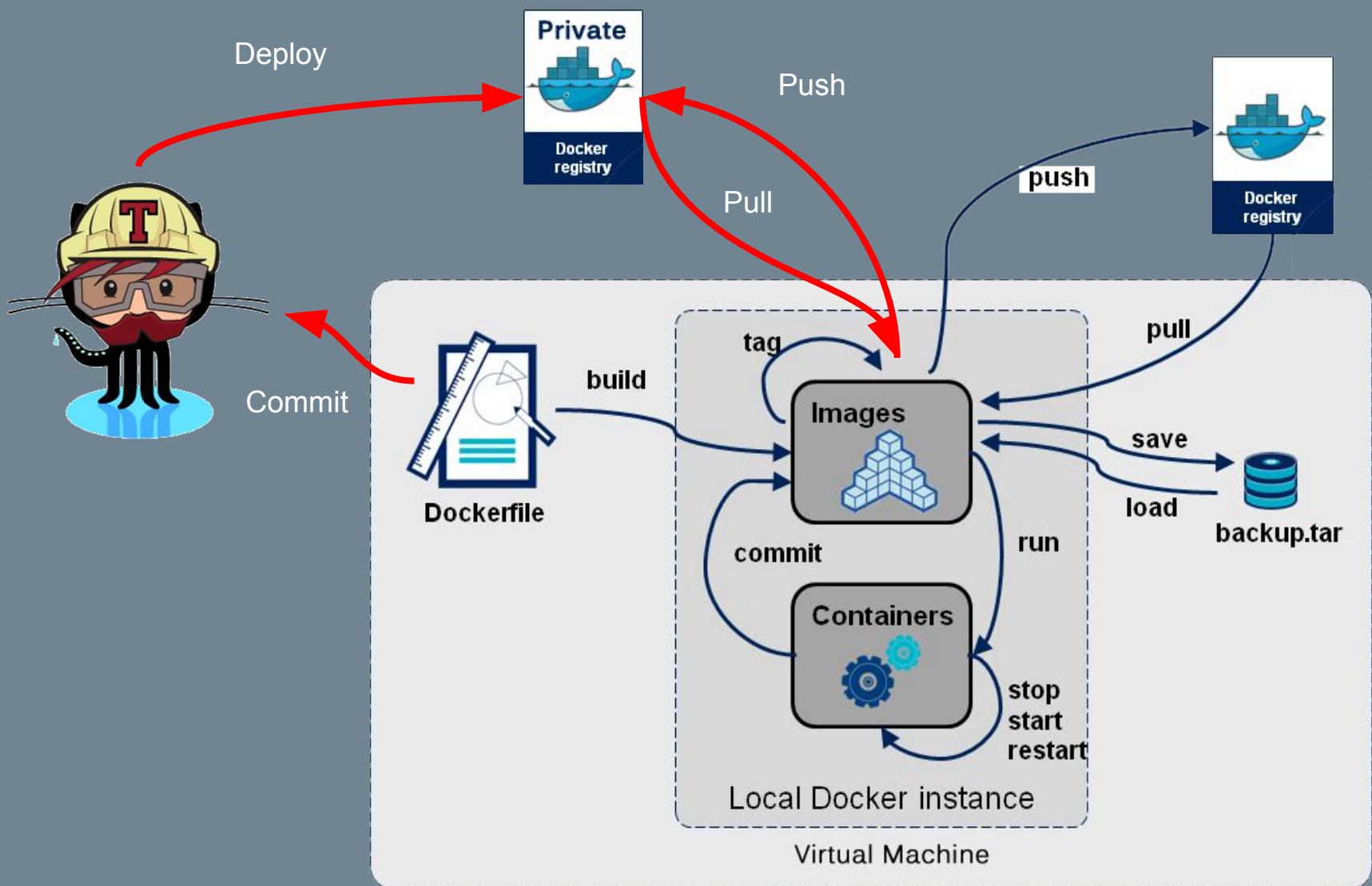


# Dockerfile Practice

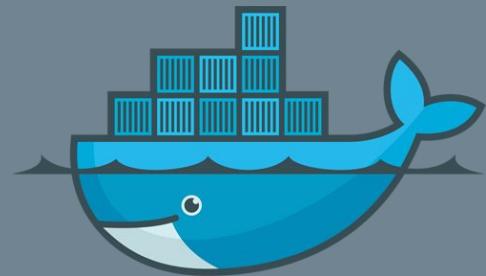
1. Must be “Dockerfile”.
2. Use a `.dockerignore` file, like `.gitignore`.
3. Minimize the number of layers
4. Sort multi-line arguments
5. ADD or COPY, in detail
6. CMD or ENTRYPOINT
7. ONBUILD
8. EXPOSE and USER
9. WORKDIR and ENV



# Use Scenario



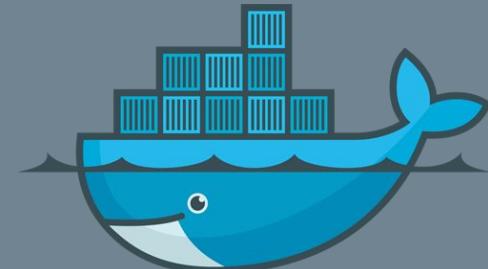
# 6.2 Docker Hub Auto-build



# Dockerfile

Sample:

```
FROM debian:jessie
MAINTAINER docker "docker@nginx.com"
RUN apt-get update && apt-get install -y nginx
CMD ["nginx", "-g", "daemon off;"]
```



# Git Workflow

1. git init or init on GitHub.
2. git add Dockerfile
3. git commit -m “First init”
4. git remote add origin  
[https://github.com/YOURNAME/docker\\_build.git](https://github.com/YOURNAME/docker_build.git)
5. git push origin master

# Create Auto-build Repo.

The screenshot shows the Docker Hub interface for creating an automated build. The top navigation bar includes links for Dashboard, Explore, Organizations, a search bar, and user account information for 'dockware'. A sidebar on the right lists options: Create Repository, Create Automated Build (which is selected), and Create Organization. The main form is titled 'Create Automated Build' and contains fields for 'Repository Namespace & Name\*', 'Visibility', 'Short Description\*', and a note about branch matching. A progress bar at the bottom indicates the process is still 'Creating...'.

Dashboard   Explore   Organizations

Search

Create

dockware

## Create Automated Build

Repository Namespace & Name\*

dockware   nginx

Visibility

private

Short Description\*

I\_Shou\_Workshop

By default Automated Builds will match branch names to Docker build tags. [Click here to customize behavior.](#)

Creating...

<https://hub.docker.com/add/automated-build/dockware/>

# Build Settings

PUBLIC | AUTOMATED BUILD

philipz/rpi-raspbian ☆

Last pushed: 3 months ago

Repo Info

Tags

Dockerfile

Build Details

Build Settings

Collaborators

Webhooks

Settings

## Build Settings

When active, builds will happen automatically on pushes.

The build rules below specify how to build your source into Docker images. The name can be a string or a regex. The Docker Tag name may contain variables. We currently support {sourceref}, which refers to the source branch/tag name. [Show more](#)



Source Repository  
philipz/docker-rpi-raspbian

Type

Name

Dockerfile Location

Docker Tag Name

Branch ▾

master



/

latest

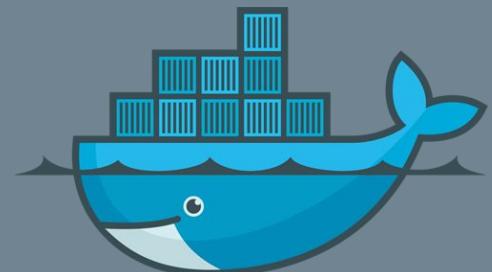


Trigger

**docker pull YOURNAME/IMAGENAME**

Save Changes

# 7. Deploy Docker image to Azure PaaS



# Azure Web App on Linux

Microsoft Azure > 新增 > Web + 行動

The screenshot shows the Azure portal interface. On the left, a sidebar lists various service categories like Dashboard, Resource Groups, and Storage. The main area is titled '新增' (Create) and shows the 'Marketplace' search bar. Under 'MARKETPLACE', categories include 計算, 網路, Storage, Web + 行動, Databases, Intelligence + analytics, 物聯網, Enterprise Integration, 安全性 + 識別, Developer tools, Monitoring + Management, Add-ons, and 容器. To the right, under 'Web + 行動', there are several options: 'Web 應用程式' (selected), 'Mobile App', '邏輯應用程式', 'Web App On Linux (預覽)' (highlighted with a red oval), and 'CDN'. The 'Web App On Linux (預覽)' card has a blue icon of a globe and the text 'Enjoy your web app natively hosted on Linux.'

新增

搜尋 Marketplace

MARKETPLACE

計算 >

網路 >

Storage >

Web + 行動 >

Databases >

Intelligence + analytics >

物聯網 >

Enterprise Integration >

安全性 + 識別 >

Developer tools >

Monitoring + Management >

Add-ons >

容器 >

Web + 行動

精選應用程式

查看全部

Web 應用程式

享受您的 Web 應用程式所提供之安全且具彈性的開發、部署及縮放選項。

Mobile App

A scalable and secure backend that can be used to power apps on any platform – iOS, Android,

邏輯應用程式

無須撰寫程式碼，就能自動化各個雲端上的資料存取與使用流程

Web App On Linux (預覽)

Enjoy your web app natively hosted on Linux.

CDN

Enjoy scalable, global distributed edge servers for fast and reliable content delivery

# Use Docker image for Web AP

Web App On Linux (預覽) □ X

建立

\* 應用程式名稱  
philipz ✓ .azurewebsites.net

\* 訂用帳戶  
Developer Program Benefit

\* 資源群組 i  
● 新建 ● 使用現有項目

App Service 方案/位置 >  
ServicePlane1acc0a9-a710(Wes...)

設定容器  
philipz/nginx >

釘選到儀表板

**建立** **自動化選項**

Docker 容器

 Docker 容器

Linux 上的 Web Apps 可善用 Docker 容器的威力，讓您從 Azure Container Registry、Docker Hub、私人容器登錄使用自訂容器，或是使用 App Service 所提供的其中一項預設容器。

影像來源  
**內建** Docker Hub 私人登錄

存放庫存取  
**公用** 私用

\* 影像及選擇性標籤 (例如 'image:tag')  
philipz/nginx

啟動檔案

**確定**

# Azure PaaS Price Models

新增 App Service 方案

為 Web 應用程式建立方案

\* App Service 方案  
請輸入您 App Service 方案的名稱

\* 位置  
Southeast Asia

\* 定價層  
S1 標準 >

確定

選擇定價層

瀏覽所提供的方案及其功能

\* Linux App Service 方案的預覽版將有 50% 的折扣。進一步了解 □

S1 標準	S2 標準	S3 標準
1 核心	2 核心	4 核心
1.75 GB RAM	3.5 GB RAM	7 GB RAM
50 GB 儲存體	50 GB 儲存體	50 GB 儲存體
自訂網域 / SSL 包括 SNI 與 IP SSL 支援	自訂網域 / SSL 包括 SNI 與 IP SSL 支援	自訂網域 / SSL 包括 SNI 與 IP SSL 支援
<input checked="" type="checkbox"/> 最多 10 個執行個體 自動調整規模	<input checked="" type="checkbox"/> 最多 10 個執行個體 自動調整規模	<input checked="" type="checkbox"/> 最多 10 個執行個體 自動調整規模
每天備份	每天備份	每天備份
5 個位置 Web 應用程式預備環境	5 個位置 Web 應用程式預備環境	5 個位置 Web 應用程式預備環境
預覽定價 下方為 50% 折扣的價格	預覽定價 下方為 50% 折扣的價格	預覽定價 下方為 50% 折扣的價格
2,308.45 TWD/月 (估計)	4,616.89 TWD/月 (估計)	9,233.78 TWD/月 (估計)
B1 基本	B2 基本	B3 基本
1 核心	2 核心	4 核心

選取

# 8.1 Docker Network command-line



# TCP/IP Foundation

www.google.com, www is hostname, google.com is domain name.

Localhost: 127.0.0.1

TCP/UDP Port: 0-65535 =  $2^{16}$ ,  
but 0 is a reserved port.

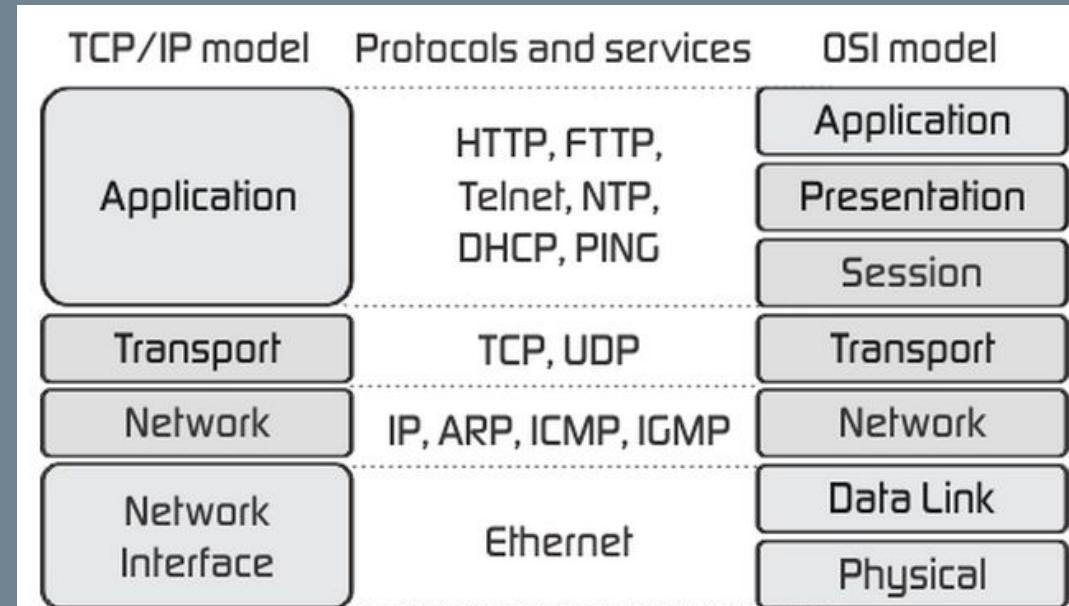
Private IP:

10.0.0.0/8

172.16.0.0/12 ~

172.31.0.0/12

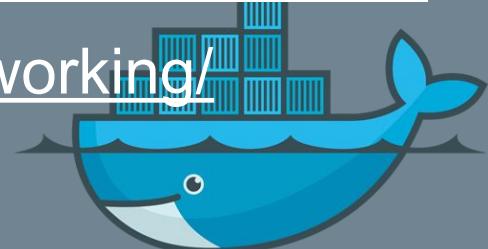
192.168.0.0/16



# Network and connectivity commands

Command	Description
<code>network connect</code>	Connect a container to a network
<code>network create</code>	Create a new network
<code>network disconnect</code>	Disconnect a container from a network
<code>network inspect</code>	Display information about a network
<code>network ls</code>	Lists all the networks the Engine daemon knows about
<code>network rm</code>	Removes one or more networks

<https://docs.docker.com/engine/userguide/networking/>

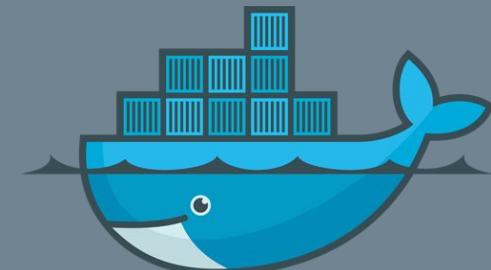


# Docker Built-In Network Drivers

- Bridge
  - Overlay
  - MACVLAN
  - Host
  - None
- Docker Plug-In Network Drivers**
- weave
  - calico
- Docker Plug-In IPAM Drivers**
- infoblox

No more “link”, just use network.

Docker Reference Architecture: Designing Scalable,  
Portable Docker Container Networks



# Exercise 1

```
$ docker network ls
```

```
$ ifconfig
```

```
$ docker run -ti --rm busybox sh
```

*cat /etc/hosts, ifconfig*

```
$ docker network inspect bridge
```

```
$ docker run -itd --name=container1 busybox
```

```
$ docker run -itd --name=container2 busybox
```

```
$ docker exec -ti container2 sh
```

*ping -w3 172.17.0.2, ping container1*



# Exercise 2

```
$ docker network create vlan_1
```

```
$ docker network inspect vlan_1
```

```
$ ifconfig | more
```

```
$ docker run --network=vlan_1 -itd --name=container3 busybox
```

```
$ docker network inspect vlan_1
```

```
$ docker run --network=vlan_1 -itd --name=container4 busybox
```

```
$ docker exec -ti container4 sh
```

*ping -w3 172.17.0.2, ping container1, ping container3*



# Exercise 3

```
$ docker network create wp_db
```

```
$ docker pull mysql:5.7
```

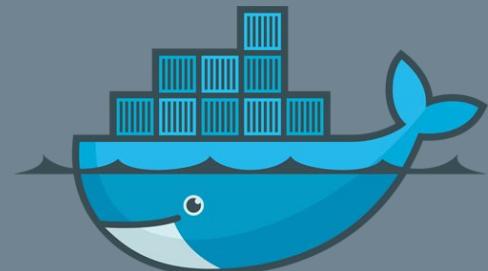
```
$ docker pull wordpress
```

```
$ docker run -d --name db --network=wp_db  
-e MYSQL_ROOT_PASSWORD=wordpress  
-e MYSQL_DATABASE=wordpress  
-e MYSQL_USER=wordpress  
-e MYSQL_PASSWORD=wordpress  
mysql:5.7
```

```
$ docker run -d --name wp -p 80:80 --network=wp_db  
-e WORDPRESS_DB_HOST=db:3306  
-e WORDPRESS_DB_PASSWORD=wordpress  
wordpress
```



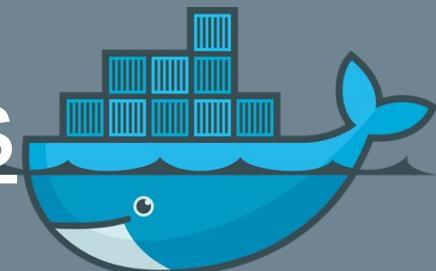
# 8.2 Docker Volume command-line



# Shared data volume commands

Command	Description
<code>volume create</code>	Creates a new volume where containers can consume and store data
<code>volume inspect</code>	Display information about a volume
<code>volume ls</code>	Lists all the volumes Docker knows about
<code>volume rm</code>	Remove one or more volumes

Manage data in containers



# Exercise

```
$ docker volume create \
  --name composewp_db_data
$ docker pull mysql:5.7
$ docker pull wordpress
$ docker run -d --name db --network=wp_db
  -e MYSQL_ROOT_PASSWORD=wordpress
  -e MYSQL_DATABASE=wordpress
  -e MYSQL_USER=wordpress
  -e MYSQL_PASSWORD=wordpress
  -v composewp_db_data:/var/lib/mysql
mysql:5.7
$ docker run -d --name wp -p 80:80 --network=wp_db
  -e WORDPRESS_DB_HOST=db:3306
  -e WORDPRESS_DB_PASSWORD=wordpress
wordpress
```



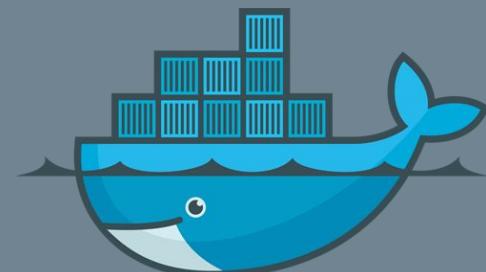
# SDS

## Software Define Storage

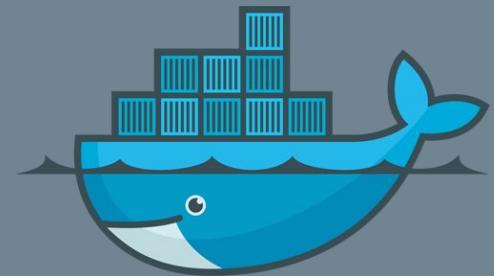
### EMC: REX-Ray

### Azure: File storage

### AWS: Elastic File System



# 9.1 Docker Compose command-line



# Install Docker Compose

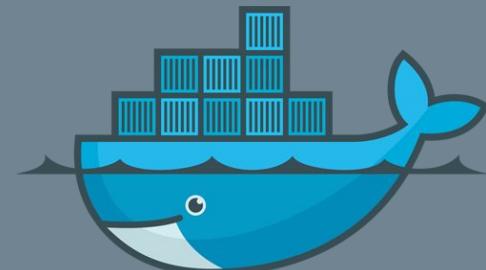
sudo curl -L

```
"https://github.com/docker/compose/releases/download/1.9.0/  
docker-compose-$(uname -s)-$(uname -m)" -o \  
/usr/local/bin/docker-compose
```

and

```
sudo chmod +x /usr/local/bin/docker-compose
```

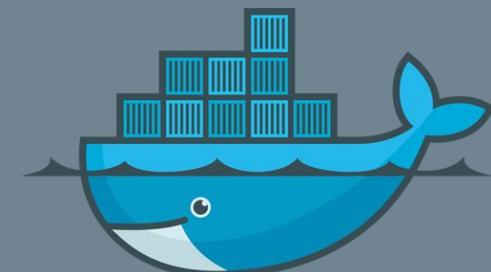
**docker-compose -v**



# Docker Compose commands (1/2)

Commands:

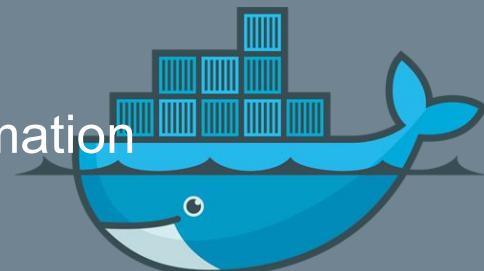
build	Build or rebuild services
bundle	Generate a Docker bundle from the Compose file
config	Validate and view the compose file
create	Create services
down	Stop and remove containers, networks, images, and volumes
events	Receive real time events from containers
exec	Execute a command in a running container
help	Get help on a command
kill	Kill containers
logs	View output from containers
pause	Pause services
port	Print the public port for a port binding



# Docker Compose commands (2/2)

Commands:

ps	List containers
pull	Pull service images
push	Push service images
restart	Restart services
rm	Remove stopped containers
run	Run a one-off command
scale	Set number of containers for a service
start	Start services
stop	Stop services
unpause	Unpause services
up	Create and start containers
version	Show the Docker-Compose version information



# Compose File Reference

Run Multi-container at the same time.

Must be docker-compose.yml

Same folder, docker-compose up -d

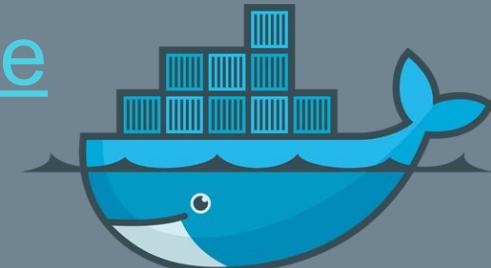
Docker will build the Dockerfile of subfolders.

Docker Network, Volume supports

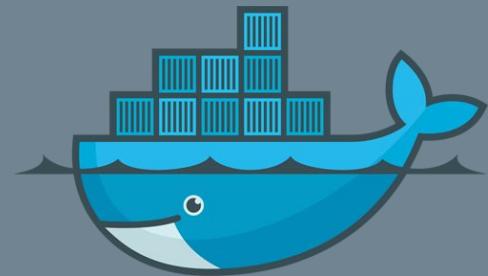
1.13 has supported Swarm mode.

Quickstart: Compose and WordPress

Kompose = Kubernetes + Compose

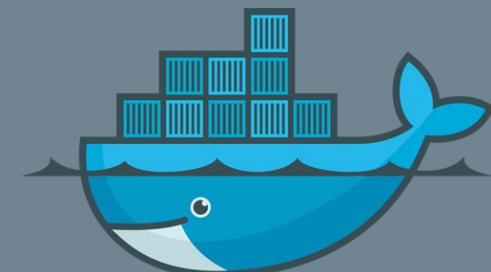


## 9.2 Using Docker Compose



# Compose File Sample (1/2)

```
version: '2'  
  
services:  
  
  db:  
  
    image: mysql:5.7  
  
    volumes:  
      - db_data:/var/lib/mysql  
  
    restart: always  
  
  environment:  
  
    MYSQL_ROOT_PASSWORD: wordpress  
  
    MYSQL_DATABASE: wordpress  
  
    MYSQL_USER: wordpress  
  
    MYSQL_PASSWORD: wordpress
```



# Compose File Sample (1/2)

```
wordpress:
```

```
  depends_on:
```

```
    - db
```

```
  image: wordpress:latest
```

```
  ports:
```

```
    - "8000:80"
```

```
  restart: always
```

```
  environment:
```

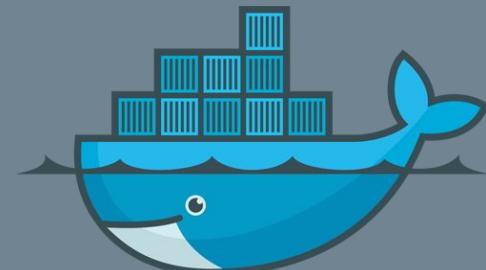
```
    WORDPRESS_DB_HOST: db:3306
```

```
    WORDPRESS_DB_PASSWORD: wordpress
```

```
  volumes:
```

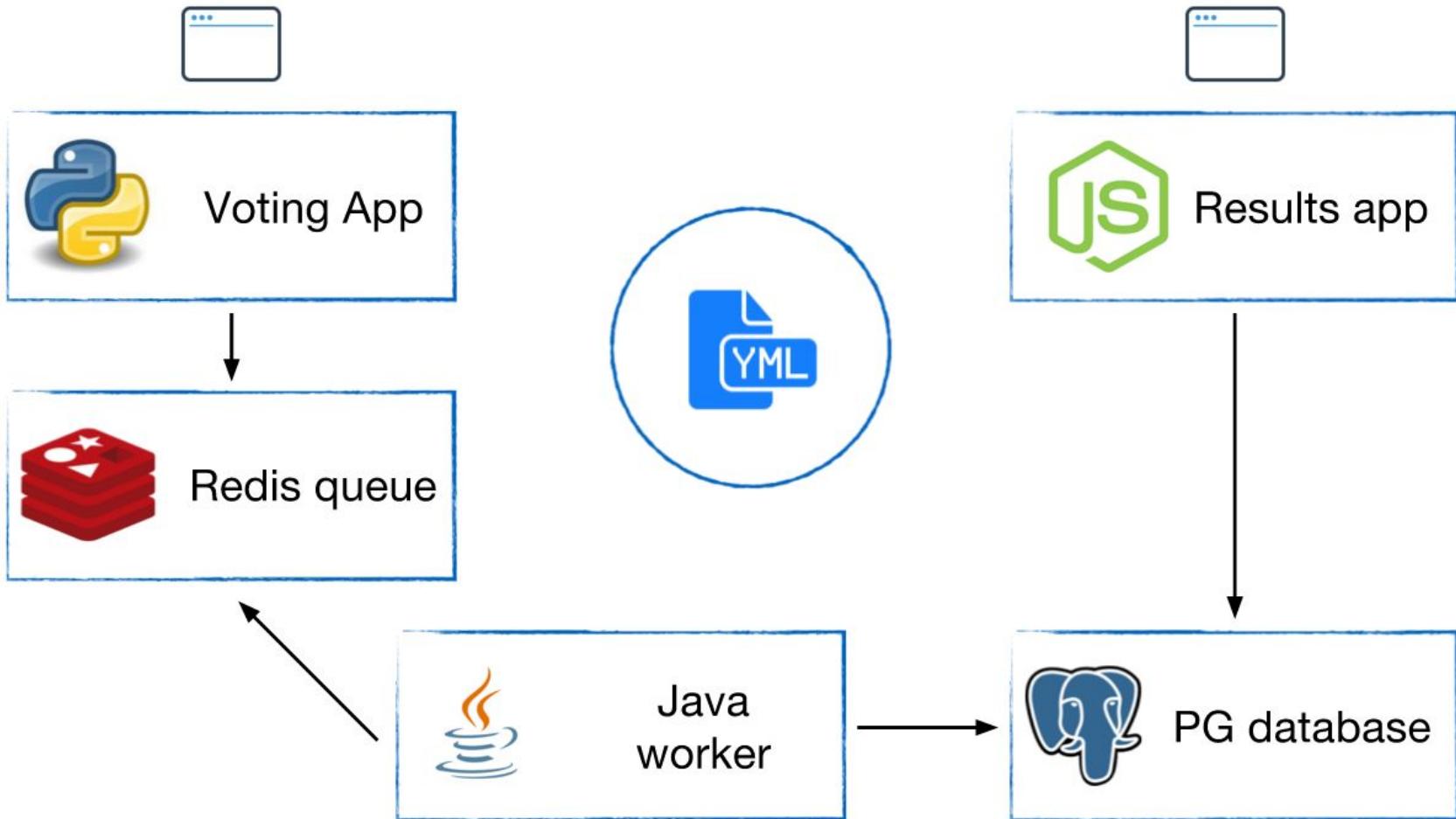
```
    db_data:
```

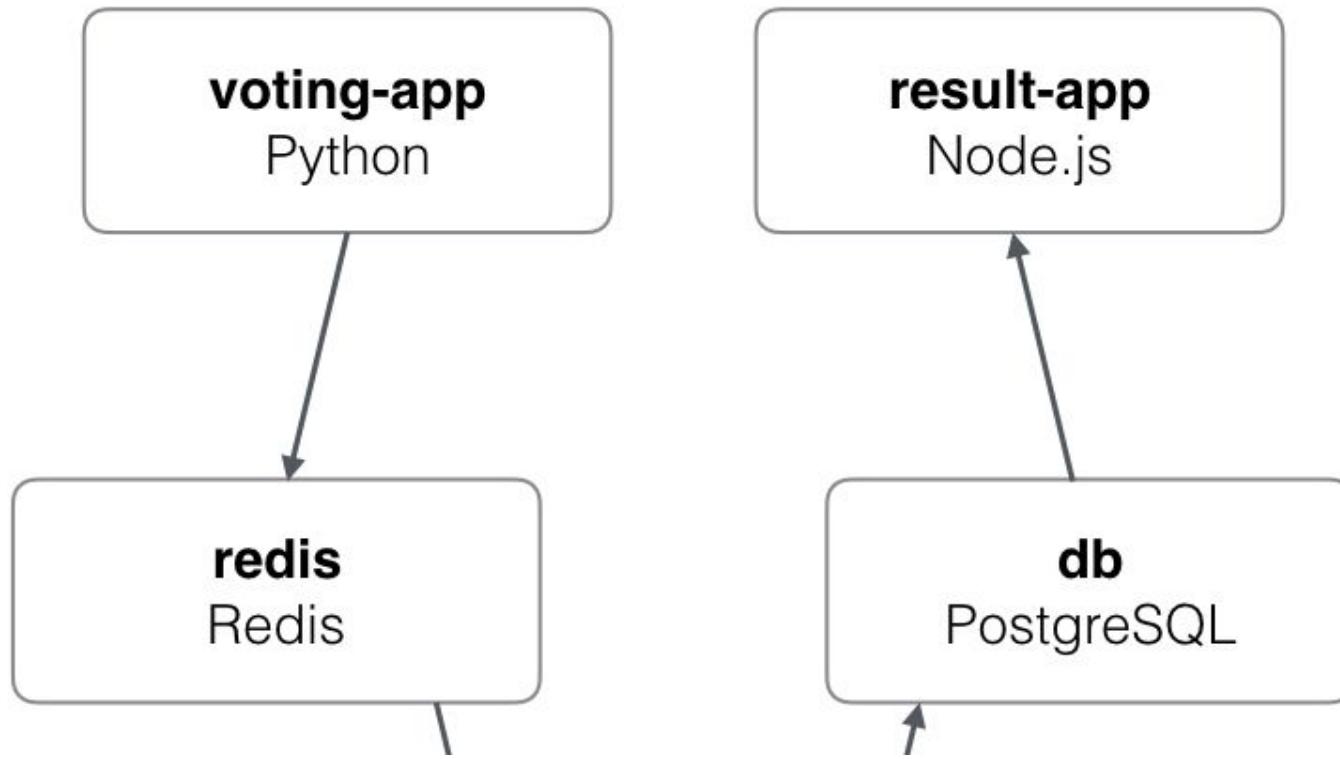
```
*** nslookup wordpress ***
```



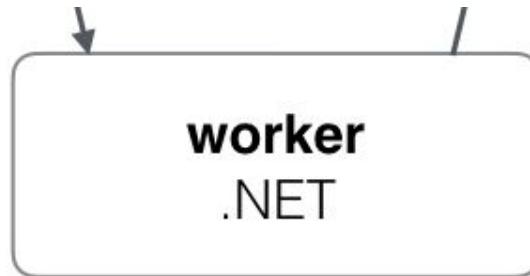
# Microservices Java Worker

## Docker Birthday #3 training





# Microservices .NET Worker

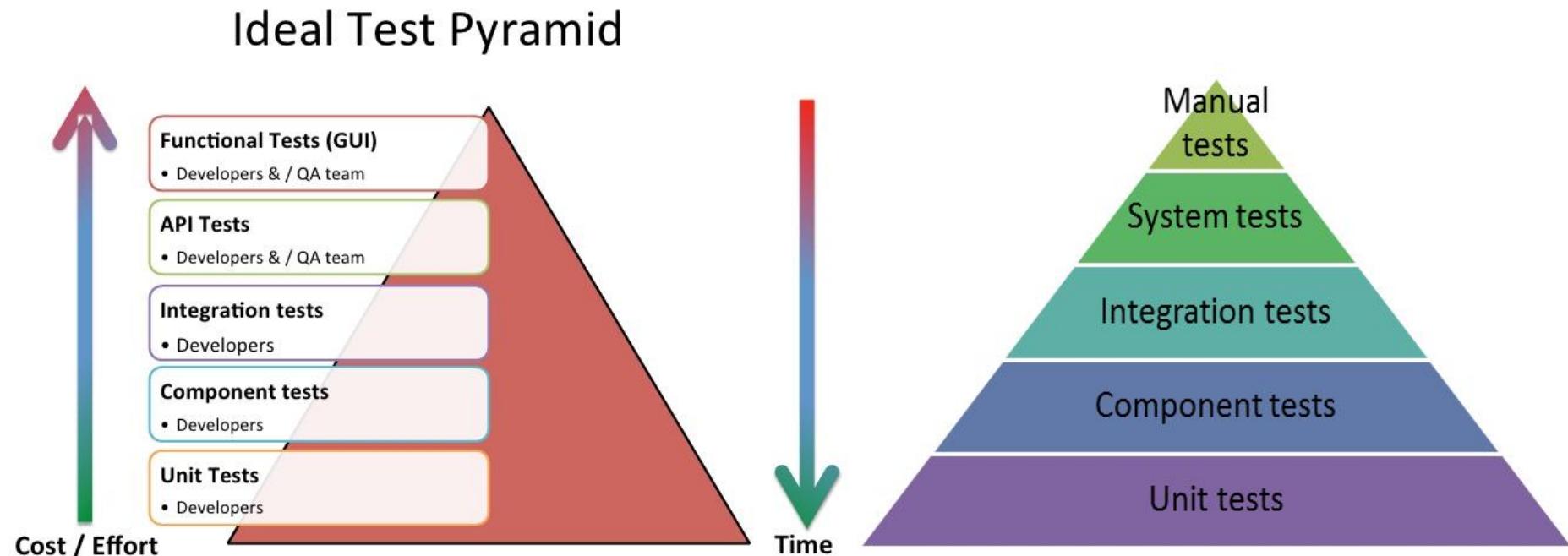


Docker Birthday #3 training

# Docker Compose & CI/CD

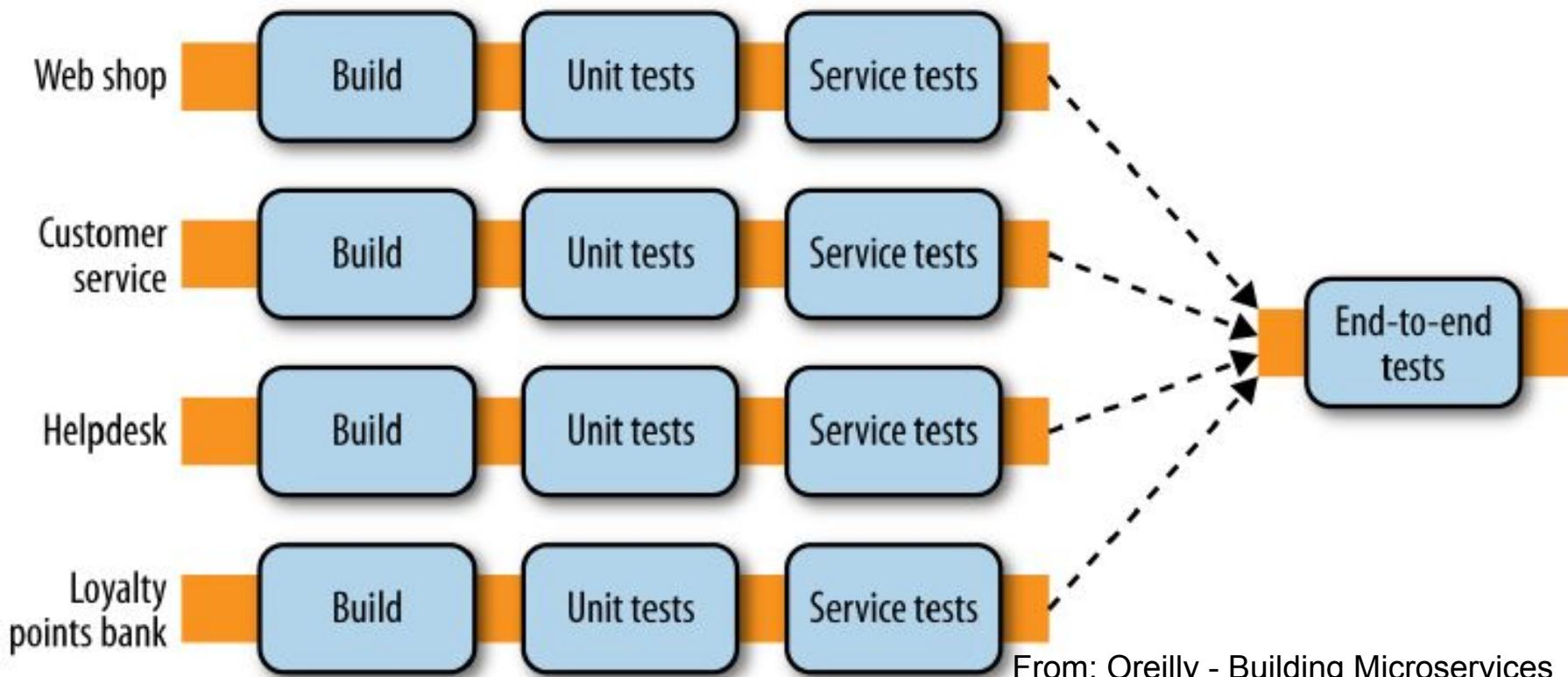
GitHub, CircleCI, Docker Hub = GitLab

Testing level? Coding effort? Env. build-up

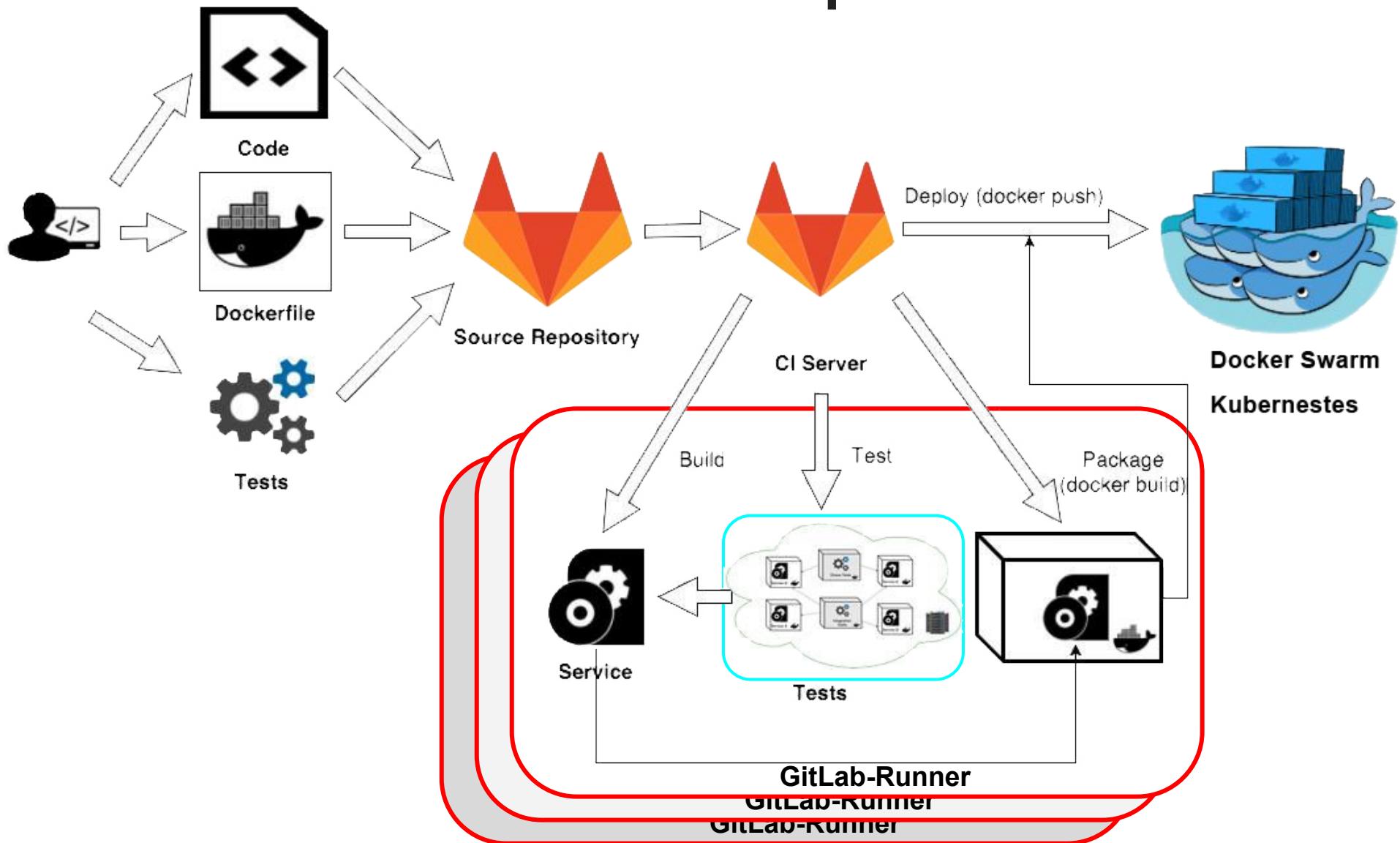


# End to End Tests

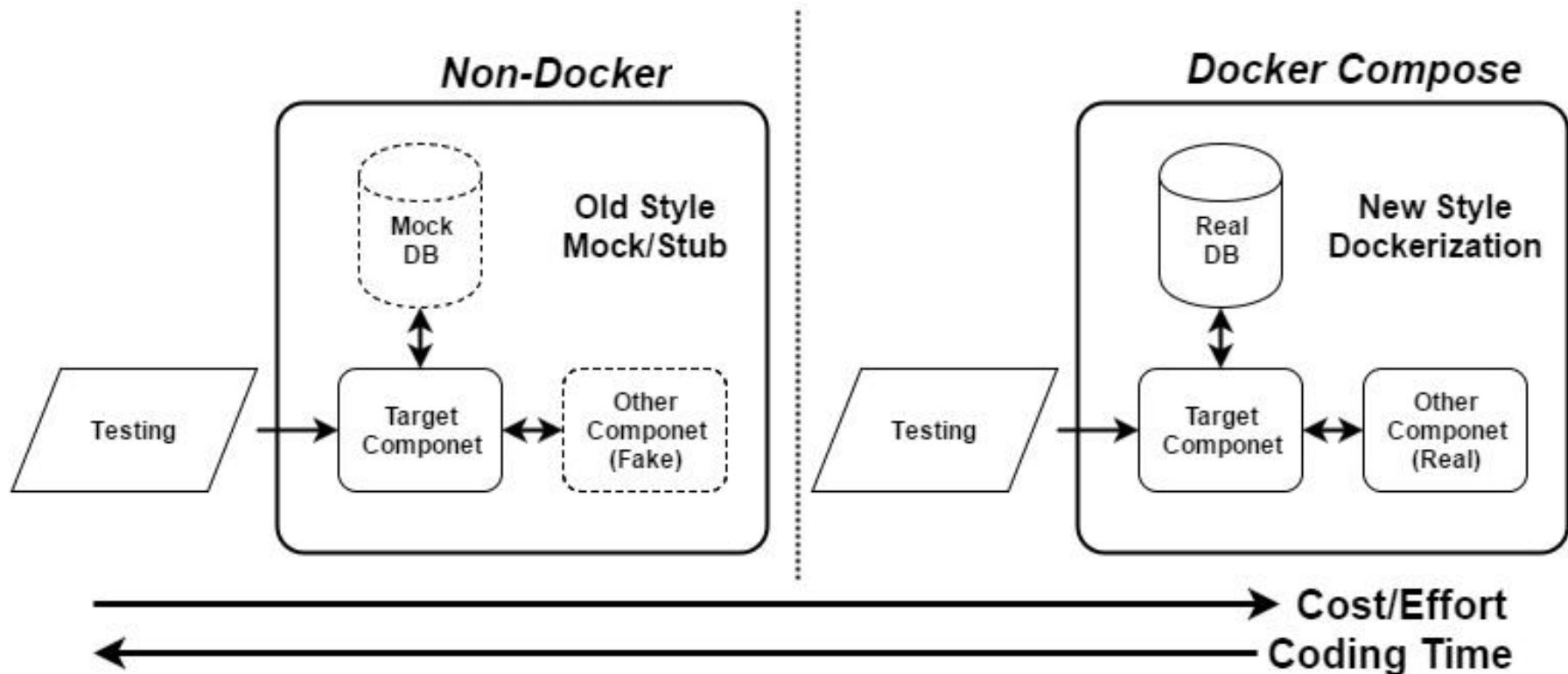
CI with Docker Compose is easy to implement.



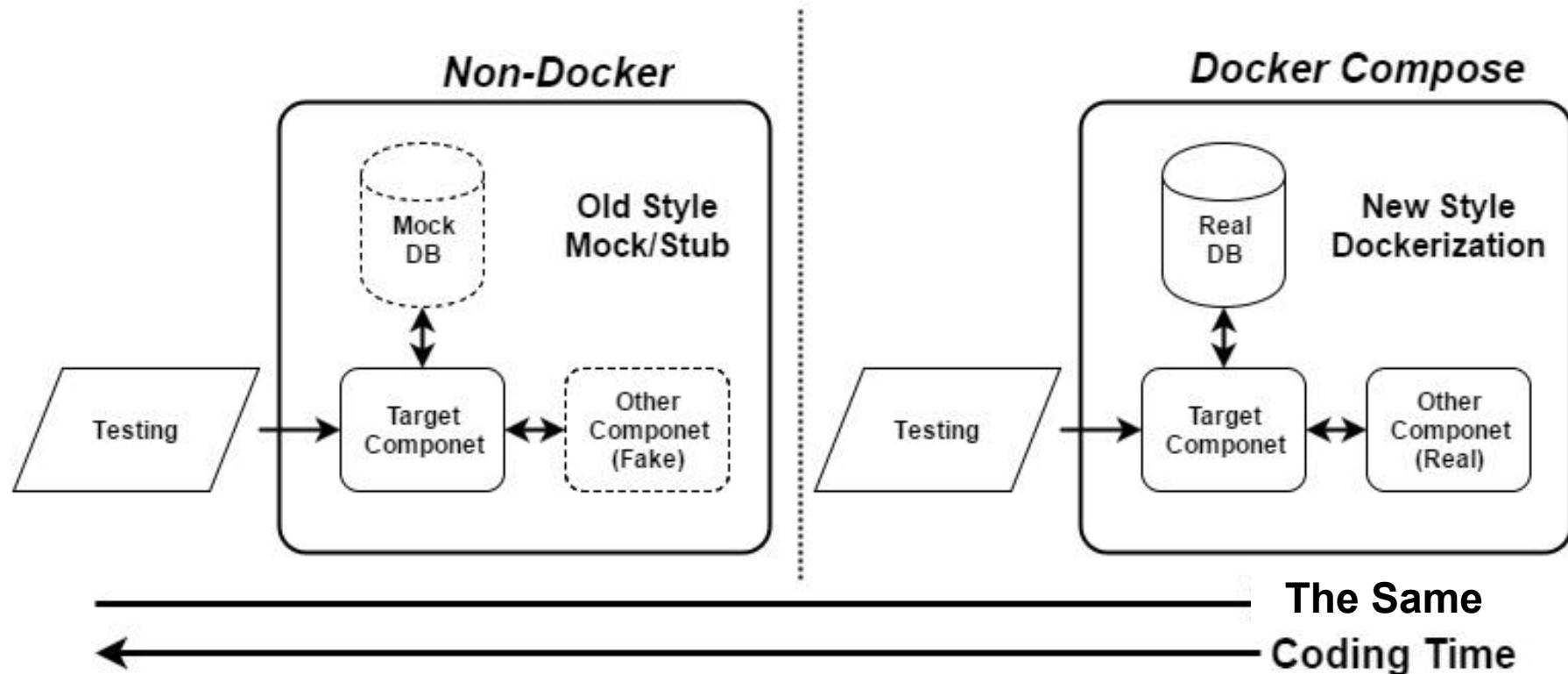
# Container Development Flow

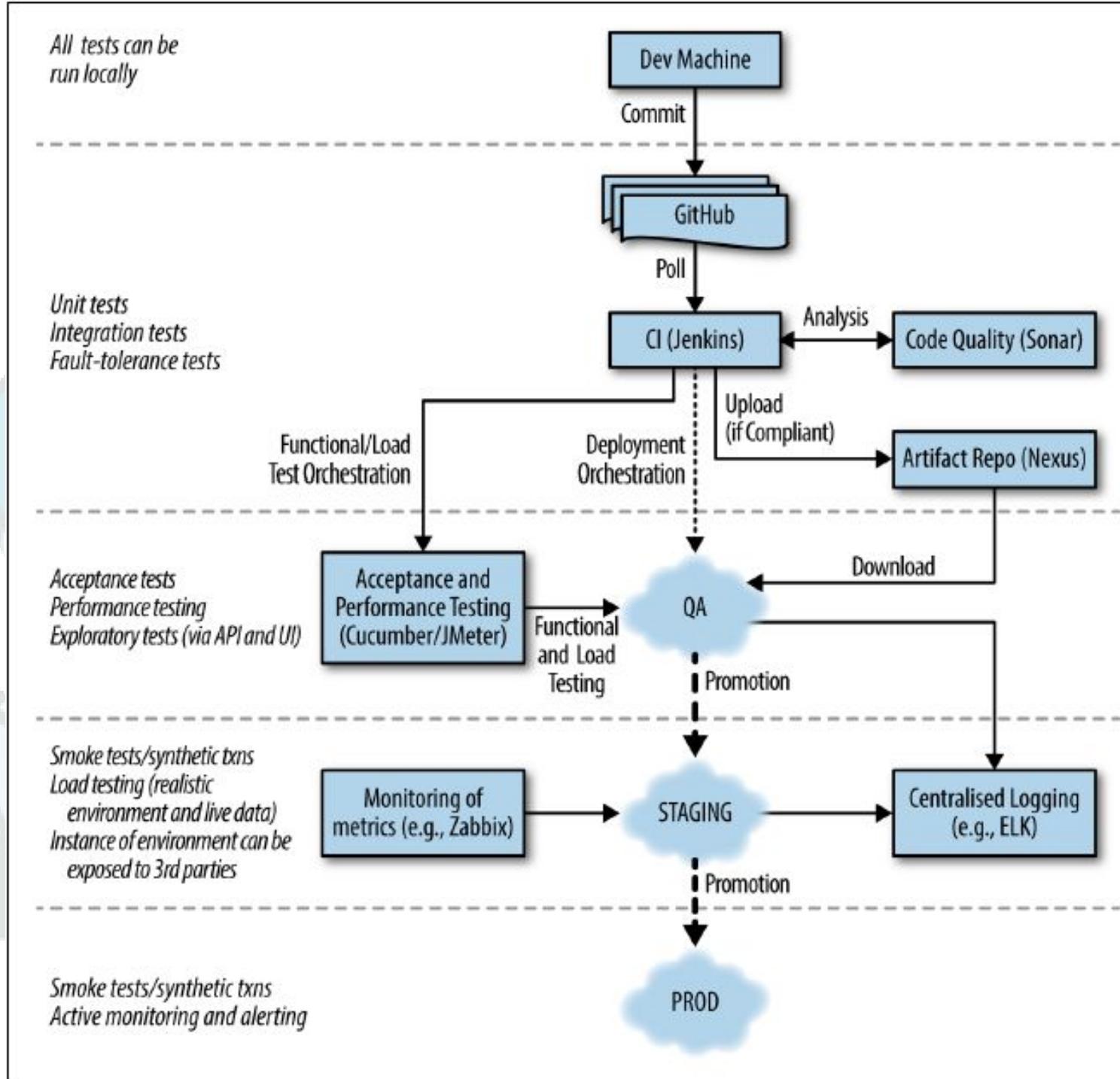


# Test Double Approach



# New Compose Test Approach





All tests can be run locally

Unit tests  
Integration tests  
Fault-tolerance tests

Acceptance tests  
Performance testing  
Exploratory tests (via API and UI)

Smoke tests/synthetic txns  
Load testing (realistic environment and live data)  
Instance of environment can be exposed to 3rd parties

Smoke tests/synthetic txns  
Active monitoring and alerting

Dev Machine

Commit

GitHub

Poll

CI (Jenkins)

Docker 1

Code Quality (Sonar)

Progress (if Compliant)

Create Container with Deployment Artifact and OS (Jenkins)

Docker 2

Deployment Orchestration

Progress (if Compliant)

Container Registry (DockerHub)

Functional/Load Test Orchestration

Acceptance and Performance Testing (Cucumber/JMeter)

Docker 3

Functional and Load Testing

QA

Download

Promotion

Centralised Logging (e.g., ELK)

STAGING

Monitoring of metrics (e.g., Zabbix)

PROD

Docker 4



```
Status: Downloaded newer image for philipz/gitlab-docker-compose:latest
$ docker-compose up -d
Creating network "dockercomposeexample_default" with the default driver
Pulling redis (redis:alpine)...
alpine: Pulling from library/redis
Digest: sha256:99105b7a83dd67a0b4a86ca5f64335801c62d4f3b685eebd4fb66fdb87c66b7b
Status: Downloaded newer image for redis:alpine
Pulling db (postgres:9.4)...
9.4: Pulling from library/postgres
Digest: sha256:9149f6309b83c9b99ae2e1ecab3e14a9662a1a8d0159320c24e34827ffe4c930
Status: Downloaded newer image for postgres:9.4
Pulling worker (philipz/votingapp_worker:latest)...
latest: Pulling from philipz/votingapp_worker
Digest: sha256:beb71b89b4b95eaca33b4ac77f1e20c0a924ab2c4d59b525d9019ba20c169707
Status: Downloaded newer image for philipz/votingapp_worker:latest
Pulling result (philipz/votingapp_result:latest)...
latest: Pulling from philipz/votingapp_result
Digest: sha256:7b89d4589099b171ad2feb96afadbdbd11b0ff9a093b1594994f3648de2fa5a8
Status: Downloaded newer image for philipz/votingapp_result:latest
Creating dockercomposeexample_redis_1
Creating dockercomposeexample_db_1
Creating dockercomposeexample_result_1
Creating dockercomposeexample_vote_1
Creating dockercomposeexample_worker_1
$ cd tests && docker build -t philipz/node-test .
Sending build context to Docker daemon 4.096 kB
```

```
Step 1 : FROM node
latest: Pulling from library/node
6a5a5368e0c2: Already exists
7b9457ec39de: Pulling fs layer
ff18e19c2db4: Pulling fs layer
```

## Build details

Duration: 7 minutes 9 seconds

Finished: a month ago

Runner: #21099

[Raw](#)[Erase](#)

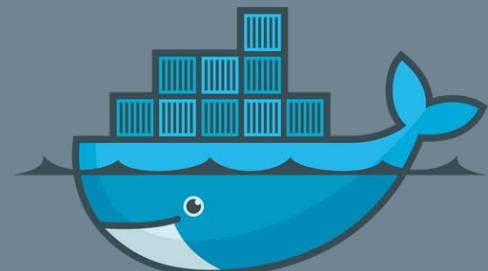
## Commit title

Remove port mapping.

build

test

# 10. Docker & Qemu & Raspberry Pi Raspbian



# RPi & Docker

How to build a base image

Cross-compiler

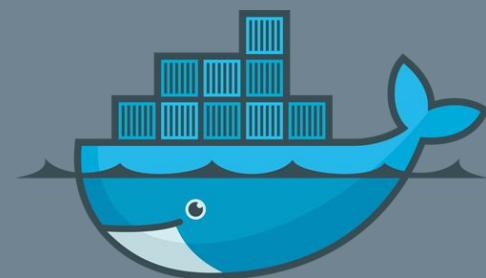
1. Building ARM containers on x86 machine

2. Qemu-static-Docker IoT CI/CD

3. Using GPIO with Docker



**JUST DO IT**



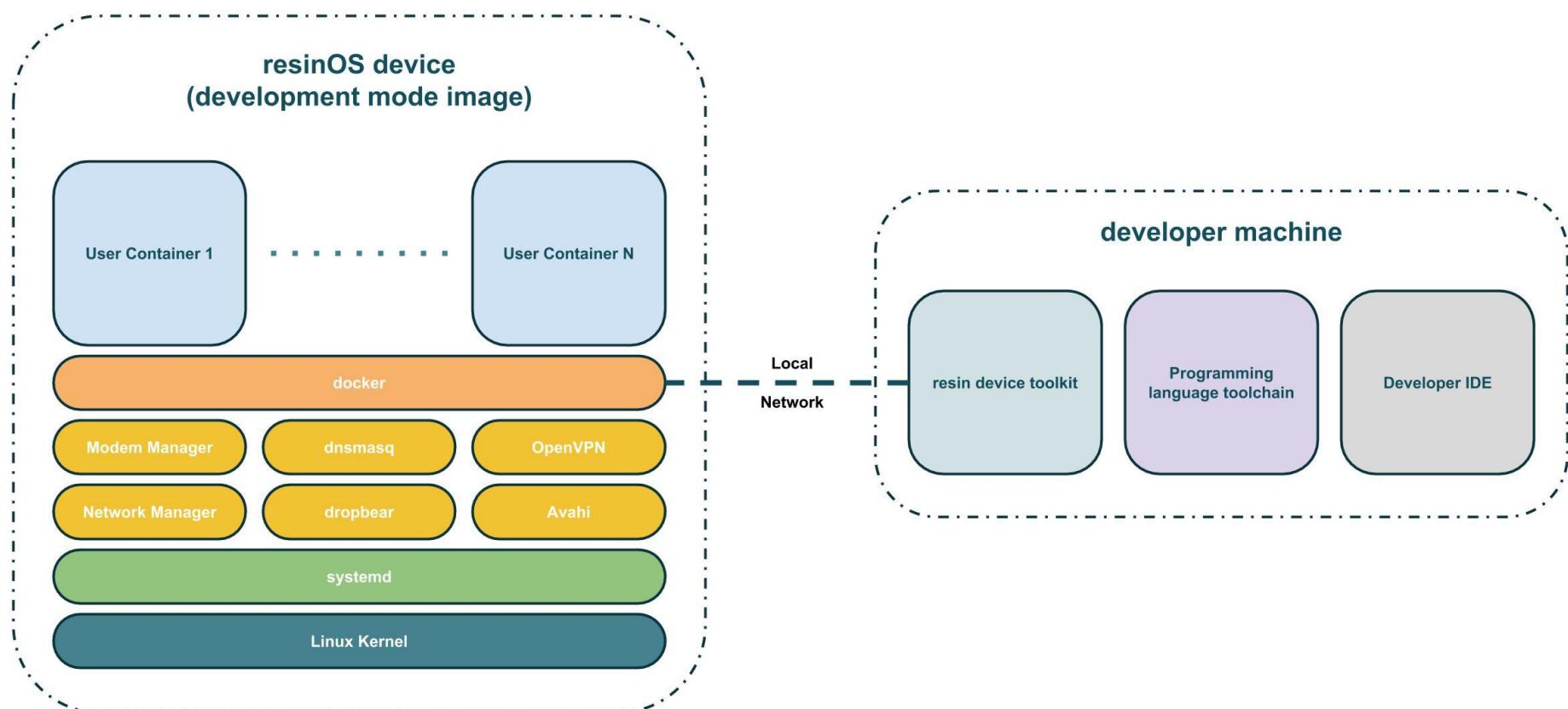
0  
(03:23)

Add Containers +

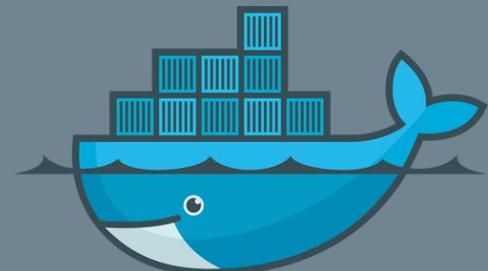
```
Step 4 : COPY qemu/cross-build-end qemu/cross-build-start qemu/qemu-arm-static qemu/sh-shim /usr/bin/
--> 6b9181f32891
Error removing intermediate container c2702bd608f7: nosuchcontainer: No such container: c2702bd608f796
2e2939b88af88f15241ee45d5d003c81105890da670df6e203
Step 5 : RUN cross-build-start
--> Running in 1d0c6ff52fd3
--> a92560a622a5
Error removing intermediate container c2702bd608f7: nosuchcontainer: No such container: c2702bd608f796
2e2939b88af88f15241ee45d5d003c81105890da670df6e203
Step 6 : RUN apt-get update && apt-get install -y mosquitto-clients
--> Running in b94e9a36c402
Get:1 http://archive.raspbian.org jessie InRelease [14.9 kB]
Get:2 http://archive.raspbian.org jessie/main armhf Packages [12.5 MB]
Fetched 12.5 MB in 12s (1019 kB/s)
Reading package lists...
Reading package lists...
Building dependency tree...
The following extra packages will be installed:
  libc-ares2 libmosquitto1 libssl1.0.0
The following NEW packages will be installed:
  libc-ares2 libmosquitto1 libssl1.0.0 mosquitto-clients
0 upgraded, 4 newly installed, 0 to remove and 34 not upgraded.
Need to get 999 kB of archives.
After this operation, 2542 kB of additional disk space will be used.
Get:1 http://archive.raspbian.org/raspbian/ jessie/main libssl1.0.0 armhf 1.0.1t-1+deb8u2 [852 kB]
Get:2 http://archive.raspbian.org/raspbian/ jessie/main libc-ares2 armhf 1.10.0-2 [71.3 kB]
Get:3 http://archive.raspbian.org/raspbian/ jessie/main libmosquitto1 armhf 1.3.4-2 [36.3 kB]
Get:4 http://archive.raspbian.org/raspbian/ jessie/main mosquitto-clients armhf 1.3.4-2 [39.3 kB]
debconf: delaying package configuration, since apt-utils is not installed
Fetched 999 kB in 1s (621 kB/s)
Selecting previously unselected package libssl1.0.0:armhf.
(Reading database... 7096 files and directories currently installed.)
```

# Why resinOS

<https://resinos.io/>

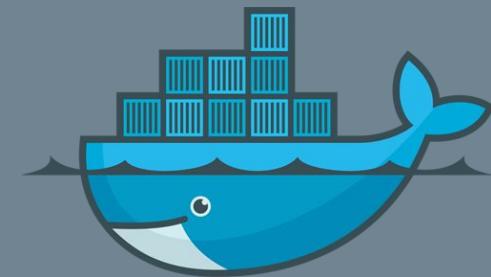


# 11. Demo TensorFlow with Docker



# Docker + TensorFlow + GPU

- Machine Learning, Deep Learning
- TensorFlow Docker images
- nvidia-docker, All-in-one DL image



# Exercise & Self-learning

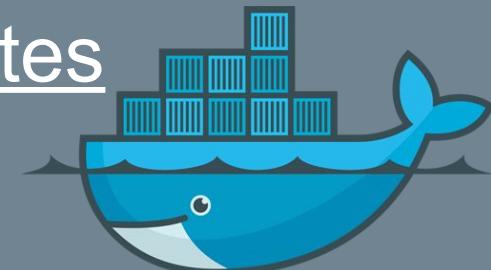
1. Docker Basic - Katacoda by Philipz
2. Docker Trainning
3. Docker Free self-paced courses
4. Docker Tutorials and Labs

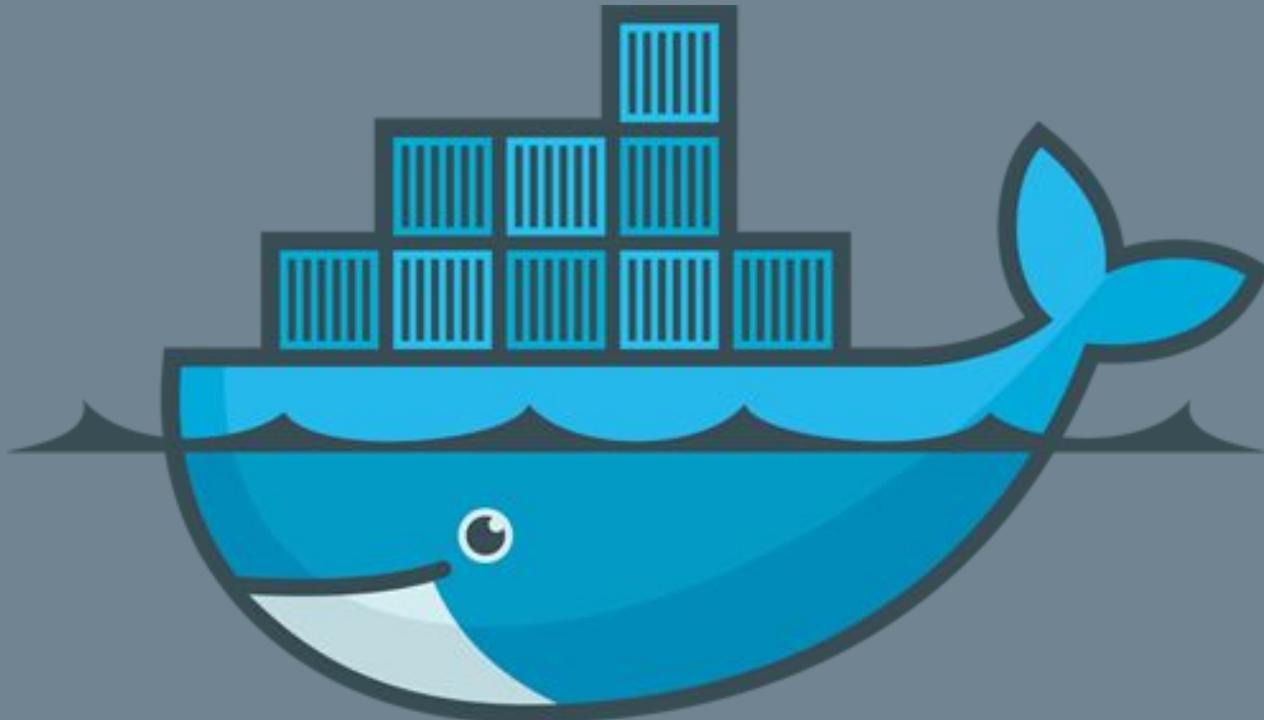
Online Self-learning

Offical Online Lab

Scalable Microservices with Kubernetes

- Udacity





Hope You Love Docker  
So long!