



Docker:Zero to Hero (Day 1)

By Praparn Luengphoonlap
Email: praparn@opcellent.com

Docker: The Next-Gen of Virtualization



Outline Day 1

- Docker principle
- Docker machine
- Image, Repository & Tag, container
- CPU, Memory and I/O
- Network
- Volume
- Inspect and Log
- Commit
- Docker Security
- Kitematic (Alpha)

Outline Day 2

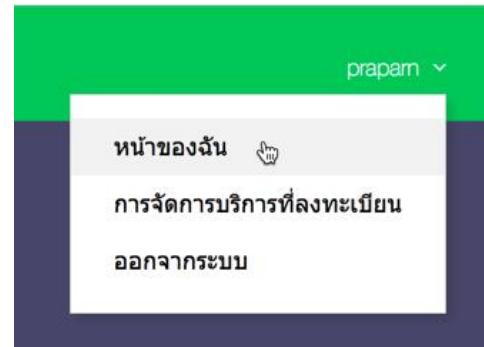
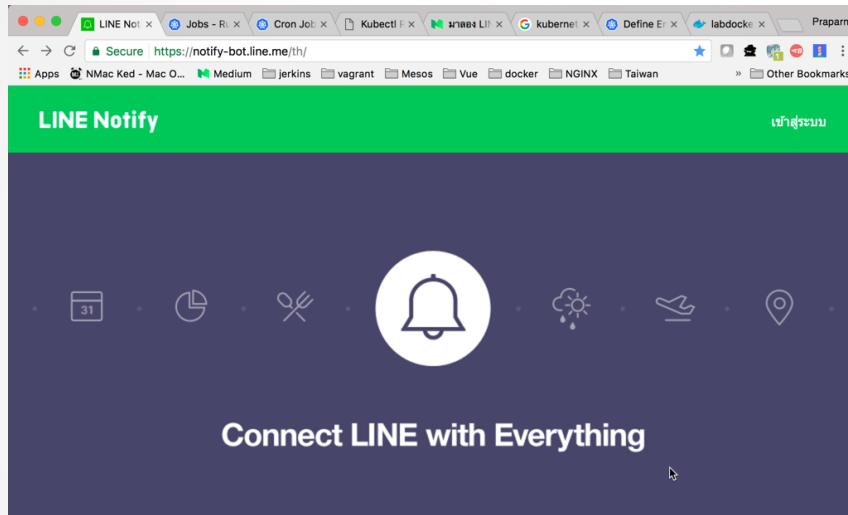
- Dockerfile and Build
- Compose
- Registry
- Swarm Mode
 - Conceptual of Swarm
 - Swarm Mode Architecture
 - Swarm init/join cluster system
 - Swarm service
 - Orchestrator Assignment
 - Config and Secret
 - Network Overlay and Ingress
 - HA Manager Role
 - Docker Stack Deploy (Compose Swarm Mode)
- portainer for Docker
- Q&A

Prerequisite

- Windows (64 bit) / Mac / Linux (64 bit) machine (ubuntu / alpine prefer)
- 1 email address (For register "hub.docker.com") / hub.docker.com account
- Line Notify Token (<https://notify-bot.line.me>)
- Tool for editor (vscode etc)
- Tool for shell (putty / terminal etc)
- Tool for transfer file (winscp / scp)
- Basic understand for linux operate
- Basic text editor skill (vim prefer) and linux structure
- Internet for download / upload image

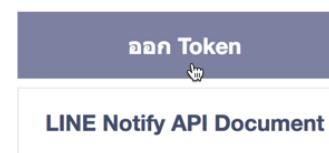
Prerequisite

- Generate LINE Token
 - <https://notify-bot.line.me>



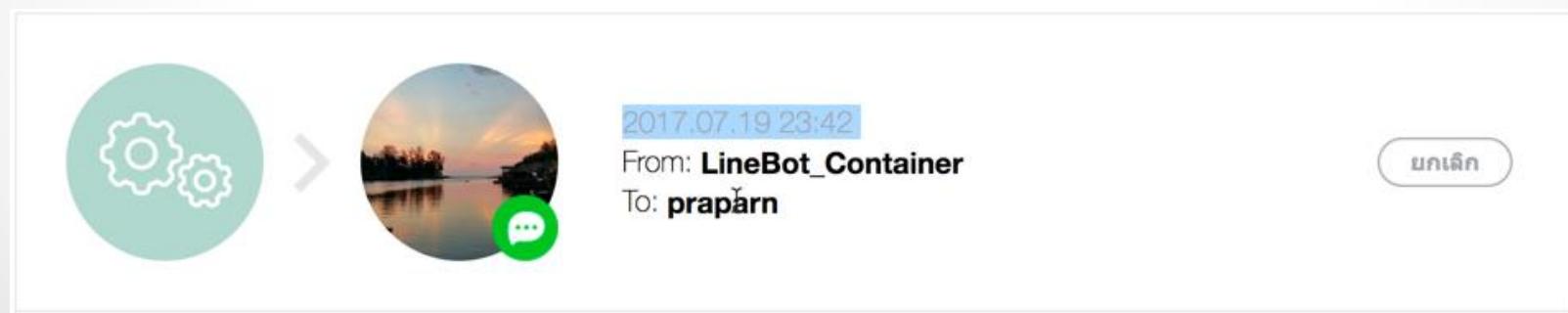
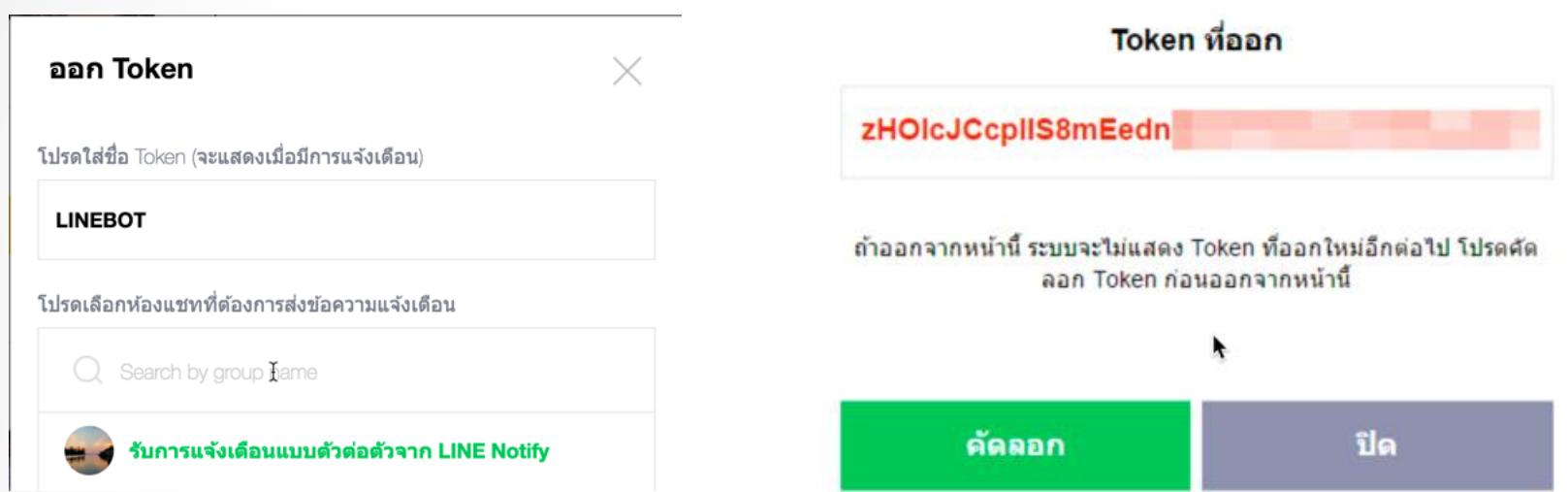
ออก Access Token (สำหรับผู้พัฒนา)

เมื่อใช้ Access Token แบบบุคคล จะสามารถตั้งค่าการแจ้งเตือนได้โดยไม่ต้องลงทะเบียนกับเว็บเซอร์วิส



Prerequisite

- Generate LINE Token
 - <https://notify-bot.line.me>



Lab Resource

- Repository for lab

The screenshot shows a web interface for managing Docker repositories. At the top, there is a navigation bar with icons for 'Explore' and 'Help', a search bar containing 'labdocker', and buttons for 'Sign up' and 'Log in'. Below the navigation bar, the title 'Repositories (7)' is displayed. A dropdown menu labeled 'All' is visible. The main content area lists seven repositories, each represented by a blue Docker icon, the repository name, its status (public), the number of stars, the number of pulls, and a 'DETAILS' button.

Repository	Status	Stars	Pulls	Details
labdocker/alpineweb	public	0	31	DETAILS
labdocker/nginx	public	0	16	DETAILS
labdocker/alpine	public	0	7	DETAILS
labdocker/centos	public	0	5	DETAILS
labdocker/debian	public	0	4	DETAILS
labdocker/fedora	public	0	3	DETAILS
labdocker/ubuntu	public	0	2	DETAILS

Lab Resource

- Software in lab

The screenshot shows a Windows file explorer window with the path: Computer > DATA (D:) > Docker_Nodejs > Workshop > Workshop_1-11_Registry. Inside this folder, there are three files: 'instruction' (Text Document, 4 KB), 'labdocker.com' (Security Certificate, 2 KB), and 'labdocker.com.key' (KEY File, 2 KB). Below the file explorer is a Notepad window titled 'instruction - Notepad'. The content of the Notepad is as follows:

Link for download:
<https://www.docker.com/products/docker-toolbox>

1. See PDF document for detail to install
2. After finished then run below command for check docker-machine (Command prompt)
 2.1 docker-machine --version ==> check version of docker machine & readiness
 2.2 docker-machine create --driver virtualbox labdocker ==> create new docker-machine for lab
 2.3 docker-machine ls ==> check ip address of new docker-machine

*Remark: default username/password for access docker-machine is docker/tcuser

3. SSH to docker-machine (labdocker)
 3.1 docker-machine ssh labdocker ==> default ssh via command prompt
 3.2 access via putty(windows) to ip address
 3.3 access via shell (mac)
 - Shell ==> New Remote Connection (Service: ssh)

4. Incase Upgrade docker-machine. Please check PDF document (Upgrade_Docker_1.10.pdf)

Lab Resource

- Download on Google Drive
 - <https://tinyurl.com/ycsln99d>
- Download on GitHub
 - git@github.com:praparn/docker_workshop_112018.git

[praparn / docker_workshop_112018](#)

Code Issues Pull requests Projects Wiki Insights Settings

docker_workshop_112018 Edit

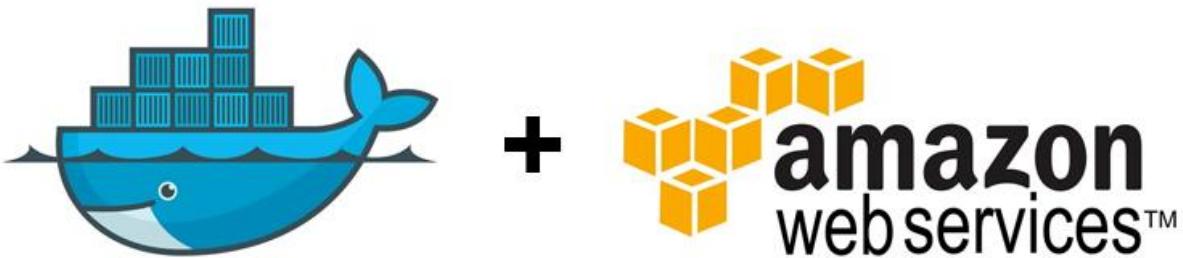
Manage topics

2 commits 1 branch 0 releases 1 contributor

Branch: master New pull request Create new file Upload files Find file Clone or download

praparn 20181020094803	Latest commit 411d8a7 13 hours ago
Workshop_1-1_Download_Install	20181020094803 13 hours ago
Workshop_1-2_Access_PoolImage	20181020094803 13 hours ago
Workshop_1-3_Create_PushImage	20181020094803 13 hours ago
Workshop_1-4_Run_Container	20181020094803 13 hours ago
Workshop_1-5_CPU_Memory_IO	20181020094803 13 hours ago
Workshop_1-6_Network	20181020094803 13 hours ago
Workshop_1-7_Volume	20181020094803 13 hours ago
Workshop_1-8_Inspect_Logs	20181020094803 13 hours ago
Workshop_1-9_Docker_Security	20181020094803 13 hours ago
Workshop_2-1_DockerFile	20181020094803 13 hours ago

Workshop 1-1: Access Docker on AWS



Docker on AWS

Docker: The Next-Gen of Virtualization



Workshop 1-1: Access Docker on AWS

https://labs.play-with-docker.com

NMac Ked - Mac ... Medium Infra NOVA jenkins vagrant Mesos Vue docker NGINX Taiwan Kubernetes PWA_PR

Contribute

The screenshot shows the Play with Docker interface. At the top left is a blue whale logo with shipping containers stacked on its back. Below it, the text "Play with Docker" is displayed, followed by the subtitle "A simple, interactive and fun playground to learn Docker". A "Login" button is visible. To the right, a session window titled "bf5kihea_bf5kikeac3u000bcr8h0" shows a timer at "03:59:29" and a "CLOSE SESSION" button. Below the timer are tabs for "Instances" and "SSH". Under "Instances", there is a single entry: "192.168.0.28 node1". Under "SSH", the command "ssh ip172-18-0-4-bf5kiheac3u000bcr8gg@direct.labs.play" is shown with a copy icon. At the bottom is a terminal window with the following text:

```
[node1] (local) root@192.168.0.28 ~
$ docker version
Client:
  Version: 18.03.1-ce
  API version: 1.37
  Go version: go1.9.2
  Git commit: 9ee9f40
  Built: Thu Apr 26 07:12:25 2018
  OS/Arch: linux/amd64
  Experimental: false
  Orchestrator: swarm

Server:
  Engine:
    Version: 18.03.1-ce
    API version: 1.37 (minimum version 1.12)
    Go version: go1.9.5
    Git commit: 9ee9f40
    Built: Thu Apr 26 07:23:03 2018
    OS/Arch: linux/amd64
    Experimental: true
[node1] (local) root@192.168.0.28 ~
```

Docker: The Next-Gen of Virtualization



Who are we ? (Opcellent)



Seamless development on cloud with K8S



● SERVICE

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s.



docker

Lorem Ipsum is simply dummy text of the printing and typesetting industry.



kubernetes

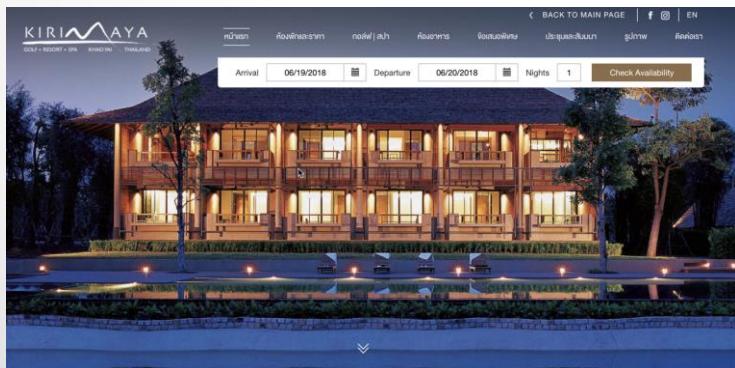
Lorem Ipsum is simply dummy text of the printing and typesetting industry.

● TRAINING

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s.



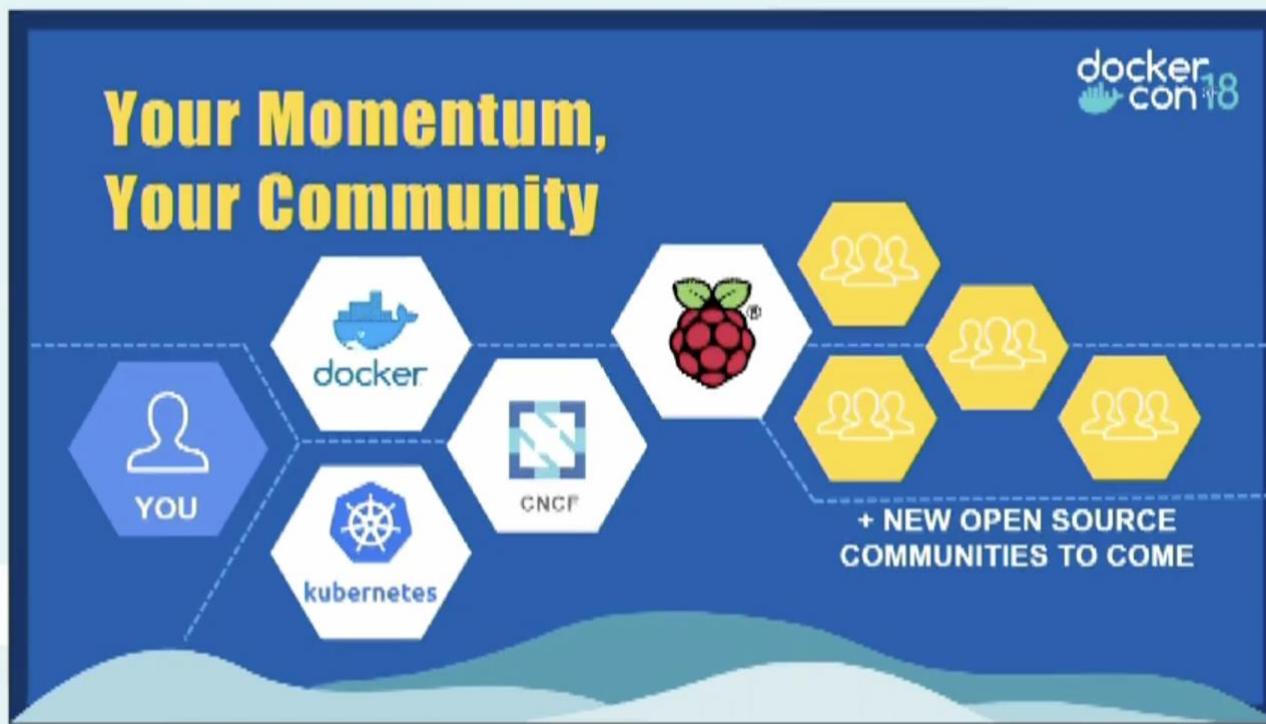
A photograph of a person sitting at a desk with a laptop, looking at a presentation slide on the screen. The slide features a blue and white design with the word 'GO' and some text.



Landscape of the world now



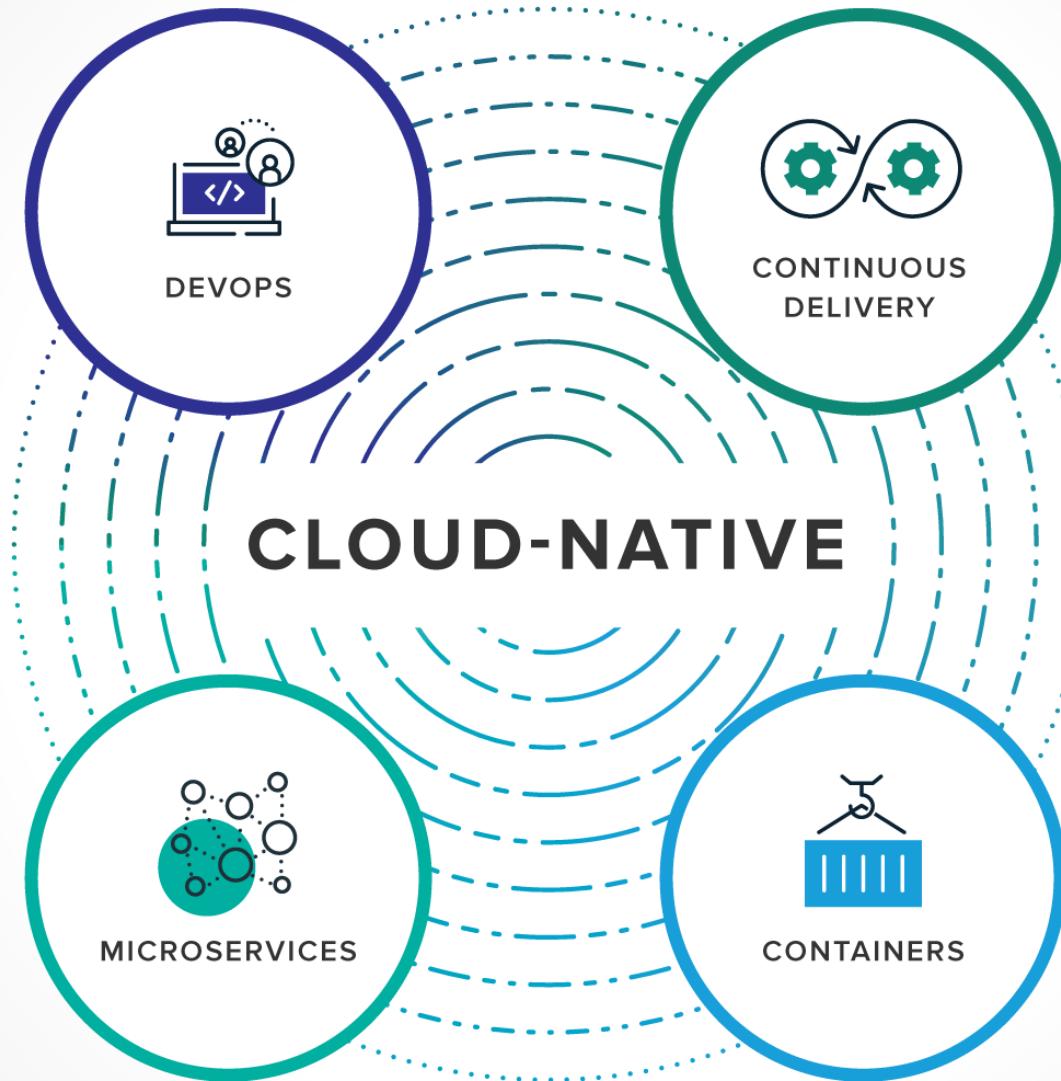
Landscape of the world now



Kubernetes: Production Workload Orchestration



Landscape of the world now



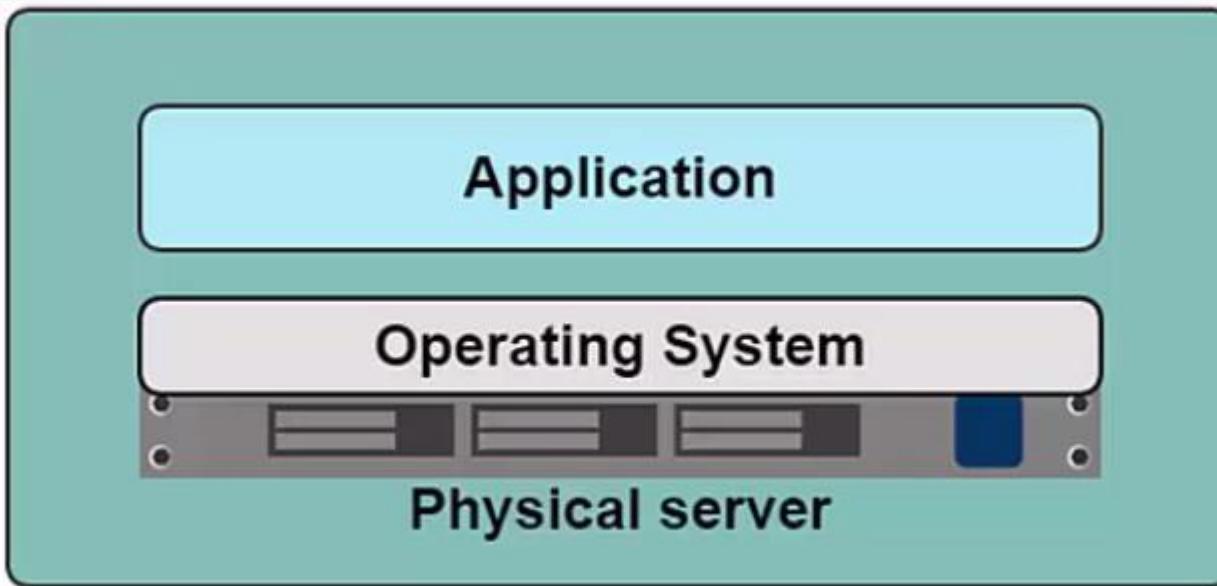
Kubernetes: Production Workload Orchestration



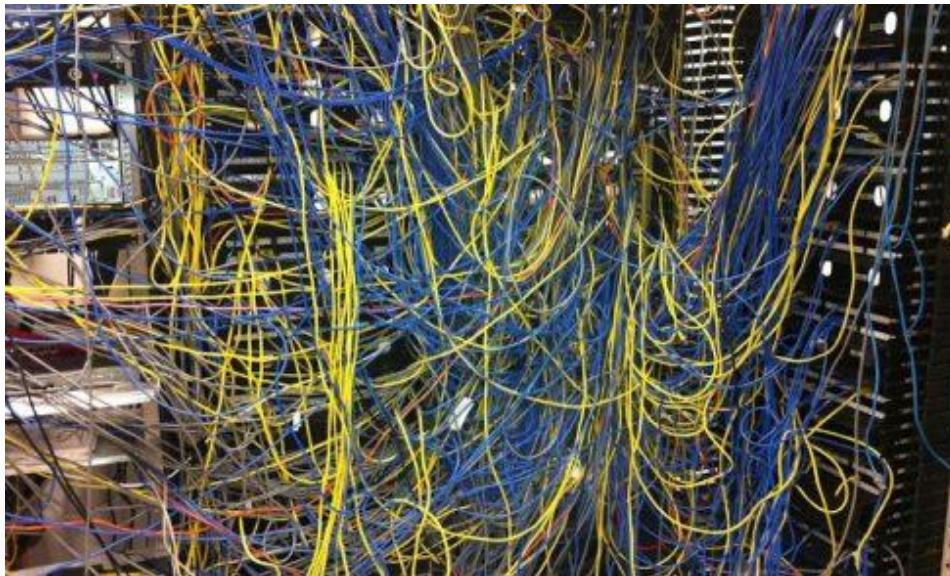
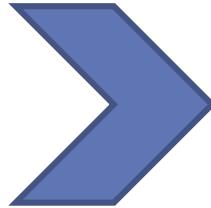
Docker Principle

• • •

What is docker ?



Existing Technology

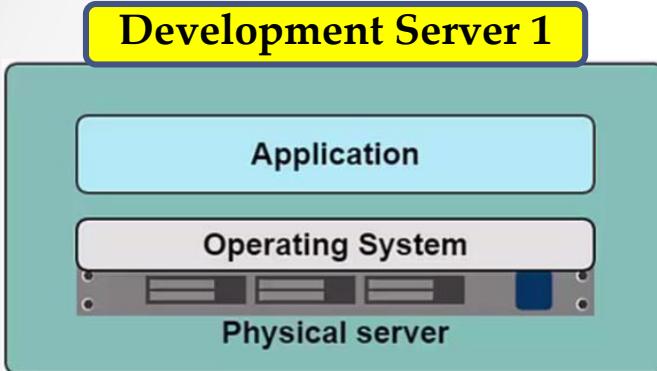


Docker: The Next-Gen of Virtualization

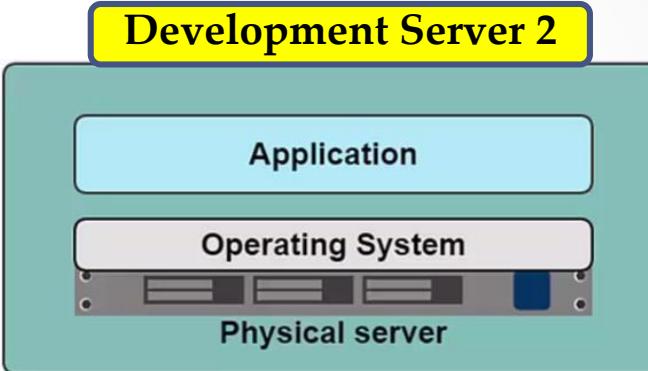


Existing Technology

- Development Environment (Mix everything possible)



- Apache 2.20 Web Server
 - PHP 5.5 Engine
 - Laravel 4.1 Framework
 - MySQL 5.1
 - Etc
- (All-in-One Server)

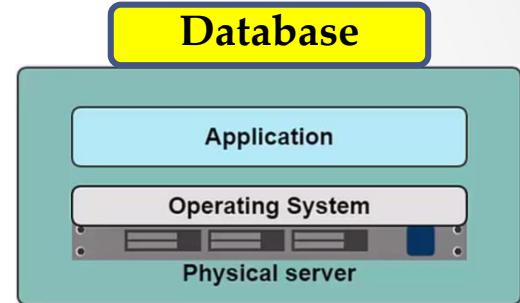
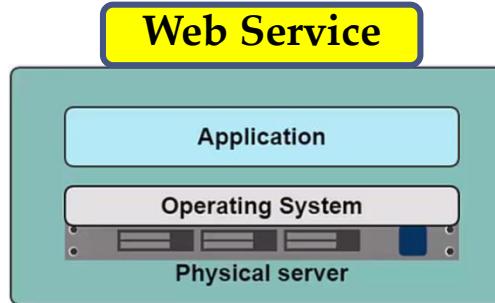
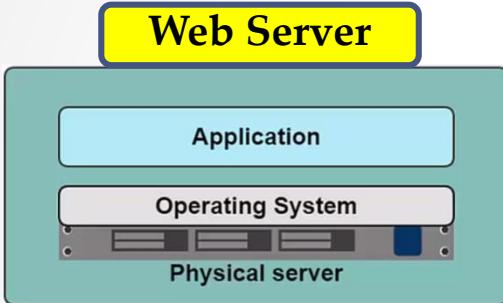


- IIS 8.0
- .Net Framework 3.5
- ASP.NET
- Etc

- Need concern about conflict component
- Survive among legacy dependency
- Lack for environment for fulfill develop & test (module test/integrate test/ UAT test / MOT test etc)
- Unexpected software conflict frequently occur
- Incomplete software's integrated test

Existing Technology

- Production Environment (Best design)
- Day 1: Application 1: Implement



- Apache 2.20 Web Server
- PHP 5.5 Engine
- Laravel 4.1 Framework

- IIS 8
- .Net Framework 3.5

- MariaDB 5.1

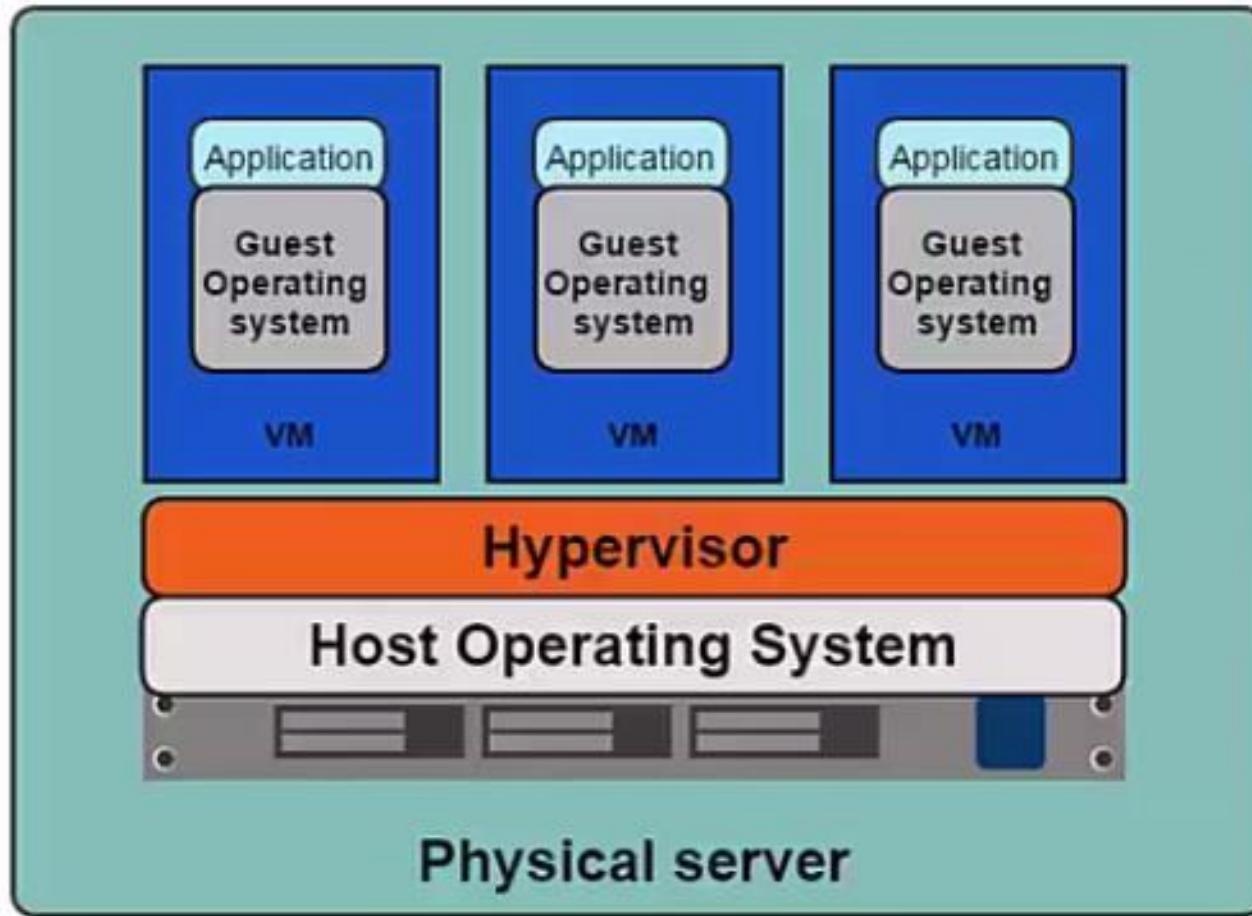
- Day 2: Application 2: Need to implement
 - Need PHP 7.0 ?
 - MariaDB 10.1.14 (Need search feature on 10.1)
- Problem ?
 - Possible to upgrade PHP to 7.0 ? / How to test existing application ?
 - What effect to MariaDB upgrade ?

Existing Technology

- What operation handle in production environment ?
 - Aware about huge of difference software component between development / production machine
 - Need to know in deep all application dependency (Wow !!!)
 - Take time for discussion and find solution to implement.
 - Possible to confusion and effect to other application that existing on share server.
 - Many unexpected problem & software bug.
 - Hard control software's quality assurance (QA)

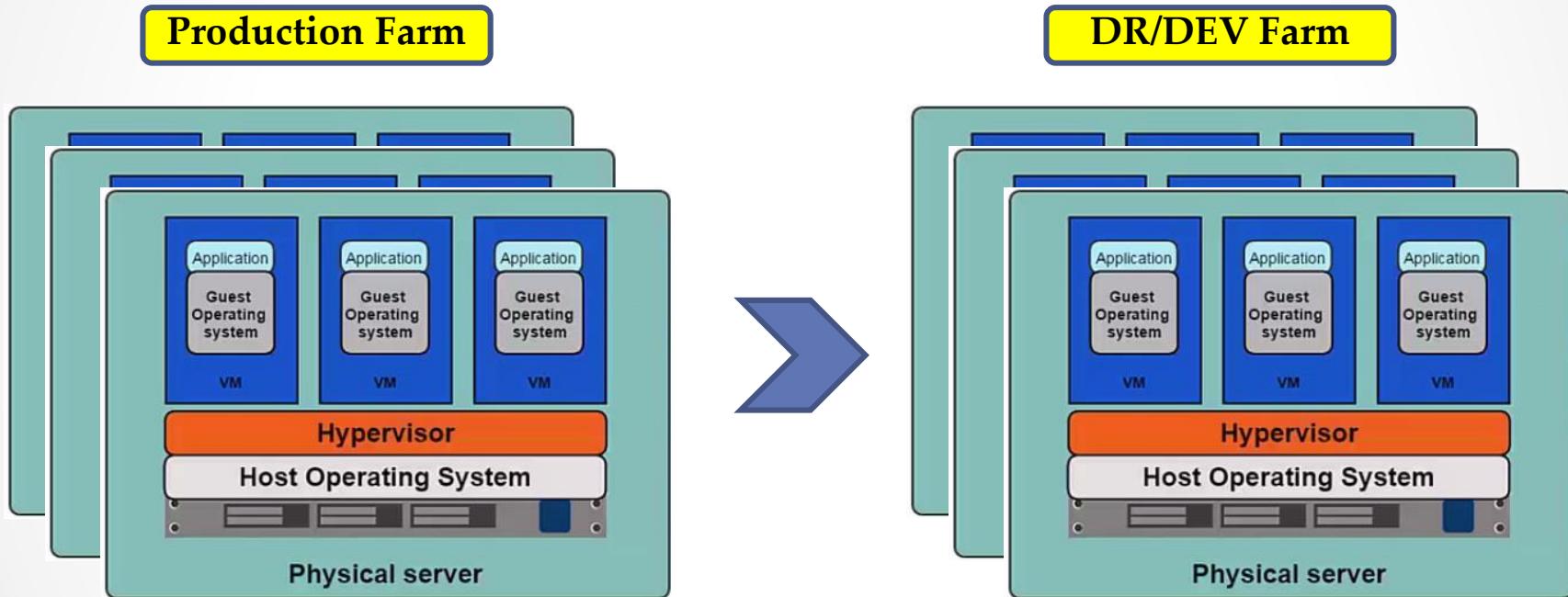


What is docker ?



Existing Technology

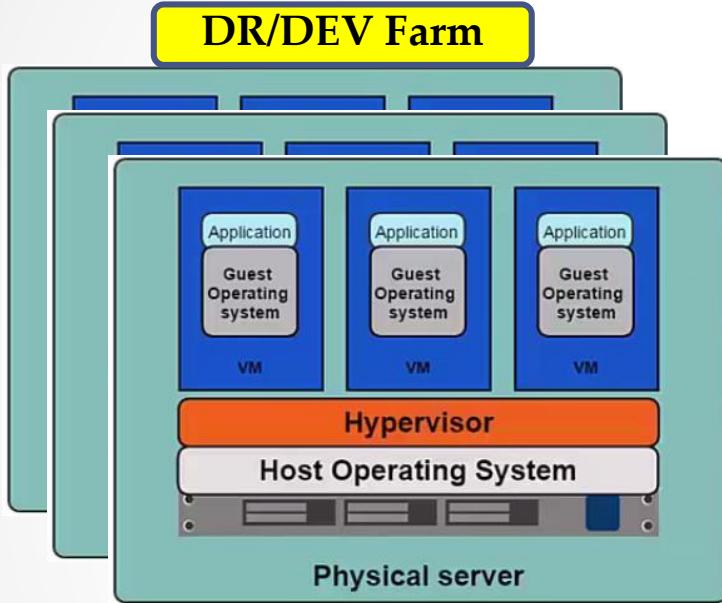
- 1 Physical server : 1 – N VMWare (&OS)
- Virtualize Hardware (CPU, Memory, Disk, Network etc)



- Kernel-base virtual machine (KVM), Vmware, Virtualbox etc

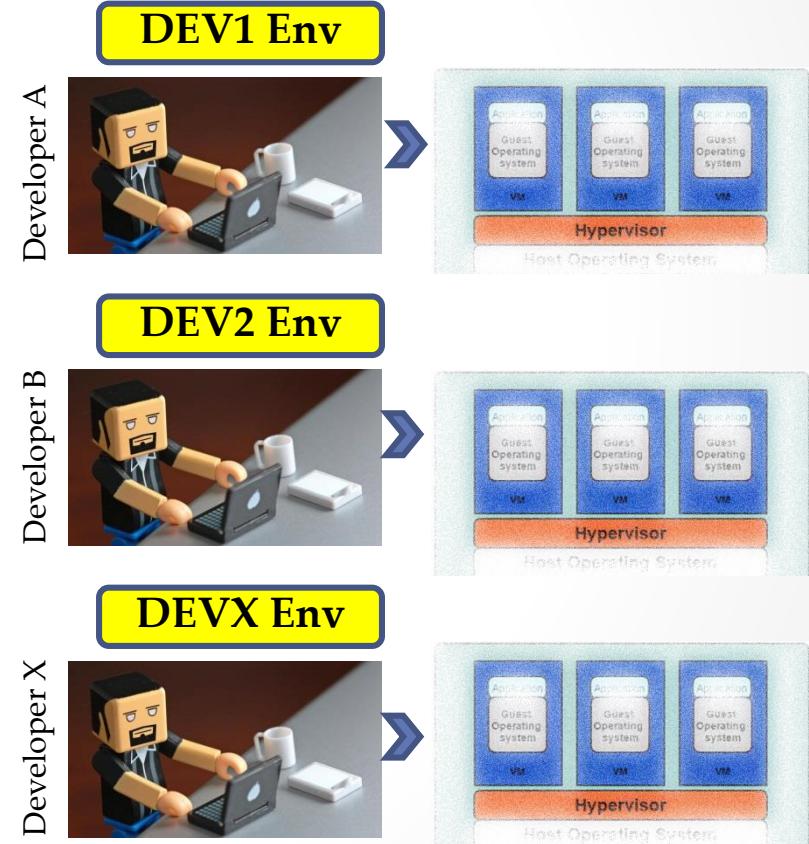
Existing Technology

- Development Environment (Clone from Production)



Virtual Machine reach limit:

- Resource insufficient (Almost from disk)
- Conflict version
- Huge of disk duplication
- Conflict & dependency still exist



Docker: The Next-Gen of Virtualization



Existing Technology

- Production Environment
- Day 1: Application 1: Implement

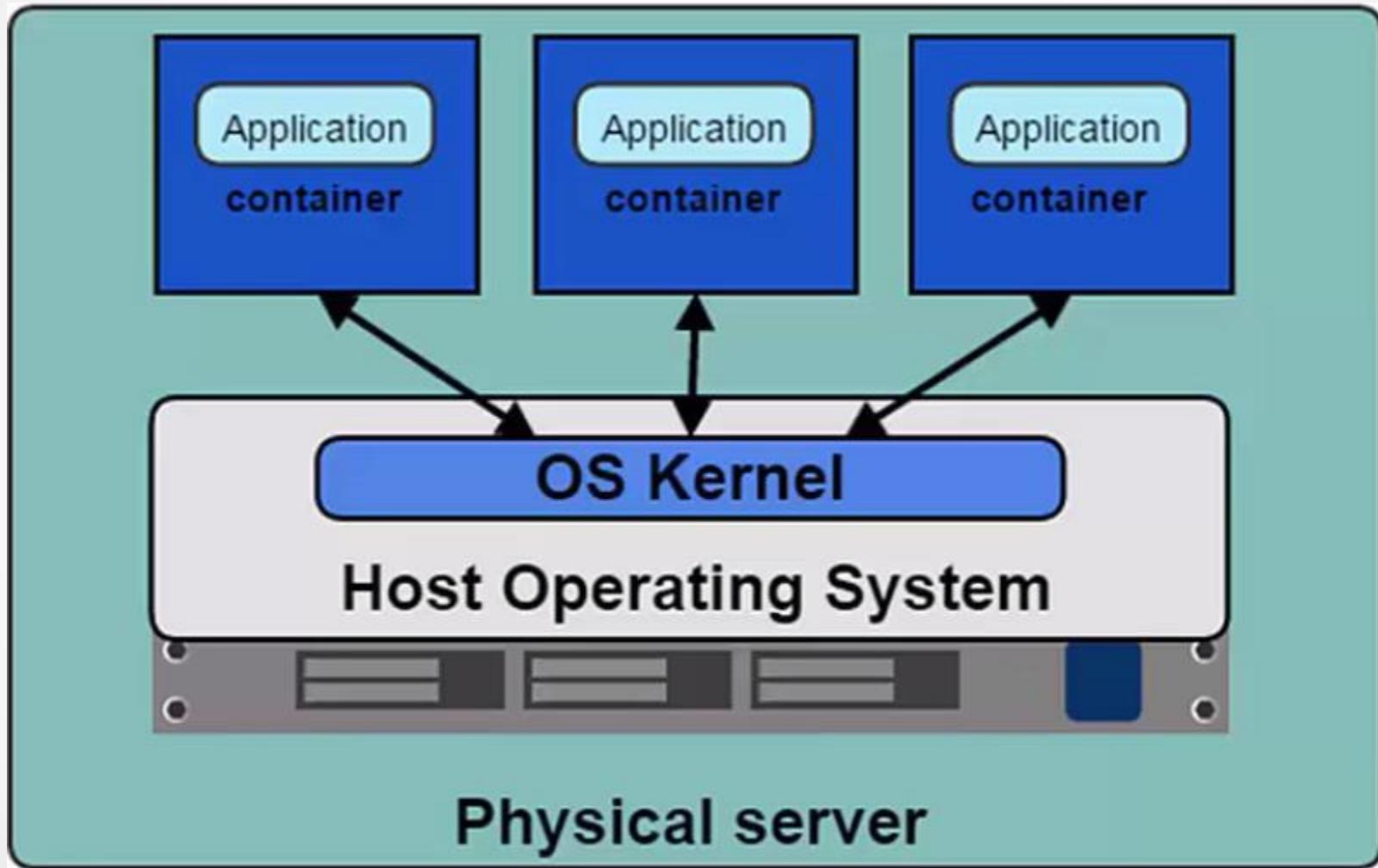


- Apache 2.20 Web Server
 - PHP 5.5 Engine
 - Laravel 4.1 Framework
- IIS 8
 - .Net FrameWork 3.5

- MariaDB 5.1

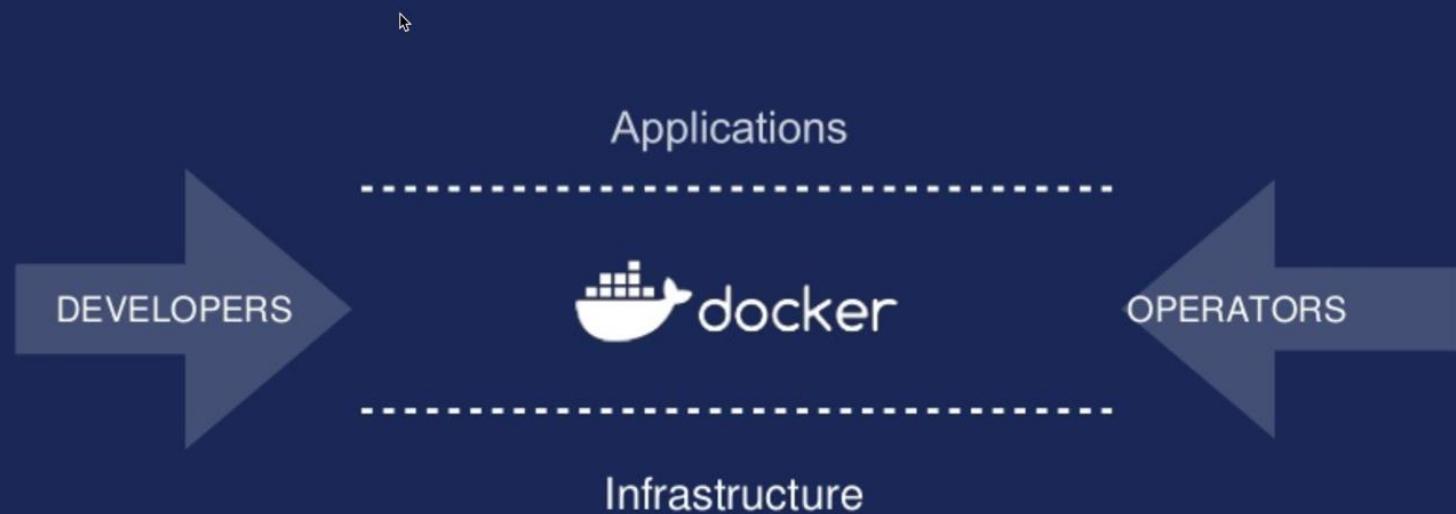
- Day 2: Application 2: Need to implement
 - Need PHP 7.0 ?
 - MariaDB 10.1.14 (Need search feature on 10.1)
- So... The problem still exist.

What is docker ?



What is docker ?

The Docker Platform in a nutshell



What is docker ?

Core Principles of the Docker Platform

INDEPENDENCE



OPENNESS



SIMPLICITY



 dockercon¹⁷ EU

What is docker ?

- Docker คือ open platform solution ที่ทำงานภายใต้คอนเซ็ปต์ของ container virtualize technology (operating -system –level virtualization)
- ผู้ใช้สามารถสร้างสภาพแวดล้อมเพื่อใช้ในพัฒนาโปรแกรมและส่งมอบเพื่อใช้งานในสภาพเดียวกัน
- Build, Ship, Run
- เหมาะสำหรับ Developer, DevOps, Architecture, Engineer
- เขียนด้วยภาษา Go
- รองรับการติดตั้งบนบนลินุกซ์ 64 บิต (kernel 3.1.0) (Official)
 - Ubuntu (X64/ARM/ARM64)
 - Debian (X64/ARM/ARM64)
 - CentOS
 - Fedora
 - Red Hat Enterprise Linux (Enterprise Edition)
 - Microsoft Windows Server 2016 (Enterprise Edition)
 - Microsoft Windows Server 1709 (Enterprise Edition)
 - Microsoft Windows Server 1803 (Enterprise Edition)
 - Suse Linux Enterprise Server (Enterprise Edition)

Operating-system-level virtualization

Mechanism	Operating system	License	Available since/between	Features									
				File system isolation	Copy on Write	Disk quotas	I/O rate limiting	Memory limits	CPU quotas	Network isolation	Nested virtualization	Partition checkpointing and live migration	Root privilege isolation
chroot	most UNIX-like operating systems	varies by operating system	1982	Partial ^[3]	No	No	No	No	No	No	Yes	No	No
Docker Linux-VServer (security context)	Linux ^[7]	Apache License 2.0	2013	Yes	Yes	Not directly	Not directly	Yes	Yes	Yes	Yes	No	No
		GNU GPLv2	2001	Yes	Yes	Yes	Yes ^[8]	Yes	Yes	Partial ^[9]	?	No	Partial ^[9]
lxc LXC	Linux	Apache License 2.0	2013	Yes	Yes	Yes	Yes ^[9]	Yes	Yes	Partial ^[9]	?	No	Partial ^[9]
		GNU GPLv2	2008	Yes ^[9]	Yes	Partial ^[9]	Partial ^[9]	Yes	Yes	Yes	Yes	No	Yes ^[9]
LXD	Linux	Apache License 2.0	2015	Yes	Yes	Partial (see LXC)	Partial (see LXC)	Yes	Yes	Yes	Yes	Partial ^[9]	Yes
OpenVZ	Linux	GNU GPLv2	2005	Yes	No	Yes	Yes ^[10]	Yes	Yes	Yes ^[10]	Partial ^[10]	Yes	Yes ^[10]
Virtuozzo	Linux, Windows	Proprietary	2000 ^[14]	Yes	Yes	Yes	Yes ^[11]	Yes	Yes	Yes ^[11]	Partial ^[11]	Yes	Yes
Solaris Containers (Zones)	illumos (OpenSolaris), Solaris	CDDL, Proprietary	2004	Yes	Yes (ZFS)	Yes	Partial ^[12]	Yes	Yes	Yes ^{[12][17][18]}	Partial ^[12]	Partial ^[12]	Yes ^[8]
FreeBSD jail	FreeBSD	BSD License	2000 ^[20]	Yes	Yes (ZFS)	Yes ^[13]	No	Yes ^[21]	Yes	Yes ^[22]	Yes	No	Yes ^[23]
sysjail	OpenBSD, NetBSD	BSD License	2006–2009 (As of March 3, 2009, it is no longer supported)	Yes	No	No	No	No	No	Yes	No	No	?
				Yes	No	Yes	Yes	Yes	Yes	Yes ^[14]	No	Yes ^[25]	?
WPARs	AIX	Proprietary	2007	Yes	No	Yes	Yes	Yes	Yes	Yes ^[14]	No	Yes ^[25]	?
HP-UX Containers (SRP) ^[9]	HPUX	Proprietary	2007	Yes	No	Partial ^[14]	Yes	Yes	Yes	Yes	?	Yes	?
iCore Virtual Accounts	Windows XP	Proprietary/Freeware	2008	Yes	No	Yes	No	No	No	No	?	No	?
Sandboxie Spoon	Windows	Proprietary/Shareware	2004	Yes	Yes	Partial	No	No	No	Partial	No	No	Yes
			2012	Yes	Yes	No	No	No	No	Yes	No	No	Yes
VMware ThinApp	Windows	Proprietary	2008	Yes	Yes	No	No	No	No	Yes	No	No	Yes

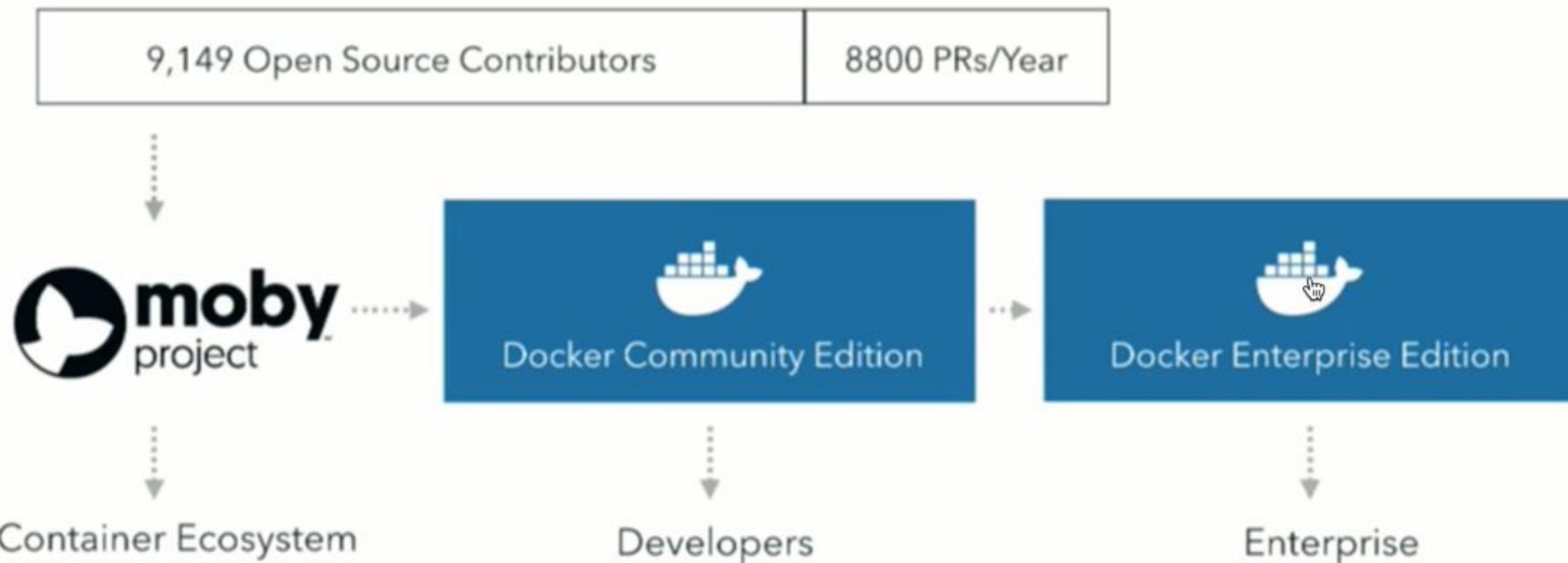
Reference: https://en.wikipedia.org/wiki/Operating-system-level_virtualization

Docker: The Next-Gen of Virtualization



What is docker ?

The Docker Innovation Model



What is docker ?

The chart compares the supported platforms for Docker EE and CE. Both editions support Windows Server, Oracle Linux, and SUSE Linux Enterprise Server. Docker EE also supports Amazon Web Services, Microsoft Azure, and Red Hat. Docker CE supports Apple, Amazon Web Services, CentOS, Microsoft Azure, Ubuntu, and Red Hat.

Enterprise Edition (EE)	Community Edition (CE)
Windows Server	Windows Server
ORACLE LINUX	ORACLE LINUX
SUSE Linux Enterprise Server	SUSE Linux Enterprise Server
amazon web services	amazon web services
Microsoft Azure	Microsoft Azure
ubuntu	ubuntu
redhat	redhat

Docker Enterprise Edition (EE) and Community Edition (CE)

Enterprise Edition (EE)	Community Edition (CE)
<ul style="list-style-type: none">CaaS enabled platform subscription (integrated container orchestration, management and security)Enterprise class supportQuarterly releases, supported for one year each with backported patches and hotfixes.Certified Infrastructure, Plugins, Containers	<ul style="list-style-type: none">Free Docker platform for "do it yourself" dev and opsMonthly Edge release with latest features for developersQuarterly release with maintenance for ops

Lifecycle

Squaring the circle: Faster releases and better stability



Docker EE Availability

Partners providing Docker EE support include:

- From Docker: Docker Sales Team
- OEM: Direct L2 / L2 Support Included: Alibaba Cloud, Hewlett Packard Enterprise, Amazon Web Services, CANONICAL, IBM, cloudera, Microsoft, and Microsoft Azure.
- Cloud Marketplaces: Docker store, DVAR, and Windows Server 2016.

<https://blog.docker.com/2018/07/extending-support-cycle-docker-community-edition/>

Docker: The Next-Gen of Virtualization



What is docker ?



EXTENDING SUPPORT CYCLE FOR DOCKER COMMUNITY EDITION

By [Sebastiaan Van Stijn](#) July 18, 2018

CE, docker, Docker 18.06 CE, Docker Community Edition, engine

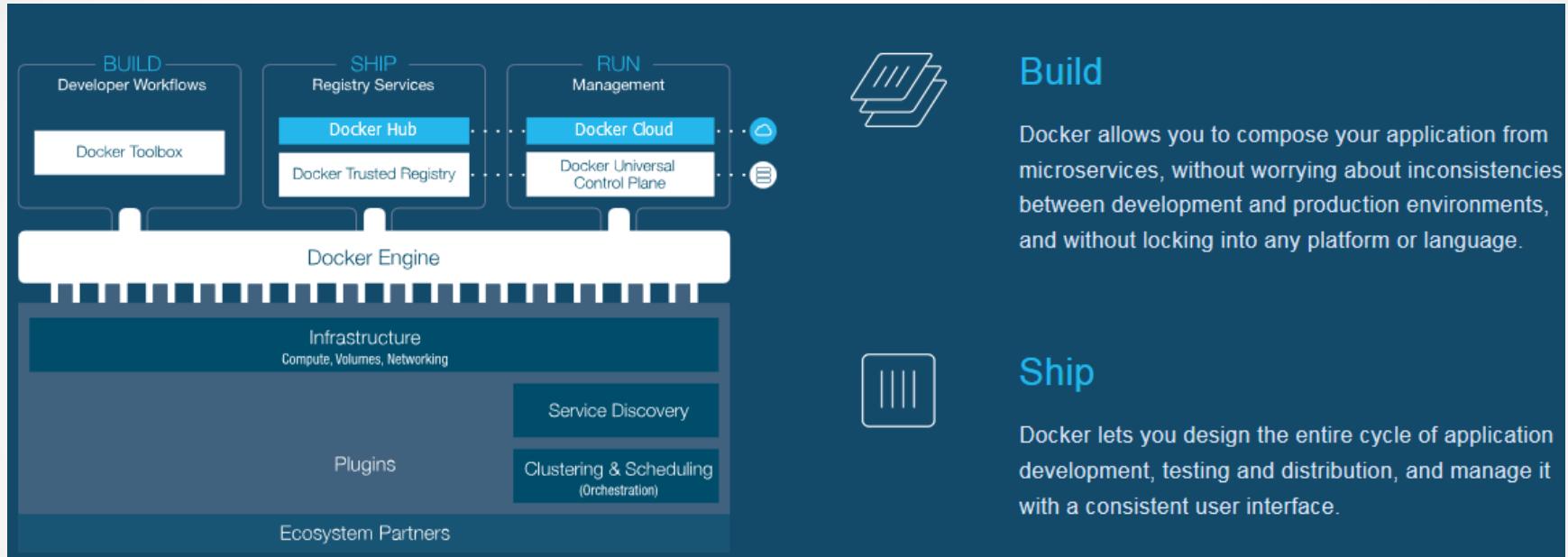
We're excited to share the release of [Docker 18.06 Community Edition \(CE\)](#) and also share some changes that will be implemented in the next release. Based on feedback we've been hearing from the community, we are implementing some changes to deliver higher quality Community Edition (CE) releases, while also providing faster access to new features as they get added:

- [Docker CE Stable is changing to twice-a-year release cadence](#)
- [Docker CE Edge is deprecated in favor of a nightly build channel](#)

As a result of these changes, Docker 18.06 CE will be the last release with a 4-month maintenance lifecycle. The planned Docker 18.09 CE release will be supported for 7 months with Docker 19.03 CE being the next release in line. Further, the monthly Edge releases of Docker CE (Engine) are now replaced with nightly builds. Note that Docker Desktop ([Docker for Mac](#), [Docker for Windows](#)) edge channels will still provide monthly updates.

<https://blog.docker.com/2018/07/extending-support-cycle-docker-community-edition/>

What is docker ?



What is docker ?



Docker Machine

Automated Docker provisioning



Docker Swarm

Host clustering and container scheduling



Docker Compose

Define multi-container applications



Docker Registry

Open source Docker image distribution (not included)



Docker Engine

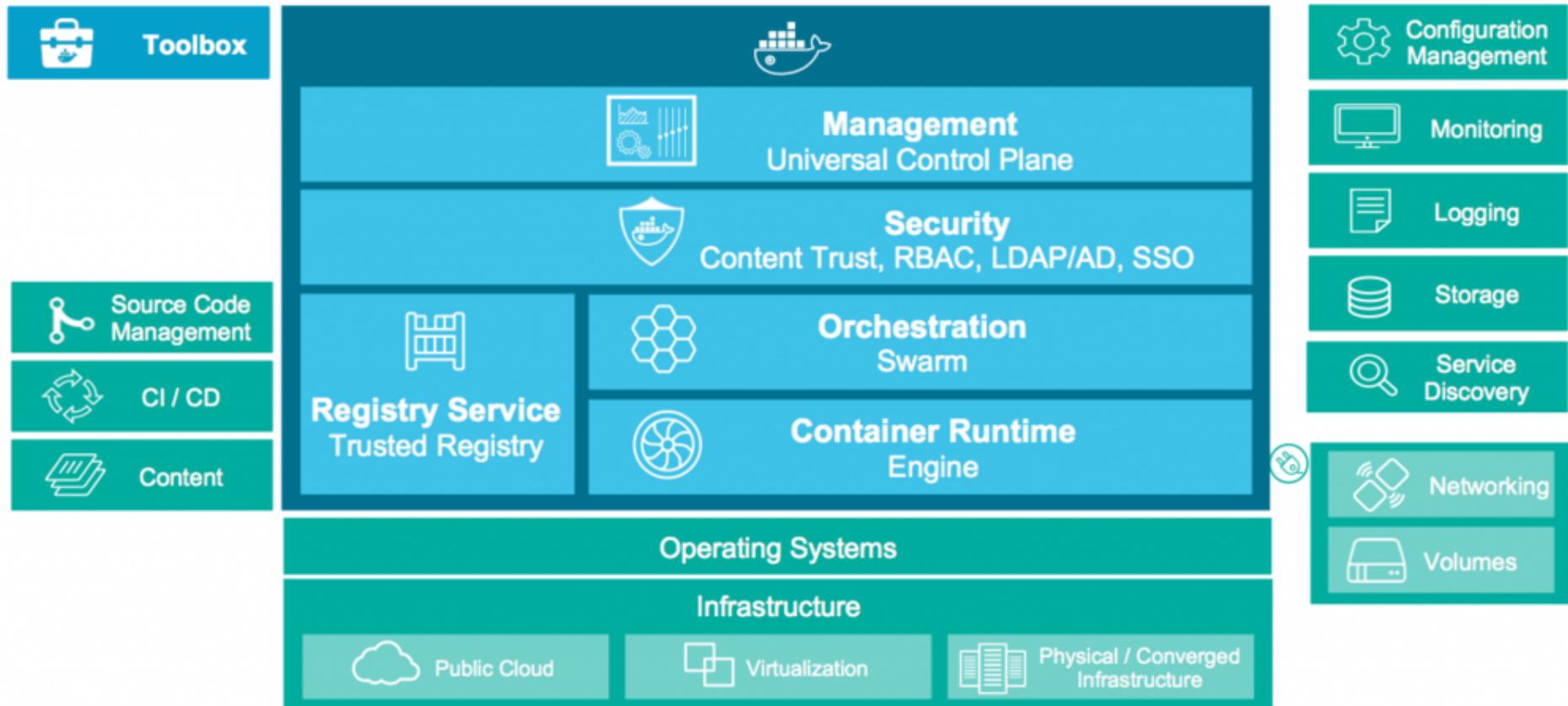
Creates and runs Docker containers



Kitematic

Desktop GUI for Docker

What is docker ?



Docker: The Next-Gen of Virtualization



What is docker ?

- DOCKER CE

DESKTOP		x86_64			
Platform					
Docker for Mac (macOS)		✓			
Docker for Windows (Microsoft Windows 10)		✓			
SERVER					
Platform	x86_64 / amd64	ARM	ARM64 / AARCH64	IBM Power (ppc64le)	IBM Z (s390x)
CentOS	✓		✓		
Debian	✓	✓	✓		
Fedor a	✓				
Ubuntu	✓	✓	✓	✓	✓

What is docker ?

- DOCKER EE

On-premises

These are the operating systems where you can install Docker EE.

Platform	x86_64 / amd64	IBM Power (ppc64le)	IBM Z (s390x)
CentOS	✓		
Oracle Linux	✓		
Red Hat Enterprise Linux	✓	✓	✓
SUSE Linux Enterprise Server	✓	✓	✓
Ubuntu	✓	✓	✓
Microsoft Windows Server 2016	✓		
Microsoft Windows Server 1709	✓		
Microsoft Windows Server 1803	✓		

When using Docker EE Standard or Advanced

IBM Power is not supported as managers or workers. Microsoft Windows Server is not supported as a manager. Microsoft Windows Server 1803 is not supported as a worker.

Docker Certified Infrastructure

Docker Certified Infrastructure is Docker's prescriptive approach to deploying Docker Enterprise Edition (EE) on a range of infrastructure choices. Each Docker Certified Infrastructure includes a reference architecture, automation templates, and third-party ecosystem solution briefs.

Platform	Docker Enterprise Edition
VMware	✓
Amazon Web Services	✓
Microsoft Azure	✓
IBM Cloud	Coming soon

What is docker ?

- Compatible Matrix (Docker EE 2.0)

OS Distribution (x86_64)	Enterprise Engine	UCP	DTR	Storage Driver	Orchestration	DTR Storage Backend
RHEL 7.3	17.06 starting with 17.06.2-ee-8	3.0.x	2.5.x	overlay2, devicemapper	Swarm mode, Kubernetes	NFSv3, Amazon S3, S3 Compliant Alternatives, Azure Storage (Blob), Google Cloud Storage, OpenStack Swift, Local Filesystem
RHEL 7.4	17.06 starting with 17.06.2-ee-8	3.0.x	2.5.x	overlay2, devicemapper	Swarm mode, Kubernetes	NFSv3, Amazon S3, S3 Compliant Alternatives, Azure Storage (Blob), Google Cloud Storage, OpenStack Swift, Local Filesystem
RHEL 7.5	17.06 starting with 17.06.2-ee-14	3.0.x starting with 3.0.2	2.5.3 and higher	overlay2	Swarm mode, Kubernetes	NFSv3, Amazon S3, S3 Compliant Alternatives, Azure Storage (Blob), Google Cloud Storage, OpenStack Swift, Local Filesystem
SLES 12 SP2	17.06 starting with 17.06.2-ee-8	3.0.x	2.5.x	btrfs	Swarm mode, Kubernetes	NFSv3, Amazon S3, S3 Compliant Alternatives, Azure Storage (Blob), Google Cloud Storage, OpenStack Swift, Local Filesystem
SLES 12 SP3	17.06 starting with 17.06.2-ee-11	3.0.x starting with 3.0.1	2.5.1 and higher	btrfs, overlay2	Swarm mode, Kubernetes	NFSv3, Amazon S3, S3 Compliant Alternatives, Azure Storage (Blob), Google Cloud Storage, OpenStack Swift, Local Filesystem
Ubuntu 14.04	17.06 starting with 17.06.2-ee-8	3.0.x	2.5.x	aufs	Swarm mode, Kubernetes	NFSv3, Amazon S3, S3 Compliant Alternatives, Azure Storage (Blob), Google Cloud Storage, OpenStack Swift, Local Filesystem
Ubuntu 16.04	17.06 starting with 17.06.2-ee-8	3.0.x	2.5.x	overlay2, aufs	Swarm mode, Kubernetes	NFSv3, Amazon S3, S3 Compliant Alternatives, Azure Storage (Blob), Google Cloud Storage, OpenStack Swift, Local Filesystem
Ubuntu 18.04	17.06 starting with 17.06.2-ee-16	3.0.x starting with 3.0.3	2.5.x starting with 2.5.4	overlay2	Swarm mode, Kubernetes	NFSv3, Amazon S3, S3 Compliant Alternatives, Azure Storage (Blob), Google Cloud Storage, OpenStack Swift, Local Filesystem
CentOS 7	17.06 starting with 17.06.2-ee-8	3.0.x	2.5.x	overlay2, devicemapper	Swarm mode, Kubernetes	NFSv3, Amazon S3, S3 Compliant Alternatives, Azure Storage (Blob), Google Cloud Storage, OpenStack Swift, Local Filesystem
Oracle Linux 7.3 ¹	17.06 starting with 17.06.2-ee-8	3.0.x	2.5.x	overlay2, devicemapper ²	Swarm mode, Kubernetes	NFSv3, Amazon S3, S3 Compliant Alternatives, Azure Storage (Blob), Google Cloud Storage, OpenStack Swift, Local Filesystem
Windows Server 2016	17.06 starting with 17.06.2-ee-8	3.0.x ³	n/a ⁵	windowsfilter	Swarm mode	NFSv3, Amazon S3, S3 Compliant Alternatives, Azure Storage (Blob), Google Cloud Storage, OpenStack Swift, Local Filesystem
Windows Server, version 1709	17.06 starting with 17.06.2-ee-14	3.0.x starting with 3.0.2 ³	n/a ⁵	windowsfilter	Swarm mode	NFSv3, Amazon S3, S3 Compliant Alternatives, Azure Storage (Blob), Google Cloud Storage, OpenStack Swift, Local Filesystem
Windows Server, version 1803	17.06 starting with 17.06.2-ee-16	3.0.x starting with 3.0.3 ³	n/a ⁵	windowsfilter	Swarm mode	NFSv3, Amazon S3, S3 Compliant Alternatives, Azure Storage (Blob), Google Cloud Storage, OpenStack Swift, Local Filesystem

IBM Z (s390x) ⁴	Enterprise Engine	UCP	DTR	Storage Driver	Orchestration	DTR Storage Backend
SLES 12 SP3	17.06 starting with 17.06.2-ee-16	3.0.x starting with 3.0.3		overlay2	Swarm mode, Kubernetes	NFSv3, Amazon S3, S3 Compliant Alternatives, Azure Storage (Blob), Google Cloud Storage, OpenStack Swift, Local Filesystem

What is docker ?

- Compatible Matrix (Docker Version 18.XX)

OS Distribution (x86_64)	Enterprise Engine	Storage Driver	Orchestration
RHEL 7.3	18.03.1-ee-1	overlay2, devicemapper	Swarm mode
RHEL 7.4	18.03.1-ee-1	overlay2, devicemapper	Swarm mode
RHEL 7.5	18.03.1-ee-1	overlay2, devicemapper	Swarm mode
SLES 12 SP2	18.03.1-ee-1	overlay2 with XFS, btrfs	Swarm mode
SLES 12 SP3	18.03.1-ee-1	overlay2 with XFS, btrfs	Swarm mode
Ubuntu 14.04	18.03.1-ee-1	aufs	Swarm mode
Ubuntu 16.04	18.03.1-ee-1	overlay2, aufs	Swarm mode
Ubuntu 18.04	18.03.1-ee-1	overlay2	Swarm mode
CentOS 7	18.03.1-ee-1	overlay2, devicemapper	Swarm mode
Oracle Linux 7.3 ¹	18.03.1-ee-1	overlay2	Swarm mode
Windows Server 2016	18.03.1-ee-1	windowsfilter	Swarm mode ²
Windows Server, version 1709	18.03.1-ee-1	windowsfilter	Swarm mode ²
Windows Server, version 1803	18.03.1-ee-1	windowsfilter	Swarm mode ²

What is docker ?

- EOL Announcement

Docker Engine

	EE 17.06	EE 18.03
General Availability (GA)	2017-08-16 (17.06.1-ee-1)	2018-06-27 (18.03.1-ee-1)
End of Life (EOL)	2020-04-16	2020-06-16
Release Frequency	X.Y Monthly	X.Y Monthly
Supported Lifespan	2 years from release ¹	2 years from release ¹
Patches and Update Frequency	As Needed: <ul style="list-style-type: none">• Maintenance releases• Security patches• Customer hotfixes	As Needed: <ul style="list-style-type: none">• Maintenance releases• Security patches• Customer hotfixes

¹ Refer to the *Supported Lifecycle* section later in this document for details.

End of Life (EOL) Docker Engine Versions

- CSE 1.11.z (EOL 2017-03-02)
- CSE 1.12.z (EOL 2017-11-14)
- CSE 1.13.z (EOL 2018-02-07)
- EE 17.03.z (EOL 2018-03-01)

Supported Lifecycle

GA to 12 months	12 to 18 months	18 to 24 months
Full support as outlined in the Docker Scope of Support	Full Support as outlined in the Docker Scope of Support ¹	Limited Support for existing installations ²

What is docker ?

- EOL Announcement

Universal Control Plane (UCP)

	2.2.z	3.0.z
General Availability (GA)	2017-08-16 (2.2.0)	2018-04-17 (3.0.0)
End of Life (EOL)	2019-08-15	2020-04-16
Release Frequency	X.Y Monthly	X.Y Monthly
Supported Lifespan	2 years from release ¹	2 years from release ¹
Patches and Update Frequency	As Needed: <ul style="list-style-type: none">• Maintenance releases• Security patches• Customer hotfixes	As Needed: <ul style="list-style-type: none">• Maintenance releases• Security patches• Customer hotfixes

¹ Refer to the *Supported Lifecycle* section later in this document for details.

End of Life (EOL) UCP Versions

- 2.0.z (EOL 2017-08-16)
- 2.1.z (EOL 2018-02-07)

Docker Trusted Registry (DTR)

	2.3.z	2.4.z	2.5.z
General Availability (GA)	2017-08-16 (2.3.0)	2017-11-02 (2.4.0)	2018-04-17 (2.5.0)
End of Life (EOL)	2019-02-15	2019-11-01	2020-04-16
Release Frequency	X.Y Monthly	X.Y Monthly	X.Y Monthly
Supported Lifespan	18 months from release ¹	2 years from release ¹	2 years from release ¹
Patches and Update Frequency	As Needed: <ul style="list-style-type: none">• Maintenance releases• Security patches• Customer hotfixes	As Needed: <ul style="list-style-type: none">• Maintenance releases• Security patches• Customer hotfixes	As Needed: <ul style="list-style-type: none">• Maintenance releases• Security patches• Customer hotfixes

¹ Refer to the *Supported Lifecycle* section later in this document for details.

End of Life (EOL) DTR Versions

- 2.1.z (EOL 2017-08-16)
- 2.2.z (EOL 2018-02-07)

Ref: <https://success.docker.com/article/maintenance-lifecycle>

Docker: The Next-Gen of Virtualization



Docker History

- A dotCloud (PaaS provider) project
- Initial commit January 18, 2013
- Docker 0.1.0 released March 25, 2013
- 18,600+ github stars, 3800+ forks, 740 Contributors.... and continues
- dotCloud pivots to docker inc. October 29, 2013



A circular portrait of Solomon Hykes, a man with dark hair and a beard, wearing a black t-shirt. The portrait is set against a white background and has a gold-colored circular border.

Solomon Hykes

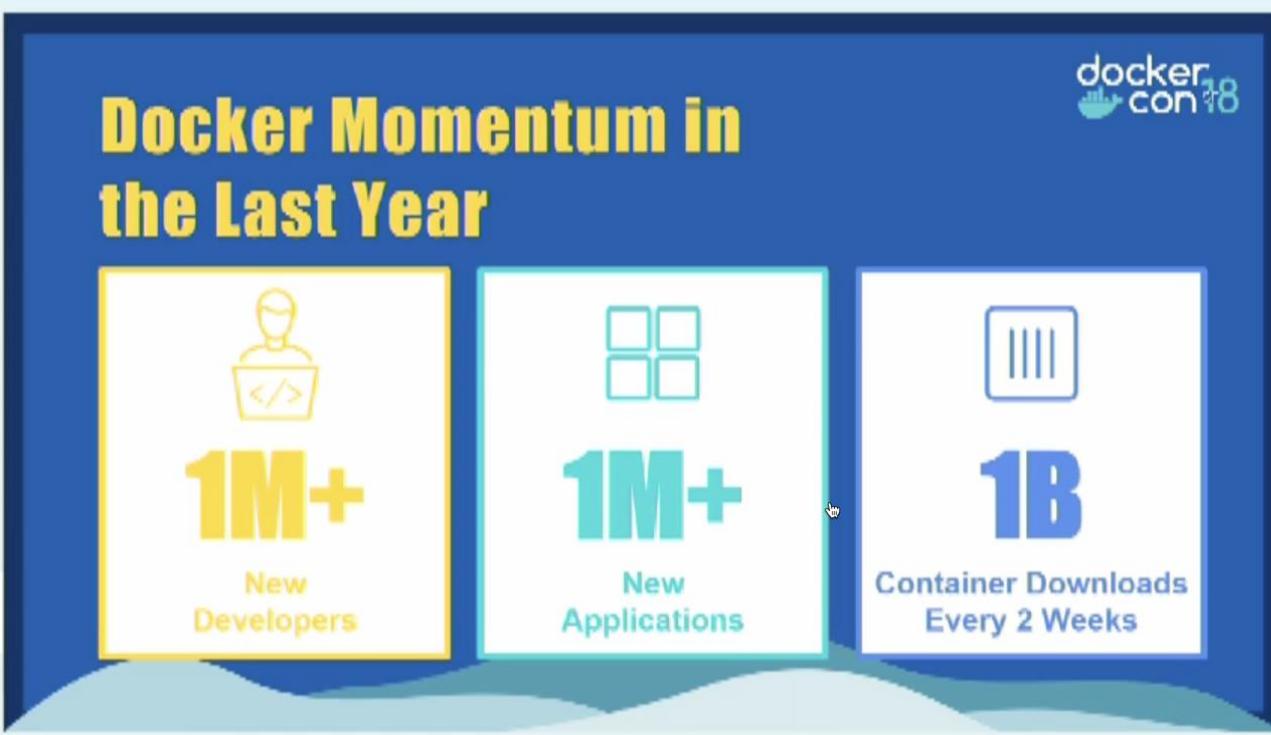
CTO and Founder

 dockercon¹⁷ EU

Docker: The Next-Gen of Virtualization



Growth of Docker (2017 stats)



Growth of Docker (Now)

docker
con
2018 SF

500+ COMMERCIAL CUSTOMERS

BAKER HUGHES
BCG GAMMA
BOSCH

Davita
EQUIFAX
Franklin stratus surfaces

GE
JABIL
jcpenney

LendingClub
Liberty Mutual INSURANCE
INSURANCE MARTIN AIR

MCKESSON
MetLife
PayPal

SOCIETE GENERALE
splunk>

Docker: The Next-Gen of Virtualization



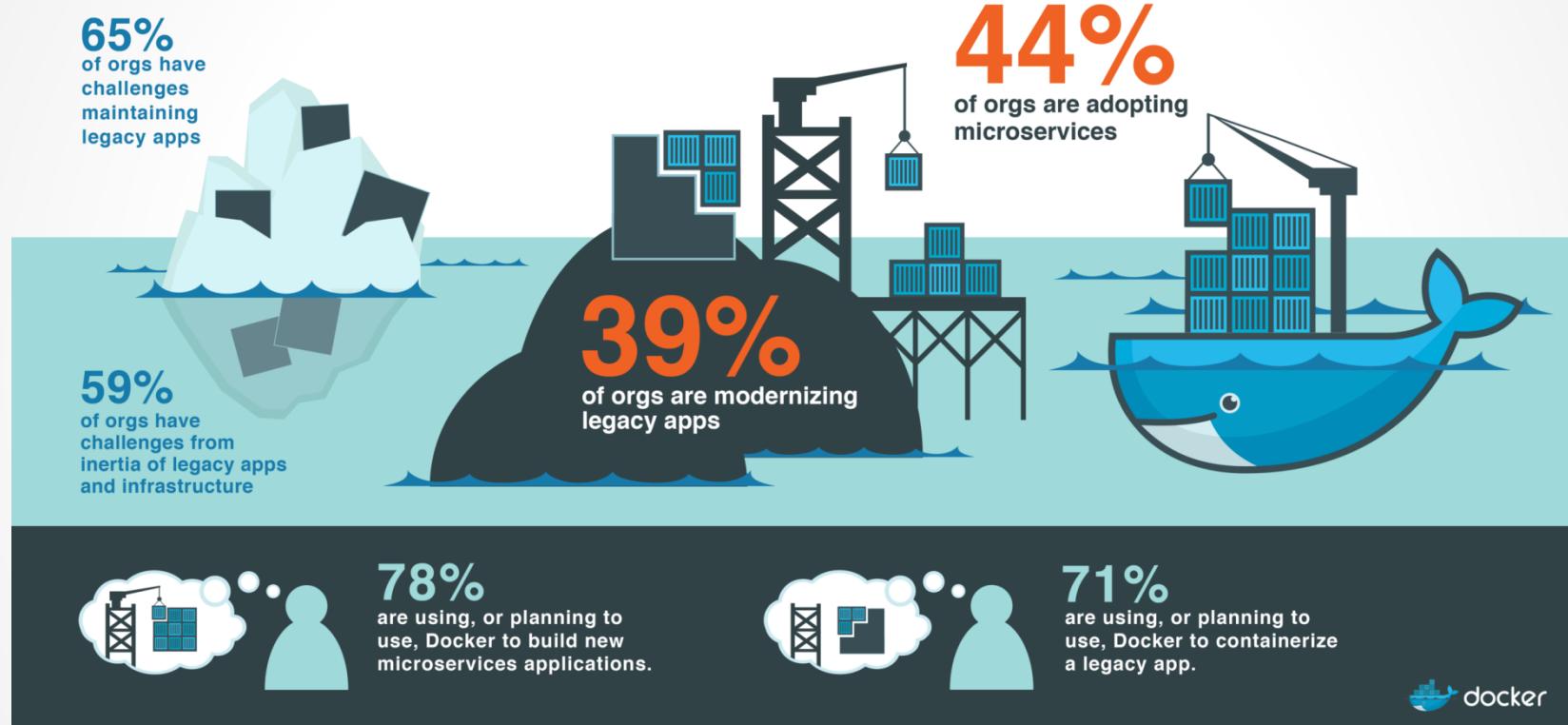
Growth of Docker (Now)



Docker: The Next-Gen of Virtualization



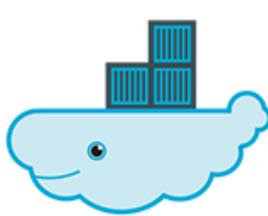
Trend of Modernize Application



Cloud Strategy Survey

80%

say Docker is part
of cloud strategy



60%

plan to use Docker to
migrate workloads to cloud

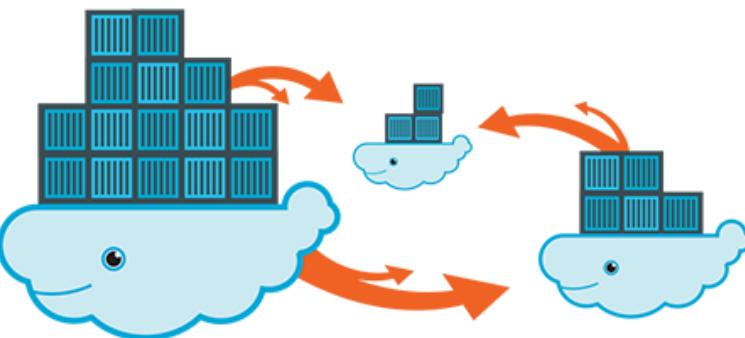


41%

want application
portability across
environments

35+%

want to avoid
cloud vendor
lock-in



Docker in Thailand

- 3 Enterprise Communication on Implement Phase
- 2 Bank on R&D

Docker: The Next-Gen of Virtualization



Who use docker now ?



Who use docker now ?

McKesson at a Glance

Fortune 500
**#6 in
2018**

184-year

history of serving
customers



\$205B
in revenues



Operations
in more than
16
countries



Deliver
1/3
of all prescription
medicine in
North America



Nearly
70,000
employees worldwide



\$3.7B
in operating cash flow



Operational data reflected as of March 2, 2017. Financial data presented for the fiscal year ended March 31, 2016.

MCKESSON

docker
con'18



Who use docker now ?

Expanding the Role of the Pharmacist as a Provider in a Low-Cost Care Setting



Who use docker now ?

And Created McKesson Labs

 **15**
locations globally

 **23**
dev Teams

 **300+**
developers represented

 **700**
data points

 Single brand for Dev
at McKesson

 Docker is a key enabler
Tools and Ingredients in our recipe to customers

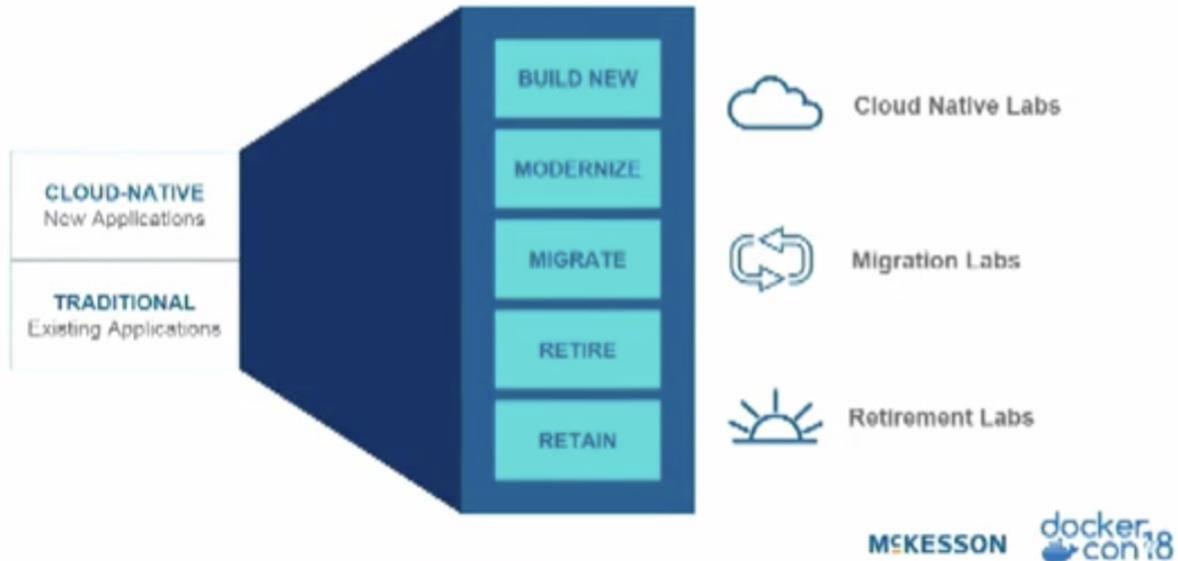
MCKESSON

docker
con'18



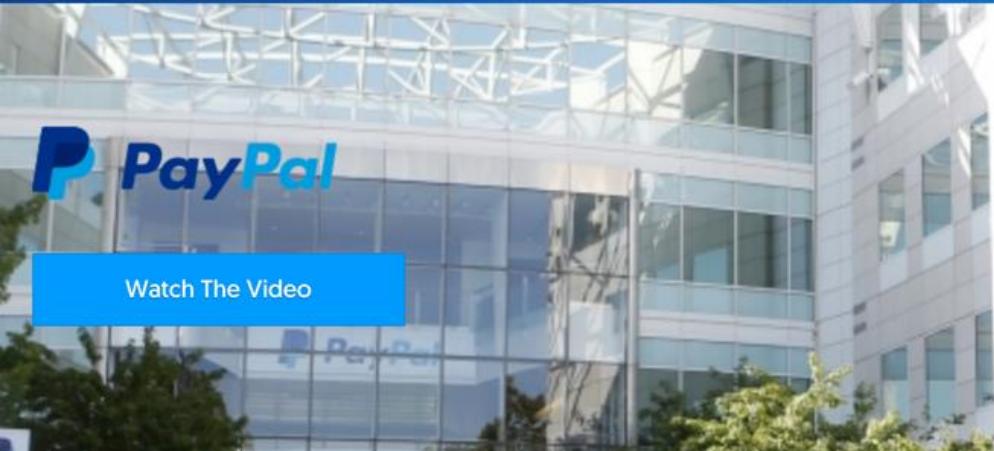
Who use docker now ?

Docker is Central to Application Transformation

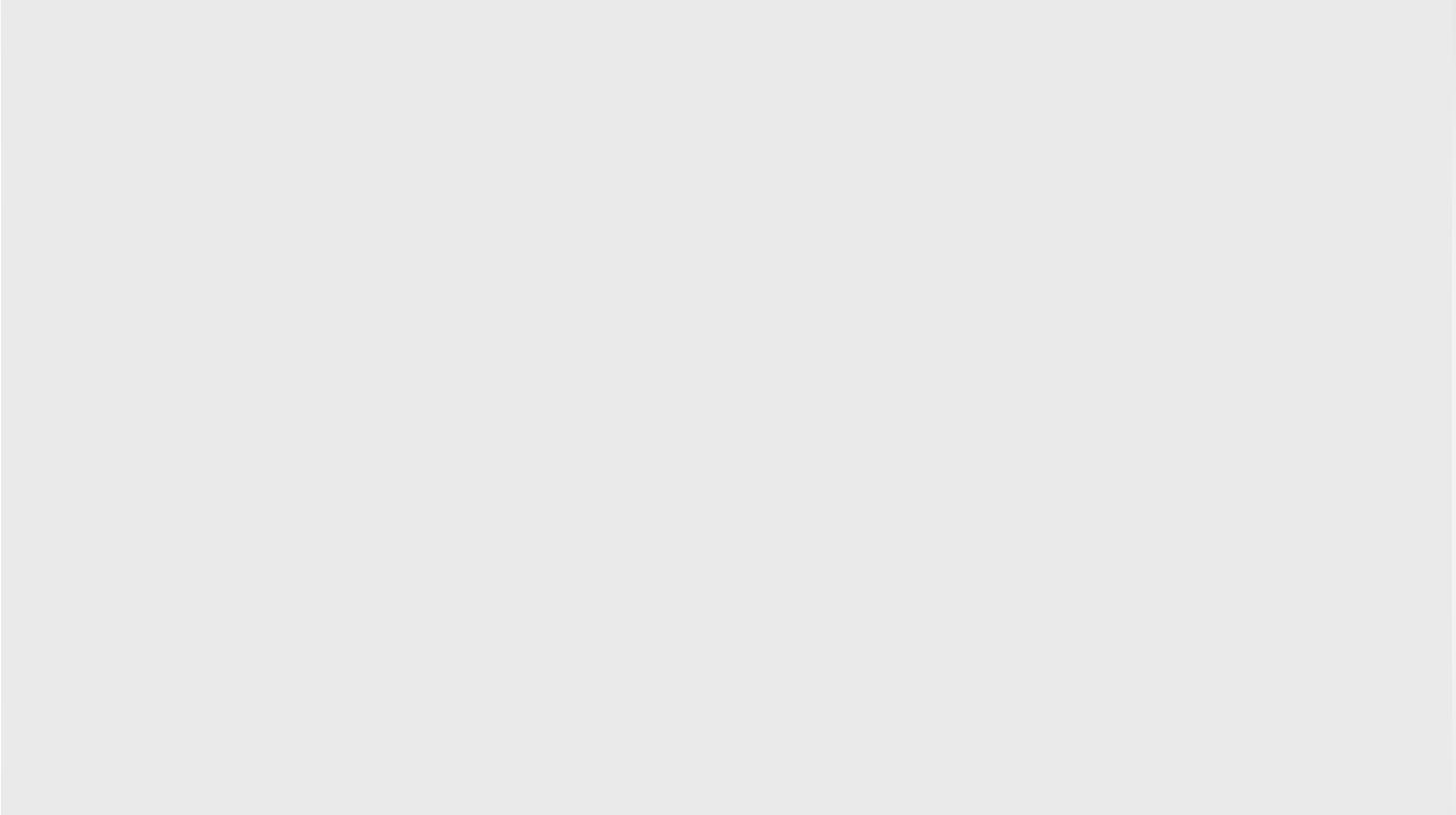


Who use docker now ?

PAYPAL USES DOCKER TO CONTAINERIZE EXISTING APPS, SAVE MONEY AND BOOST SECURITY



Who use docker now ?



Docker: The Next-Gen of Virtualization



Who use docker now ?



Stage 1 Benefits

Decouple

Standardize deployments around Docker containers, rather than individual processes built around app stacks.

Modernize

With apps & dependencies Dockerized, move to modern OS & kernel.

10-20% performance boost to some apps for free.

150,000 containers

700+ apps

18 months

0 code changes

Who use docker now ?

Later Stage Benefits



1 Resource Efficiency

50% fewer vCPUs in QA

25% fewer vCPUs in Prod

2 Security

Revoke access to Prod

Automate patching

3 No VM Provisioning

Scale containers (fast)

Not VMs (slow)

4 Availability

Increased resiliency

"Anything can run anywhere"

5 Availability Zones

Application deployments of an entire AZ in a few hours.

6 Consistent Platform

Consistent tooling

Standard playbook across app stacks

> 50% boost

to developer local
build-deploy-test
speed

Developer
Freedom

for greenfield apps

Who use docker now ?



U B E R

Workshop Automate Software Development



Who use docker now ?



Who use docker now ?

About MetLife



Life



P&C



Annuity



Dental



Disability



Health



Legal

- Global Fortune 500® 128 Company
- 100 Million Customers
- \$500 Billion total assets under investment
- \$63 Billion in revenue in 2017



Even Old Elephants Can Dance

- Concept to production in 5 months
- The spark of innovation is spreading at MetLife
- Docker and DevOps change your culture
- Check out Tim Tyler's session

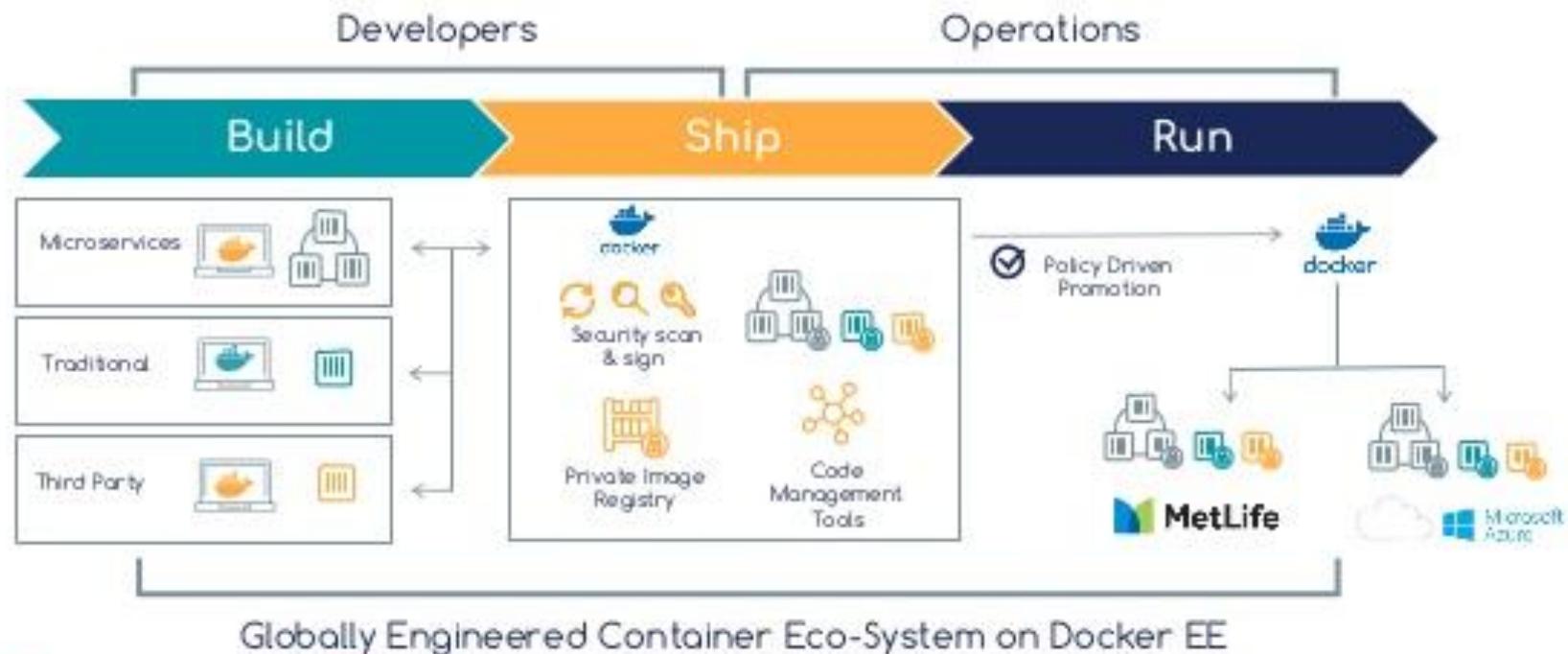
11:15 AM - Docker 0 to 60 in 5 Months:
How a Traditional Fortune 40 Company
Turns on a Dime

docker17
con



Who use docker now ?

The MetLife Platform



Who use docker now ?

US Infrastructure Reduction Forecast

593 Applications

10%

Of the total portfolio

-70%

VMs

+

-67%

Cores

+

10x

Average CPU
Utilization

=

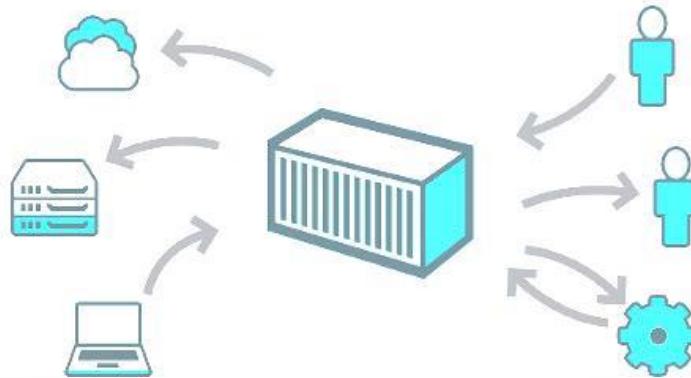
-66%

Cost
Reduction

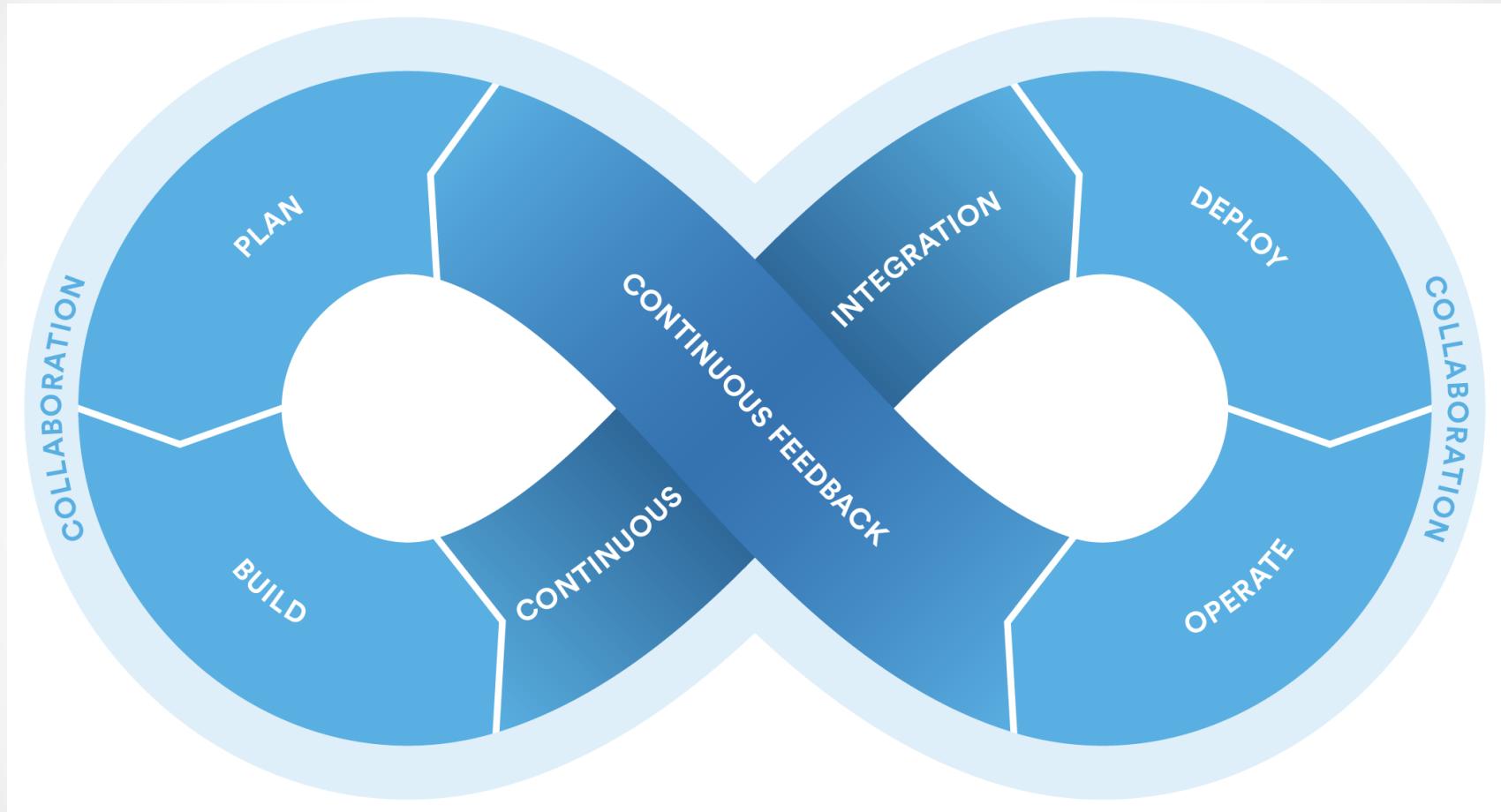


Container Life Cycle

- Developer
 - สร้าง template สำหรับโปรแกรม (build)
 - เริ่มรัน template สำหรับพัฒนาโปรแกรม (run)
 - พัฒนาเสร็จเรียบร้อย สั่ง save version เพื่อเตรียมนำขึ้นใช้งาน (commit)
- Operation
 - เริ่มรัน template ที่ได้รับจาก developer (run) บนเครื่อง production
 - หยุดการให้บริการโปรแกรม (stop, kill)
 - ลบโปรแกรมออกจากเครื่อง production (rm, rmi)



Container Life Cycle



What Cool of Docker ?

- รวบรวมทุกสิ่งที่จำเป็นต้องใช้ในการรันโปรแกรมไว้ใน container (component, library etc)
- ขนาดไฟล์ container มีขนาดเล็กมาก (เทียบกับขนาดไฟล์ของ virtual machine หรือ os)
- มี overhead ในการรันโปรแกรมทรัพยากรต่ำ
- ลดระยะเวลาในการติดตั้งและทดสอบโปรแกรม
- ส่งมอบโปรแกรมไปทำงานบนเครื่องแม่ข่าย production ได้โดยไม่มีความจำเป็นต้องปรับแต่งระบบใหม่ (zero configure)
- สามารถรันโปรแกรมได้บนเครื่องแม่ข่ายทุกๆระบบปฏิบัติการฯ ที่ติดตั้ง docker ได้
- สามารถ scale-out ได้ง่ายในอนาคต
- World open for docker !!!

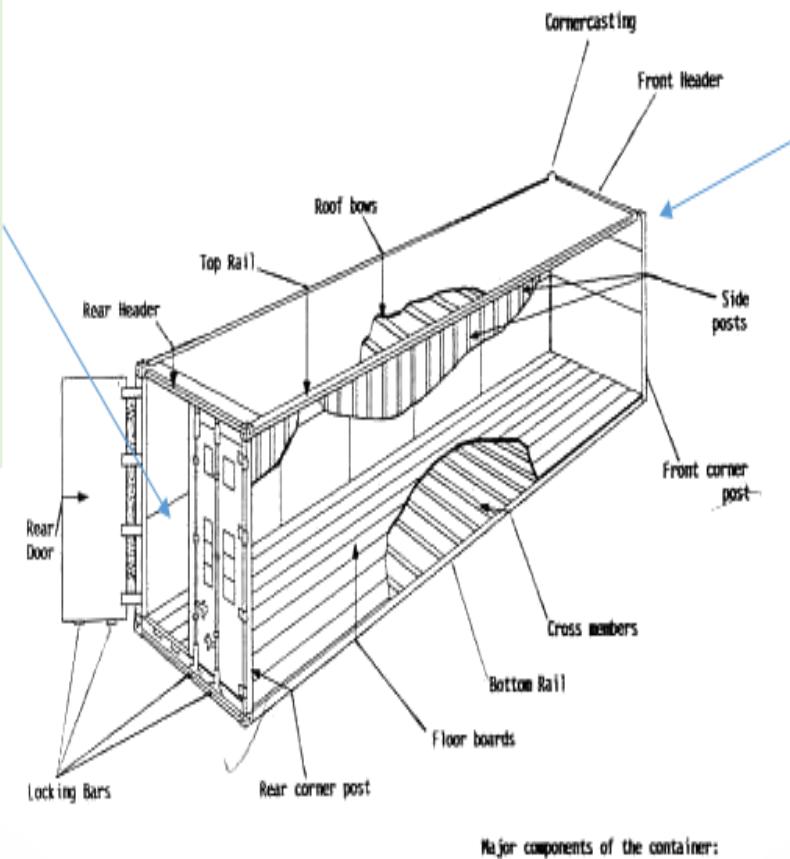
Separate of Concern

Dan the Developer

- Worries about what's "inside" the container
 - His code
 - His Libraries
 - His Package Manager
 - His Apps
 - His Data
- All Linux servers look the same

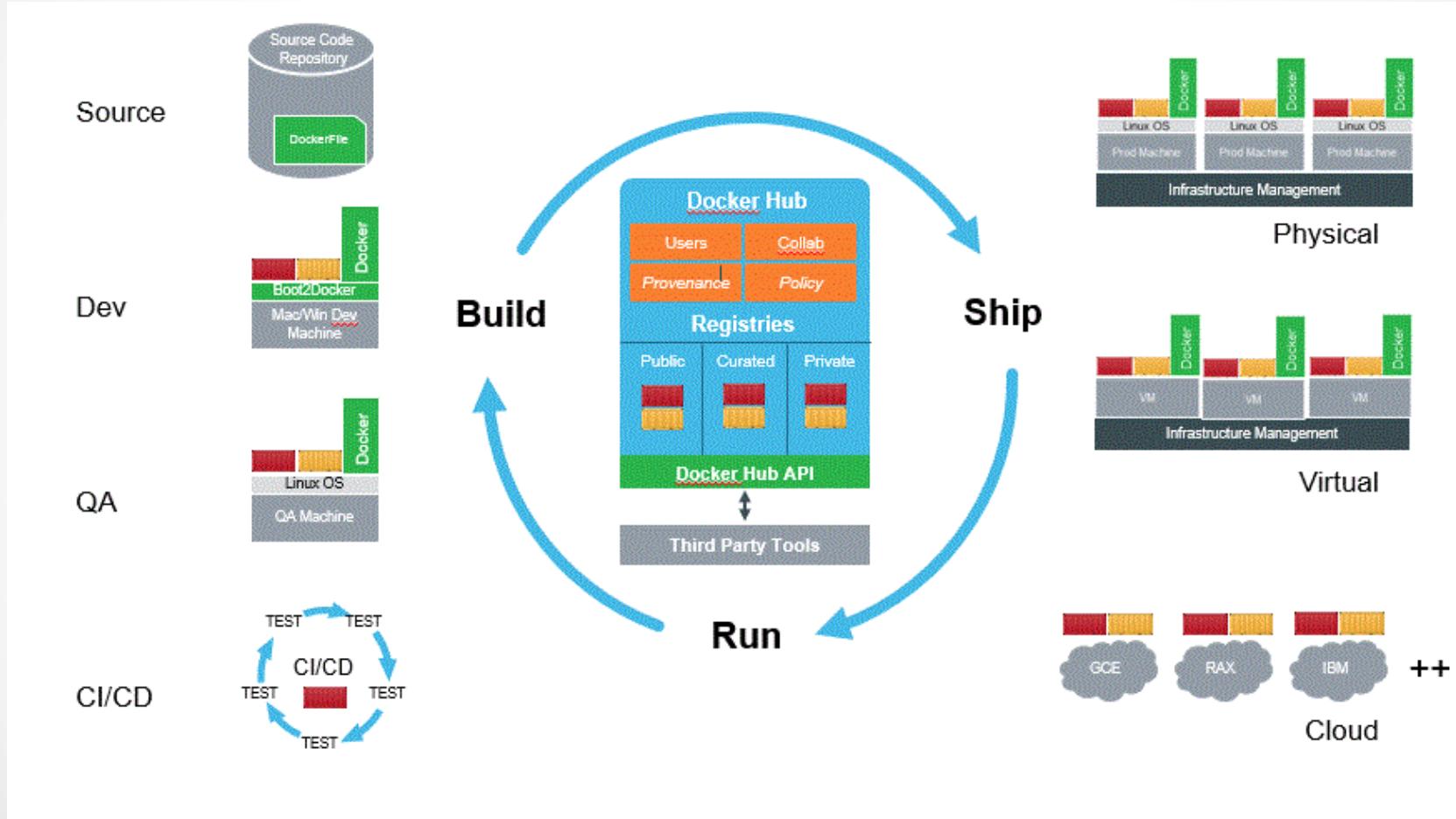
Oscar the Ops Guy

- Worries about what's "outside" the container
 - Logging
 - Remote access
 - Monitoring
 - Network config
- All containers start, stop, copy, attach, migrate, etc. the same way

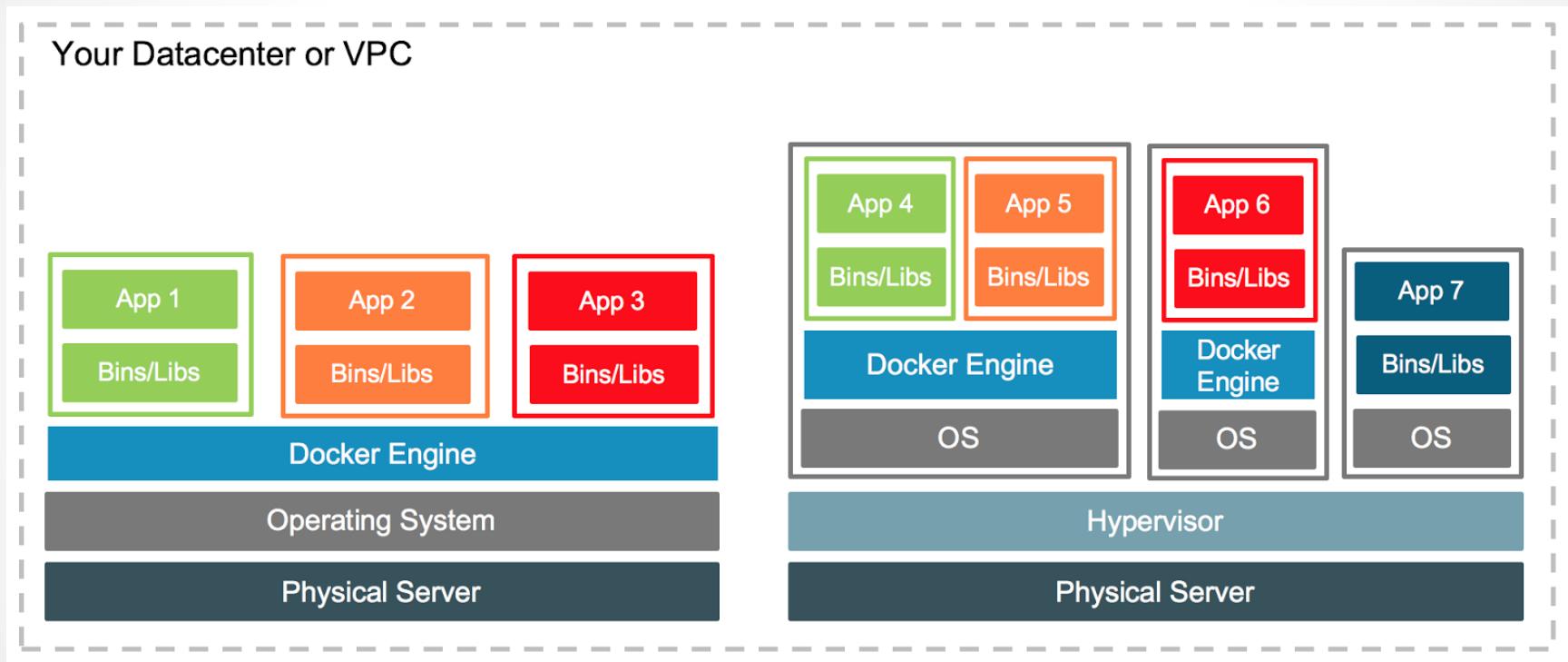


Benefit of docker for DevOps

- Build-Ship-Run



Docker vs VMWare



Docker: The Next-Gen of Virtualization



Docker vs VMWare



Docker: The Next-Gen of Virtualization



Docker Version

```
praparns-MacBook-Pro:nodejs praparn$ docker version
Client:
  Version: 18.06.1-ce
  API version: 1.38
  Go version: go1.10.3
  Git commit: e68fc7a
  Built: Tue Aug 21 17:21:31 2018
  OS/Arch: darwin/amd64
  Experimental: false

Server:
  Engine:
    Version: 18.06.1-ce
    API version: 1.38 (minimum version 1.12)
    Go version: go1.10.3
    Git commit: e68fc7a
    Built: Tue Aug 21 17:29:02 2018
    OS/Arch: linux/amd64
    Experimental: true
```

```
praparns-MacBook-Pro:nodejs praparn$ docker system info
Containers: 0
  Running: 0
  Paused: 0
  Stopped: 0
Images: 9
  Server Version: 18.06.1-ce
  Storage Driver: overlay2
    Backing Filesystem: extfs
    Supports d_type: true
    Native Overlay Diff: true
  Logging Driver: json-file
  Cgroup Driver: cgroups
  Plugins:
    Volume: local
    Network: bridge host ipvlan macvlan null overlay
    Log: awslogs fluentd gcplogs gelf journald json-file logentries splunk syslog
  Swarm: inactive
  Runtimes: runc
  Default Runtime: runc
  Init Binary: docker-init
  containerd version: 468a545b9edcd5932818eb9de8e72413e616e86e
  runc version: 69663f0bd4b60df09991c08812a60108003fa340
  init version: fec3683
  Security Options:
    seccomp
      Profile: default
  Kernel Version: 4.9.93-linuxkit-aufs
  Operating System: Docker for Mac
  OSType: linux
  Architecture: x86_64
  CPUs: 4
  Total Memory: 1.952GiB
  Name: linuxkit-025000000001
  ID: VGYC:SXNI:FNTR:QTRT:NR2M:GDBL:N6VQ:UDPO:FGPC:PHNM:TX3U:DTOP
  Docker Root Dir: /var/lib/docker
  Debug Mode (client): false
  Debug Mode (server): true
  File Descriptors: 26
  Goroutines: 53
  System Time: 2018-10-20T17:06:08.527801472Z
  EventsListeners: 2
  HTTP Proxy: gateway.docker.internal:3128
  HTTPS Proxy: gateway.docker.internal:3129
  Registry: https://index.docker.io/v1/
  Labels:
  Experimental: true
  Insecure Registries:
    127.0.0.0/8
  Live Restore Enabled: false
praparns-MacBook-Pro:nodejs praparn$
```

Cloud Native Computing Foundation

Cloud Native Landscape

v2.1

See the interactive landscape at landscape.cnfc.io

Greyed logos are not open source

App Definition & Development



Database & Data Warehouse



Streaming



Source Code Management



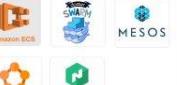
Continuous Integration / Continuous Delivery (CI/CD)



Orchestration & Management



Scheduling & Orchestration



Coordination & Service Discovery



Service Management



Runtime



Cloud-Native Storage



Container Runtime



Cloud-Native Network



Provisioning



Host Management / Tooling



Infrastructure Automation



Container Registries



Secure Images



Key Management



Cloud



Public



Private



github.com/cnfc/landscape

This landscape is intended as a map through the previously uncharted terrain of cloud native technologies. There are many routes to deploying a cloud native application, with CNCF Projects representing a particularly well-traveled path.



Kubernetes Certified Service Provider

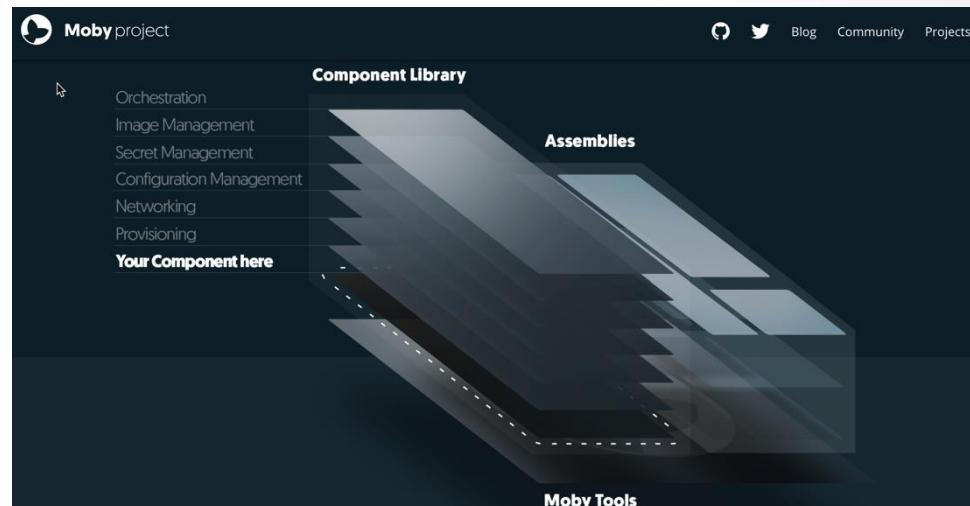
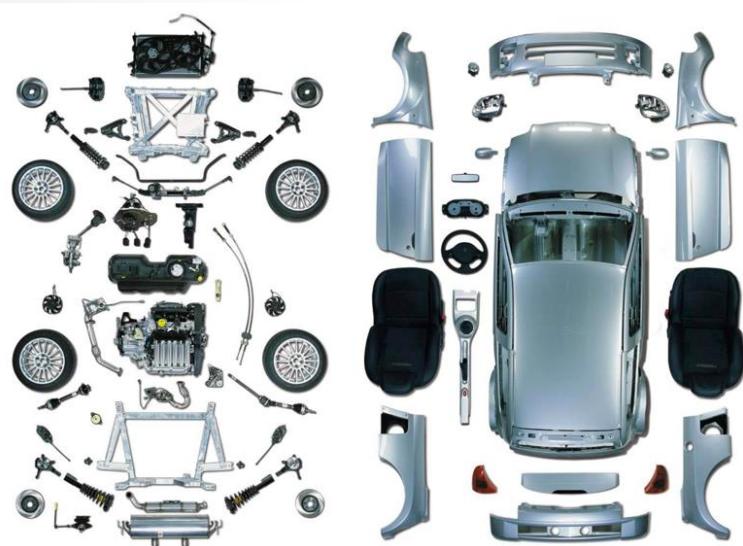
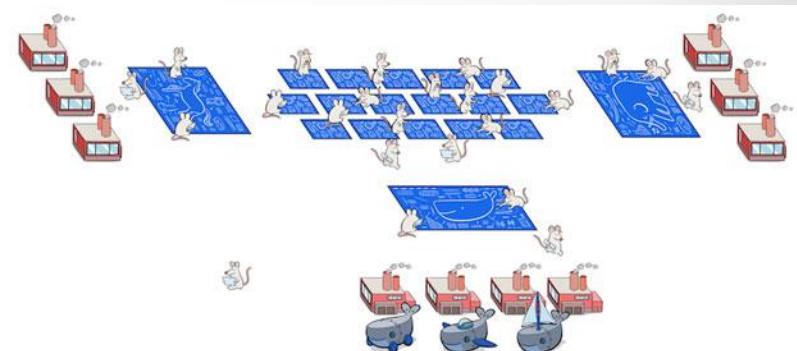
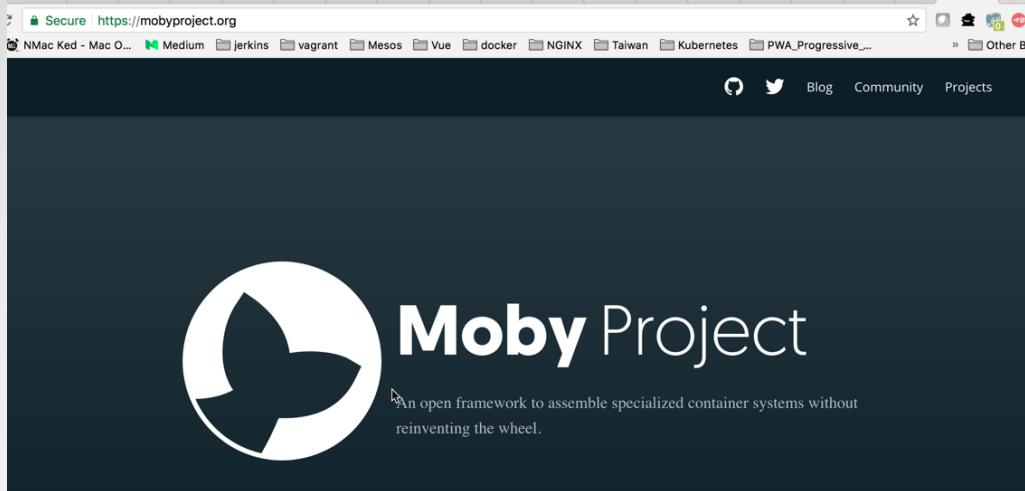
Special



Kubernetes: Production Workload Orchestration



What's New



Docker: The Next-Gen of Virtualization



What's New

Docker Pricing Plans

	COMMUNITY EDITION	ENTERPRISE EDITION BASIC	ENTERPRISE EDITION STANDARD	ENTERPRISE EDITION ADVANCED
Container engine and built in orchestration, networking, security	✓	✓	✓	✓
Docker Certified Infrastructure, Plugins and ISV Containers		✓	✓	✓
Image Management (private registry, caching)	Cloud hosted repos		✓	✓
Docker Datacenter Integrated container app management			✓	✓
Docker Datacenter Multi-tenancy with RBAC, LDAP/AD support		✓	✓	✓
Integrated secrets mgmt, image signing policy			✓	✓
Image security scanning	Preview			✓
Support	Community Support	Business Day or Business Critical	Business Day or Business Critical	Business Day or Business Critical

Docker: The Next-Gen of Virtualization



What's News

Docker CE release notes

Estimated reading time: 63 minutes

For Docker Enterprise Edition, see [Docker EE](#).

For Docker releases prior to 17.03.0, see [Docker Engine release notes](#).

[Learn about Docker releases](#).

Release notes for stable versions are listed first. You can go straight to the Edge release notes or [learn more about Stable and Edge releases](#).

Stable releases

18.06.1-ce (2018-08-21)

Builder

- Fix no error if build args are missing during docker build. [docker/engine#25](#)
- Set BuildKit's ExportedProduct variable to show useful errors. [docker/engine#21](#)

Client

- Various shell completion script updates: [docker/cli#1229](#), [docker/cli#1268](#), and [docker/cli#1272](#)
- Fix DOCKER_CONFIG warning message and fallback search. [docker/cli#1241](#)
- Fix help message flags on docker stack commands and sub-commands. [docker/cli#1267](#)

Runtime

- Disable CRI plugin listening on port 10010 by default. [docker/engine#29](#)
- Update containerd to v1.1.2. [docker/engine#33](#)
- Windows: Do not invoke HCS shutdown if terminate called. [docker/engine#31](#)
- Windows: Select polling-based watcher for Windows log watcher. [docker/engine#34](#)

Swarm Mode

- Fix the condition used for skipping over running tasks. [docker/swarmkit#2677](#)
- Fix task sorting. [docker/swarmkit#2712](#)

Docker EE Engine release notes

Estimated reading time: 28 minutes

This document describes the latest changes, additions, known issues, and fixes for Docker Enterprise Edition (Docker EE).

Docker EE is functionally equivalent to the corresponding Docker CE that it references. However, Docker EE also includes back-ported fixes (security-related and priority defects) from the open source. It incorporates defect fixes that you can use in environments where new features cannot be adopted as quickly for consistency and compatibility reasons.

18.03.1-ee-3 (2018-08-30)

Important notes about this release

If you're deploying UCP or DTR, use Docker EE Engine 17.06.

Builder

- Fix: no error if build args are missing during docker build. [docker/engine#25](#)
- Ensure RUN instruction to run without healthcheck. [moby/moby#37413](#)

Client

- Fix manifest list to always use correct size. [docker/cli#1156](#)
- Various shell completion script updates. [docker/cli#1159](#) [docker/cli#1227](#)
- Improve version output alignment. [docker/cli#1204](#)

Runtime

- Disable CRI plugin listening on port 10010 by default. [docker/engine#29](#)
- Update containerd to v1.1.2. [docker/engine#33](#)
- Windows: Pass back system errors on container exit. [moby/moby#35967](#)
- Windows: Fix named pipe support for hyper-v isolated containers. [docker/engine#2](#) [docker/cli#1165](#)
- Register OCI media types. [docker/engine#4](#)

Swarm Mode

- Clean up tasks in dirty list for which the service has been deleted. [docker/swarmkit#2694](#)
- Propagate the provided external CA certificate to the external CA object in swarm. [docker/cli#1178](#)

What's News

Microsoft | Windows IT Pro Center Explore Docs Downloads Scripts Support
Docs / Windows Server / Get Started / What's New in Windows Server / What's New in Windows Server, version 1803 [Feedback](#) [Edit](#) [Link](#)

Filter by title

What's New in Windows Server, version 1809

What's New in Windows Server, version 1803

What's New in Windows Server, version 1709

> What's New in Windows Server 2016

Windows Server Semi-Annual Channel Overview

Installation and Upgrade of Windows Server Releases

What's New in Windows Server version 1803

05/07/2018 • 4 minutes to read • Contributors  

Applies To: Windows Server (Semi-Annual Channel)



The content in this section describes what's new and changed in Windows Server, version 1803. The new features and changes listed here are the ones most likely to have the greatest impact as you work with this release. Also see [Windows Server Semi-Annual Channel update](#).

Server Core container image

A smaller Server Core container with better application compatibility is now available. Detailed information is available [here](#).

- Unused optional features and roles have been removed. For more information see [Roles, Role Services, and Features not in Server Core containers](#).
 - Decreased download size to **1.58 GB**, 30% reduction from Windows Server, version 1709.
 - Decreased size on disk to 3.61 GB, 20% reduction from Windows Server, version 1709.
- Nano Server container image is below 100MB

Windows Server release strategy

Windows Server version 1709 was released in September 2017 as the first release in the Semi-Annual Channel. The Semi-Annual Channel has a faster release cadence and addresses feedback from those who want rapid innovation every few months. This complements the Long-Term Servicing Channel where the release cadence is every 2-3 years.

Based on telemetry and feedback, these channels have demonstrated that they conform well to the following general strategy:

- The Semi-Annual Channel is ideal for modern applications and innovation scenarios, such as containers and micro-services.
- The Long-Term Servicing Channel is the preferred release for core infrastructure scenarios such as software-defined datacenter and hyper-converged infrastructure (HCI).

The specific scenarios for the Semi-Annual Channel and the Long-Term Servicing Channel are as follows:

	Long Term Servicing Channel	Semi-Annual Channel
Recommended scenarios	General purpose file servers, first and third-party workloads, traditional apps, infrastructure roles, software-defined Datacenter, and Hyper-converged infrastructure	Containerized applications, container hosts, and application scenarios benefiting from faster innovation
New releases	Every 2-3 years	Every 6 months
Support	5 years of Mainstream support + 5 years of Extended support	18 months
Editions	All available Windows Server editions	Standard and Datacenter

Docker: The Next-Gen of Virtualization



What's News

microsoft/windowsservercore ★

Last pushed: 11 days ago

Repo Info Tags

Short Description

Short description is empty for this repo.

Full Description

Supported Windows Server 2016 (Long Term Support Channel) Tags

```
docker pull microsoft/windowsservercore:ltsc2016
```

- ltsc2016, latest
- 10.0.14393.<Build Number>
- 10.0.14393.<Build Number>_<Language Code>

Supported Windows Server, version 1709 (Fall Creators Update) Tags

```
docker pull microsoft/windowsservercore:1709
```

- 1709
- 1709_KB<Knowledge Base ID>

Supported Windows Server, version 1803 (April 2018 Update) Tags

```
docker pull microsoft/windowsservercore:1803
```

- 1803
- 1803_KB<Knowledge Base ID>

Docker Pull Command



```
docker pull microsoft/windowsservercc
```

Owner



microsoft

What's News

Docker with Swarm and Kubernetes

1 →

The best enterprise
container security and
management



2 ←

The best container
development workflow

3 →

Native Kubernetes
integration provides full
ecosystem
compatibility

4 ←

Industry-standard
container runtime

What's News

Docker Enterprise Edition

Management for Swarm and Kubernetes

Features		Swarm Support	Kubernetes Support
100% Interoperability	<ul style="list-style-type: none">- <i>Clean upstream integration</i>- <i>Full ecosystem compatibility</i>	✓	✓
Secure Cluster Lifecycle	<ul style="list-style-type: none">- <i>Easy High Availability provisioning</i>- <i>Cryptographic node identity</i>	✓	✓
Secure Supply Chain	<ul style="list-style-type: none">- <i>Registry</i>- <i>Content Trust</i>- <i>Secure Scanning</i>	✓	✓
Secure Multi-tenancy	<ul style="list-style-type: none">- <i>Role Based Access Control</i>- <i>Authorization, Authentication</i>- <i>Node Segmentation</i>	✓	✓
Management Dashboard		✓	✓
Supported and Certified on Windows Server and Major Linux Distributions		✓	✓

Docker Machine

• • •

Platform of Docker

- Docker Toolbox (Obsolete) for Desktop (Docker CE)
 - Install docker toolbox on machine will Create VM (Oracle VirtualBox)
 - Dockertoolbox for MAC
 - Dockertoolbox for Windows (7,8,10)
- Docker Native for Desktop (Docker CE)
 - Docker for MAC (Moby Linux (xhyve engine))
 - Docker for Windows (Moby Linux (hyper-v engine))
- Docker Native for Server (Docker CE/EE)
 - Docker for Windows 2016 (EE)
 - Docker for Ubuntu,CENTOS (CE/EE)
 - Docker for Debian (EE/ARM)
 - Docker for Red Hat Enterprise/SUSE/Oracle Linux (EE)
 - Docker for Fedora (CE)

Platform of Docker

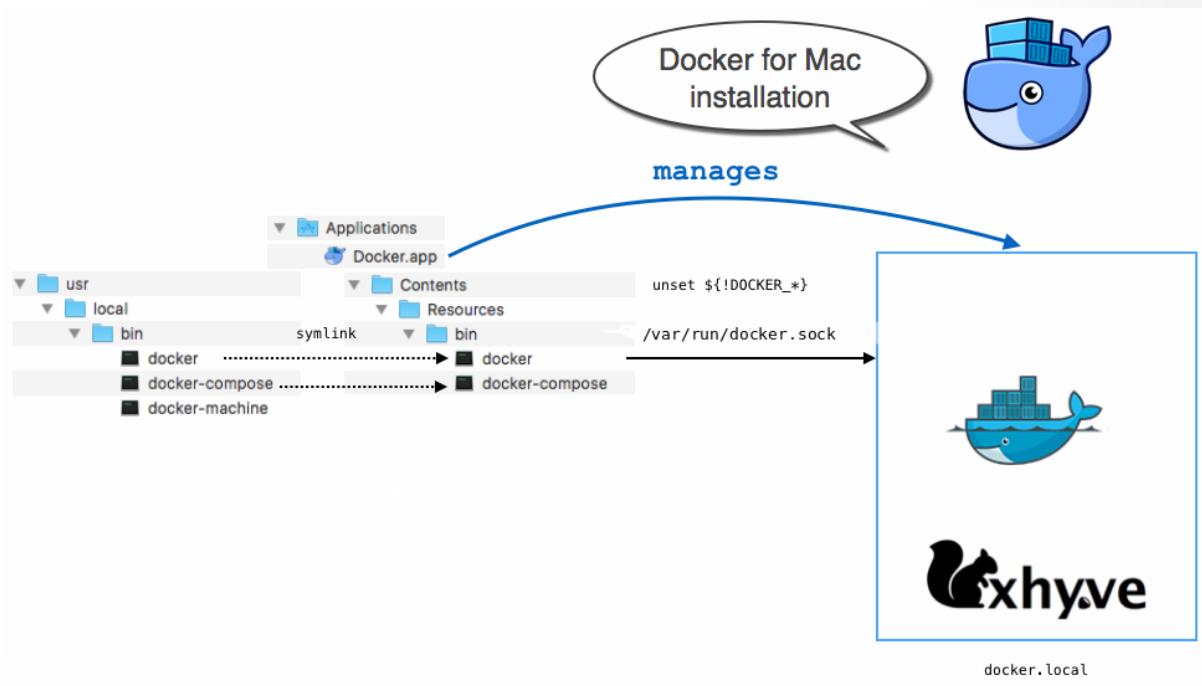
- Pros and Cons.
- Docker Toolbox
 - Pros
 - Compatible for All (Windows 7, 8, 10, MAC OS X)
 - Limit result for operate
 - Suitable for simulate multiple node or swarm
 - Cons
 - Multiple step for transfer source code and debug
 - Overhead for VirtualBox

Platform of Docker

- Pros and Cons.
- Docker for Windows or MAC
 - Pros
 - Full integrate with Machine (No need virtualbox)
 - Auto-update
 - File sharing (Host $\leftarrow \rightarrow$ Container)
 - Http-proxy
 - Custom docker-registry
 - Cons
 - Single Node all case
 - Limit for

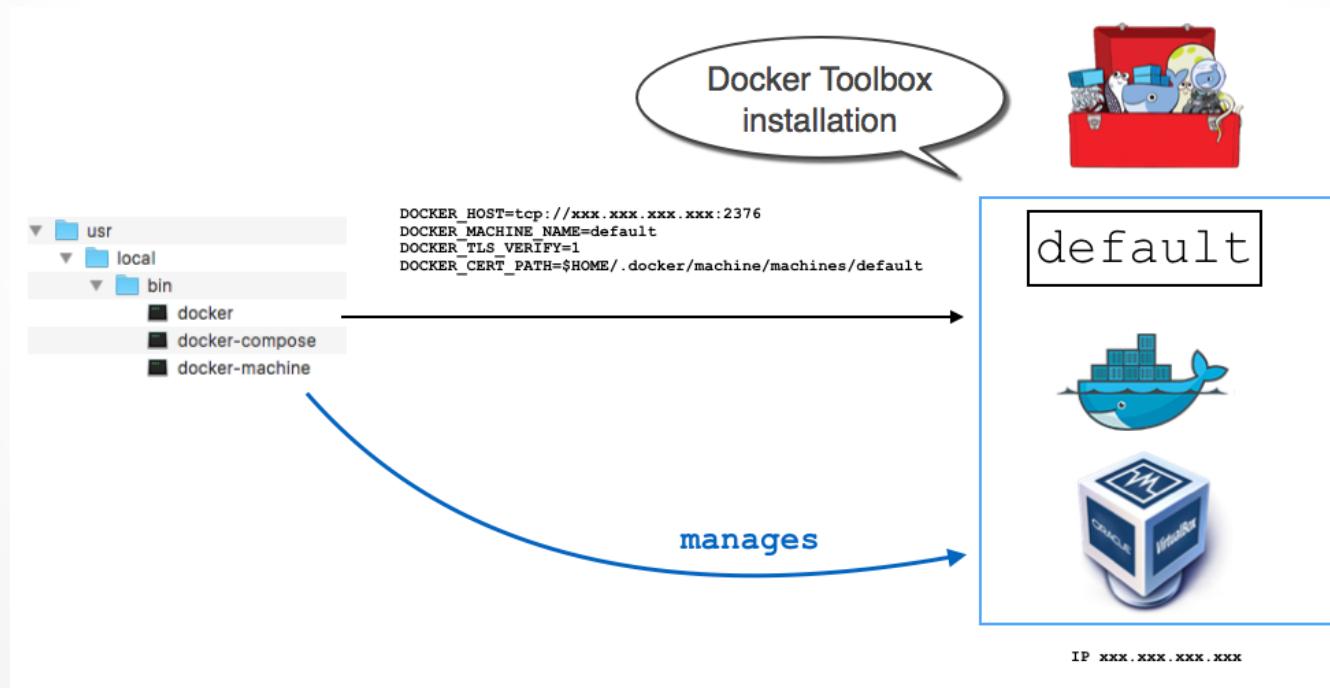
Platform of Docker

- **Docker Native for Desktop/Server**
- การติดตั้ง docker ลงบนเครื่อง linux server / windows (non-toolbox) / mac (non-toolbox) เมื่อเกิดการรัน container ขึ้น docker daemon จะให้บริการบนเครื่อง linux server ที่ทำการติดตั้งโดยตรง ซึ่งเราสามารถเรียกใช้งานโปรแกรมบนเครื่องได้ทันที



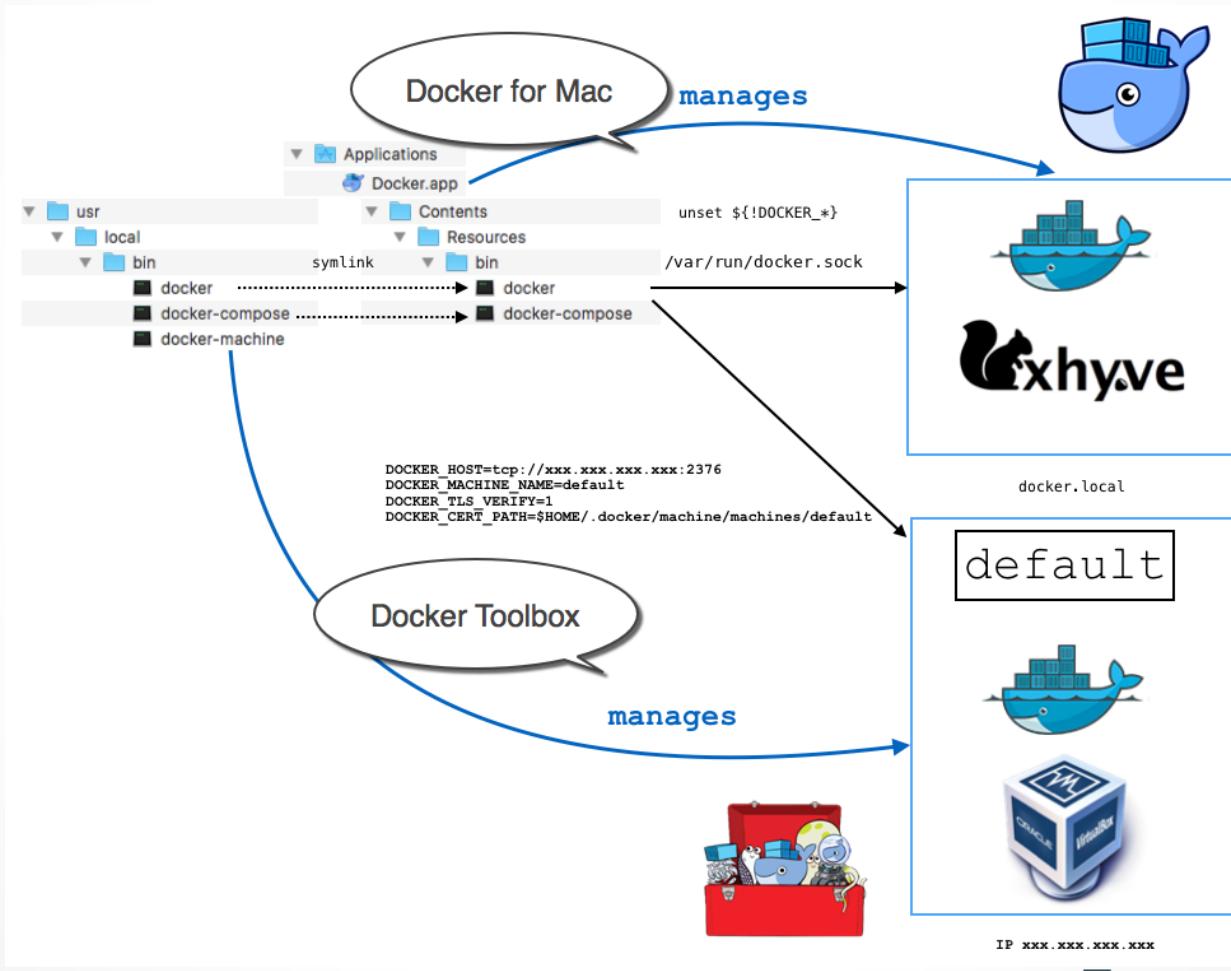
Linux vs Windows vs MAC OS

- **Docker ToolBox**
- การติดตั้ง docker ลงบนเครื่อง windows หรือ mac os จะมีการติดตั้ง virtual machine ใหม่ขึ้นบนเครื่องเพื่อให้บริการสำหรับ docker โดยเฉพาะดังนั้นมีการสั่งรันเจ็งเกิดการ remote run ไปยังเครื่อง virtual machine (docker-machine)



Linux vs Windows vs MAC OS

- Special for MAC (Coexist)



Linux vs Windows vs MAC OS

Not Secure | boot2docker.io

Apps NMac Ked - Mac ... Medium Infra NOVA jenkins vagrant Mesos Vue docker NGINX Taiwan Kubernetes

State of boot2docker

The `boot2docker` CLI tool is *long*-since officially deprecated in favor of [Docker Machine](#).

On the other hand, the `boot2docker` distribution (as in, `boot2docker.iso`) is in "maintenance mode". See <https://github.com/boot2docker/boot2docker> for more details.

It is **highly** recommended that users transition from Boot2Docker over to [Docker for Mac](#) or [Docker for Windows](#) instead.

 docker docs Search the docs Guides Product manuals Glossary Reference Samples

Docker Enterprise Edition

Docker Cloud

Docker Compose

Docker for Mac

Docker for Windows

Docker ID accounts

Docker Machine

- Machine overview
- Install Machine

Get started with Docker Machine and a local VM

Estimated reading time: 13 minutes

Let's take a look at using `docker-machine` to create, use and manage a Docker host inside of a local virtual machine.

Prerequisite Information

With the advent of [Docker for Mac](#) and [Docker for Windows](#) as replacements for [Docker Toolbox](#), we recommend that you use these for your primary Docker workflows. You can use these applications to run Docker natively on your local system without using Docker Machine at all. (See [Docker for Mac vs. Docker Toolbox](#) for an explanation on the Mac side.)

For now, however, if you want to create *multiple* local machines, you still need Docker Machine to create and manage machines for multi-node experimentation. Both Docker for Mac and Docker for Windows include the newest version of Docker Machine, so when you install either of these, you get `docker-machine`.

Docker-machine CLI

- ตรวจสอบสถานะ docker-machine

```
docker-machine ls
```

- สั่งเริ่มการทำงานของ docker-machine

```
docker-machine start <machine name>
```

- สั่งหยุดการทำงานของ docker-machine

```
docker-machine stop <machine name>
```

- สั่งสร้าง docker-machine เครื่องใหม่

```
docker-machine create --driver virtualbox/hyperv <name>
```

Workshop 1-2: Pull Image

ทำการ download image ubuntu ลงมาที่เครื่อง boot2docker โดยใช้คำสั่งดังนี้

```
docker image pull labdocker/alpine:latest
```

```
docker image pull labdocker/alpineweb:latest
```

Compare with Ubuntu image
docker image pull labdocker/cadvisor:latest

```
docker images/docker image ls
```

```
Administrator: Windows PowerShell
PS C:\WINDOWS\system32> docker image ls
REPOSITORY          TAG      IMAGE ID      CREATED       SIZE
labdocker/alpineweb latest   5770233f642f  12 hours ago  50.5MB
labdocker/ubuntu    latest   ea4c82ded15a  2 days ago   85.8MB
labdocker/alpine     latest   196d12cf6abl  5 weeks ago  4.41MB
labdocker/cadvisor   latest   75f88e3ec333  10 months ago 62.2MB
PS C:\WINDOWS\system32>
```

General Command in Docker

Command	Description
docker attach	Attach local standard input, output, and error streams to a running container
docker build	Build an image from a Dockerfile
docker checkpoint	Manage checkpoints
docker commit	Create a new image from a container's changes
docker config	Manage Docker configs
docker container	Manage containers
docker cp	Copy files/folders between a container and the local filesystem
docker create	Create a new container
docker deploy	Deploy a new stack or update an existing stack
docker diff	Inspect changes to files or directories on a container's filesystem
docker events	Get real time events from the server
docker exec	Run a command in a running container
docker export	Export a container's filesystem as a tar archive
docker history	Show the history of an image
docker image	Manage images
docker images	List images
docker import	Import the contents from a tarball to create a filesystem image
docker info	Display system-wide information
docker inspect	Return low-level information on Docker objects
docker kill	Kill one or more running containers
docker load	Load an image from a tar archive or STDIN
docker login	Log in to a Docker registry
docker logout	Log out from a Docker registry
docker logs	Fetch the logs of a container
docker network	Manage networks
docker node	Manage Swarm nodes
docker pause	Pause all processes within one or more containers
docker plugin	Manage plugins
docker port	List port mappings or a specific mapping for the container
docker ps	List containers
docker pull	Pull an image or a repository from a registry
docker push	Push an image or a repository to a registry
docker rename	Rename a container
docker restart	Restart one or more containers
docker rm	Remove one or more containers
docker rmi	Remove one or more images
docker run	Run a command in a new container
docker save	Save one or more images to a tar archive (streamed to STDOUT by default)
docker search	Search the Docker Hub for images
docker secret	Manage Docker secrets

<https://docs.docker.com/engine/reference/commandline/docker/>

Docker: The Next-Gen of Virtualization



General Command in Docker

docker secret	Manage Docker secrets
docker service	Manage services
docker stack	Manage Docker stacks
docker start	Start one or more stopped containers
docker stats	Display a live stream of container(s) resource usage statistics
docker stop	Stop one or more running containers
docker swarm	Manage Swarm
docker system	Manage Docker
docker tag	Create a tag TARGET_IMAGE that refers to SOURCE_IMAGE
docker top	Display the running processes of a container
docker trust	Manage trust on Docker images (experimental)
docker unpause	Unpause all processes within one or more containers
docker update	Update configuration of one or more containers
docker version	Show the Docker version information
docker volume	Manage volumes
docker wait	Block until one or more containers stop, then print their exit codes

<https://docs.docker.com/engine/reference/commandline/docker/>

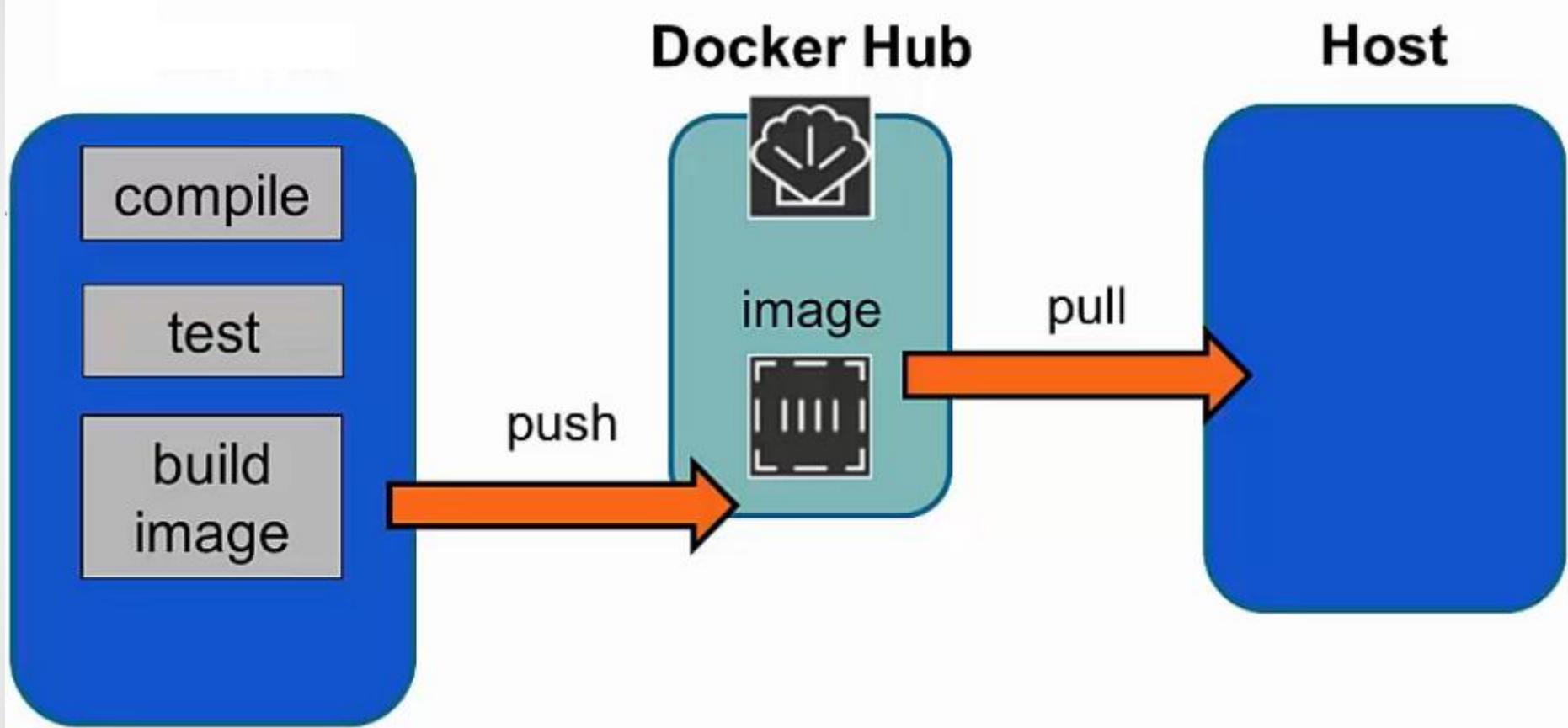
Docker: The Next-Gen of Virtualization



Image, Repository and Container

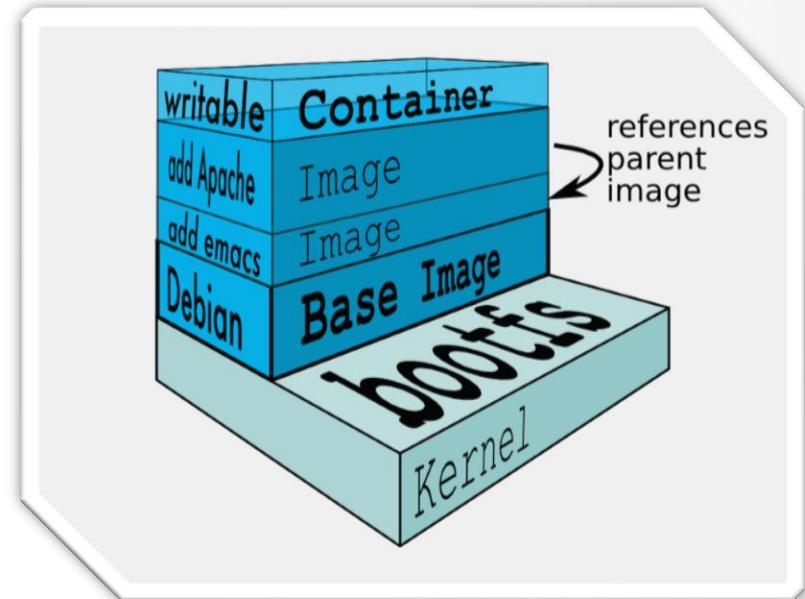
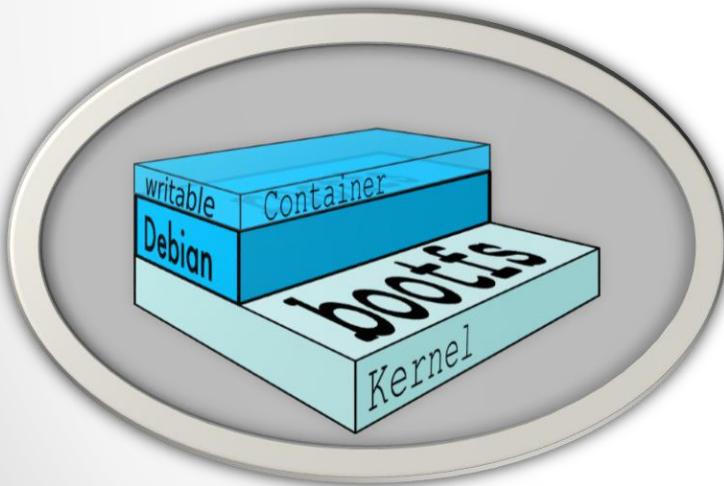
• • •

Docker Deployment



Docker Image

- Image คือ template ที่ถูกสร้างขึ้นเพื่อเตรียมใช้ในการรัน container
- เป็นไฟล์ที่อ่านได้อย่างเดียว
- ถูกสร้างโดยผู้ใช้งานเอง หรือผู้อื่น
- จัดเก็บไว้ใน repository (hub.docker.com, registry, trust registry)



Docker Image

- เรียกดู image ที่อยู่ภายในเครื่อง

```
docker images / docker image ls
```

```
Administrator: Windows PowerShell
PS C:\WINDOWS\system32> docker image ls
REPOSITORY          TAG      IMAGE ID      CREATED       SIZE
labdocker/alpineweb latest   5770233f642f  12 hours ago  50.5MB
labdocker/ubuntu    latest   ea4c82dcd15a  2 days ago   85.8MB
labdocker/alpine    latest   196d12cf6ab1  5 weeks ago  4.41MB
labdocker/cadvisor  latest   75f88e3ec333  10 months ago 62.2MB
PS C:\WINDOWS\system32>
```

- tag image ใหม่เพื่อให้ตรงตามความต้องการในการใช้งาน

```
docker image tag <image id> <acc name/imagename: version>
```

```
Ex: docker image tag 14f89d0e6257 labdocker/alpinelab:1.0
```

Repository (Registry)

- component ในการจัดเก็บ docker (image) ต่างๆ รวมไปที่ศูนย์กลาง เพื่อให้ docker engine บนเครื่องต่างๆ มาเรียก image ไปใช้งาน
- รองรับการเก็บ version ของ image อย่างเป็นระบบ
- สามารถให้ข้อมูลหรือคุณมีอ่อนน้ำในการใช้งานกับผู้ download image ได้
- มี official image ที่สร้างจากผู้พัฒนาโปรแกรมเอง
- Docker มีให้บริการ repository บน cloud แก่ผู้ใช้งาน
- Url: <https://hub.docker.com/>
- Free registry without cost
- Private repository (registry, trust registry)

Repository (Registry)

- Hub.docker.com

The screenshot shows a web browser window with the URL <https://hub.docker.com> in the address bar. The page has a dark blue header with a search bar containing 'Search' and a navigation bar with icons for back, forward, search, and other functions. Below the header, there's a top navigation bar with a ship icon, 'Explore', 'Help', a search bar with 'windows', and a 'Log In' button. The main content area features a large call-to-action section with the text 'Build, Ship, & Run Any App, Anywhere' in white and blue. Below this, a sub-section highlights 'Dev-test pipeline automation, 100,000+ free apps, public and private registries'. To the right, there's a 'New to Docker?' section with a sub-section 'Create your free Docker ID to get started.' It includes fields for 'Choose a Docker Hub ID', 'Enter your email address', and 'Choose a password'.

Repository (Registry)

PUBLIC REPOSITORY

[labdocker/cadvisor](#) ☆

Last pushed: 3 days ago

[Repo Info](#) [Tags](#) [Collaborators](#) [Webhooks](#) [Settings](#)

Short Description

cadvisor



Docker Pull Command



docker pull labdocker/cadvisor

Full Description

```
docker run \
--volume=/var/run:/var/run:rw \
--volume=/sys:/sys:ro \
--volume=/var/lib/docker/:/var/lib/docker:ro \
--publish=8080:8080 \
--detach=true \
--name=cadvisor \
labdocker/cadvisor:latest
```



Owner



labdocker

Repository (Registry)

- download image ออกจาก registry

```
docker image pull <account name/image name: tags>
```

Ex: docker image pull labdocker/alpine:latest

- Upload image ขึ้นไปเก็บบน registry

- Login

```
docker login <url> -u <username> -p <password> -e <email>
```

Ex: docker login -u labdocker -p xxxxxxxx -e xxxx@xxx.xxx

Ex: docker login 10.38.7.248:8080 -u labdocker

Repository (Registry)

- Push image ขึ้นไปเก็บบน registry

```
docker image push<account name/image name: tags>
```

Ex: docker image push labdocker/alpinelab:1.0

- Logoff ออกจาก Repository

```
docker logoff
```

*Docker 1.10 (upper) จะทำการ push แบบ parallel layer ทำให้สามารถ push image ได้เร็วขึ้น สาม
เท่าจากเดิม

Workshop 1-3: Create & Push Image

- ใน workshop นี้สอนการสร้าง image file สำหรับใช้เป็น web server และจัดเก็บ image file ลงใน repository ส่วนตัวเพื่อเตรียมไว้ใช้งาน
- สมัครใช้งาน <http://hub.docker.com> และแจ้งยืนยันอีเมล์เพื่อเริ่มใช้งาน
- ทำการสร้าง repository ชื่อ alpineweb ตามตัวอย่างด้านล่าง

The screenshot shows a 'Create Repository' form on the Docker Hub website. At the top left, there is a dropdown menu with 'labdocker' selected. To its right is a text input field containing 'alpineweb'. Below these fields is a large text area with the placeholder text 'alpineweb'. Underneath this area is a 'Visibility' section with a dropdown menu set to 'public'. At the bottom left of the form is a prominent blue 'Create' button.

Workshop 1-3: Build & Push Image

- ตรวจสอบ image id ด้วยคำสั่ง

```
docker image ls
```

- tag image ใหม่เป็น <accountname>/alpineweb:latest

```
docker image tag labdocker/alpineweb:latest \
<accountname>/alpineweb:latest
```

- Login เข้า hub.docker.com ด้วย username/password ที่ตั้งไว้

```
docker login -u <xxxx>
```

- Push image alpine ที่ tag ไว้ขึ้น repository

```
docker image push <accountname>/alpineweb:latest
```

Store.docker.com



The screenshot shows the Docker Store homepage with a teal header. On the left is the "docker store" logo. On the right are links for "Explore", "Publish", and "Feedback", followed by a user profile icon for "praparn". Below the header, the title "The Docker Store" is centered, with the subtitle "Find Trusted and Enterprise Ready Containers, Plugins, and Docker Editions" underneath. A search bar with the placeholder "Search for great content (e.g. mysql)" is positioned below the subtitle. At the bottom of the page, there are navigation links for "Docker EE", "Docker CE", "Containers", and "Plugins".

Docker: The Next-Gen of Virtualization



Store.docker.com

The Docker Store

Find Trusted and Enterprise Ready Containers, Plugins, and Docker Editions

Docker EE Docker CE **Containers** Plugins

Filters (1) [Clear All](#)

1 - 10 of 26 available images.

Most Popular

Store

Community (Docker Hub)

Docker Certified

Analytics

Application Frameworks

Application Infrastructure

Application Services

Databases

 Neo4j Community
Neo Technology, Inc.
Neo4j is a highly scalable, robust, native graph database. It is used in mission-critical apps by thousands of ...
[Databases](#)

 Oracle Database Enterprise Edition
Oracle
Oracle Database 12c Enterprise Edition
[Databases](#)

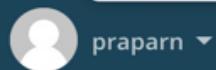
 docker

Docker: The Next-Gen of Virtualization

Store.docker.com



Explore Publish Feedback



Sysdig Cloud Monitoring Agent-Commercial

By Sysdig, Inc.

Sysdig Cloud is a commercial tool that provides docker monitoring and kubernetes monitoring for container-native apps

Categories: [Featured Images](#), [DevOps Tools](#), [Analytics](#), [Monitoring](#)



Basic Cloud

\$25 / mo

Pro Cloud

\$38 / mo

Pro Cloud is per-host, per-month, with up to 30 containers and 500 custom metrics per host. Contact us for customized plans.

[Terms of Service](#) | 1 Month Free Trial available

[Proceed to Checkout](#)



DESCRIPTION

REVIEWS

RESOURCES

What are your containers doing? Sysdig is the most powerful docker monitoring and troubleshooting solution. Built on ContainerVision™, Sysdig can see inside your docker containers without intrusive instrumentation. We natively integrate with the entire docker ecosystem, giving you deep visibility into your applications and microservices. Free trial at www.sysdig.com.

Please note that this container is the agent specifically for our commercial docker monitoring solution.

Our core features include:

*Cloud and On-premise: Deploy Sysdig the way you want. Let us collect your data in the cloud or run our back end behind your firewall.

Docker: The Next-Gen of Virtualization



Get this product by clicking on "Get Content" button above.



Store.docker.com

Docker Certified Trusted & supported products

- ✓ Certified Containers provide ISV apps available as containers.
- ✓ Certified Plugins for networking and volumes in containers.
- ✓ Certified Infrastructure delivers an optimized and validated Docker platform for enterprise OS and Cloud Providers.

[View Certified Images](#)



POPULAR EDITIONS



Docker Enterprise Edition for IBM
POWER
Server



Docker Enterprise Edition for IBM
Z
Server



Docker Enterprise Edition for AWS
Cloud



Docker Enterprise Edition for IBM
Cloud (Private Beta)
Cloud



Docker Enterprise Edition for
Azure
Cloud



Docker Enterprise Edition for
CentOS
Server

Cloud.docker.com (Obsolete)

[Pricing](#)[Resources](#)[Sign up](#)[Sign in](#)

Docker Cloud

The official cloud service to build and ship your Docker applications.

New to Docker?

Create your free Docker ID to get started.

Choose your Docker ID

Email address

Choose a password

[Sign up](#)



! **IMPORTANT MESSAGE** - Cluster Management in Docker Cloud will be discontinued on May 25. See our migration guides and view the notification.

Docker: The Next-Gen of Virtualization



Status.docker.com

Docker System Status

Current system status information.

 docker

[SUBSCRIBE](#)

All Systems Operational Updated a minute ago

Our system status page is a real-time view of the performance and uptime of Docker products and services.

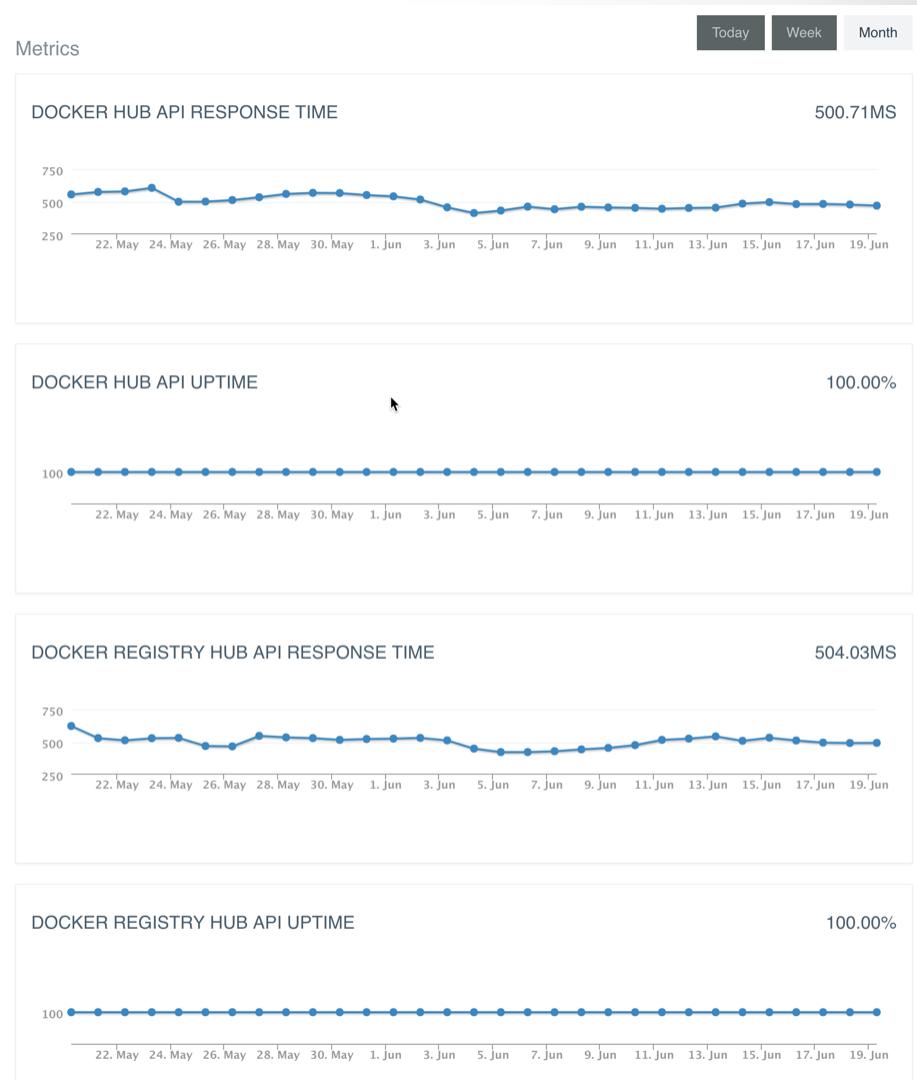
Docker Package Repositories AWS Operational

Docker Support Site NA11 Operational

Docker Community Forums FMT2 Operational

Docker.com Web IA03 Operational

<https://status.docker.com/>

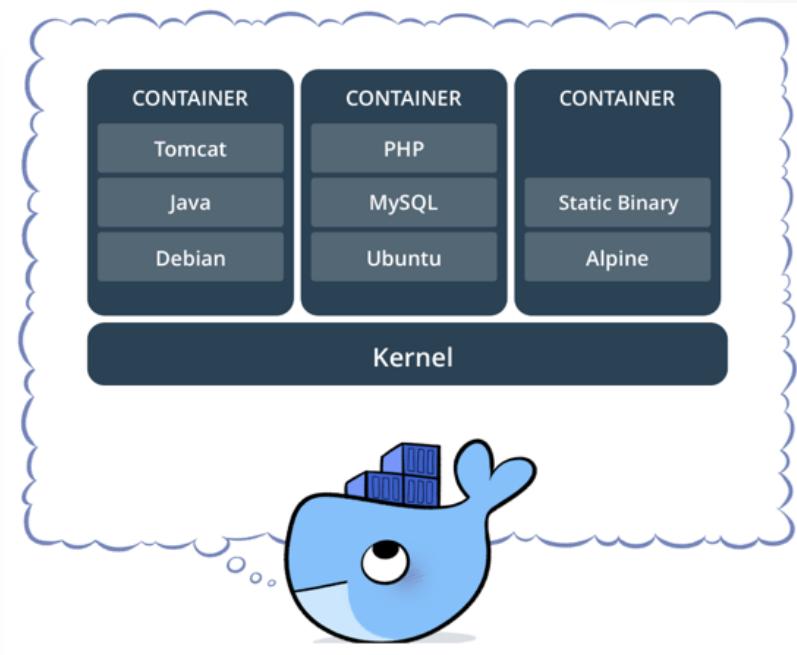
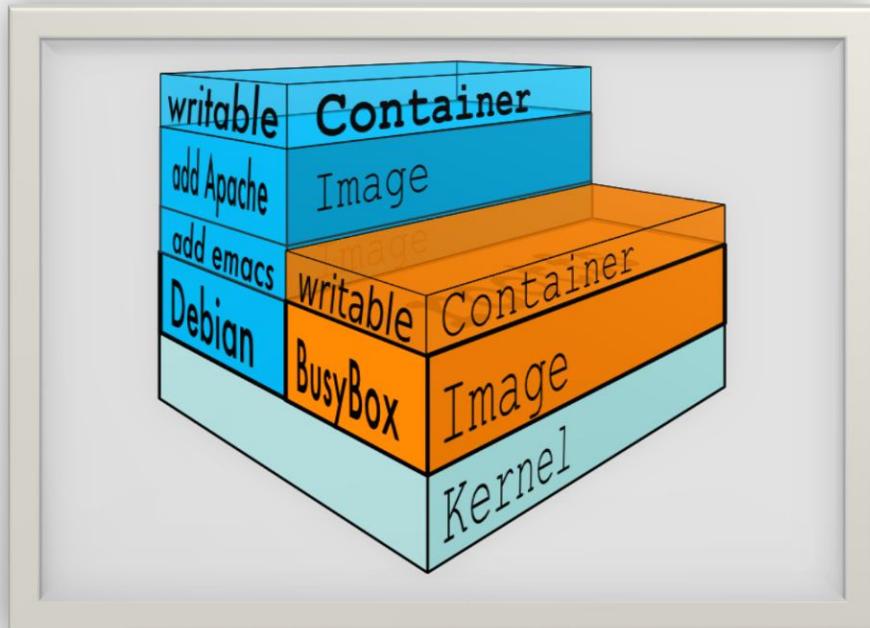


Docker: The Next-Gen of Virtualization



Docker Container

- Container คือชุดของ software layer ที่รันอิสระโดยอ้างอิงจาก image (run)
- ใช้สำหรับสร้างสภาพแวดล้อมที่จำเป็นต้องใช้ในการรันหรือพัฒนาโปรแกรม
- เมื่อพัฒนาโปรแกรมเสร็จเรียบร้อยแล้ว หรือต้องการ backup container สามารถสั่งเก็บ container ชุดปัจจุบันไปเป็น image เพื่อรอการเรียกใช้งานต่อไป (commit)



Docker Container

- สั่ง run docker เพื่อสร้าง container จาก image file

```
docker container run <option> <image id/name> <command>
```

Ex: docker container run -i -t --rm -p 3000:3000 \ labdocker/alpineweb:latest node hello.js

- สั่ง exec command ไปบน container ที่รันอยู่

```
docker container exec -it <container id/name> <command>  
docker container attach <container id/name>
```

- เริ่ม/หยุด container

```
docker container <start/stop> <container id/name>
```

- ลบ container

```
docker container rm <containerid/name>
```

Workshop 1-4: Run Container

- Ex 1: Interactive NODEJS
- สั่งรัน container แบบ Interactive โดยตั้งชื่อว่า nodejs และ map network 3000:3000

```
docker container run -i -t --rm --name nodejs -p 3000:3000 \
labdocker/alpineweb:latest node hello.js
```

- ทดสอบเปิดหน้าเว็บบน url: http://<ip address>:3000
- ตรวจสอบ container ด้วยคำสั่ง

```
docker container ps -a
```

- Exist with Ctrl+C

Workshop 1-4: Run Container

- Ex 2: DAEMON LINE BOT



ข้อมูลที่นำไปใช้ของ API
ประเภทของ API : REST
ประเภทของข้อมูล : JSON
ความปลอดภัย : PUBLIC

คำแนะนำที่อยู่ของ API
https://iapi.bot.or.th/Stat/Stat-ReferenceRate/DAILY_REF_RATE_V1/

พารามิเตอร์ใน Header
api-key : xk3h6ash-ahw5l3mch6hl3-2hhg6l4mng2346l34l6

ข้อมูลพารามิเตอร์สำหรับการใช้งาน API

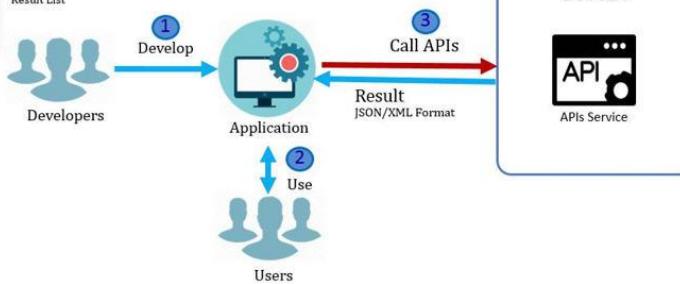
ชื่อตัวแปร	ประเภท	ค่าอธิบาย
start_period	Datetime	วันที่เริ่มคุณสมบัติอยู่ (YYYY-MM-DD) เช่น 2017-06-30
end_period	Datetime	วันที่สิ้นสุดคุณสมบัติ (YYYY-MM-DD) เช่น 2017-06-30

ข้อมูลพารามิเตอร์ที่จะได้รับ

ชื่อ	ประเภท	ค่าอธิบาย
report_name_eng	String	ชื่อรายงาน ภาษาอังกฤษ
rereport_name_th	String	ชื่อรายงาน ภาษาไทย
report_uoq_name_eng	String	หน่วย ภาษาอังกฤษ
report_uoq_name_th	String	หน่วย ภาษาไทย
report_source_of_data	String	แหล่งมาของข้อมูล
report_remark_eng	String	หมายเหตุ รายงานภาษาอังกฤษ
report_remark_th	String	หมายเหตุ รายงานภาษาไทย
last_updated	String	วันที่แก้ไขข้อมูลล่าสุด
period	String	วันที่รวมคุณสมบัติ
rate	String	อัตรา



BOT API Services Process



Workshop 1-4: Run Container

• Ex 2: DAEMON LINE BOT

```
[docker@labdocker:~$ docker container run --rm --name linebot -e "TITLE=Line BOT (Bank of Thailand)" -e "TOKEN=EIeqyillzkzpCo1ROrebZTcGbIyzDZhP0jcd0t6CX" labdocker/linenotify:bot_v1
null
{"statusCode":200,"body":"{\\"status\\":200,\\"message\\":\\"ok\\","headers":{"server":"nginx","date":"Sat, 20 Jan 2018 03:12:22 GMT","content-type":"application/json; charset=UTF-8","transfer-encoding":"chunked","connection":"keep-alive","keep-alive":true,"timeout":3,"x-ratelimit-limit":1000,"x-ratelimit-imagelimit":50,"x-ratelimit-remaining":998,"x-ratelimit-imageerror":50,"x-ratelimit-reset":1516421345}, "request": {"uri": {"protocol": "https:", "slashes": true, "auth": null, "host": "notify-api.line.me", "port": 443, "hostname": "notify-api.line.me", "hash": null, "search": null, "query": null, "pathname": "/api/notify", "path": "/api/notify", "href": "https://notify-api.line.me/api/notify"}, "method": "POST", "headers": {"Content-Type": "application/x-www-form-urlencoded", "authorization": "Bearer EIeqyillzkzpCo1ROrebZTcGbIyzDZhP0jcd0t6CX", "content-length": 466}}}
{\\"status\\":200,\\"message\\":\\"ok\\"
docker@labdocker:~$
```

```
1 const request = require('request');
2 const qs = require('querystring');
3 const TITLE=process.env.TITLE;
4 const KEY =process.env.KEY; //Input API-KEY to Header
5 //const KEY="U9G1L457H60CugT7VmBaEacbHV9RX0PyS005cYaGsm";
6 const URL =process.env.URL; //Input URL for Request
7 //const URL="https://api.bot.or.th/Stat/Stat-ReferenceRate/DAILY_REF_RATE_V1/?";
8 //const TOKEN = 'EIeqyillzkzpCo1ROrebZTcGbIyzDZhP0jcd0t6CX';
9 const TOKEN=process.env.TOKEN; //Input TOKEN LINE
10 //Set Date
11 var dateFormat = require('dateformat');
12 var daynow = new Date(); // Today!
13 daynow.setDate(daynow.getDate() - 1); // Yesterday!
14 var daynow=dateFormat(daynow, "yyyy-mm-dd");
15 var DAYSTARTPERIOD=daynow;
16 var DAYENDPERIOD=daynow;
17 //Set Date
18 var stickerPkg=2; //stickerPackageId
19 var stickerId=161; //stickerId
20 var Message="";
21 //Setup QueryString
22 var queryString = qs.stringify({
23   start_period: DAYSTARTPERIOD,
24   end_period: DAYENDPERIOD
25 });
26 //Setup QueryString
27 function callback(error, response, body) {
28   //console.log(URL+queryString);
29   //console.log('Response Code: '+ response.statusCode);
30   //console.log(body)
31   if (!error && response.statusCode == 200) {
32     var info = JSON.parse(body);
```



LineBot_Container: TITLE:Line BOT (Bank of Thailand)
==Report from BOT API==
Report Name:Rates of Exchange of Commercial Banks in Bangkok Metropolis (2002-present)
Report Source:Bank of Thailand
Data:33.2770000 THD/USD
Last Update Timestamp:2017-08-10 16:39:04
Remark:Daily Weighted-average Interbank Exchange Rate - THB / USD



16:39



Workshop 1-4: Run Container

- Ex 2: DAEMON LINE BOT
- สร้าง container แบบ Daemon โดยตั้งชื่อว่า linebot เพื่อทำการดึงข้อมูลจาก API ของธนาคารแห่งประเทศไทยอອກมาแล้วส่ง LINE Notify ไปหา Token Key ที่กำหนด

```
docker container run -it --rm --name linebot \
-e TITLE="Line BOT (Bank of Thailand)" \
-e "TOKEN=<LINE TOKEN>" \
labdocker/linenotify:bot_v1
```

Workshop 1-4: Run Container

- Ex 3: Detach Mode NODEJS
- สร้าง container แบบ detach (daemon) โดยตั้งชื่อว่า nodejs และ map network 3000:3000

```
docker container run -d -t --name nodejs -p 3000:3000 \
labdocker/alpineweb:latest node hello.js
```

- ทดสอบเปิดหน้าเว็บบน url: http://<ip address>:3000
- ตรวจสอบ container ด้วยคำสั่ง

```
docker container ps -a
```

- ทดสอบ Shell เข้าสู่ container ด้วยคำสั่ง

```
docker container exec -i -t nodejs sh
```

Workshop 1-4: Run Container

- Ex 3: Detach Mode NODEJS

- สั่งหยุดเริ่ม container ด้วยคำสั่ง stop

```
docker container stop nodejs
```

```
docker container start nodejs
```

- ทำการตรวจสอบ container offline ด้วยคำสั่ง

```
docker container ps -a
```

- ทดสอบ Shell เข้าสู่ container ด้วยคำสั่ง

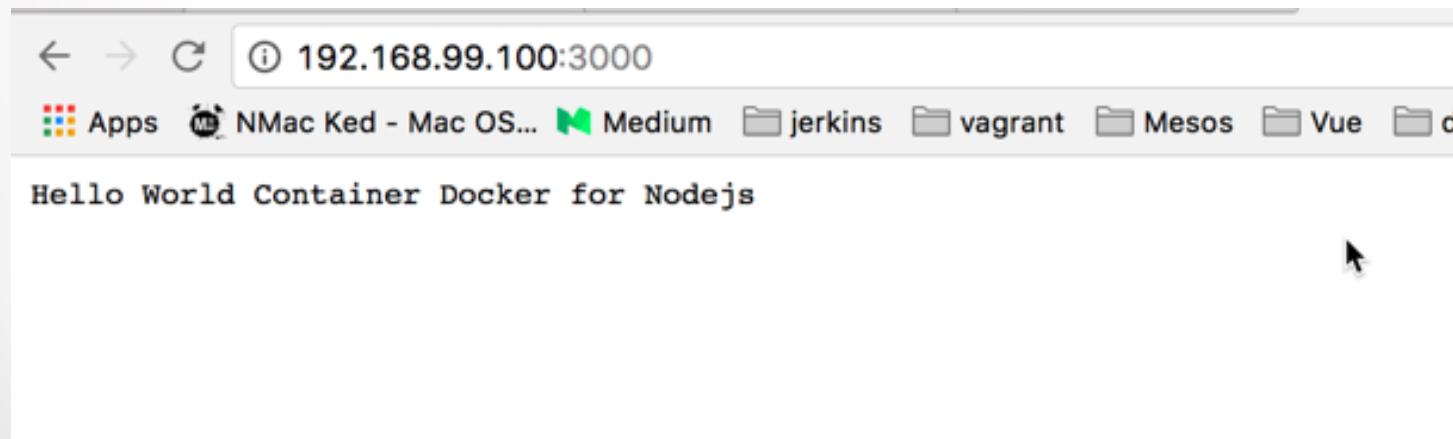
```
docker container exec -i -t nodejs sh
```

```
docker container attach nodejs
```

Workshop 1-4: Run Container

- Ex 3: Detach Mode NODEJS

```
1 var http = require('http');
2
3 http.createServer(function (req, res) {
4     res.writeHead(200, {'Content-Type': 'text/plain'});
5     res.end('Hello World Container Docker for Nodejs\n');
6 }).listen(3000, '0.0.0.0');
7
8 console.log('Server running at http://0.0.0.0:3000/');
```



Workshop 1-4: Run Container

- Ex 4: Detach Mode Python
- สร้าง container แบบ detach (daemon) โดยตั้งชื่อว่า python และ map network 5000:5000

```
docker container run -d -t --name python -p 5000:5000 \
labdocker/cluster:webservicelite
```

- ทดสอบเปิดหน้าเว็บบน url: http://<ip address>:5000
- ทดสอบ Shell เข้าสู่ container ด้วยคำสั่ง

```
docker container exec -i -t python sh
```

```
docker container attach python
```

Workshop 1-4: Run Container

- Ex 4: Detach Mode PYTHON

```
1  from flask import Flask
2  import os
3  import time
4  app = Flask(__name__)
5
6  @app.route('/')
7  def hello():
8      return '<H1> Welcome Page from Container Python Lab </H1>Checkpoint Date/Time: ' + time.strftime("%c") + '\n'
9
10 if __name__ == "__main__":
11     app.run(host="0.0.0.0", port=5000, debug=True)
```



CPU, Memory & I/O

• • •

CPU

- Default container จะมองเห็น CPU resource ทั้งหมดในเครื่องระหว่าง runtime และ share เวลาการทำงานให้ทุก container เท่ากัน
- config share time ในการใช้งาน (default: 1024)
`--cpu-shares, -c =“<0 (default): Ratio>”`
- config share core cpu
`--cpuset-cpus=“1, 3”`
- กำหนด cpu quota & period (default: 100 ms)
`--cpu-period=100000 --cpu-quota=500000`
- NUMA (non-uniform memory access) architecture
`--cpuset-mems=“1,3”`

Workshop 1-5: CPU Configure

- Download cAdvisor

```
docker image pull labdocker/cadvisor:latest
```

- สร้าง cAdvisor ตาม command ดังนี้

```
docker container run \
    --mount type=bind,source=/var/run,target=/var/run \
    --mount type=bind,source=/sys,target=/sys,readonly \
    --mount
    type=bind,source=/var/lib/docker,target=/var/lib/docker,readonly \
    --publish=8080:8080 \
    --detach=true \
    --name=cadvisor \
    labdocker/cadvisor:latest
```

Workshop 1-5: CPU Configure

- cAdvisor

The screenshot shows the cAdvisor web interface. At the top is a logo featuring a grey owl with large white eyes and a yellow beak, above the word "cAdvisor". Below the logo is a navigation bar with a single item: "/". Underneath the bar is a "root" button. The main content area has three sections: "Docker Containers", "Subcontainers", and "Isolation". The "Isolation" section is expanded, showing a "CPU" tab which is highlighted with a blue background and white text. Below the tab, the text "Shares 1024 shares" is visible.

Workshop 1-5: CPU Configure

- Download busybox เพื่อใช้ในการทดสอบ

```
docker image pull labdocker/busybox:latest
```

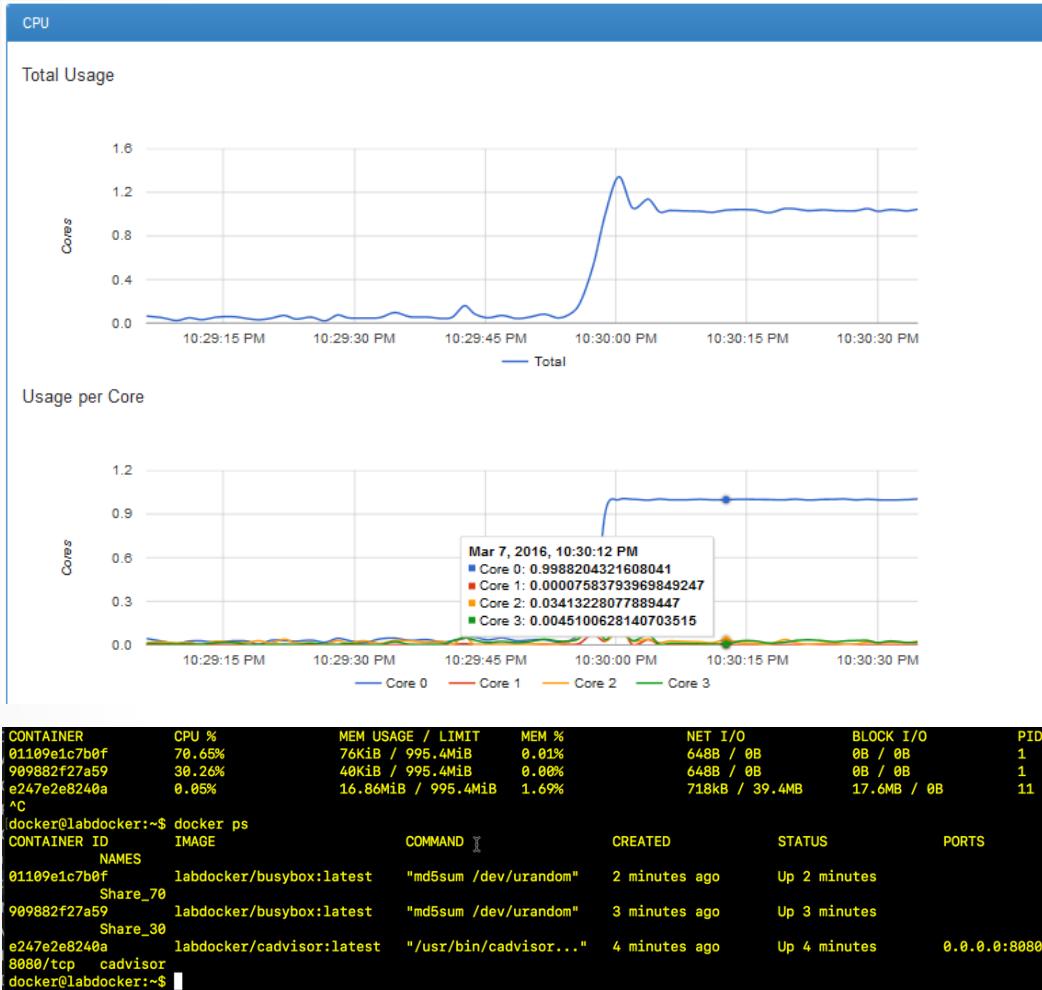
- Scenario 1 (Single CPU: 70/30)

```
docker container run -d \
--name='Share_30' \
--cpuset-cpus=0 \
--cpu-shares=30 \
labdocker/busybox:latest md5sum /dev/urandom
```

```
docker container run -d \
--name='Share_70' \
--cpuset-cpus=0 \
--cpu-shares=70 \
labdocker/busybox:latest md5sum /dev/urandom
```

Workshop 1-5: CPU Configure

- Result



Quiz ?

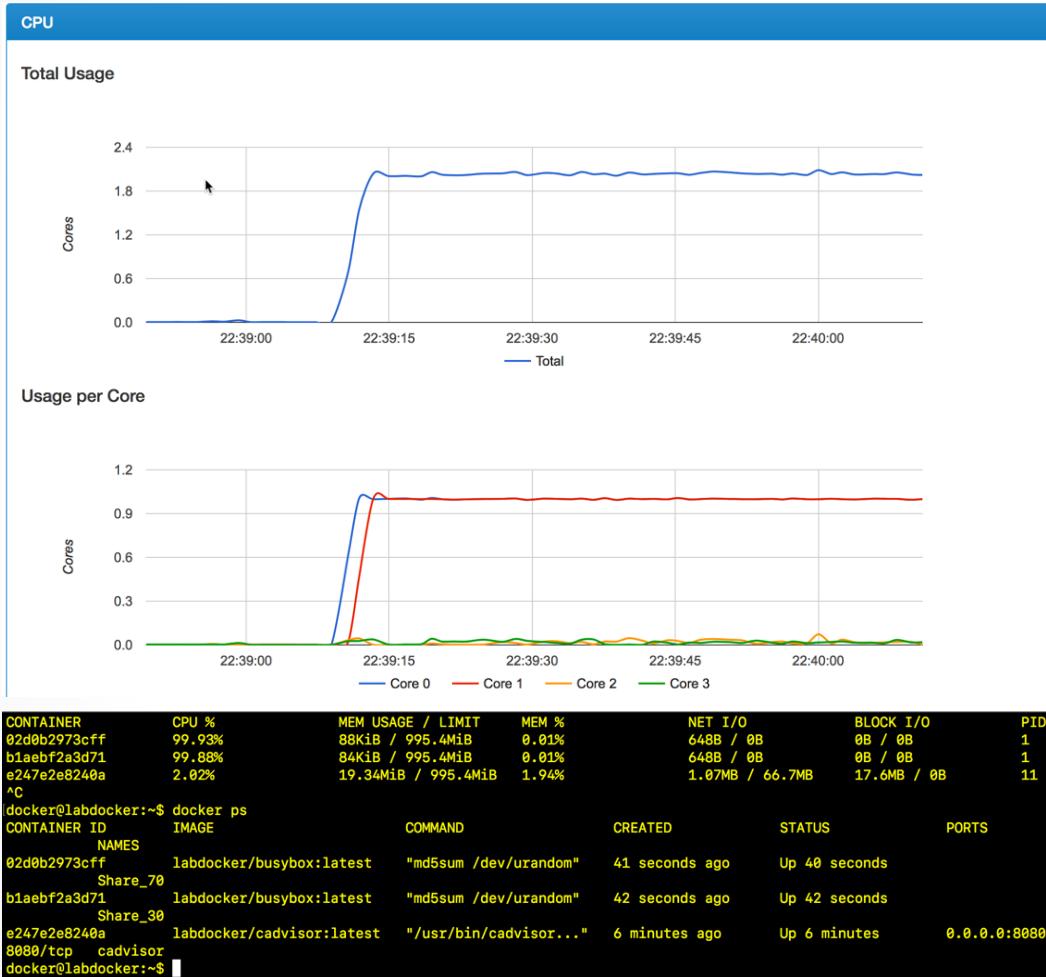
- Scenario (Multiple CPU: 70/30)

```
docker container run -d \
--name='Share_30' \
--cpuset-cpus=0 \
--cpu-shares=30 \
labdocker/busybox:latest md5sum /dev/urandom
```

```
docker container run -d \
--name='Share_70' \
--cpuset-cpus=1 \
--cpu-shares=70 \
labdocker/busybox:latest md5sum /dev/urandom
```

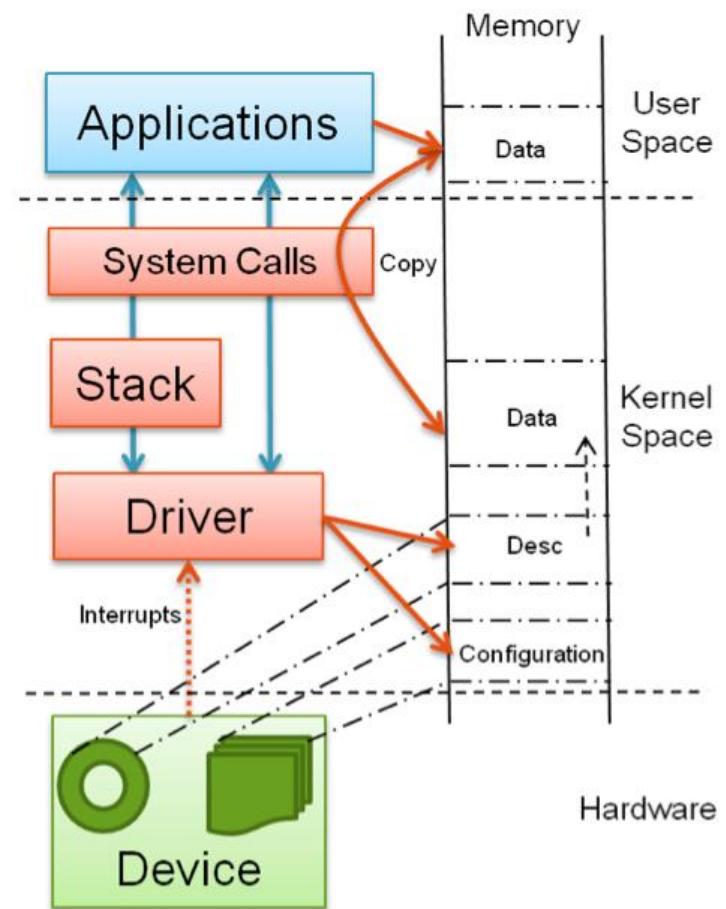
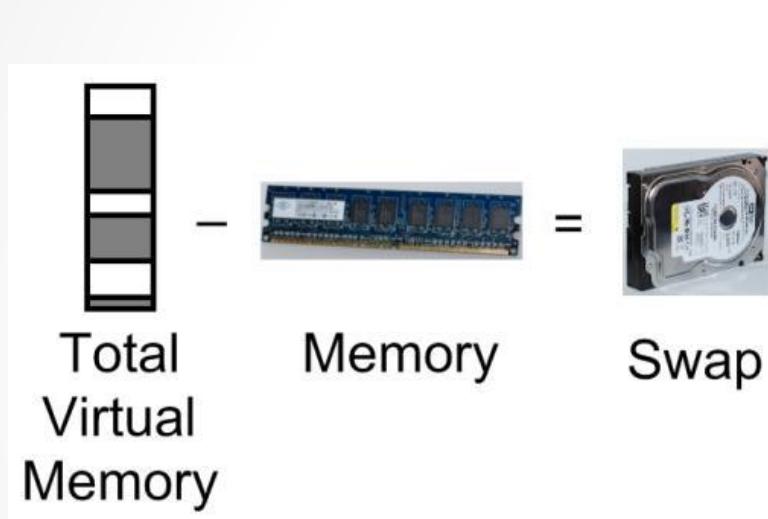
Quiz ?

- Result



Memory

- Default container จะมองเห็น memory resource ทั้งหมดในเครื่องระหว่าง runtime และสามารถใช้งาน memory resource ทั้งหมดเท่าที่ต้องการ



Memory

- การคอนฟิก memory บน container สามารถกำหนดได้ดังนี้
- memory limit and reservation (default: unlimit) (B:byte, K:kilobyte, M:megabyte, G:gigabyte)

```
--memory, -m =10G --memory-reservation= 1G
```

- memory swap (default: unlimit)

```
-- memory-swap= -1
```

- kernel memory

```
-- kernel-memory= 500M
```

- memory swappiness (kernel) (0=no swap)

```
-- memory-swappiness =0
```

Memory

- memory ใน production system
- กำหนด memory-swappiness = 0 (no swappiness for kernel memory)
- กำหนด memory swap (--memory-swap) เป็น 2 เท่าของ memory (Limit)
- monitor footprint ในการใช้งาน memory ของ application system และกำหนด เป็น “--memory-reservation”
- ทำ capacity testing เพื่อกำหนด maximum memory ที่ต้องการในการใช้งาน (+30%) และกำหนด “--memory“ (Limit)
- สำหรับ critical system ให้กำหนด “--memory-reservation” เท่ากับ “--memory“ (Limit)
- Memory Priority ? (Still waiting)

I/O

- โดย default ทุก container จะมีโอกาสใช้ I/O เท่าๆกัน (500 weight)
- กำหนดค่า weight ในการใช้งาน I/O (สำหรับ Direct IO เท่านั้น)
`--blkio-weight 300, --blkio-weight-device "/s01/tdb:500"`
- จำกัดการอ่านข้อมูล เป็น bps (kb: kilobyte, mb: megabyte, gb: gigabyte) / iops
`--device-read-bps /s01/tdb:1mb`
`--device-read-iops /s01/tdb:20000`
- จำกัดการเขียนข้อมูล เป็น bps , iops
`--device-write-bps /s01/tdb:1mb`
`--device-write-iops /s01/tdb:20000`

I/O

- I/O configure in production system
- weight I/O ควรกำหนดเฉพาะในกรณีที่ application ต้องทำงานกับ direct i/o (slow than cache)
- Monitor การใช้งาน I/O ของ container (IOSTAT -xtc) และปรับแต่ง

device	extended device statistics								tty			cpu			
	r/s	w/s	kr/s	kw/s	wait	actv	svc_t	%w	%b	tin	tout	us	sy	wt	id
fd0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0	0	0	0	0	0	100
sd0	0.0	0.0	0.4	0.4	0.0	0.0	49.5	0	0	0	0	0	0	0	0
sd6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0	0	0	0	0	0	0
nfs1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0	0	0	0	0	0	0
nfs49	0.0	0.0	0.0	0.0	0.0	0.0	15.1	0	0	0	0	0	0	0	0
nfs53	0.0	0.0	0.4	0.0	0.0	0.0	24.5	0	0	0	0	0	0	0	0
nfs54	0.0	0.0	0.0	0.0	0.0	0.0	6.3	0	0	0	0	0	0	0	0
nfs55	0.0	0.0	0.0	0.0	0.0	0.0	4.9	0	0	0	0	0	0	0	0

- Web server / Application server → IOOPS ต่ำ / Transfer สูง
- Database server → IOOPS ต่ำ / Transfer สูง
- คำนึงถึง physical storage performance

I/O

- Normal disk iops

Disk Speed	IOPS	RAID	Write Penalty
15,000	175	0	1
10,000	125	1	2
7200	75	5	4
5400	50	6	6
		DP	2
		10	2

- Raw disk performance: disk iops x unit of disk
 - **Ex:** SAS 128GB (15K) x 6 units => $175 \times 6 = 1060$ iops
- Raid disk performance:
 - $((\text{raw iops} \times \% \text{write}) / \text{penalty}) + (\text{raw} \times \% \text{read})$
 - Normal situation (write 30%, read 70%)
 - **Ex:** Raid 5 $((1060 \times .3) / 4) + (1060 \times .7) \rightarrow 821.6$ iops
 - **Ex:** Raid 1 $((1060 \times .3) / 2) + (1060 \times .7) \rightarrow 901$ iops

Network

• • •

Network

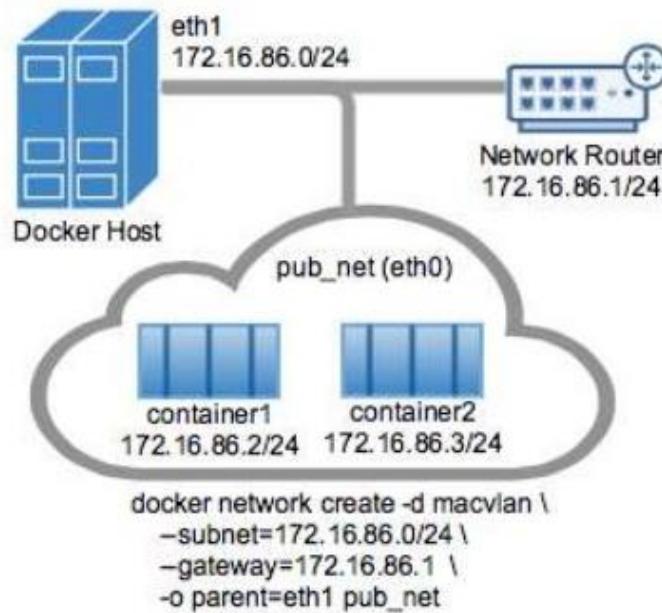
- Software define network by design (virtual switch)
- default docker จะจัดเตรียม network มาให้สามรูปแบบ

```
[docker@labdocker:~$ docker network ls
NETWORK ID      NAME      DRIVER      SCOPE
0dc1bbf00498    bridge    bridge      local
91d6dd4056a9    host      host       local
fa786a00adda    none      null       local
docker@labdocker:~$ ]
```

- **Bridge/User Defined** คือ default network สำหรับให้ container เชื่อมต่อออกสู่โลกภายนอกผ่าน virtual switch “docker0” โดยใช้ network stack ของ container เองในการเชื่อมต่อ (route mode)
- **none** คือ network loopback (127.0.0.1) สำหรับ container ที่ไม่มีการเชื่อมต่อออกไปด้านนอก
- **host** คือ network host ที่ container ใช้งาน host network stack ในการทำงาน (ใช้ในกรณี ต้องการ network performance สูงสุด) (security concern)

Network

- Additional network type
- **MACVLAN**



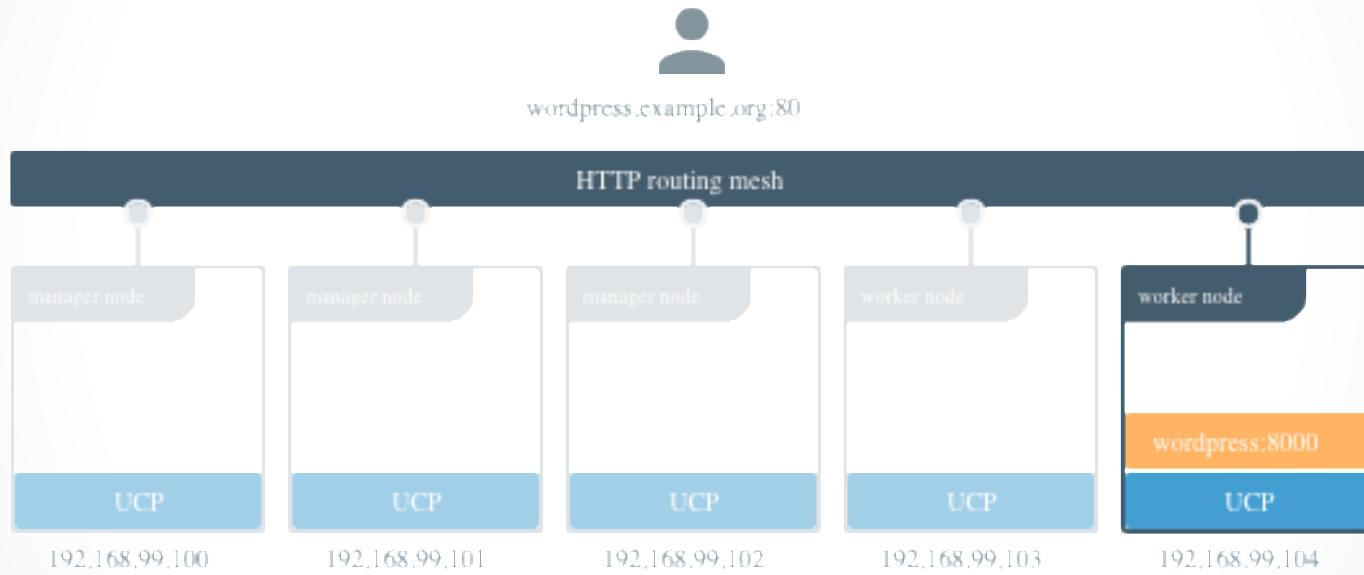
- **Overlay (Swarm Mode)**
- **3rd Party Network Plugin (Swarm Mode)**

Network

- Summary
 - **Bridge/User Defined Network:** best when you need multiple containers to communicate on the same Docker host.
 - **Host networks:** best when the network stack should not be isolated from the Docker host, but you want other aspects of the container to be isolated.
 - **Overlay networks:** best when you need containers running on different Docker hosts to communicate, or when multiple applications work together using swarm services.
 - **Macvlan networks:** best when you are migrating from a VM setup or need your containers to look like physical hosts on your network, each with a unique MAC address
 - **Third-party network:** allow you to integrate Docker with specialized network stacks.

Network

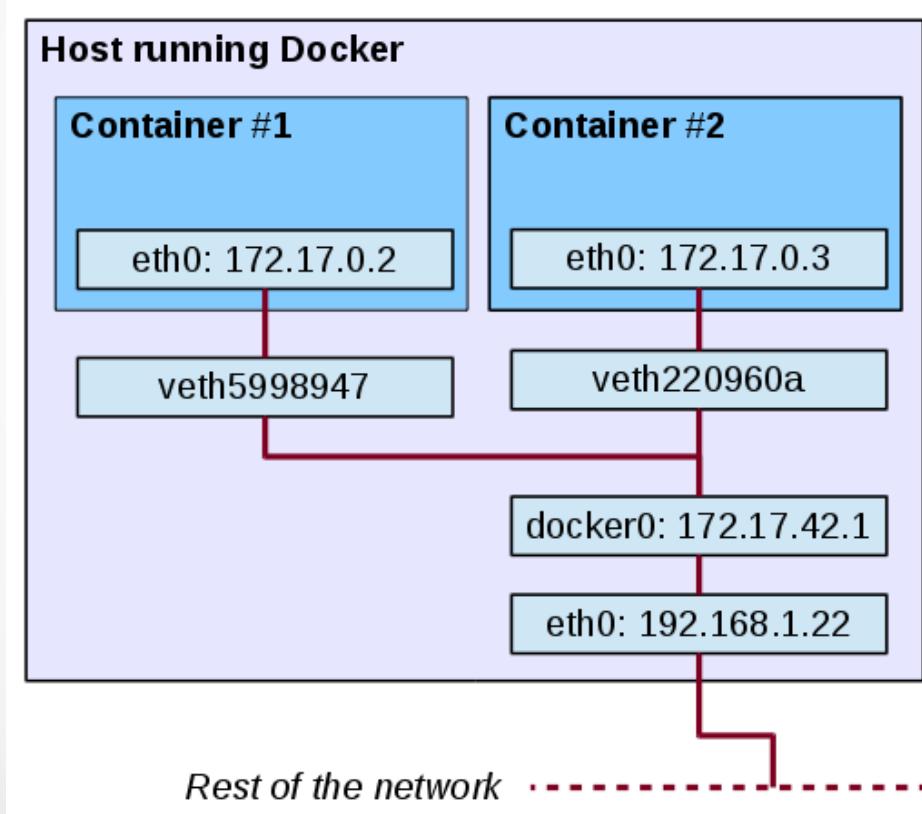
- Docker EE Network Feature:
 - **HTTP Route Mesh:** load balance L4 to target container on specific docker host



- **Session Stickiness:** (Cookies Base) for re-route traffic from original to same end-point container (Stateful)

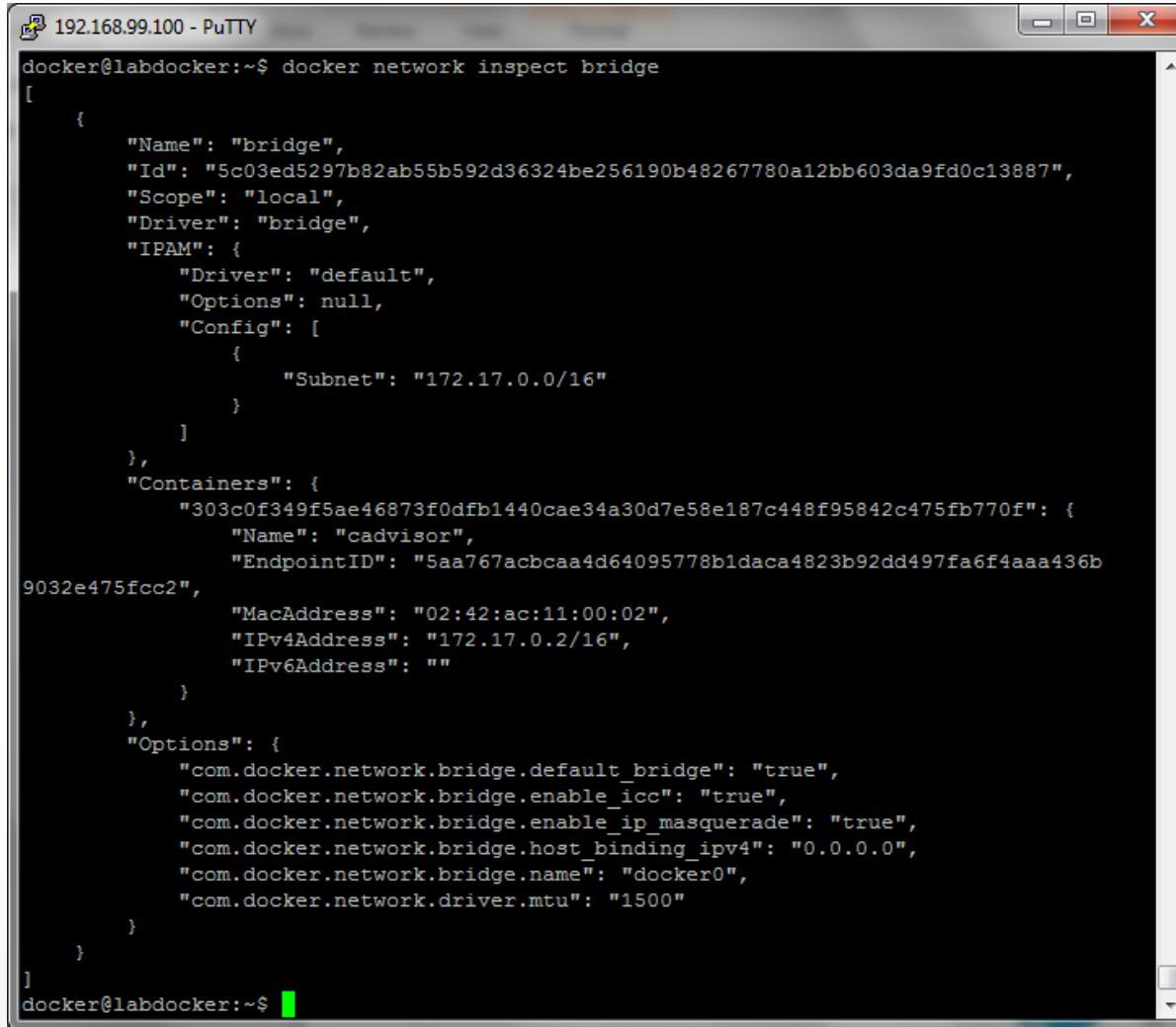
Network

- bridge (default) เมื่อสั่งรัน container ในระบบโดยไม่ได้ระบุ network ใดๆ container จะถูกเพิ่ม ipaddress, subnet เข้าสู่ bridge network โดยอัตโนมัติ



Network

- docker network inspect bridge



```
192.168.99.100 - PuTTY
docker@labdocker:~$ docker network inspect bridge
[
    {
        "Name": "bridge",
        "Id": "5c03ed5297b82ab55b592d36324be256190b48267780a12bb603da9fd0c13887",
        "Scope": "local",
        "Driver": "bridge",
        "IPAM": {
            "Driver": "default",
            "Options": null,
            "Config": [
                {
                    "Subnet": "172.17.0.0/16"
                }
            ]
        },
        "Containers": {
            "303c0f349f5ae46873f0dfb1440cae34a30d7e58e187c448f95842c475fb770f": {
                "Name": "cadvisor",
                "EndpointID": "5aa767acbc当地4d64095778b1daca4823b92dd497fa6f4aaa436b9032e475fcc2",
                "MacAddress": "02:42:ac:11:00:02",
                "IPv4Address": "172.17.0.2/16",
                "IPv6Address": ""
            }
        },
        "Options": {
            "com.docker.network.bridge.default_bridge": "true",
            "com.docker.network.bridge.enable_icc": "true",
            "com.docker.network.bridge.enable_ip_masquerade": "true",
            "com.docker.network.bridge.host_binding_ipv4": "0.0.0.0",
            "com.docker.network.bridge.name": "docker0",
            "com.docker.network.driver.mtu": "1500"
        }
    }
]
docker@labdocker:~$
```

Network

- Option เกี่ยวกับ network สำหรับสั่ง run container

```
--dns= x.x.x.x (Default --name,--net-alias,--link also  
dns internal docker)  
--net=<bridge/none/host/custom>  
--net-alias = "xxxx"  
--add-host="xxxx"  
--mac-address="xx:xx:xx:xx"  
--ip="x.x.x.x"  
--ipv6="xx:xx:xx:xx" <new in 1.11>  
-p, --publish = 9999:9999  
-P, --publish-all → Auto map network
```

```
docker container run -i -t --rm --name nodejs -p 3000:3000 \  
labdocker/alpineweb:latest node nodejs/hello.js
```

Network

- สร้าง custom virtual switch เพื่อจัดระเบียบและแบ่งแยก network ของ application ออกจากกัน (Recommend for Production)

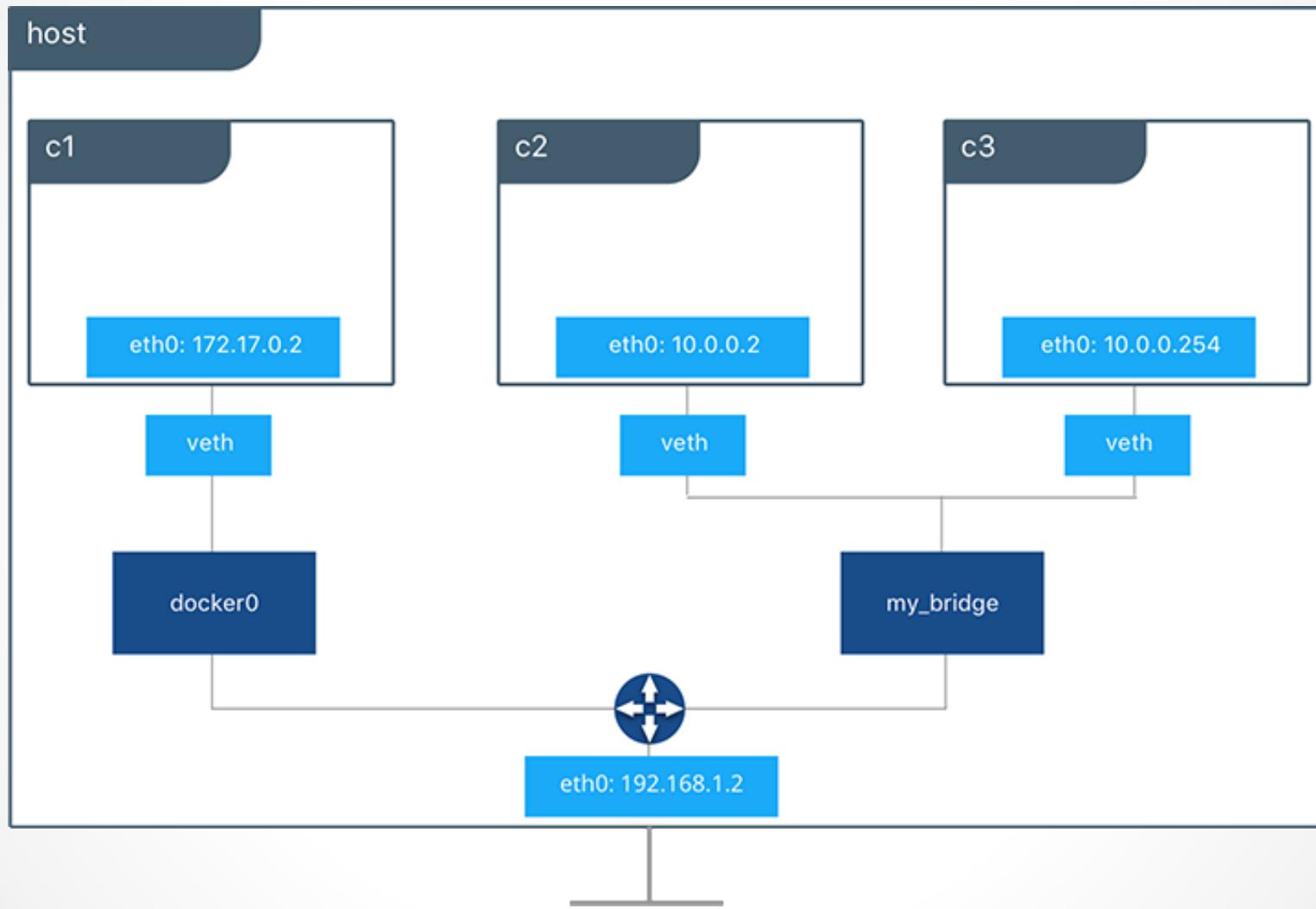
```
docker network create --driver <default/null/host> name
```

```
Ex: docker network create --driver default netweb
```

```
Ex: docker network rm netweb
```

- สามารถเพิ่มเติม option ในการสร้าง virtual switch เพิ่มเติม
 - d macvlan -o parent = กำหนด vlan tagging
 - subnet = xx (ระบุ ip address ของ virtual switch (Ex: 192.168.100.0/24))
 - ip-range = xx (ระบุ ip address ที่จะส่งให้ container ทำงาน) (Ex: 192.168.100.128/25)
 - gateway = xx (ระบุ ip address gateway) (Ex: 192.168.100.5)
 - opt = custom option
 - opt="com.docker.network.mtu"="1500"
 - opt="com.docker.network.bridge.host_binding_ipv4"=x.x.x.x

Network



Network

- การเพิ่ม network ใน container

```
docker network connect <network> <container>
```

Ex: docker network connect webnet web1

- การลด network ใน container

```
docker network disconnect <network> <container>
```

Ex: docker network disconnect webnet web1

Network

- Link container (add on /etc/hosts) (Legacy)

```
docker -dt --name web1 labdocker/alpineweb sh
```

```
docker -dt --name web2 --link web1:webmaster \  
labdocker/alpineweb sh
```

Legacy container links

Estimated reading time: 14 minutes

✖ Warning: The `--link` flag is a legacy feature of Docker. It may eventually be removed. Unless you absolutely need to continue using it, we recommend that you use user-defined networks to facilitate communication between two containers instead of using `--link`. One feature that user-defined networks do not support that you can do with `--link` is sharing environmental variables between containers. However, you can use other mechanisms such as volumes to share environment variables between containers in a more controlled way.

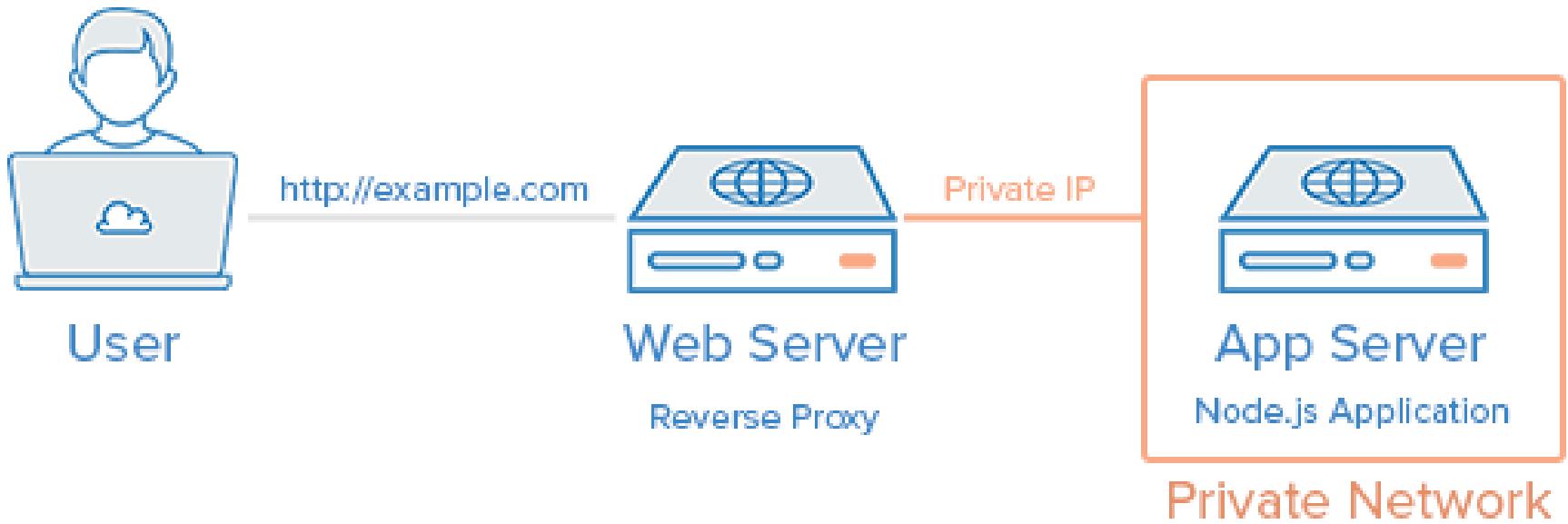
- DNS resolve on docker network (Recommend for Production)

Workshop 1-6: Network Configure

- Part 1: Reverse Proxy Network (DNS)
- Purpose: ทำการ Deploy nodejs webserver ด้วย solution proxy ของ nginx (load balance / reverse proxy)
- **public network** เพื่อให้บุคคลภายนอกเข้าใช้งานผ่าน nginx server (reverse proxy)
 - IP Address: 192.168.100.0/24
 - Range Address: 192.168.100.128 – 192.168.100.256
 - MTU: 1500
- **private network** เพื่อให้ nginx server เข้าเรียกใช้งาน nodejs web server
 - IP Address: 192.168.101.0/24
 - Range Address: 192.168.101.128 – 192.168.101.256
 - MTU: 9000

Workshop 1-6: Network Configure

Reverse Proxy to Node.js Application



Workshop 1-6: Network Configure

WEB1 (No Map)

DNS Name: web1 (Private)

WEB2 (No Map)

DNS Name: web2 (Private)

vSwitch: Webinternal

IP Address: 192.168.101.0/24

NGINX (Map Port:80:8080)

Join Network: Webinternal

Join Network: Webpublic

vSwitch: Webpublic

IP Address: 192.168.100.0/24

Map Port
(80:8080)

Public
Network

Workshop 1-6: Network Configure

- ทำการสร้าง Switch สำหรับเชื่อมต่อ Private / Public Network
 - docker network create --driver bridge \
--subnet=192.168.100.0/24 --ip-range=192.168.100.128/25 \
--gateway=192.168.100.5 --opt="com.docker.network.mtu"="1500"
webpublic
 - docker network create --driver bridge \
--subnet=192.168.101.0/24 --ip-range=192.168.101.128/25 \
--gateway=192.168.101.5 --opt="com.docker.network.mtu"="9000"
webinternalCreate Server

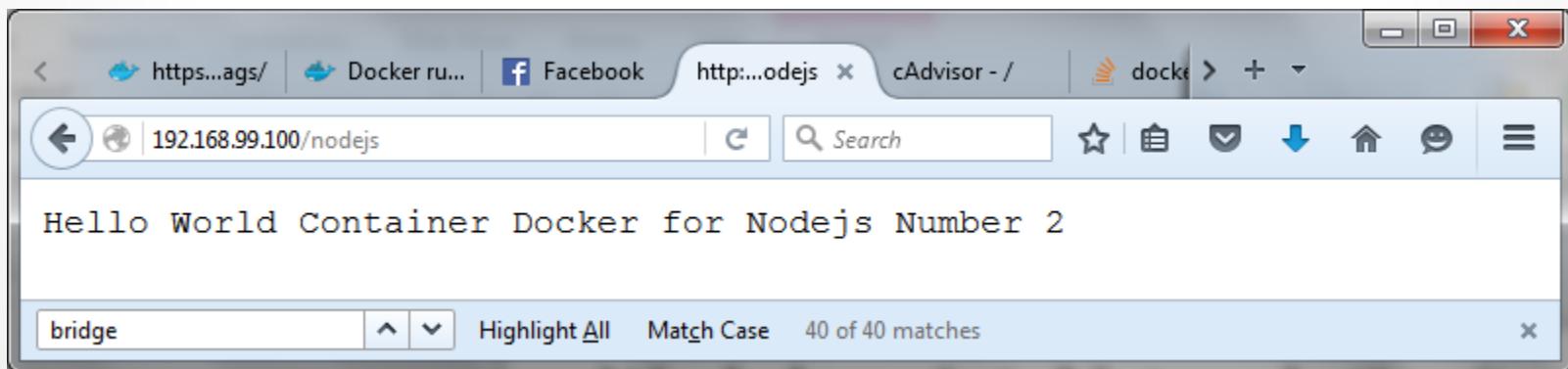
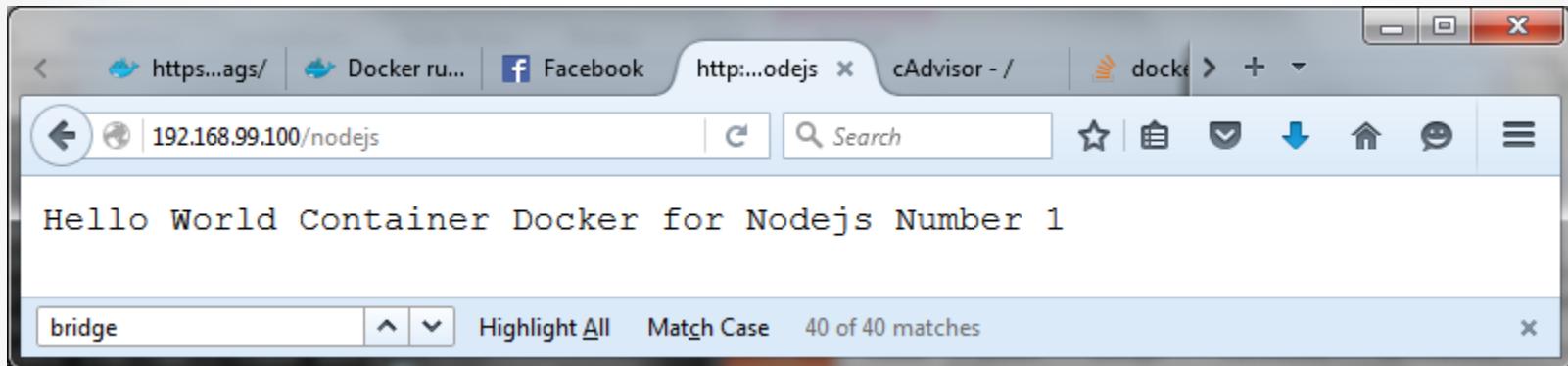
Workshop 1-6: Network Configure

- สร้างเครื่อง Webserver (nodejs) และ Nginx
 - Web1**
 - docker run -dt --name web1 --net webinternal \
--net-alias web1 labdocker/alpineweb:web1 node hello.js
 - Web2**
 - docker run -dt --name web2 --net webinternal \
--net-alias web2 labdocker/alpineweb:web2 node hello.js
 - NGINX**
 - docker run -dt --name nginx --net webinternal \
-p 80:8080 labdocker/nginx:lab

docker network connect webpublic nginx

Workshop 1-6: Network Configure

- url: http://<ip address of docker-machine>/nodejs



Workshop 1-6: Network Configure

- curl result:

```
/usr/sbin # curl http://web1:3000
Hello World Container Docker for Nodejs Number 1
/usr/sbin # curl http://web2:3000
Hello World Container Docker for Nodejs Number 2
/usr/sbin #
```

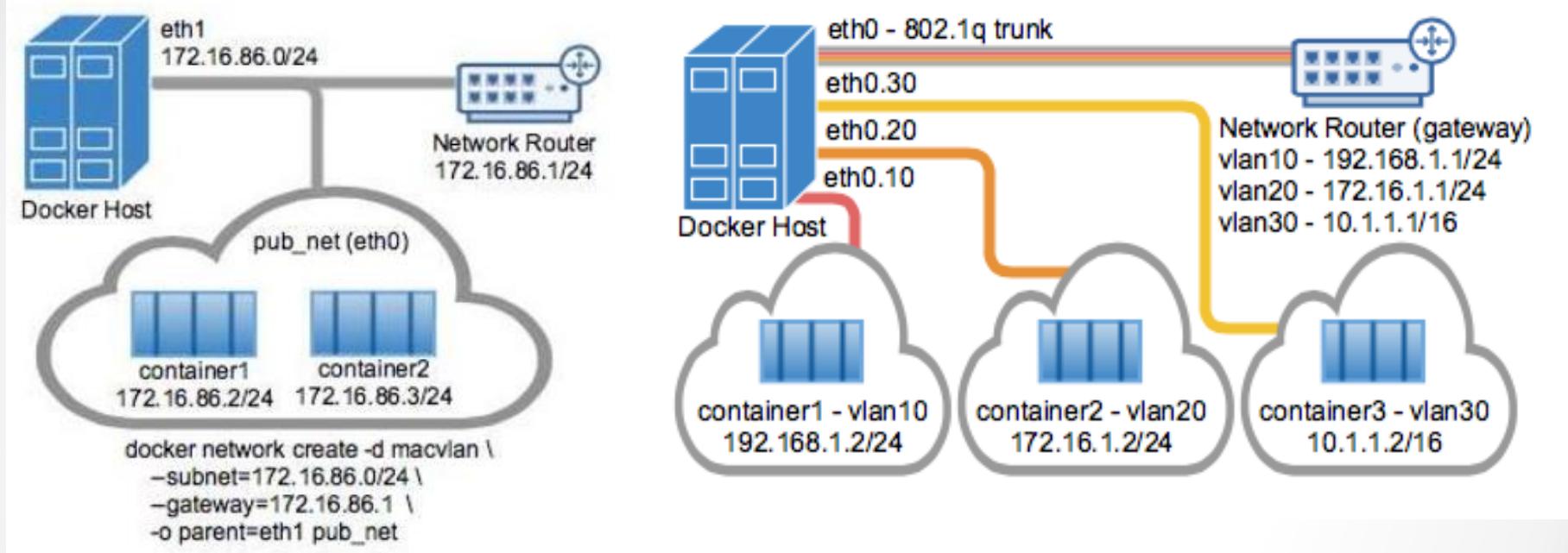
- nginx.conf

```
upstream nodejs_web {
    server web1:3000;
    server web2:3000;
}

server {
    listen 8080;
    location /nodejs{
        proxy_pass http://nodejs_web;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
    }
}
```

Network

- 802.1q Tagging (MACVLAN)



DEMO: MACVLAN

- Reverse Proxy Network (MACVLAN) (This need docker-machine)
- Purpose: ทำการ Deploy nodejs webserver ด้วย solution proxy ของ nginx ผ่าน Macvlan
- **Macvlan:**
 - Interface: eth1
 - IP Address: 192.168.99.0/24
 - Gateway: 192.168.99.1 → Your Machine
 - Labdocker: 192.168.99.100 → Default
 - Range Address: 192.168.99.192/28

100.1.0.00-100 - 100.1.0.00-001

<https://github.com/docker/libnetwork/blob/master/docs/macvlan.md>

Medium Jenkins vagrant Mesos Vue docker NGINX Taiwan Kubernetes PWA_Progressive_W... MYSQL_Cluster GeneralKB Node

- Note: Linux Macvlan interface types are not able to ping or communicate with the default namespace IP address. For example, if you create a container and try to ping the Docker host's eth0 it will not work. That traffic is explicitly filtered by the kernel to offer additional provider isolation and security. This is a common gotcha when a user first uses those Linux interface types since it is natural to ping local addresses when testing.

DEMO: MACVLAN

Hostname: labdocker

WEB1:

IP Address: 192.168.99.193

DNS Name: web1

Service Port: 3000

WEB2:

IP Address: 192.168.99.194

DNS Name: web2

Service Port: 3000

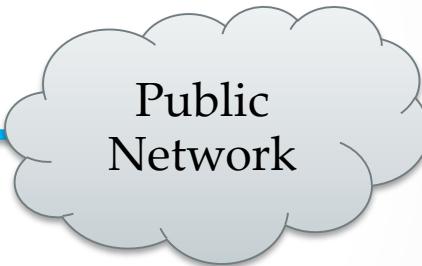
NGINX:

IP Address: 192.168.99.195

DNS Name: nginx

Service Port: 8080

Network Card: Eth1
IP Address: 192.168.99.100



DEMO: MACVLAN

- ทำการสร้าง Switch สำหรับเชื่อมต่อ Macvlan
 - docker network create -d macvlan \
--subnet=192.168.99.0/24 \
--ip-range=192.168.99.192/28 \
--gateway=192.168.99.1 \
-o parent=eth1 macvlanlab
- ในการนี้ต้องการทำ Tag VLAN (802.1q) จาก Switch/Router เข้าสู่ Docker Host และทำการสร้าง Sub Interface สามารถทำได้ดังนี้
 - docker network create -d macvlan \
--subnet=192.168.99.0/24 \
--ip-range=192.168.99.192/28 \
--gateway=192.168.99.1 \
-o parent=eth1.<vlan id> macvlanlab

DEMO: MACVLAN

- สร้างเครื่อง Webserver (nodejs) และ Nginx
 - Web1**
 - docker run -dt --name web1 --net macvlanlab \--net-alias web1 labdocker/alpineweb:web1 node hello.js
 - Web2**
 - docker run -dt --name web2 --net macvlanlab \--net-alias web2 labdocker/alpineweb:web2 node hello.js
 - NGINX**
 - docker run -dt --name nginx --net macvlanlab \labdocker/nginx:lab

DEMO: MACVLAN

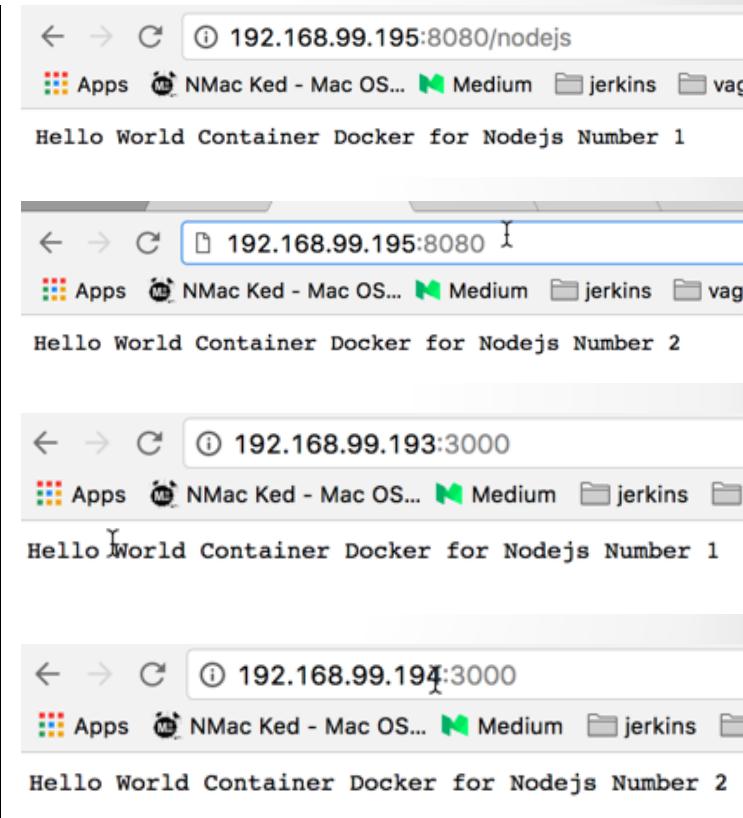
- Internal test:

```
[docker@labdocker:~$ docker exec -it nginx curl http://web1:3000
Hello World Container Docker for Nodejs Number 1
[docker@labdocker:~$ docker exec -it nginx curl http://web2:3000
Hello World Container Docker for Nodejs Number 2
[docker@labdocker:~$ docker exec -it ping web1
Error: No such container: ping
[docker@labdocker:~$ docker exec -it nginx ping web1
PING web1 (192.168.99.193): 56 data bytes
64 bytes from 192.168.99.193: seq=0 ttl=64 time=0.253 ms
64 bytes from 192.168.99.193: seq=1 ttl=64 time=0.405 ms
^C
--- web1 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 0.253/0.329/0.405 ms
[docker@labdocker:~$ docker exec -it nginx ping web2
PING web2 (192.168.99.194): 56 data bytes
64 bytes from 192.168.99.194: seq=0 ttl=64 time=0.256 ms
^C
--- web2 ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 0.256/0.256/0.256 ms
```

DEMO: MACVLAN

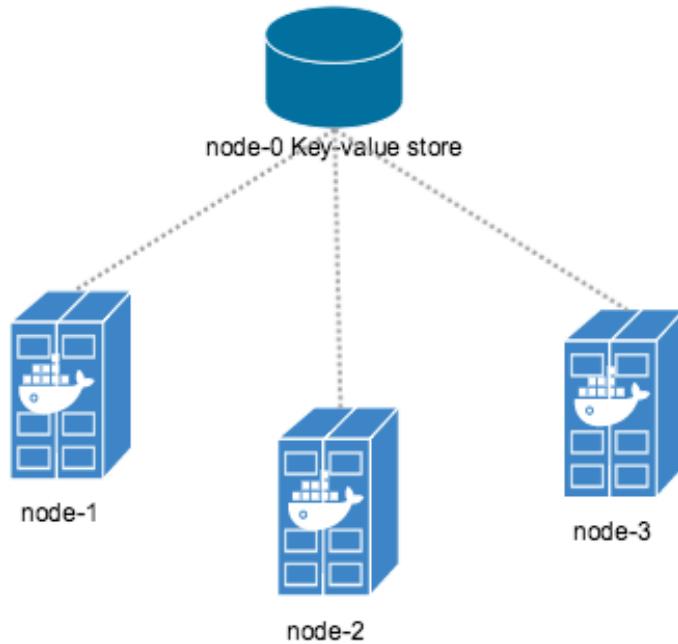
- External Test:

```
praparns-MacBook-Pro:~ praparn$ curl http://192.168.99.195:8080/nodejs
Hello World Container Docker for Nodejs Number 1
praparns-MacBook-Pro:~ praparn$ curl http://192.168.99.195:8080/nodejs
Hello World Container Docker for Nodejs Number 2
praparns-MacBook-Pro:~ praparn$ curl http://192.168.99.194:3000
Hello World Container Docker for Nodejs Number 2
praparns-MacBook-Pro:~ praparn$ curl http://192.168.99.193:3000
Hello World Container Docker for Nodejs Number 1
praparns-MacBook-Pro:~ praparn$ ping 192.168.99.195
PING 192.168.99.195 (192.168.99.195): 56 data bytes
64 bytes from 192.168.99.195: icmp_seq=0 ttl=64 time=0.471 ms
^C
--- 192.168.99.195 ping statistics ---
1 packets transmitted, 1 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 0.471/0.471/0.471/0.000 ms
praparns-MacBook-Pro:~ praparn$ ping 192.168.99.194
PING 192.168.99.194 (192.168.99.194): 56 data bytes
64 bytes from 192.168.99.194: icmp_seq=0 ttl=64 time=0.392 ms
^C
--- 192.168.99.194 ping statistics ---
1 packets transmitted, 1 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 0.392/0.392/0.392/0.000 ms
praparns-MacBook-Pro:~ praparn$ ping 192.168.99.193
PING 192.168.99.193 (192.168.99.193): 56 data bytes
64 bytes from 192.168.99.193: icmp_seq=0 ttl=64 time=0.495 ms
^C
--- 192.168.99.193 ping statistics ---
1 packets transmitted, 1 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 0.495/0.495/0.495/0.000 ms
praparns-MacBook-Pro:~ praparn$
```



Network

- Network across host (Overlay Network)



Network

- 3rd Party Network Plugin

Driver	Description
contiv	An open source network plugin led by Cisco Systems to provide infrastructure and security policies for multi-tenant microservices deployments. Contiv also provides integration for non-container workloads and with physical networks, such as ACI. Contiv implements remote network and IPAM drivers.
weave	A network plugin that creates a virtual network that connects Docker containers across multiple hosts or clouds. Weave provides automatic discovery of applications, can operate on partially connected networks, does not require an external cluster store, and is operations friendly.
calico	An open source solution for virtual networking in cloud datacenters. It targets datacenters where most of the workloads (VMs, containers, or bare metal servers) only require IP connectivity. Calico provides this connectivity using standard IP routing. Isolation between workloads – whether according to tenant ownership or any finer grained policy – is achieved via iptables programming on the servers hosting the source and destination workloads.
kuryr	A network plugin developed as part of the OpenStack Kuryr project. It implements the Docker networking (libnetwork) remote driver API by utilizing Neutron, the OpenStack networking service. Kuryr includes an IPAM driver as well.

Docker Remote IPAM Drivers

Community and vendor created IPAM drivers can also be used to provide integrations with existing systems or special capabilities.

Driver	Description
infoblox	An open source IPAM plugin that provides integration with existing Infoblox tools.

Volume

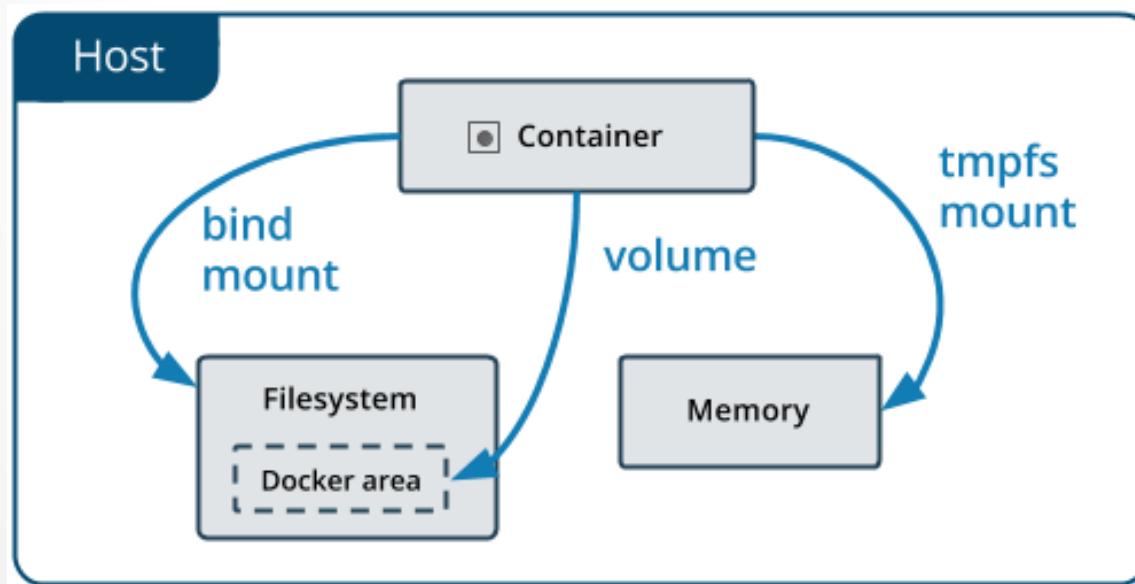
• • •

Volume

- ตามปกติแล้ว docker จะเก็บข้อมูลทุกอย่างเอาไว้ภายใน container
 - /var/log
 - /sys/data
 - /etc/nagios/
 - /etc/mysql
 - Etc.
- สำหรับการใช้งาน container ใน production environment docker ได้สร้าง tool สำหรับอำนวยความสะดวกในการใช้งานข้อมูลร่วมกันในหลายๆ กรณี อาทิเช่น
 - การ share ข้อมูลร่วมกันระหว่าง container
 - การ share ข้อมูลร่วมกันระหว่าง container / host
 - จัดเก็บ configuration / log / data ของทุกๆ container ไว้ที่ศูนย์กลาง
 - data migration
 - backup / restore

Volume

- รูปแบบการใช้งาน Volume บน Docker
 - Bind Mount (Map path with Host Directly)
 - Volumes Mount(Local, Special Driver)
 - Tmpfs Mount (Memory Only)
- พารามิเตอร์ในการใช้งาน (`--mount`), (`-v (Obsolete)`)



Volume

- 1. Bind Mount: map volume ระหว่าง host directory และ container directory

```
-v /volume-host:/volume-container:(ro/rw)
```

```
--mount type=bind,source=/volume-host  
,target=/volume-container, (readonly)
```

Ex: docker container run -dt -v /etc/nginx.conf:/etc/nginx.conf:ro

Ex: docker container run -dt --mount type=bind, \
source=/etc/nginx.conf,target=/etc/nginx.conf,readonly

- ตรวจสอบการคอนฟิกบัน container (inspect)

```
"HostConfig": {  
    "Binds": [  
        "/var/log:/var/log:ro"],
```

Volume

- 2. Volumes สร้าง volume กลางเพื่อใช้ในการจัดเก็บข้อมูล หรือ share volume ระหว่าง host/container/container (local):
/var/lib/docker/volumes,driver: 3rd Party
 - สร้าง Volume สำหรับใช้เก็บข้อมูล

```
docker volume create --driver <xxx> --opt <xxx> <name>
```

Ex: docker volume create --driver local datavol

Ex: docker volume create --driver local \
--opt type=nfs --opt o=addr=192.168.99.100,rw \
--opt device=/nfs_share datavol

Volume

- ການໃຊ້ Volumes (-v option is not allow for service level (Swarm/Compose))

Ex: docker volume create --driver local datavol

Ex: docker container run -dt --name web1 \
-v datavol:/data labdocker/alpine sh

Ex: docker container run -dt --name web2 \
-v datavol:/data labdocker/alpine sh

Ex: docker container run -dt --name web1 \
--mount source=datavol,target=/data labdocker/alpine sh

Ex: docker container run -dt --name web2 \
--mount source=datavol,target=/data labdocker/alpine sh

Volume

- การลบ volume ที่ออกจาก container ให้ใช้คำสั่ง docker volume rm เพื่อลบ volume

```
docker volume rm <volume name>
```

- Third Party Volume Plugin (Since 1.8)

```
docker volume create --driver=<third party> <volume name>
```

Ex: docker volume create --driver=gce --name gce-disk
-o SizeGb=90

Ex: docker run -dt -v gce-disk:/store/database alpine sh

Volume

Plugin	Description
Azure File Storage plugin	Lets you mount Microsoft Azure File Storage shares to Docker containers as volumes using the SMB 3.0 protocol. Learn more .
BeeGFS Volume Plugin	An open source volume plugin to create persistent volumes in a BeeGFS parallel file system.
Blockbridge plugin	A volume plugin that provides access to an extensible set of container-based persistent storage options. It supports single and multi-host Docker environments with features that include tenant isolation, automated provisioning, encryption, secure deletion, snapshots and QoS.
Contiv Volume Plugin	An open source volume plugin that provides multi-tenant, persistent, distributed storage with intent based consumption. It has support for Ceph and NFS.
Convoy plugin	A volume plugin for a variety of storage back-ends including device mapper and NFS. It's a simple standalone executable written in Go and provides the framework to support vendor-specific extensions such as snapshots, backups and restore.
DigitalOcean Block Storage plugin	Integrates DigitalOcean's block storage solution into the Docker ecosystem by automatically attaching a given block storage volume to a DigitalOcean droplet and making the contents of the volume available to Docker containers running on that droplet.
DRBD plugin	A volume plugin that provides highly available storage replicated by DRBD . Data written to the docker volume is replicated in a cluster of DRBD nodes.
Flocker plugin	A volume plugin that provides multi-host portable volumes for Docker, enabling you to run databases and other stateful containers and move them around across a cluster of machines.
Fuxi Volume Plugin	A volume plugin that is developed as part of the OpenStack Kuryr project and implements the Docker volume plugin API by utilizing Cinder, the OpenStack block storage service.
gce-docker plugin	A volume plugin able to attach, format and mount Google Compute persistent-disks .
GlusterFS plugin	A volume plugin that provides multi-host volumes management for Docker using GlusterFS.
Horcrux Volume Plugin	A volume plugin that allows on-demand, version controlled access to your data. Horcrux is an open-source plugin, written in Go, and supports SCP, Minio and Amazon S3.
HPE 3Par Volume Plugin	A volume plugin that supports HPE 3Par and StoreVirtual iSCSI storage arrays.
IPFS Volume Plugin	An open source volume plugin that allows using an ipfs filesystem as a volume.
Keywhiz plugin	A plugin that provides credentials and secret management using Keywhiz as a central repository.
Local Persist Plugin	A volume plugin that extends the default local driver's functionality by allowing you specify a mountpoint anywhere on the host, which enables the files to <i>always persist</i> , even if the volume is removed via docker volume rm.
NetApp Plugin(nDVP)	A volume plugin that provides direct integration with the Docker ecosystem for the NetApp storage portfolio. The nDVP package supports the provisioning and management of storage resources from the storage platform to Docker hosts, with a robust framework for adding additional platforms in the future.
Netshare plugin	A volume plugin that provides volume management for NFS 3/4, AWS EFS and CIFS file systems.
Nimble Storage Volume Plugin	A volume plug-in that integrates with Nimble Storage Unified Flash Fabric arrays. The plug-in abstracts array volume capabilities to the Docker administrator to allow self-provisioning of secure multi-tenant volumes and clones.
OpenStorage Plugin	A cluster-aware volume plugin that provides volume management for file and block storage solutions. It implements a vendor neutral specification for implementing extensions such as CoS, encryption, and snapshots. It has example drivers based on FUSE, NFS, NBD and EBS to name a few.
Portworx Volume Plugin	A volume plugin that turns any server into a scale-out converged compute/storage node, providing container granular storage and highly available volumes across any node, using a shared-nothing storage backend that works with any docker scheduler.
Quobyte Volume Plugin	A volume plugin that connects Docker to Quobyte 's data center file system, a general-purpose scalable and fault-tolerant storage platform.
REX-Ray plugin	A volume plugin which is written in Go and provides advanced storage functionality for many platforms including VirtualBox, EC2, Google Compute Engine, OpenStack, and EMC.
Virtuozzo Storage and Ploop plugin	A volume plugin with support for Virtuozzo Storage distributed cloud file system as well as ploop devices.
VMware vSphere Storage Plugin	Docker Volume Driver for vSphere enables customers to address persistent storage requirements for Docker containers in vSphere environment

Docker: The Next-Gen of Virtualization



Volume

- ການ backup volume ຈາກ container

Ex:

```
docker container run -it --rm --mount source=datavol,target=/data \
--mount type=bind,source=$(pwd) ,target=/backup \
tar cvf /backup/vol001.tar /data
```

- ການ restore volume ຈາກ container

Ex:

```
docker container run -it --rm --mount source=datavol,target=/data \
--mount type=bind,source=$(pwd) ,target=/backup \
bash -c "cd /data && tar xvf /backup/vol001.tar --strip 1"
```

Volume

- 3. tmpfs mount ใช้เพื่อจัดเก็บข้อมูลใน memory ไว้บน non-persistency space โดย tmpfs ไม่สามารถ Share ระหว่าง container และไม่สามารถใช้งานบน Windows ได้

```
docker volume create --driver <xxx> --opt <xxx> <name>
```

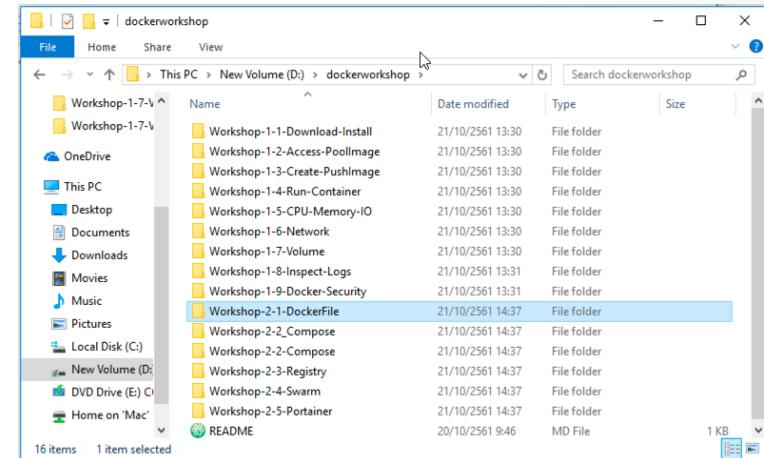
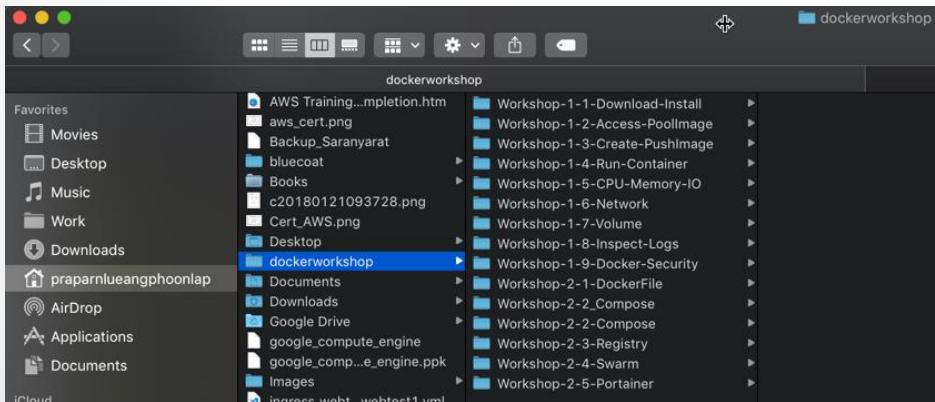
Ex: docker volume create --driver local --opt type=tmpfs \ --opt device=tmpfs --opt o=size=100m,uid=1000 tmptest

Ex: docker container run -dt --name tmptest –mount type=tmpfs,destination=/app nginx:latest

Ex: docker container run -dt --name tmptest --tmpfs /app \ nginx:latest

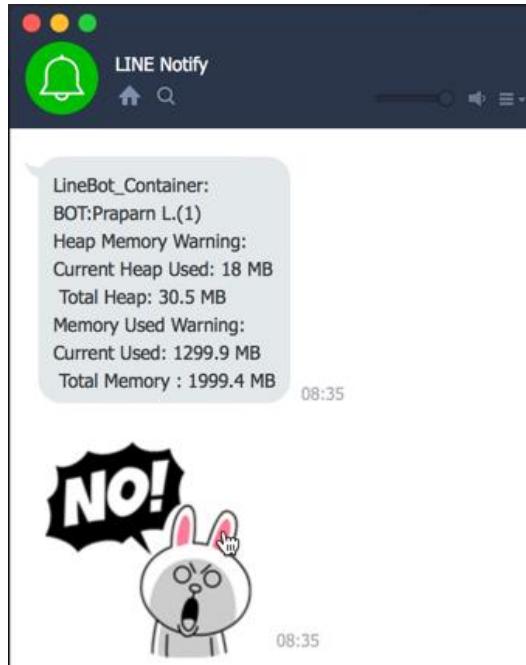
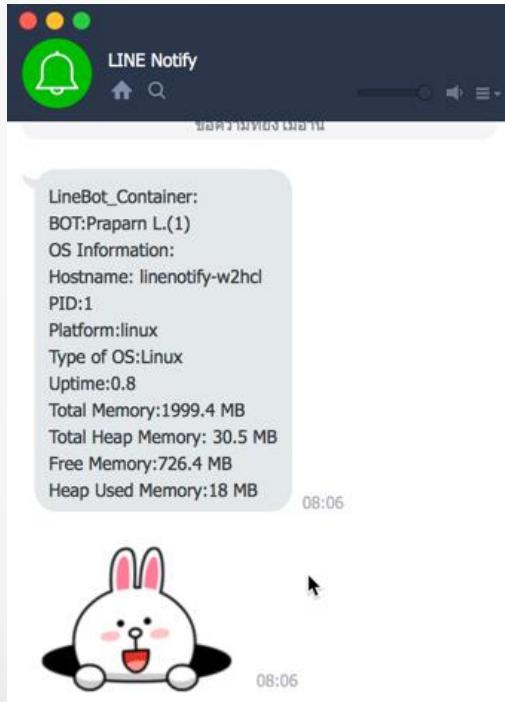
Workshop 1-7: Share Volume

- Before Workshop
- Check folder “dockerworkshop” and copy all workshop to here
 - MAC OS X: /Users/<Current User>/dockerworkshop or ~/dockerworkshop
 - Windows 10: d:\dockerworkshop



Workshop 1-7: Share Volume

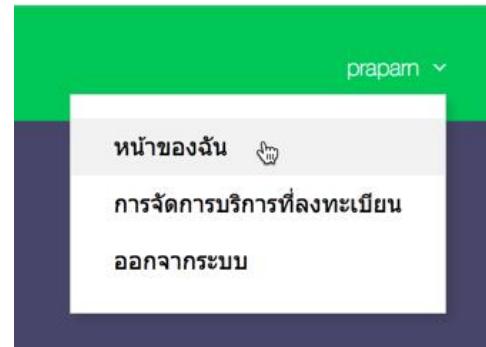
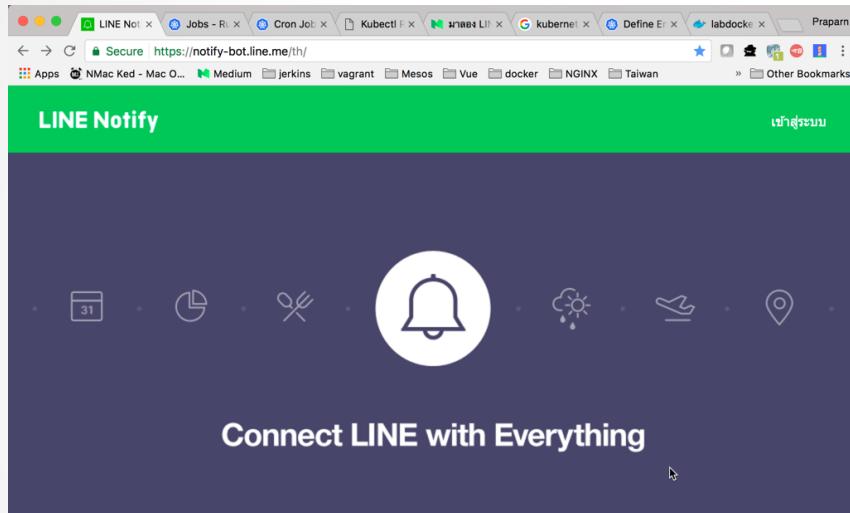
- Part 1: Host Path debug program
- Purpose: ทำการ Map volume เพื่อนำ source code เข้าไปทำการ debug/run บน container
- Example on WorkShop: "Monitor and LINE notify container"



LINE

Workshop 1-7: Share Volume

- Generate LINE Token
 - <https://notify-bot.line.me>



ออก Access Token (สำหรับผู้พัฒนา)

เมื่อใช้ Access Token แบบบุคคล จะสามารถตั้งค่าการแจ้งเตือนได้โดยไม่ต้องลงทะเบียนกับเว็บเซอร์วิส



Workshop 1-7: Share Volume

- Generate LINE Token
 - <https://notify-bot.line.me>

ออก Token

โปรดใส่ชื่อ Token (จะแสดงเมื่อมีการแจ้งเตือน)

LINEBOT

โปรดเลือกห้องแชทที่ต้องการส่งข้อความแจ้งเตือน

Search by group Name

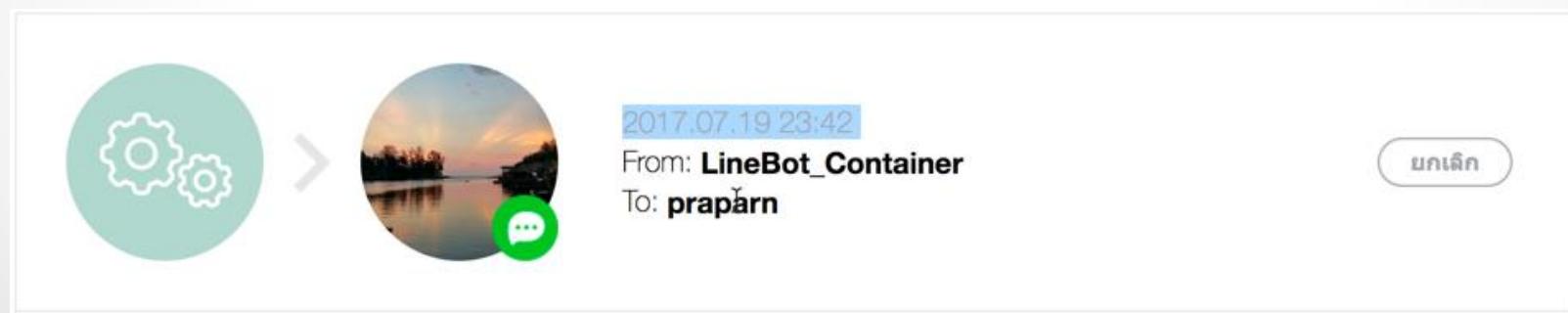
 รับการแจ้งเตือนแบบตัวต่อตัวจาก LINE Notify

Token ที่ออก

zHOlcJCcpIIS8mEedn [REDACTED]

ถ้าออกจากหน้านี้ ระบบจะไม่แสดง Token ที่ออกใหม่ล่าสุดไป โปรดคัดลอก Token ก่อนออกจากหน้านี้

คัดลอก **ปิด**



Workshop 1-7: Share Volume

```
1 const Monitor = require('monitor');
2 const request = require('request');
3 const TITLE=process.env.TITLE;
4 const INTERVAL=process.env.INTERVAL;
5 const HEAP_HIGH =process.env.HEAP_HIGH;
6 const MEM_HIGH =process.env.MEM_HIGH;
7 const SH_OS =process.env.SH_OS;
8 const TOKEN = process.env.TOKEN;
9 const critical_StickerPkg = 2;
10 const critical_StickerId = 39;
11 const information_StickerPkg = 2;
12 const information_StickerId = 34;
13 var options = {
14   probeClass: 'Process',
15   initParams: {
16     pollInterval: INTERVAL
17   }
18 }
19 var processMonitor = new Monitor(options);
20 var stickerPkg=0; //stickerPackageId
21 var stickerId=0; //stickerId
22 processMonitor.on('change', () => {
23   var sendNotify="N" //final decision to send notify
24   var heapWarning="N" //flag for heap memory warning
25   var memoryWarning="N" //flag for memory warning
26   if(HEAP_HIGH<((heapUsed/heapTotal)*100)){heapWarning="Y";}
27   if(MEM_HIGH<((usedMem/totalMem)*100)){memoryWarning="Y";}
28 }
```

```
19 var processMonitor = new Monitor(options);
20 var stickerPkg=0; //stickerPackageId
21 var stickerId=0; //stickerId
22 processMonitor.on('change', () => {
23   var sendNotify="N" //final decision to send notify
24   var heapWarning="N" //flag for heap memory warning
25   var memoryWarning="N" //flag for memory warning
26   //collect OS information
27   var hostName = processMonitor.get('hostname');
28   var pID = processMonitor.get('pid');
29   var platForm = processMonitor.get('platform');
30   var upTime = processMonitor.get('uptime');
31   var type = processMonitor.get('type');
32   var totalMem = processMonitor.get('totalmem');
33   totalMem=Number(((totalMem/1024)/1024).toFixed(1)); //Convert to MB
34   var heapTotal = processMonitor.get('heapTotal');
35   heapTotal=Number(((heapTotal/1024)/1024).toFixed(1)); //Convert to MB
36   var titleMSG="\n"+TITLE+"("+pID+)"
37   var osMSG="";
38   var warnMSG="";
39   //collect OS information
40   //collect memory information
41   var heapUsed = processMonitor.get('heapUsed');
42   heapUsed=Number(((heapUsed/1024)/1024).toFixed(1)); //Convert to MB
43   if(HEAP_HIGH<((heapUsed/heapTotal)*100)){heapWarning="Y";}
44   var freeMem = processMonitor.get('freemem');
45   freeMem=Number(((freeMem/1024)/1024).toFixed(1));
46   var usedMem = totalMem-freeMem;
47   usedMem=Number(usedMem.toFixed(1)); //Convert to MB
48   if(MEM_HIGH<((usedMem/totalMem)*100)){memoryWarning="Y";}
49   //collect memory information
```

Workshop 1-7: Share Volume

The screenshot shows a Mac desktop with three Terminal windows and a LINE Notify application.

Terminal Window Content:

```
mit":50,"x-ratelimit-remaining":997,"x-ratelimit-imageremaining":50,"x-ratelimit-reset":1501996936}, "request": {"uri": {"protocol": "https:", "slashes": true, "auth": null, "host": "notify-api.line.me", "port": 443, "hostname": "notify-api.line.me", "hash": null, "search": null, "query": null, "pathname": "/api/notify", "path": "/api/notify", "href": "https://notify-api.line.me/api/notify"}, "method": "POST", "headers": {"Content-Type": "application/x-www-form-urlencoded", "authorization": "Bearer EIEqyillzkzpaCo1ROrebZTcGbIyzDZH0jcd0t6CX", "content-length": 343}}, "status": 200, "message": "ok"}  
^C  
/nodejs # export HEAP_HIGH=20  
/nodejs # export MEM_HIGH=20  
/nodejs # export SH_OS=N  
/nodejs # node index.js  
null  
{ "statusCode": 200, "body": { "status": 200, "message": "ok", "headers": { "server": "nginx", "date": "Sun, 06 Aug 2017 04:30:41 GMT", "content-type": "application/json; charset=UTF-8", "transfer-encoding": "chunked", "connection": "keep-alive", "keep-alive": "timeout=3", "x-ratelimit-limit": "1000", "x-ratelimit-imagerelimit": "50", "x-ratelimit-remaining": "996", "x-ratelimit-imageremaining": "50", "x-ratelimit-reset": "1501996936"}, "request": {"uri": {"protocol": "https:", "slashes": true, "auth": null, "host": "notify-api.line.me", "port": 443, "hostname": "notify-api.line.me", "hash": null, "search": null, "query": null, "pathname": "/api/notify", "path": "/api/notify", "href": "https://notify-api.line.me/api/notify"}, "method": "POST", "headers": {"Content-Type": "application/x-www-form-urlencoded", "authorization": "Bearer EIEqyillzkzpaCo1ROrebZTcGbIyzDZH0jcd0t6CX", "content-length": 290}}}, "status": 200, "message": "ok"}  
^C  
/nodejs #
```

LINE Notify Application:

LineBot_Container:
NODEJS_BOT_NOTIFY(67 Edit)
LINENOTIFY)
Heap Memory Warning:
Current Heap Used: 18.2 MB
Total Heap: 30.5 MB
Memory Used Warning:
Current Used: 477.6 MB
Total Memory : 995.8 MB

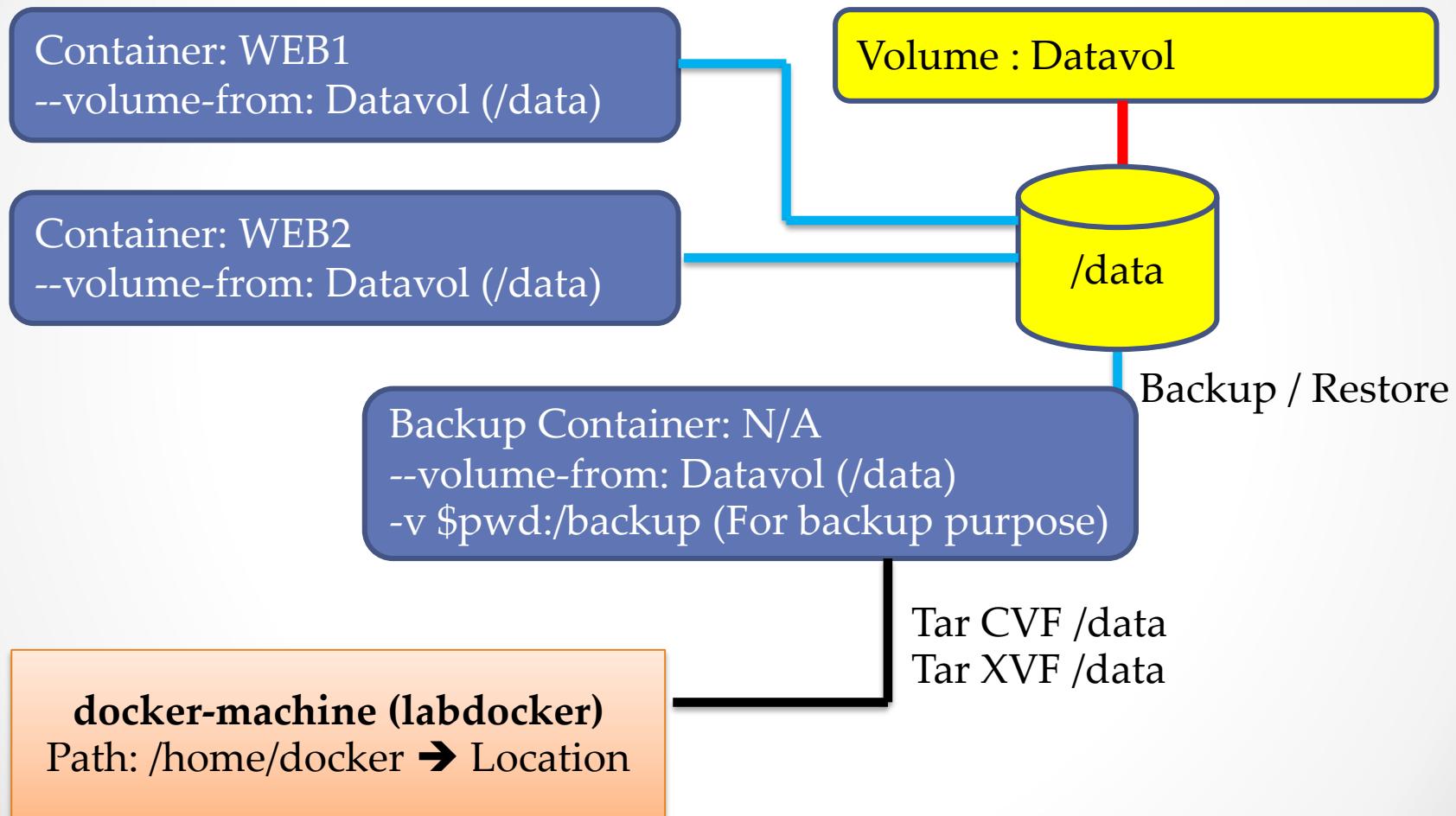
11:30

NO!

11:30

Workshop 1-7: Share Volume

- Part 2: Container Volume



Log and Inspect

• • •

Log

- เพื่อตรวจสอบ log การทำงานบน container อย่างต่อเนื่อง docker ได้เตรียม tool ในการตรวจสอบ log การทำงาน

```
docker container logs <options> <container name>
```

- options ในการดูล็อกไฟล์
 - f, --follow เพื่อตรวจสอบล็อกไฟล์ใหม่ตลอดเวลา
 - t, --timestamps กำหนดให้สั่นและเวลาในล็อกไฟล์
 - since= <unix timestamps> กำหนดเวลาเริ่มตรวจสอบล็อกไฟล์ หรือกำหนด yyyy-mm-dd
 - tail = "all" / number of line
 - until แสดง log ก่อนเวลาที่กำหนด (Ex: 20180619220001) หรือ Relative (Ex: 42m = 42 minute before)
 - details แสดงรายละเอียดเพิ่มเติมของล็อก เช่น Env variable, labels โดยจะเป็นรายละเอียดเพิ่มเติมจากการใส่ option --log-opt ตอนสร้าง container

Inspect

- การตรวจสอบค่าคอนฟิกของ container / network / volume etc

```
docker inspect <container>
```

```
docker network inspect
```

```
docker volume inspect
```

Workshop 1-8: Log & Inspect

- ทดสอบสร้าง container nginx และ map port 80 ดังนี้

```
docker container run -dt --name nginx \
-p 80:8080 labdocker/nginx: badversion
```

- ตรวจสอบค่าคอนฟิกภายใน container ด้วยคำสั่ง inspect

```
docker container inspect nginx
```

- ตรวจสอบล็อกการทำงานภายใน container ด้วยคำสั่ง logs

```
docker container logs nginx
```

Commit

• • •

Commit

- เมื่อต้องการเก็บ container ที่ทำการรันเรียบร้อยเป็น snap image เป็น generation สามารถทำได้ผ่านคำสั่ง “commit”

```
docker container commit <options> <container> <image name:tag>
```

- options
 - a, --author=“ ” ระบุผู้จัดทำ image
 - c, --change = [] เพิ่มเติม command พิเศษที่จะจัดเก็บภายใน image
 - m, --message = “ ” ระบุ comment ใน image ที่จัดเก็บ
 - p, --pause=true กำหนดให้หยุดการใช้งาน container ชั่วคราวในระหว่างทำการ commit

Docker Security

• • •

Docker Security

- เพื่อให้การรันงานบน docker container มีความปลอดภัยสูง docker มีการพิจารณาเรื่อง security ในการทำงานแบ่งออกได้ดังนี้
- Control Group (CG Group) / Name Space
 - Docker container run in separate namespace (Process independence, Not release / touch with other container)
 - Separate network stack
 - CG group will create for separate resource (CPU, Memory, I/O) and limit for protect single container run process and make host fail (denied-of-service)
- Docker daemon operation (root privilege)
 - Beaware for allow user who operate with docker daemon
 - Some operation quite risk for security (docker pull, docker run –v with share host path etc)
 - Docker API should be aware for RESTFUL connection with secure method

Docker Security

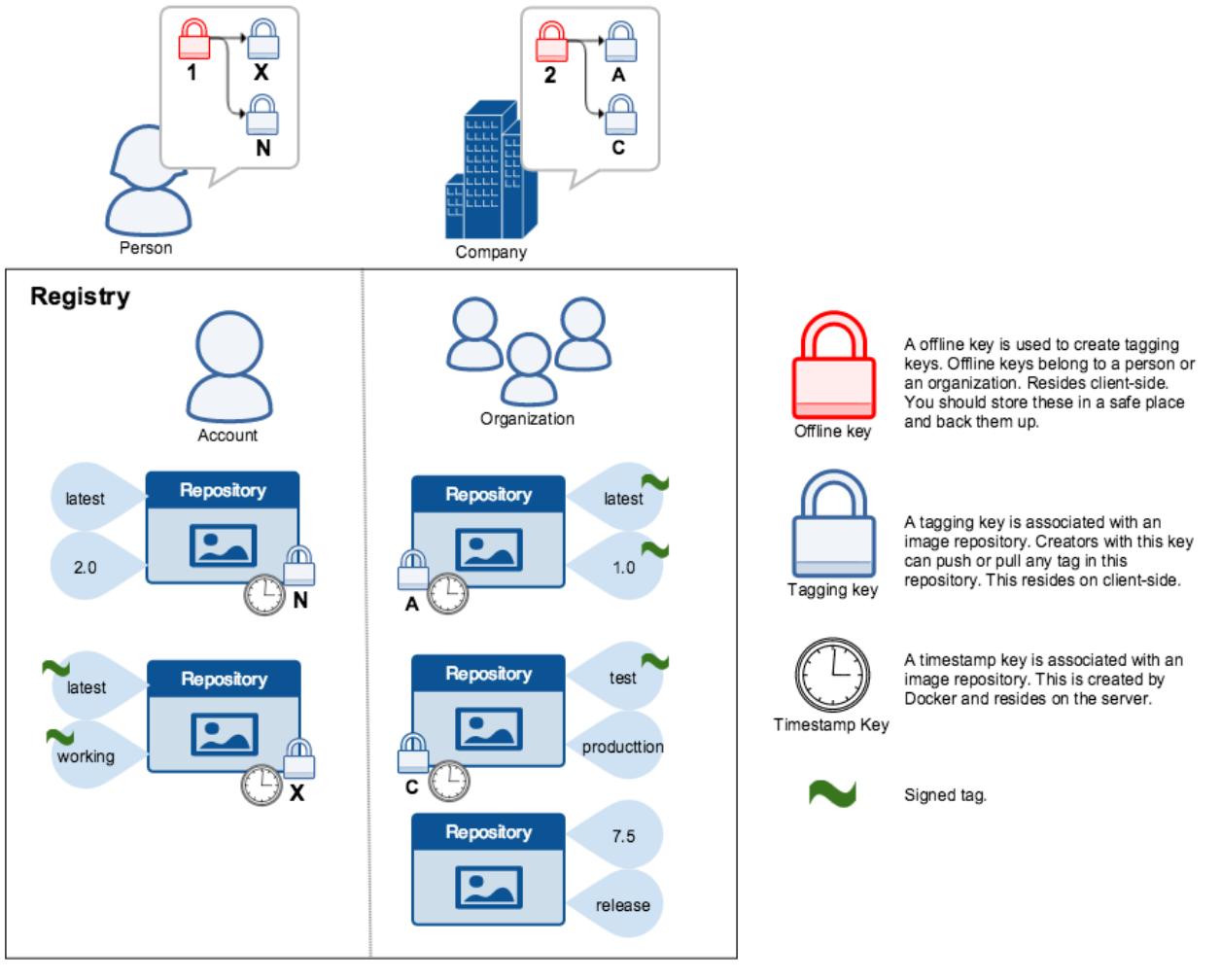
- Secure enhancement by limit capability of container inside
 - Normally user will be “root” inside container
 - Almost container job (web server, restful server, database server) don’t need high privilege as “root” for operate
 - Some activity should be prohibit on container
 - Mount disk operation
 - Access system path (/bin, /usr/bin, /proc etc)
 - Create network socket
 - Execute shell
 - etc

Docker Security

- Recommendation for enhance docker security
 - ระวังการจัดการ User ที่มีหน้าที่ทำงานกับ docker daemon และ privilege คำสั่งบน docker command
 - ใช้ non-root user ในการจัดการ docker daemon operation (docker run etc)
 - จัดการ Image Content Trust (Sign Image) ในการสั่ง pull/push image
 - เปลี่ยนการจัดการ docker daemon socket เป็น HTTP socket (with TLS verify and authentication)
 - จำกัดสิทธิและขอบเขตการทำงานของ User บน container (Secure compute mode (seccomp))

Docker Security

- Image Content Trust



Workshop 1-9: Docker Security

- Content Trust
- Purpose: ทำการ enable feature "DOCKER_CONTENT_TRUST" และใช้งาน image แบบ sign/unsigned

```
[root@labdocker:~# docker push paparn/alpine:sign
The push refers to a repository [docker.io/paparn/alpine]
6102f0d2ad33: Layer already exists
sign: digest: sha256:65242e8220a341cec40628caa77eb4acd2fc252329aa853526fde15a4a1d85 size: 528
Signing and pushing trust metadata
You are about to create a new root signing key passphrase. This passphrase
will be used to protect the most sensitive key in your signing system. Please
choose a long, complex passphrase and be careful to keep the password and the
key file itself secure and backed up. It is highly recommended that you use a
password manager to generate the passphrase and keep it safe. There will be no
way to recover this key. You can find the key in your config directory.
Enter passphrase for new root key with ID ca6ab4b:
Repeat passphrase for new root key with ID ca6ab4b:
Enter passphrase for new repository key with ID 68d88cd (docker.io/paparn/alpine):
Repeat passphrase for new repository key with ID 68d88cd (docker.io/paparn/alpine):
```

Kitematic

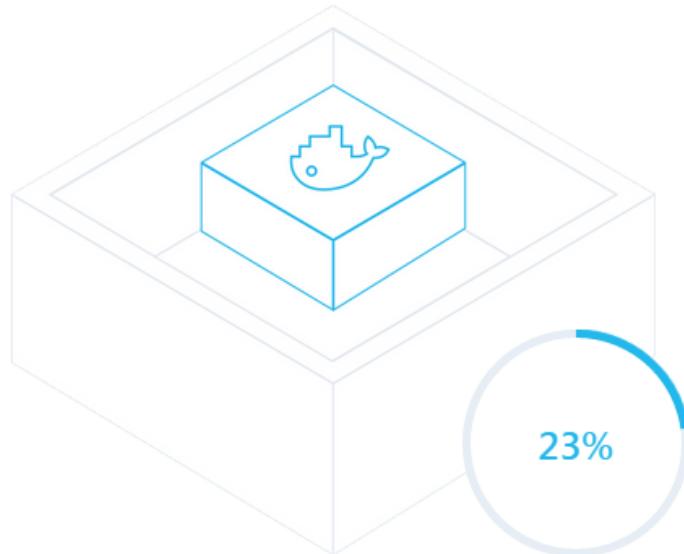
• • •

Docker: The Next-Gen of Virtualization



Kitematic

- GUI tool for operate docker

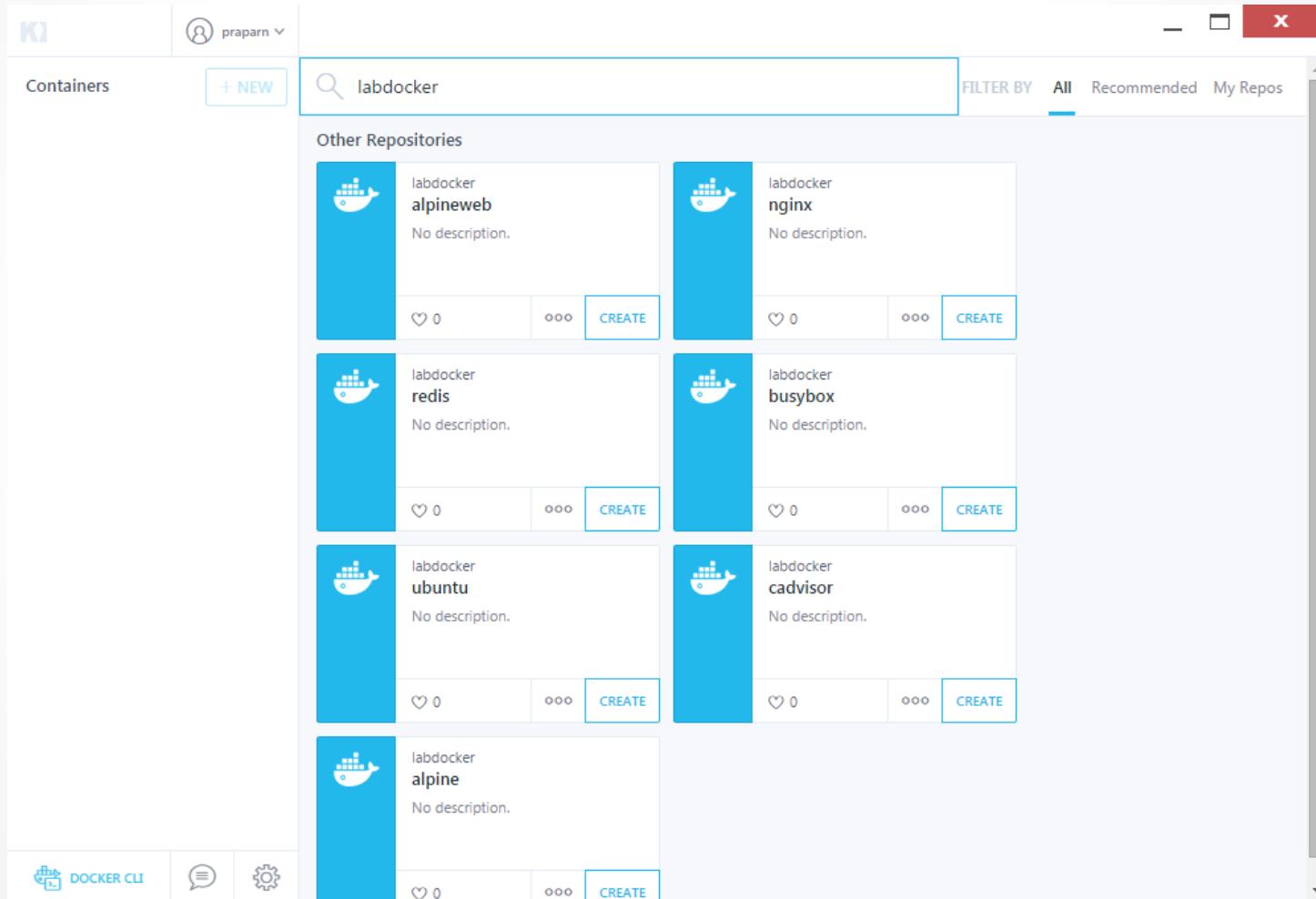


Starting Docker VM

To run Docker containers on your computer, Kitematic is starting a Linux virtual machine. This may take a minute...

Kitematic

- GUI tool for operate docker



Recapture for Day 1

- Docker principle
- Docker machine
- Image, Repository & Tag, container
- CPU, Memory and I/O
- Network
- Volume
- Inspect and Log
- Commit
- Docker Security
- Kitematic

Q&A for Day 1

• • •



Docker:Zero to Hero (Day 2)

By Praparn Luengphoonlap
Email: eva10409@gmail.com

Docker: The Next-Gen of Virtualization



Outline Day 2

- Dockerfile and Build
- Compose
- Registry
- Swarm Mode
 - Conceptual of Swarm
 - Swarm Mode Architecture
 - Swarm init/join cluster system
 - Swarm service
 - Orchestrator Assignment
 - Config and Secret
 - Network Overlay and Ingress
 - HA Manager Role
 - Docker Stack Deploy (Compose Swarm Mode)
- Portainer for Docker
- Q&A

Dockerfile and Build

• • •

Dockerfile

- docker file คือการคำสั่งที่ใช้สร้าง image ขึ้นเพื่อใช้งานโดยเนพาะ และสามารถกำหนดเป็นมาตรฐานสำหรับการทำงานในองค์กรโดยสร้าง text file ตาม syntax ที่ทาง docker กำหนดไว้
 - FROM
 - MAINTAINER
 - RUN (shell), [“exec”, “parameter1”, “parameter2”, “etc”]
 - CMD (shell), [“exec”, “parameter1”, “parameter2”, “etc”]
 - EXPOSE
 - ARG
 - ENV
 - COPY /ADD (source) (Destination), [(source) (destination)]
 - ENTRYPOINT
 - WORKDIR
 - Etc (<https://docs.docker.com/engine/reference/builder>)
- เมื่อสร้าง dockerfile เสร็จแล้วสามารถสั่ง image ผ่านคำสั่ง build

```
docker build <option> <path of dockerfile>
```

Dockerfile

- **Best practice** ในการสร้าง **dockerfile**
- simple is the best
- 1 container / 1 service
- เลือก source image จาก official owner
- สร้าง dockerfile โดยพยายามให้มี footprint ต่ำที่สุด
 - สร้าง dockerfile บน path ที่สร้างขึ้นเฉพาะงาน
 - วางเฉพาะไฟล์ที่จำเป็นต้องใช้งานบน path
 - install component เพื่อที่จำเป็นต้องใช้งานบน container
- คำสั่ง run เพื่อติดตั้ง software รวมยอดใน 1 ชุดคำสั่ง
- ควรมีการลบ temp ที่เกิดขึ้นจากการติดตั้ง software เพื่อลดขนาดของ image
- จัดเรียง dockerfile ให้อ่านง่าย
 - apt-update && apt-install -y \
 - curl \
 - automake \
- ควรระบุ EXPOSE port ที่ออกแบบให้ใช้งานไว้ในทุกๆครั้ง

Dockerfile

```
RUN apt-get update && apt-get install -y \
    aufs-tools \
    automake \
    build-essential \
    curl \
    dpkg-sig \
    libcap-dev \
    libsqlite3-dev \
    mercurial \
    reprepro \
    ruby1.9.1 \
    ruby1.9.1-dev \
    s3cmd=1.1.* \
&& rm -rf /var/lib/apt/lists/*
```

Dockerfile

```
FROM php:5.6-apache
MAINTAINER Volker Wiegand <volker.wiegand@cvw.de>

RUN a2enmod rewrite

RUN apt-get update && apt-get install -y \
    libpng12-dev \
    libjpeg-dev \
    libpq-dev \
    libxml2-dev \
    vim-tiny \
    && docker-php-ext-configure gd --with-png-dir=/usr --with-jpeg \
    && docker-php-ext-install gd mbstring mysql mysqli pdo_mysql \
    && rm -rf /var/lib/apt/lists/* /var/www/html/index.html

ENV MANTIS_VER 1.3.2
ENV MANTIS_MD5 f30acb6d41ba757b7c09f922a0f68b06
ENV MANTIS_URL https://sourceforge.net/projects/mantisbt/files/mantis-
ENV MANTIS_FILE mantisbt.tar.gz

RUN mkdir -p /var/lib/mantisbt && cd /var/lib/mantisbt \
    && curl -fSL ${MANTIS_URL} -o ${MANTIS_FILE} \
    && echo "${MANTIS_MD5} ${MANTIS_FILE}" | md5sum -c \
    && tar -xz --strip-components=1 -f ${MANTIS_FILE} \
    && rm ${MANTIS_FILE} \
    && chown -R www-data:www-data .
```

Dockerfile

- ควรระบุ WORKDIR เพื่อกำหนด path เริ่มต้นในการทำงานทุกครั้ง (before ENTRYPOINT, RUN, CMD, COPY etc)
- ADD/COPY operation
 - COPY เป็น basic transfer simple file
 - ADD ใช้ในกรณี remote file URL, self extract (.tar) (Not recommend)
- ENV PATH ควรระบุ path ของ executable ที่จำเป็นต้องใช้งาน
- พิจารณาการใช้ cache ของ image layer โดยสามารถ ignore cache ด้วย option “--no-cache=true”
 - ตามปกติ docker จะปรับเทียบ instruction ที่กำหนดไว้กับ image layer ที่
 - Consider: instruction, base image

How can check dockerfile ?

Secure <https://www.fromlatest.io/#/>

FROM:latest

```
1 FROM labdocker/alpine:latest
2 MAINTAINER Praparn Lueangphoonglap (eva10409@gmail.com)
3 LABEL Description="NodeJS/NGINX Build Container" Version="1.0"
4 ENV NODE_VERSION=v4.3.0 NPM_VERSION=2.14.12
5 RUN apk update && \
6     apk add nginx
7 COPY nginx.conf /etc/nginx/nginx.conf
8 WORKDIR /usr/sbin
9 ENTRYPOINT ["nginx", "-c", "/etc/nginx/nginx.conf"]
10 EXPOSE 8080
```

Line 5: Consider `--no-cache or --update with rm -rf /var/cache/apk/*` (Optimization)

Line 1: Base Image Latest Tag (Clarity)

2 issues found Made by REPLICATED

Instruction_Kubernetes.txt ... | Instruction_Kubernetes.txt ... | Show All X

Workshop 2-1: DockerFile

- Part1: NGINX/NODEJS

Container Name: NODEJS

IP Address: X.X.X.X (Port: 3000:3000)

Container Name: NGINX

IP Address: X.X.X.X (Port: 8080:80)

AWS's Machine

Path: ~/dockerworkshop/Workshop-2-1-DockerFile

/nodejs

dockerfile



/nginx

dockerfile



Workshop 2-1: DockerFile

- ตรวจสอบ dockerfile ตามตัวอย่างด้านล่าง

```
FROM labdocker/alpine:latest
MAINTAINER Praparn Lueangphoonlap (eva10409@gmail.com)
LABEL Description="NodeJS/NGINX Build Container" Version="1.0"
ENV NODE_VERSION=v4.3.0 NPM_VERSION=2.14.12
RUN apk update && \
    apk add nodejs
RUN mkdir /nodejs
COPY hello.js /nodejs/
WORKDIR /nodejs
ENTRYPOINT ["node","hello.js"]
EXPOSE 3000
```

```
docker image build -t labdocker/node:1.0 /Share_DockerToolbox/nodejs
```

Workshop 2.1: DockerFile

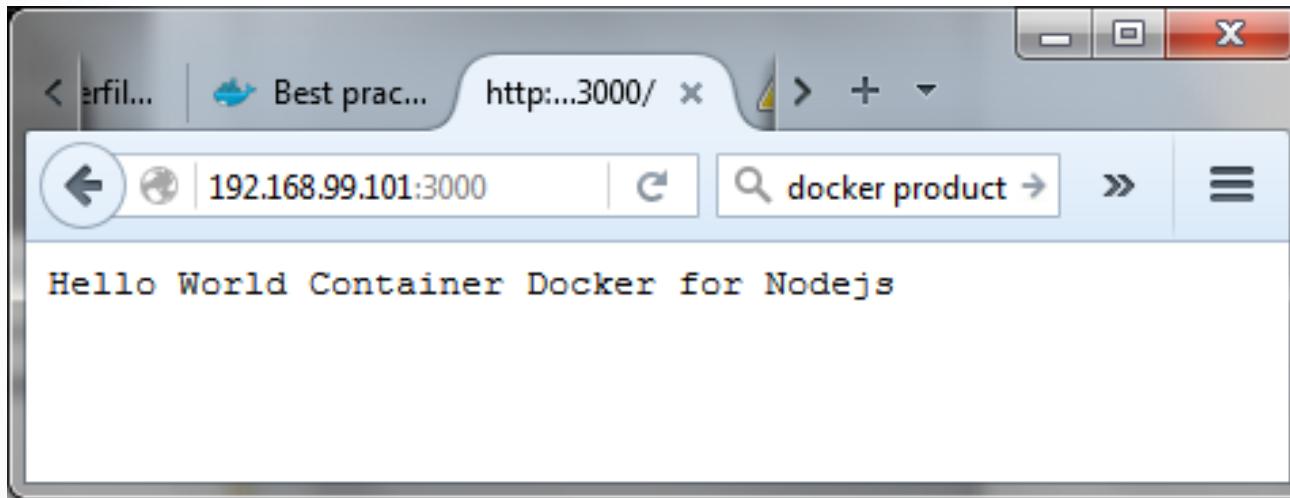
- Output

```
Sending build context to Docker daemon 3.072 kB
Step 1 : FROM labdocker/alpine:latest
--> 14f89d0e6257
Step 2 : MAINTAINER Praparn Lueangphoonlap (eval0409@gmail.com)
--> Using cache
--> 10f80fb4997d
Step 3 : LABEL Description "NodeJS/NGINX Build Container" Version "1.0"
--> Using cache
--> 59bc5a927e19
Step 4 : ENV NODE_VERSION v4.3.0 NPM_VERSION 2.14.12
--> Using cache
--> ce23d23959b7
Step 5 : RUN apk update &&     apk add nodejs
--> Running in b12c752ef6a0
fetch http://dl-4.alpinelinux.org/alpine/v3.3/main/x86_64/APKINDEX.tar.gz
fetch http://dl-4.alpinelinux.org/alpine/v3.3/community/x86_64/APKINDEX.tar.gz
v3.3.1-99-gc7d905f [http://dl-4.alpinelinux.org/alpine/v3.3/main]
v3.3.1-59-g48b0368 [http://dl-4.alpinelinux.org/alpine/v3.3/community]
OK: 5857 distinct packages available
(1/4) Installing libgcc (5.3.0-r0)
(2/4) Installing libstdc++ (5.3.0-r0)
(3/4) Installing libuv (1.7.5-r0)
(4/4) Installing nodejs (4.3.0-r0)
Executing busybox-1.24.1-r7.trigger
OK: 29 MiB in 15 packages
--> 8f2bf208ab86
Removing intermediate container b12c752ef6a0
Step 6 : RUN mkdir /nodejs
--> Running in 2ee51d1af642
--> d3edf2c2f511
Removing intermediate container 2ee51d1af642
Step 7 : COPY hello.js /nodejs/
--> f703ccbc7dd4
Removing intermediate container 952935d69cce
Step 8 : CMD nginx -c /etc/nginx/nginx.conf
--> Running in 67940385f5ac
--> edfba609e20a
```

Workshop 2-1: DockerFile

- Test run

```
docker container run -dt --name NODEJS \
-p 3000:3000 labdocker/node:1.0
```



Workshop 2-1: DockerFile

- ตรวจสอบ dockerfile ตามตัวอย่างด้านล่าง

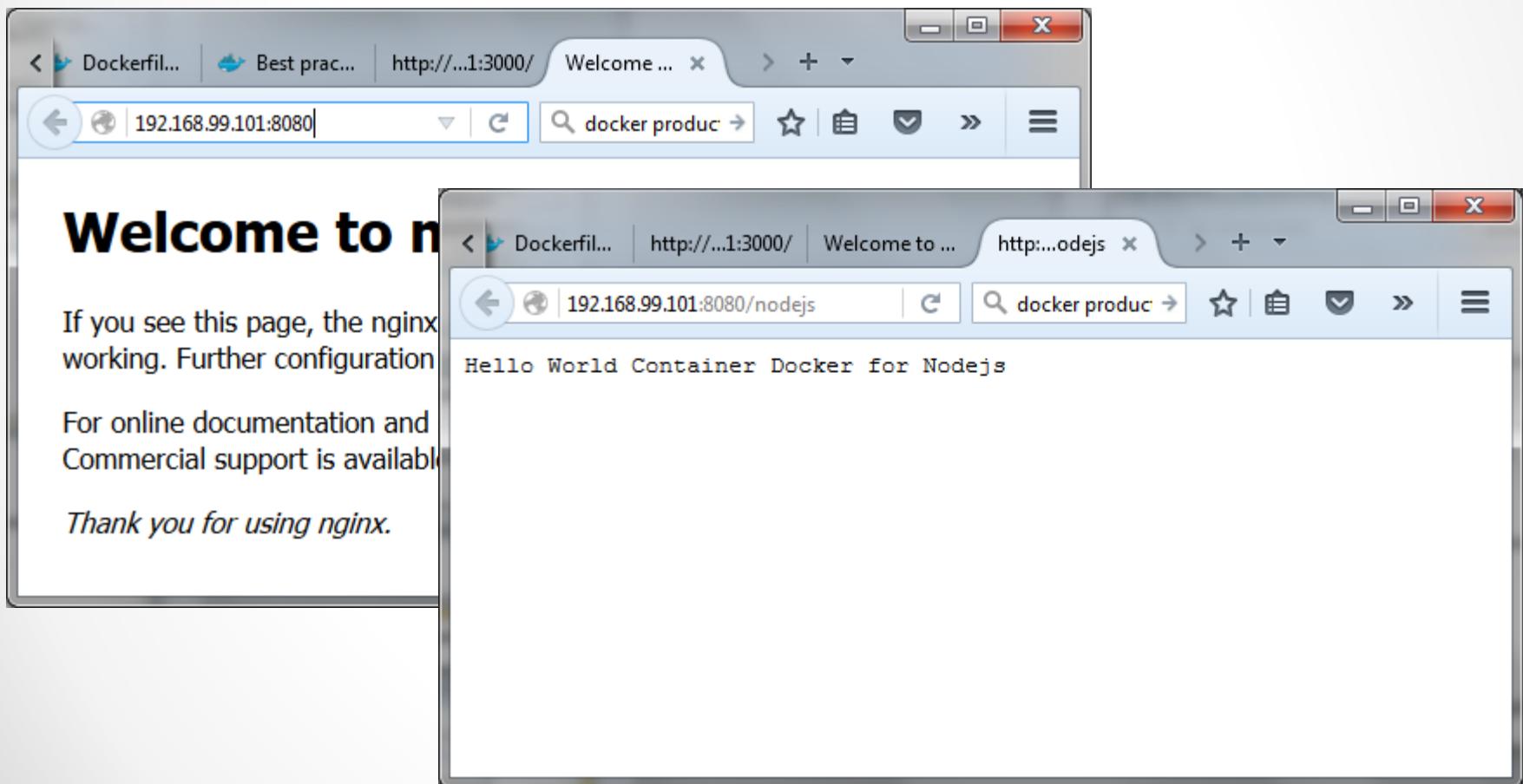
```
FROM labdocker/alpine:latest
MAINTAINER Praparn Lueangphoonlap (eva10409@gmail.com)
LABEL Description="NodeJS/NGINX Build Container" Version="1.0"
RUN apk update && \
    apk add nginx
COPY nginx.conf /etc/nginx/nginx.conf
WORKDIR /usr/sbin
ENTRYPOINT ["nginx","-c","/etc/nginx/nginx.conf"]
EXPOSE 8080
```

```
docker build -t labdocker/web:1.0 /Share_DockerToolbox/nginx
```

Workshop 2-1: DockerFile

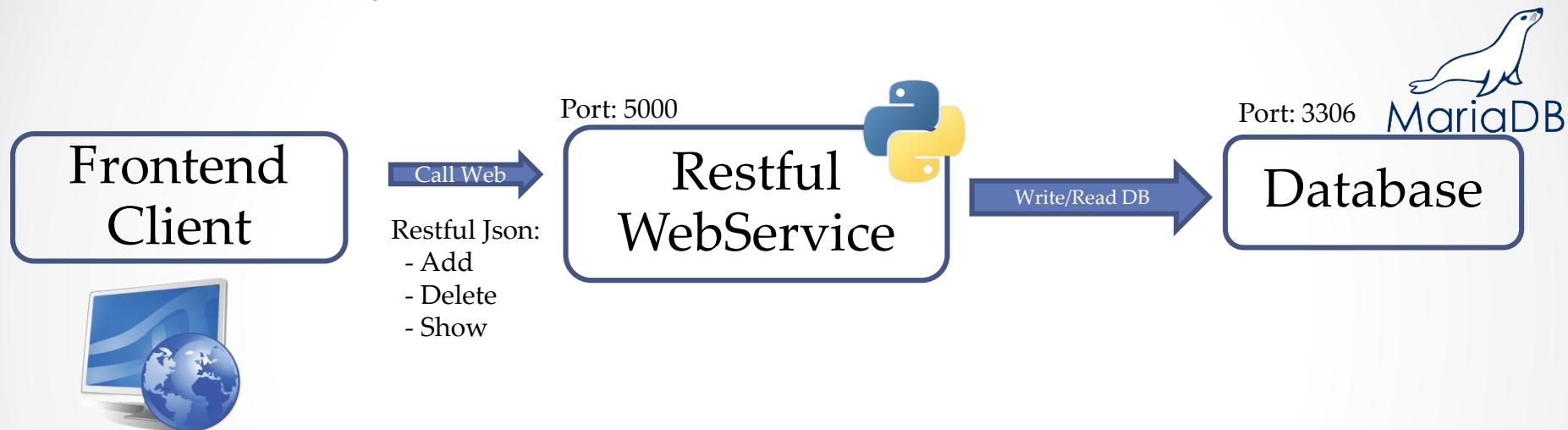
- Test run

```
docker container run -dt --name NGINX -p 8080:8080 labdocker/web:1.0
```



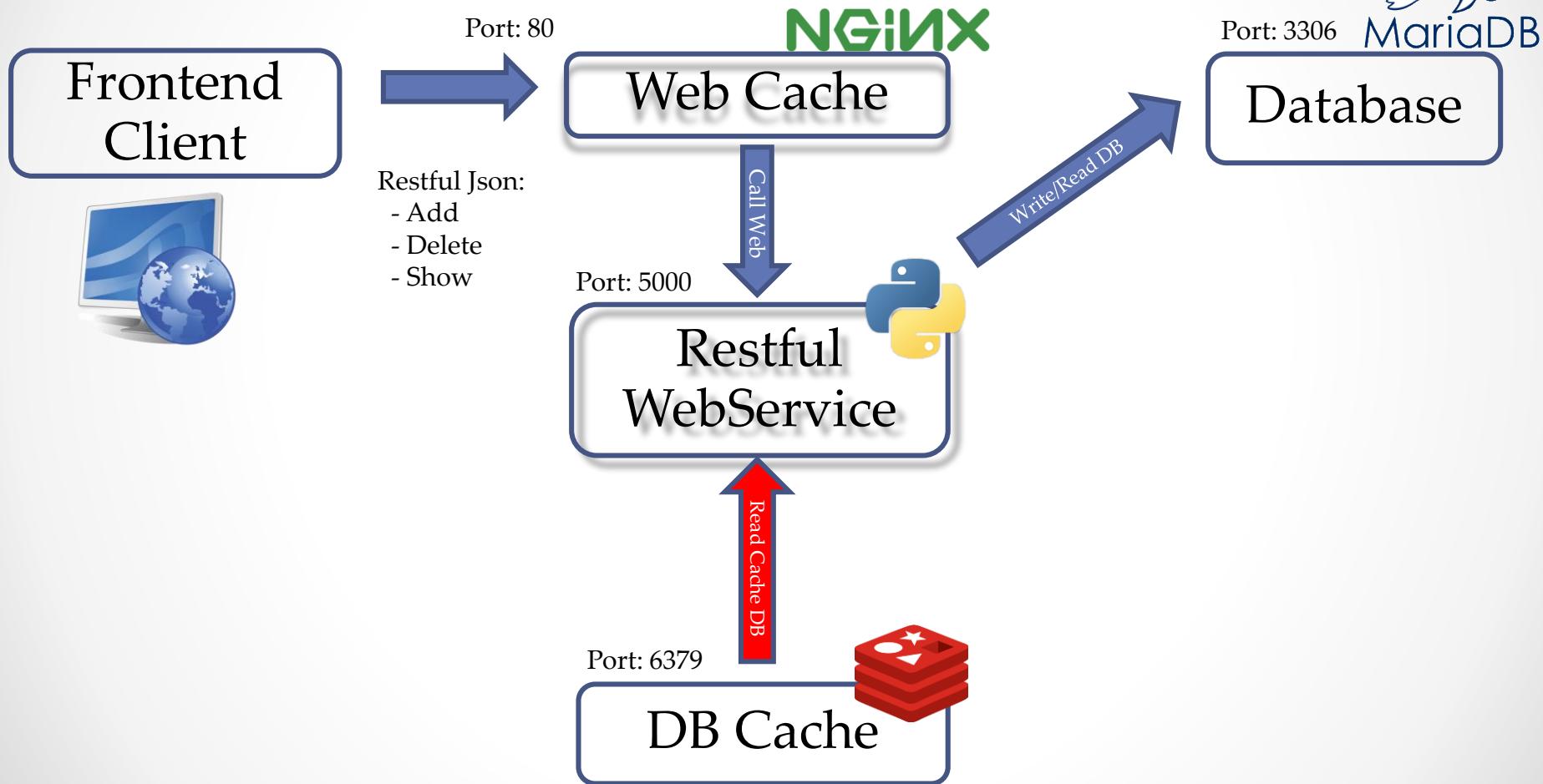
Workshop 2-1: DockerFile

- Part2: Python RESTFUL
- Basic Component



Workshop 2-1: DockerFile

- Optimize for WorkLoad



Workshop 2-1: DockerFile

- Restful WebService
 - /init

```
@app.route('/init')
def init():
    MAIN_DB.execute("DROP DATABASE IF EXISTS ACCTABLE")
    MAIN_DB.execute("CREATE DATABASE ACCTABLE")
    MAIN_DB.execute("USE ACCTABLE")
    sql = """CREATE TABLE users (
        ID int,
        USER char(30),
        DESCRIBE char(250)
    )"""
    MAIN_DB.execute(sql)
    db.commit()
    return "##### Database Create New Account Table Done #####"
```

- /insertuser

```
@app.route("/users/insertuser", methods=['POST'])
def add_users():
    req_json = request.get_json()
    MAIN_DB.execute("INSERT INTO ACCTABLE.users (ID, USER, DESCRIBE) VALUES (%s,%s,%s)", (req_json['uid'], req_json['user'], req_json['descripe']))
    #curl -i -H "Content-Type: application/json" -X POST -d '{"uid": "1", "user":"Praparn Luangphoonlap", "descripe":"System Engineer"}' http://<IP>
    db.commit()
    return Response("##### Record was added #####", status=200, mimetype='application/json')
```

Workshop 2-1: DockerFile

- Restful WebService
 - /removeuser/<uid>

```
@app.route('/users/removeuser/<uid>')
def remove_users(uid):
    hash = hashlib.sha224(str(uid)).hexdigest()
    key = "sql_cache:" + hash
    MAIN_DB.execute("DELETE FROM ACCTABLE.users WHERE ID =" + str(uid))
    db.commit()
    #curl http://<IP Host>;<Port>/users/removeuser/<uid>
    if (CACHE_DB.get(key)):
        CACHE_DB.delete(key)
        return Response("##### Record was deleted (Both Database Cache) #####", status=200, mimetype='application/json')
    else:
        return Response("##### Record was deleted #####", status=200, mimetype='application/json')
```

Workshop 2-1: DockerFile

- Restful WebService
 - /users/<uid>

```
@app.route('/users/<uid>')
def get_users(uid):
    hash = hashlib.sha224(str(uid)).hexdigest()
    key = "sql_cache:" + hash
    #curl http://<IP Host>:<Port>/users/<uid>
    if (CACHE_DB.get(key)):
        return CACHE_DB.get(key) + "(Database Cache)"
    else:
        MAIN_DB.execute("select USER from ACCTABLE.users where ID=" + str(uid))
        data = MAIN_DB.fetchone()
        if data:
            CACHE_DB.set(key,data[0])
            CACHE_DB.expire(key, 36);
            return CACHE_DB.get(key)
        else:
            return "##### Record not found #####"
```

Workshop 2-1: DockerFile

- Web Cache (NGINX)

```
http {
    client_max_body_size 500M;
    client_body_timeout 3000s;
    include /etc/nginx/mime.types;
    default_type application/octet-stream;
    log_format main '$remote_addr - $remote_user [$time_local] "$request" '
    '$status $body_bytes_sent "'.$http_referer"
    '"$http_user_agent" "$http_x_forwarded_for"';

    access_log /var/log/nginx/access.log main;
    tcp_nopush on;

    keepalive_timeout 65;
    gzip on;

    include /etc/nginx/conf.d/*.conf;
server {
    listen 80;
    client_body_buffer_size 50M;
    index index.html      index.htm;
    location / {
        proxy_pass http://webservice:5000;
        proxy_next_upstream error timeout invalid_header http_500 http_502 http_503 http_504;
        proxy_redirect off;
        proxy_buffering off;
        proxy_set_header      Host          $host;
        proxy_set_header      X-Real-IP     $remote_addr;
        proxy_set_header      X-Forwarded-For $proxy_add_x_forwarded_for;
    }
}
```

Workshop 2-1: DockerFile

- Database / Database Cache

```
maindb:  
  image: labdocker/mysql:latest  
  container_name: maindb  
  environment:  
    MYSQL_ROOT_PASSWORD: password  
  
cachedb:  
  image: labdocker/redis:latest  
  container_name: cachedb  
  
webservice:  
  build: .  
  dockerfile: dockerfile_python  
  container_name: webservice  
  ports:  
    - "5000:5000"  
  links:  
    - cachedb:cachedb  
    - maindb:maindb  
  
webcache:  
  build: .  
  dockerfile: dockerfile_nginx  
  container_name: webcache  
  ports:  
    - "80:80"  
  links:  
    - webservice:webservice
```



Workshop 2-1: DockerFile

- Dockerfile: WebService

```
FROM labdocker/alpinepython:2.7-onbuild
MAINTAINER Praparn Lueangphoonlap (eva10409@gmail.com)
LABEL Description="Python Special Build for Alpine (On Build)" Version="1.0"
EXPOSE 5000
CMD [ "python", "main.py" ]
```

```
docker build -t labdocker/webservice:1.0 \
-f dockerfile_python Path: ~/dockerworkshop/Workshop-2-
1-DockerFile/python_restfulset/
```

Workshop 2-1: DockerFile

- Dockerfile: Webcache

```
FROM labdocker/alpine:latest
MAINTAINER Praparn Lueangphoonlap (eva10409@gmail.com)
LABEL Description="NGINX Build Container" Version="1.0"
RUN apk update && \
apk add nginx
COPY nginx.conf /etc/nginx/nginx.conf
WORKDIR /usr/sbin
ENTRYPOINT ["nginx","-c","/etc/nginx/nginx.conf"]
EXPOSE 8080
```

```
docker build -t labdocker/webcache:1.0 \
-f dockerfile_nginx ~/dockerworkshop/Workshop-2-1-
DockerFile/python_restfulset/
```

Workshop 2-1: DockerFile

- Test run container

```
praparns-MacBook-Pro:python_restfulset praparn$ docker container ps
CONTAINER ID        IMAGE               COMMAND      CREATED          STATUS           PORTS          NAMES
0c8e81f04982        labdocker/webcache:1.0   "nginx -c /etc/nginx..."   5 seconds ago   Up 3 seconds   0.0.0.0:80->8080/tcp   webcache
702b554adb42        labdocker/webservice:1.0  "python main.py"       10 seconds ago  Up 9 seconds   0.0.0.0:5000->5000/tcp   webservice
f259cf6fe88a        labdocker/redis:latest    "docker-entrypoint.s..."  22 seconds ago  Up 20 seconds  6379/tcp          cachedb
b0dbab1856d6        labdocker/mariadb:latest  "/docker-entrypoint..."  48 seconds ago  Up 47 seconds   3306/tcp          maindb
praparns-MacBook-Pro:python_restfulset praparn$
```

```
python_restfulset — Terminal MAC Pro — bash — 155x29
...ockerFile/python_restfulset — Terminal MAC Pro — bash
praparns-MacBook-Pro:python_restfulset praparn$ export Server_IP=127.0.0.1
praparns-MacBook-Pro:python_restfulset praparn$ export Server_Port=80
praparns-MacBook-Pro:python_restfulset praparn$ curl http://$Server_IP:$Server_Port/
<H1> Welcome Page from Container Python Lab </H1>Checkpoint Date/Time: Sun Oct 21 09:36:04 2018
praparns-MacBook-Pro:python_restfulset praparn$
```

The screenshot shows a Mac OS X desktop environment. In the foreground, a Safari browser window is open, displaying the "Welcome Page from Container Python Lab". The URL in the address bar is "127.0.0.1". The browser interface includes a toolbar with various icons and a bookmarks bar at the bottom.

Workshop 2-1: DockerFile

- Init / Insert Data

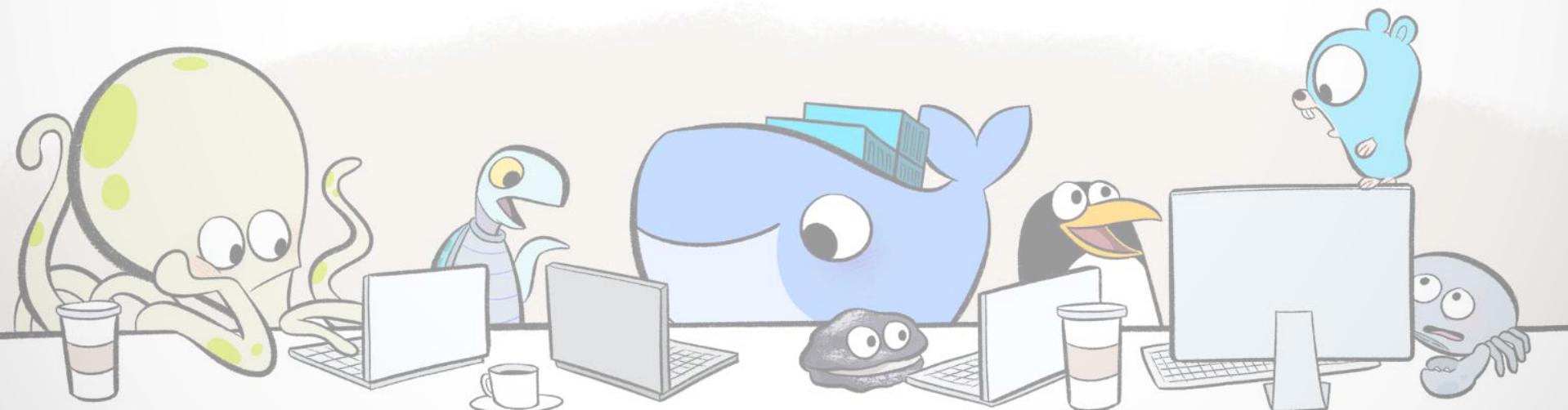
```
praparns-MacBook-Pro% curl http://$Server_IP:$Server_Port/init
#####
Database Create New Account Table Done #####
praparns-MacBook-Pro% curl -i -H "Content-Type: application/json" -X POST -d '{"uid": "1", "user":"Praparn Luangphoonlap", "desribe":"Slave"}' http://$Server_IP:$Server_Port/users/insertuser
curl -i -H "Content-Type: application/json" -X POST -d '{"uid": "2", "user":"Somchai Sunsukwan", "desribe":"Security Guard"}' http://$Server_IP:$Server_Port/users/insertuser
curl -i -H "Content-Type: application/json" -X POST -d '{"uid": "3", "user":"Sanyachan Panrudee", "desribe":"House Keeping"}' http://$Server_IP:$Server_Port/users/insertuser
curl -i -H "Content-Type: application/json" -X POST -d '{"uid": "4", "user":"Sakkan Yanyicharoen", "desribe":"Messenger"}' http://$Server_IP:$Server_Port/users/insertuser
curl -i -H "Content-Type: application/json" -X POST -d '{"uid": "5", "user":"Chatchai Mounrang", "desribe":"Programmer"}' http://$Server_IP:$Server_Port/users/insertuser
curl -i -H "Content-Type: application/json" -X POST -d '{"uid": "6", "user":"Anusit Kannaphat", "desribe":"DevOps Manager"}' http://$Server_IP:$Server_Port/users/insertuser
curl -i -H "Content-Type: application/json" -X POST -d '{"uid": "7", "user":"Meelarp Maisanuk", "desribe":"System Engineer"}' http://$Server_IP:$Server_Port/users/insertuser
curl -i -H "Content-Type: application/json" -X POST -d '{"uid": "8", "user":"Pansa Bunsong", "desribe":"Security Guard"}' http://$Server_IP:$Server_Port/users/insertuser
curl -i -H "Content-Type: application/json" -X POST -d '{"uid": "9", "user":"Wiphanee Wongsaaisawan", "desribe":"Administrator"}' http://$Server_IP:$Server_Port/users/insertuser
curl -i -H "Content-Type: application/json" -X POST -d '{"uid": "10", "user":"Nopparat Nopparat", "desribe":"Administrator"}' http://$Server_IP:$Server_Port/users/insertuser

HTTP/1.1 200 OK
Server: nginx/1.8.1
Date: Sun, 06 Aug 2017 16:06:09 GMT
Content-Type: application/json
Content-Length: 41
Connection: keep-alive
#####
Record was added #####
HTTP/1.1 200 OK
Server: nginx/1.8.1
Date: Sun, 06 Aug 2017 16:06:09 GMT
Content-Type: application/json
Content-Length: 41
Connection: keep-alive
#####
Record was added #####
HTTP/1.1 200 OK
Server: nginx/1.8.1
Date: Sun, 06 Aug 2017 16:06:09 GMT
Content-Type: application/json
Content-Length: 41
Connection: keep-alive
#####
Record was added #####
HTTP/1.1 200 OK
Server: nginx/1.8.1
Date: Sun, 06 Aug 2017 16:06:09 GMT
Content-Type: application/json
Content-Length: 41
Connection: keep-alive
```

Workshop 2-1: DockerFile

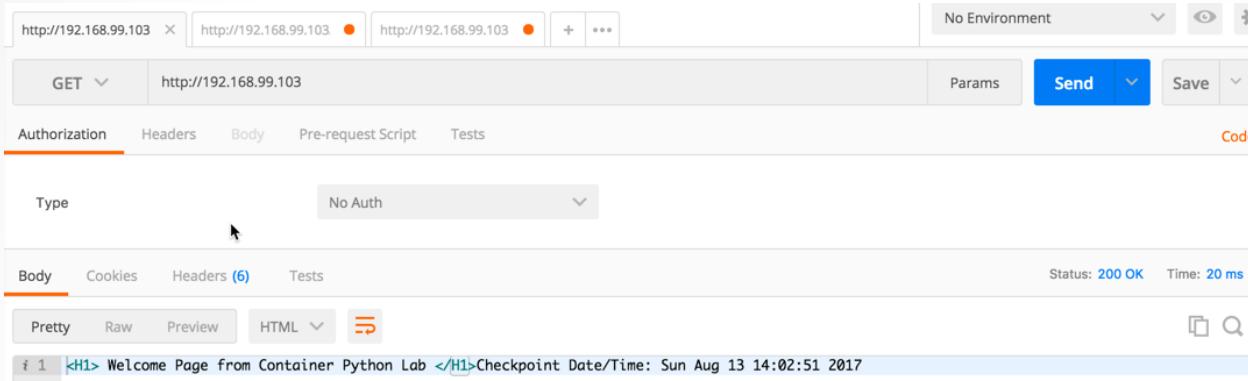
- Get Data (Direct/Cache) and Delete Data

```
~ — Terminal MAC Pro — ssh < docker-machine ssh labdocker
praparns-MacBook-Pro% curl http://$Server_IP:$Server_Port/users/1
Praparn Luangphoonlap(Database Direct)
praparns-MacBook-Pro% curl http://$Server_IP:$Server_Port/users/1
Praparn Luangphoonlap(Database Cache)
praparns-MacBook-Pro% curl http://$Server_IP:$Server_Port/users/4
Sakkan Yanyicharoen(Database Direct)
praparns-MacBook-Pro% curl http://$Server_IP:$Server_Port/users/4
Sakkan Yanyicharoen(Database Cache)
praparns-MacBook-Pro% curl http://$Server_IP:$Server_Port/users/removeuser/1
#####
Record was deleted (Both Database Cache) #####
praparns-MacBook-Pro% curl http://$Server_IP:$Server_Port/users/removeuser/2
#####
Record was deleted #####
praparns-MacBook-Pro% curl http://$Server_IP:$Server_Port/users/removeuser/3
#####
Record was deleted #####
praparns-MacBook-Pro% curl http://$Server_IP:$Server_Port/users/removeuser/4
#####
Record was deleted (Both Database Cache) #####
praparns-MacBook-Pro%
```



Workshop 2-1: DockerFile

- Case for POSTMAN



http://192.168.99.103 X http://192.168.99.103 ● http://192.168.99.103 ● + *** No Environment Send Save

GET http://192.168.99.103 Params Code

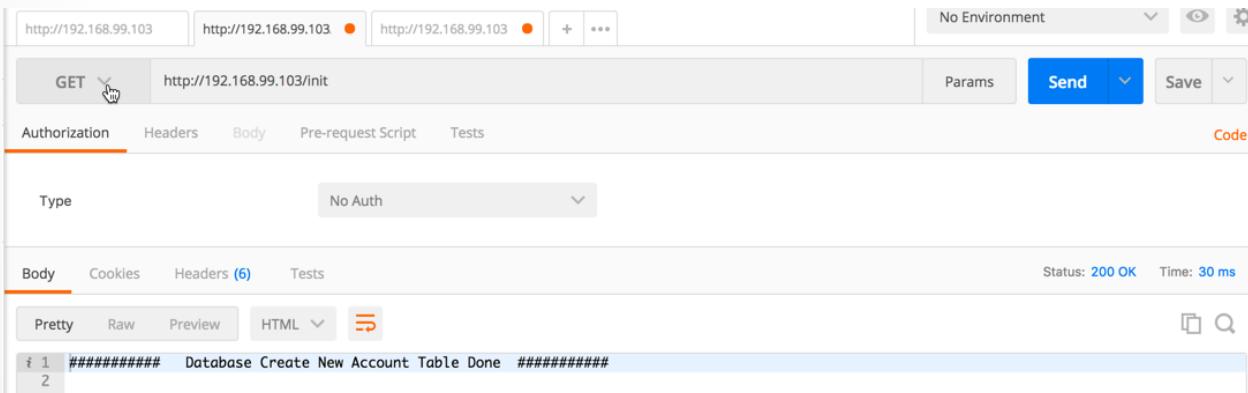
Authorization Headers Body Pre-request Script Tests

Type: No Auth

Body Cookies Headers (6) Tests Status: 200 OK Time: 20 ms

Pretty Raw Preview HTML Copy Search

i 1 <H1> Welcome Page from Container Python Lab </H1>Checkpoint Date/Time: Sun Aug 13 14:02:51 2017



http://192.168.99.103 X http://192.168.99.103 ● http://192.168.99.103 ● + *** No Environment Send Save

GET http://192.168.99.103/init Params Code

Authorization Headers Body Pre-request Script Tests

Type: No Auth

Body Cookies Headers (6) Tests Status: 200 OK Time: 30 ms

Pretty Raw Preview HTML Copy Search

i 1 ##### Database Create New Account Table Done #####

Workshop 2-1: DockerFile

- Case for POSTMAN

The screenshot shows the Postman application interface. At the top, there are three tabs for URLs: http://192.168.99.103, http://192.168.99.103 (highlighted with a red dot), and http://192.168.99.103. To the right of the tabs are buttons for '+', '...', 'No Environment' (with dropdown arrows), and settings (eye and gear icons). Below the tabs, the method is set to 'POST' and the URL is 'http://192.168.99.103/users/insertuser'. On the right, there are 'Params', 'Send' (highlighted with a blue button), 'Save' (with dropdown arrows), and 'Code' buttons. Under the 'Body' tab, which is selected (indicated by an orange underline), the content type is set to 'raw' and 'JSON (application/json)'. The request body contains the following JSON:

```
1 {"uid": "1", "user": "Praparn Luangphoonlap", "desripe": "Slave"}  
2
```

Below the body, the status bar shows 'Status: 200 OK' and 'Time: 25 ms'. The 'Body' tab is also underlined in orange. At the bottom, there are buttons for 'Pretty', 'Raw', 'Preview', and 'HTML' (with a dropdown arrow). To the right are icons for copy (square with arrow) and search (magnifying glass).

Workshop 2-1: DockerFile

- Case for POSTMAN

The image displays two separate instances of the Postman application interface, each showing a list of API requests and their corresponding responses.

Screenshot 1 (Top):

- Request URL: `http://192.168.99.103/users/1`
- Method: GET
- Authorization: No Auth
- Body: Pretty (selected), Raw, Preview, HTML
- Response Status: 200 OK, Time: 22 ms
- ResponseBody:

```
i 1 Praparn Luangphoonlap(Database Direct)
2
```

Screenshot 2 (Bottom):

- Request URL: `http://192.168.99.103/users/1`
- Method: GET
- Authorization: No Auth
- Body: Pretty (selected), Raw, Preview, HTML
- Response Status: 200 OK, Time: 24 ms
- ResponseBody:

```
i 1 Praparn Luangphoonlap(Database Cache)
```

Workshop 2-1: DockerFile

- Case for POSTMAN

The screenshot shows the Postman application interface with two requests made to the same endpoint but with different parameters.

Request 1: GET http://192.168.99.103/users/removeuser/1

Request 2: GET http://192.168.99.103/users/removeuser/3

In both requests, the Body tab is selected, showing the response body:

```
i 1 ##### Record was deleted #####
2
```

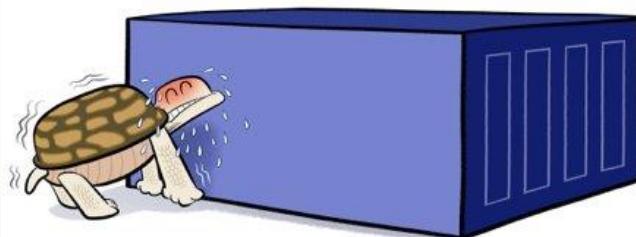
The status for both requests is 200 OK with a time of 22 ms for the first and 25 ms for the second.

Dockerfile

- Multi-State was introduced on April 2017(Since 17.06)
- Write multiple “FROM” on same dockerfile
- Reduce duplicate workload by copy “intermediate state” during build to new image

**Build smaller images with
Multi-stage builds**

**First stage:
complete build
environment**



**Second stage:
minimal runtime
environment** ❤️



One Dockerfile, one build

Dockerfile

Dockerfile1:

```
FROM golang:1.7.3
WORKDIR /go/src/github.com/alexellis(href-counter/
RUN go get -d -v golang.org/x/net/html
COPY app.go .
RUN go get -d -v golang.org/x/net/html \
&& CGO_ENABLED=0 GOOS=linux go build -a -installsuffix cgo -o app .
Output: image 1 unit
```

Dockerfile2:

```
FROM alpine:latest
RUN apk --no-cache add ca-certificates
WORKDIR /root/
COPY app .
CMD ["./app"]
Output: image final unit
```

Dockerfile

Multi-state dockerfile:

```
FROM golang:1.7.3 as builder
WORKDIR /go/src/github.com/alexellis(href-counter/
RUN go get -d -v golang.org/x/net/html
COPY app.go .
RUN CGO_ENABLED=0 GOOS=linux go build -a -installsuffix cgo -o app .
```

```
FROM alpine:latest
RUN apk --no-cache add ca-certificates
WORKDIR /root/
COPY --from=builder /go/src/github.com/alexellis(href-counter/app .
CMD ["./app"]
```

Output: Final Image 1 Unit

Compose

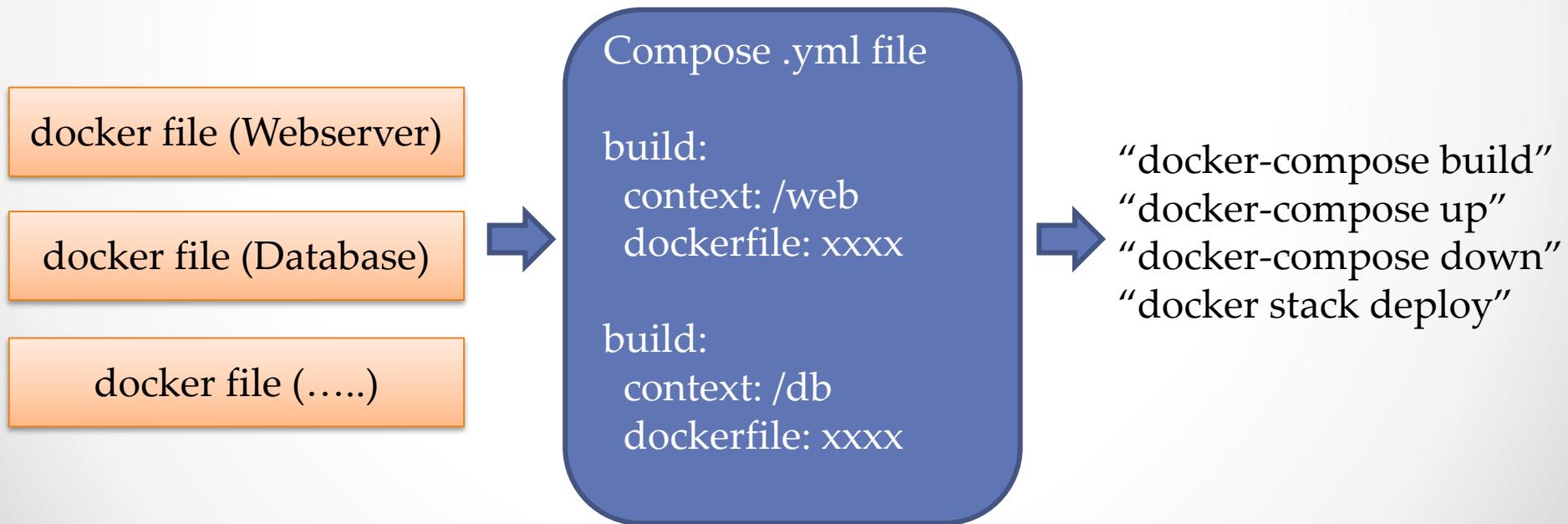
• • •

Compose

- Compose เป็นเครื่องมือที่ใช้ในการสร้าง system service ที่จะใช้ในการรันระบบงานทั้งระบบ ซึ่งตามปกติจะประกอบไปด้วยหลาย component อาทิเช่น
 - Web component service
 - Application component service
 - Database component service
 - Load balance component service
 - etc
- Compose สามารถควบคุมการ start / stop / monitor การทำงานของ service ทั้งระบบเป็น single point
- เหมาะสำหรับการสร้าง development environment / test environment / automatic deploy to production (docker-machine / swarm) (build one ➔ ship to everywhere)
- *No Support --ip now

Compose

- ขั้นตอนในการสร้าง compose
 - สร้าง docker file สำหรับแต่ละ component
 - กำหนดค่า running parameter ใน .yml/.yaml file ของ compose ซึ่งจะอ้างอิงถึง docker file แต่ละตัว
 - docker-compose up



Compose

Compose file format	Docker Engine release
3.7	18.06.0+
3.6	18.02.0+
3.5	17.12.0+
3.4	17.09.0+
3.3	17.06.0+
3.2	17.04.0+
3.1	1.13.1+
3.0	1.13.0+
2.4	17.12.0+
2.3	17.06.0+
2.2	1.13.0+
2.1	1.12.0+
2.0	1.10.0+
1.0	1.9.1.0+

Version 3.4

An upgrade of [version 3](#) that introduces new parameters. It is only available with Docker Engine version **17.09.0** and higher.

Introduces the following additional parameters:

- `target` and `network` in [build configurations](#)
- `start_period` for [healthchecks](#)
- `order` for [update configurations](#)
- `name` for [volumes](#)

Version 3.5

An upgrade of [version 3](#) that introduces new parameters. It is only available with Docker Engine version **17.12.0** and higher.

Introduces the following additional parameters:

- `isolation` in service definitions
- `name` for networks, secrets and configs
- `shm_size` in [build configurations](#)

Version 3.6

An upgrade of [version 3](#) that introduces new parameters. It is only available with Docker Engine version **18.02.0** and higher.

Introduces the following additional parameters:

- `tmpfs.size` for `tmpfs`-type mounts

Version 3.7

An upgrade of [version 3](#) that introduces new parameters. It is only available with Docker Engine version **18.06.0** and higher.

Introduces the following additional parameters:

- `init` in service definitions
- `rollback_config` in [deploy configurations](#)
- Support for extension fields at the root of service, network, volume, secret and config definitions

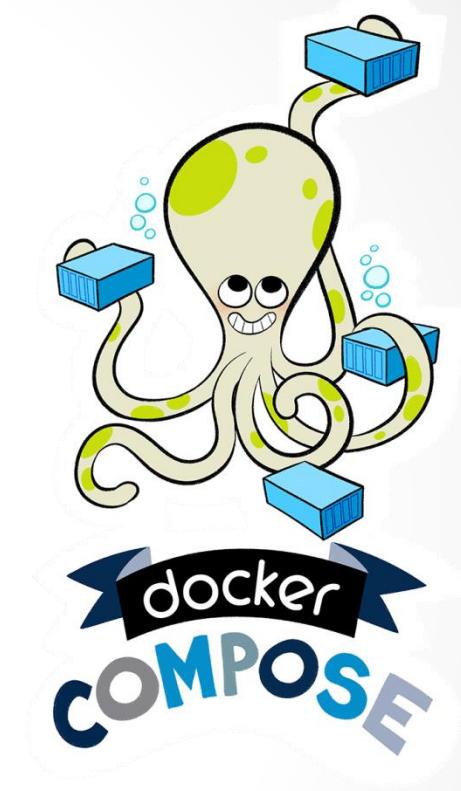
Compose

- Compose syntax: (docker compose up/down/run)
 - version: x.x (Current Version 3.4)
 - services:
 - XXX (service name):
 - image: <image name>
 - build:
 - target: /xxxx (New on Version 3.4)
 - cache_from: (New on Version 3.2)
 - <image name>
 - labels: (New on Version 3.3)
 - <label>: <value>
 - context: ./<path>
 - dockerfile: <file>
 - container_name: <name>
 - args:
 - <xxxx>:<value>
 - dns: x.x.x.x
 - dns_search: xxx.com
 - entrypoint: ["nginx","-c","/etc/nginx/nginx.conf"]
 - env_file: xxxx.xxx (VAR=VAL)
 - Environment:
 - TOKEN: XXXX
 - ports:
 - - "9999:9999"
 - Link:
 - xx:<alias>
 - healthcheck:
 - test: ["CMD-SHELL", "pg_isready"]
 - interval: 30s
 - timeout: 30s
 - retries: 3
 - start_period: 100s (New on Version 3.4)



Compose

- Compose syntax:
 - XXX (service name):
 - depends_on:
 - XXXX
 - XXXX
 - networks:
 - XXXX1
 - XXXX2



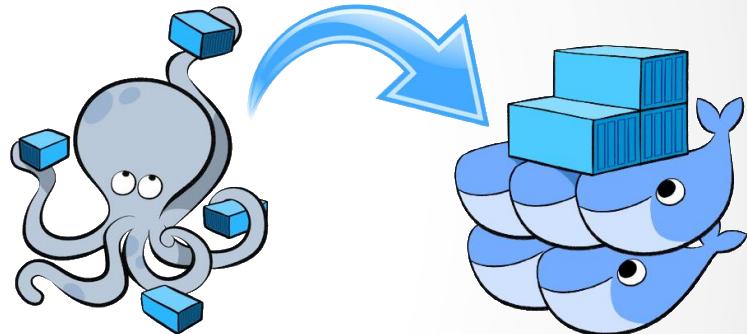
Ref:<https://docs.docker.com/compose/compose-file/#command>

Docker: The Next-Gen of Virtualization



Compose

- Compose syntax: (docker stack deploy)
 - services:
 - XXX (service name): (Current Version 3.4)
 - image: <image name>
 - **deploy: <SWARM>**
 - mode:
 - global
 - replicated
 - resource:
 - limits:
 - cpus: '0.5'
 - memory: 100M
 - reservations:
 - cpus: '0.1'
 - memory: 20M
 - restart_policy:
 - condition: on-failure
 - delay: 5s
 - max_attempts: 3
 - update_config:
 - parallelism: 2
 - delay: 10s
 - failure_action: 10ms
 - max_failure_ratio: X
 - order: (stop-first,start-first) <New on Version 3.4>



Compose

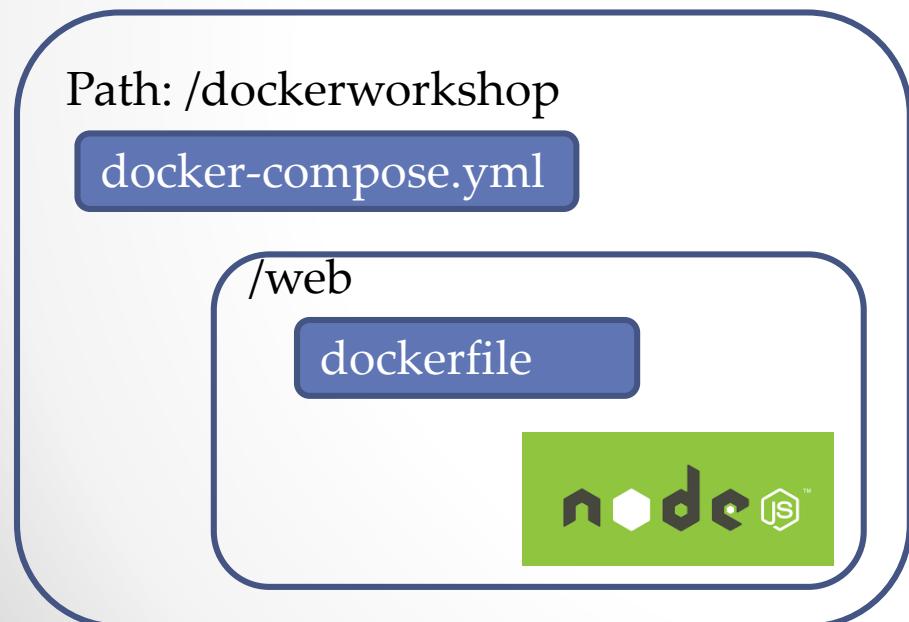
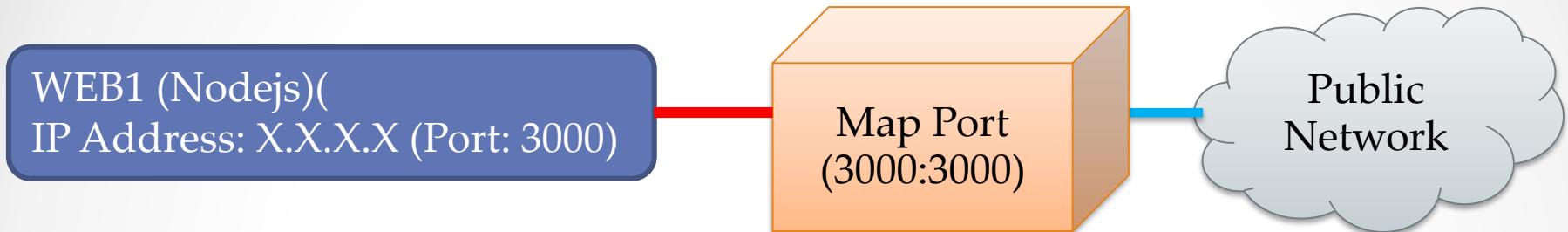
- DOCKER STACK DEPLOY

- Not Supported
 - build
 - cgroup_parent
 - container_name
 - devices
 - dns
 - dns_search
 - external_links
 - links
 - network_mode
 - security_opt
 - stop_signal
 - sysctls
 - userns_mode



Workshop 2-2: Compose

- Part1: componenodejs



Docker: The Next-Gen of Virtualization



Workshop 2-2: Compose

- dockerfile

```
FROM labdocker/alpine:latest
MAINTAINER Praparn Lueangphoonlap (eva10409@gmail.com)
LABEL Description="NodeJS/NGINX Build Container" Version="1.0"
ENV NODE_VERSION=v4.3.0 NPM_VERSION=2.14.12
RUN apk update && \
    apk add nodejs
RUN mkdir /nodejs
COPY hello.js /nodejs/
WORKDIR /nodejs
ENTRYPOINT ["node","hello.js"]
EXPOSE 3000
```

- docker-compose.yml

```
version: "3.4"
services:
  web:
    build:
      context: ./web
      dockerfile: dockerfile          # optional for specific dockerfile
      container_name: nodejs          # optional for specific container name
      command: ["node","hello.js"]     # optional for override command on dockerfile
    ports:
      - 3000:3000
```

Workshop 2-2: Compose

- Part 2: composemultinodejs

WEB1 (No Map)

IP Address: (Webinternal)(Private)

WEB2 (No Map)

IP Address: (Webinternal) (Private)

vSwitch: Webinternal
IP Address: 192.168.101.0/24

NGINX (Map Port:80:8080)
IP Address: Webpublic (Public)
IP Addrerss: Webpublic (Private)

vSwitch: Webpublic
IP Address: 192.168.100.0/24

Map Port
(80:8080)

Public
Network

Workshop 2-2: Compose

- Use case: multinodejs with nginx

Path: /dockerworkshop

docker-compose.yml

/nginx

dockerfile
(webinternal)
(webpublic)

nginx.conf
load balance:
web1
web2



/web1

dockerfile (webinternal)



/web2

dockerfile (webinternal)



Workshop 2-2: Compose

- nginx: dockerfile

```
FROM labdocker/alpine:latest
MAINTAINER Praparn Lueangphoonlap (eva10409@gmail.com)
LABEL Description="NodeJS/NGINX Build Container" Version="1.0"
RUN apk update && \
    apk add nginx
COPY nginx.conf /etc/nginx/nginx.conf
WORKDIR /usr/sbin
ENTRYPOINT ["nginx","-c","/etc/nginx/nginx.conf"]
EXPOSE 8080
```

- Web1 & Web 2: dockerfile

```
FROM labdocker/alpine:latest
MAINTAINER Praparn Lueangphoonlap (eva10409@gmail.com)
LABEL Description="NodeJS/NGINX Build Container" Version="1.0"
ENV NODE_VERSION=v4.3.0 NPM_VERSION=2.14.12
RUN apk update && \
    apk add nodejs
RUN mkdir /nodejs
COPY hello.js /nodejs/
WORKDIR /nodejs
ENTRYPOINT ["node","hello.js"]
EXPOSE 3000
```

Workshop 2-2: Compose

- Web1: Hello.js

```
var http = require('http');
http.createServer(function (req, res)
{res.writeHead(200, {'Content-Type': 'text/plain'});
res.end('Hello World Container Docker for Nodejs Node 1\n');}).listen(3000,
'0.0.0.0');
console.log('Server running at http://0.0.0.0:3000/');
```

- Web2: Hello.js

```
var http = require('http');
http.createServer(function (req, res)
{res.writeHead(200, {'Content-Type': 'text/plain'});
res.end('Hello World Container Docker for Nodejs Node 2\n');}).listen(3000,
'0.0.0.0');
console.log('Server running at http://0.0.0.0:3000/');
```

Workshop 2-2: Compose

- docker-compose.yml

```
1  version: '3.4'
2  services:
3    nginx:
4      build:
5        context: ./nginx
6        dockerfile: dockerfile          # optional for specific dockerfile
7        cache_from:
8          - alpine:latest
9      container_name: nginx
10     depends_on:
11       - web1
12       - web2
13     healthcheck:
14       test: ["CMD", "curl", "-f", "http://localhost:80"]  # option for health check and send curl for check http://localhost:3000
15       interval: 30s                                     # interval health check
16       timeout: 10s                                     # timeout for check
17       retries: 3                                       # maximum retries
18       start_period: 30s                                # start period <news on 3.4>
19     networks:
20       webpublic:
21         aliases:
22           - nginx
23       webinternal:
24         aliases:
25           - nginx
26     ports:
27       - "80:8080"
```

Workshop 2-2: Compose

- docker-compose.yml

```
29  web1:
30    build:
31      context: ./web1
32      dockerfile: dockerfile          # optional for specific dockerfile
33      cache_from:
34        - labdocker/alpineweb:latest
35    container_name: web1
36    healthcheck:
37      test: ["CMD", "curl", "-f", "http://localhost:3000"] # option for health check and send curl for check http://localhost:3000
38      interval: 30s                                     # interval health check
39      timeout: 10s                                     # timeout for check
40      retries: 3                                       # maximum retries
41      start_period: 30s                                # start period <news on 3.4>
42    networks:
43      webinternal:
44        aliases:
45        - web1
46
47  web2:
48    build:
49      context: ./web2
50      dockerfile: dockerfile          # optional for specific dockerfile
51      cache_from:
52        - labdocker/alpineweb:latest
53    container_name: web2
54    healthcheck:
55      test: ["CMD", "curl", "-f", "http://localhost:3000"] # option for health check and send curl for check http://localhost:3000
56      interval: 30s                                     # interval health check
57      timeout: 10s                                     # timeout for check
58      retries: 3                                       # maximum retries
59      start_period: 30s                                # start period <news on 3.4>
60    networks:
61      webinternal:
62        aliases:
63        - web2
```

Workshop 2-2: Compose

- docker-compose.yml

```
65 networks:
66   webpublic:
67     driver: bridge
68     ipam:
69       driver: default
70       config:
71         - subnet: 192.168.100.0/24
72   webinternal:
73     driver: bridge
74     ipam:
75       driver: default
76       config:
77         - subnet: 192.168.101.0/24
78
```

Registry

• • •

Registry

- Registry เป็น container แบบหนึ่งที่ใช้สำหรับจัดเก็บ image ที่สร้างขึ้นไว้บน private server โดยไม่จำเป็นต้องผ่าน public internet (hub.docker.com) ทำให้สามารถจัดเก็บ image ภายในองค์กรได้อย่างปลอดภัย
- Registry เป็นพื้นที่สำหรับจัดเก็บ image ที่ถูกสร้างขึ้นโดยแบ่งออกตาม image name และ tag version ที่กำหนดไว้

Ex: labdocker/alpineweb:latest

- กรณีการใช้งาน registry ระหว่าง Server ด้วยกันสามารถกำหนด TLS security ระหว่างกันได้
 - สามารถเลือก storage backend อื่นๆในการจัดเก็บ image ได้ เช่น Etc: s3 (aws), azure, gcs (google cloud) etc
 - มี option gc (garbage collect) ด้วย option “bin/registry garbage-collect [--dry-run] config.yml”



Workshop 2-3: Registry

- Part1: Registry on single docker-machine
- เริ่มใช้งาน registry โดย download image registry ตามตัวอย่าง

```
docker image pull registry:2.6.2 (registry version 2.0)
```

- Start container เพื่อเริ่มใช้งาน

```
docker container run -d -p 5000:5000 \
--restart=always --name registry \
-e REGISTRY_STORAGE_DELETE_ENABLED=true \
--mount type=bind,source=/home/docker,target=/var/lib/registry \
registry:2.5.2
```

- ดำเนินการ tag image และทดสอบ push image docker เข้า registry

```
docker image tag labdocker/alpine:latest \
localhost:5000/alpine:latest
```

```
docker push localhost:5000/alpine:latest
```

- *ต้องบันทึก digest image ไว้เพื่อใช้อ้างอิงภายหลัง

Workshop 2-3: Registry

- ในการตรวจสอบ registry image file เพื่อให้เกิดความง่ายต่อการใช้งาน docker ได้จัดเตรียม HTTP API สำหรับการเรียกใช้งานบน registry ผ่าน curl/customize application ซึ่งรองรับการใช้งาน อาทิเช่น
- การ List รายการ image ที่จัดเก็บไว้บน

```
curl -X GET http://localhost:5000/v2/_catalog
```

- ตรวจสอบ revision image tag ที่จัดเก็บไว้ในแต่ละ image

```
curl -X GET http://localhost:5000/v2/<name>/tags/list
```

```
curl -X GET http://localhost:5000/v2/alpine/tags/list
```

- สามารถทำการลบ image ที่จัดเก็บไว้โดยอ้างอิงจาก digest

```
curl -X DELETE  
http://localhost:5000/v2/alpine/manifests/<digest>
```

Workshop 2-3: Registry

- ตรวจสอบ image ทาง physical file จาก docker-machine

```
cd /home/docker  
cd /docker/registry/v2
```

- ลบ image file ออกจาก docker-machine

```
docker image rm localhost:5000/alpine:latest
```

- ทำการดึง image เพิ่มมาจาก registry

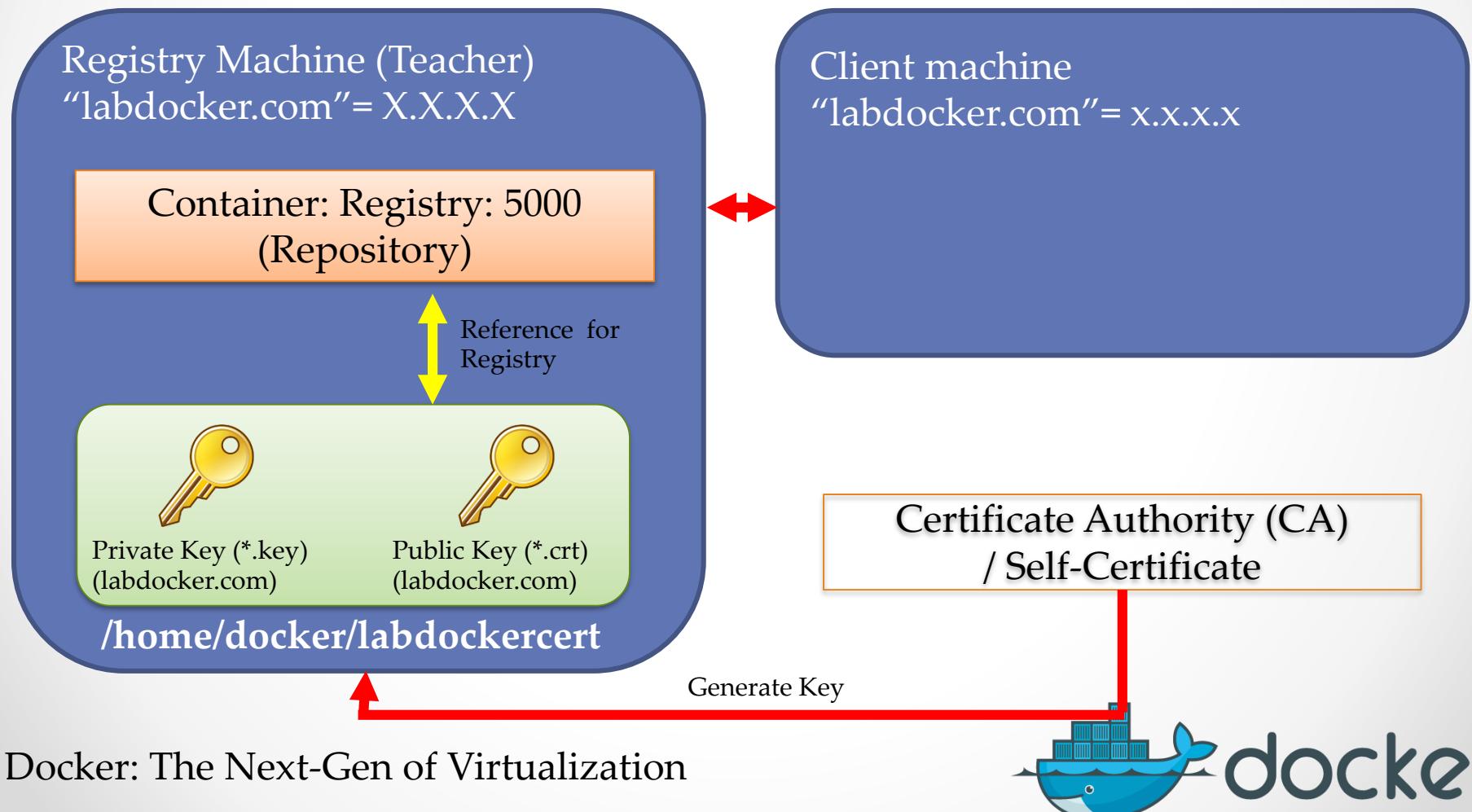
```
docker image pull localhost:5000/alpine:latest
```

- ทำการลบ container registry เพื่อยกเลิกการใช้งาน

```
docker container stop registry  
docker container rm registry
```

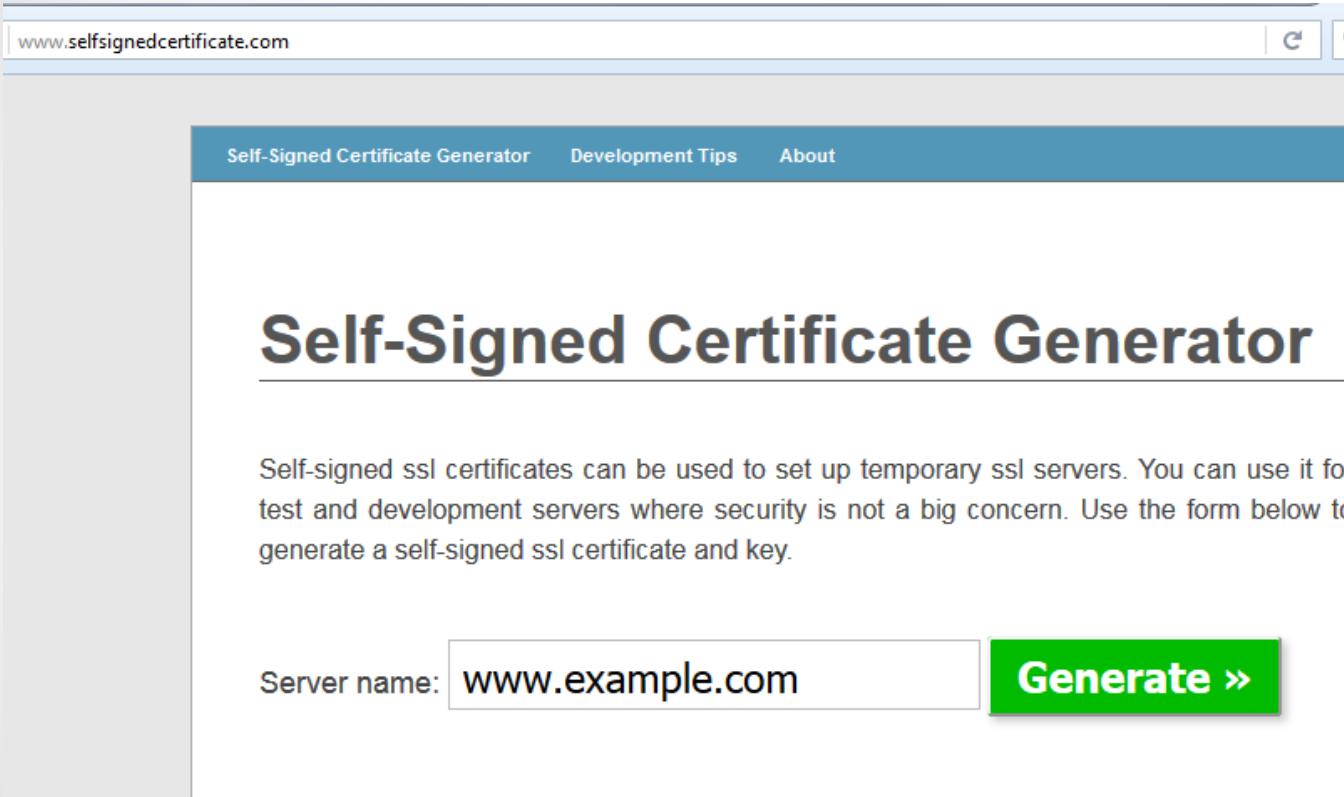
Registry on difference machine

- Part2: Registry on multiple docker-machine
- กรณีที่ต้องใช้งาน registry จาก docker server มากกว่า 1 เครื่อง เราสามารถดึงต้องสร้าง TLS certificate (next generate of SSL) เพื่อ trust registry ระหว่างกัน



Registry on difference machine

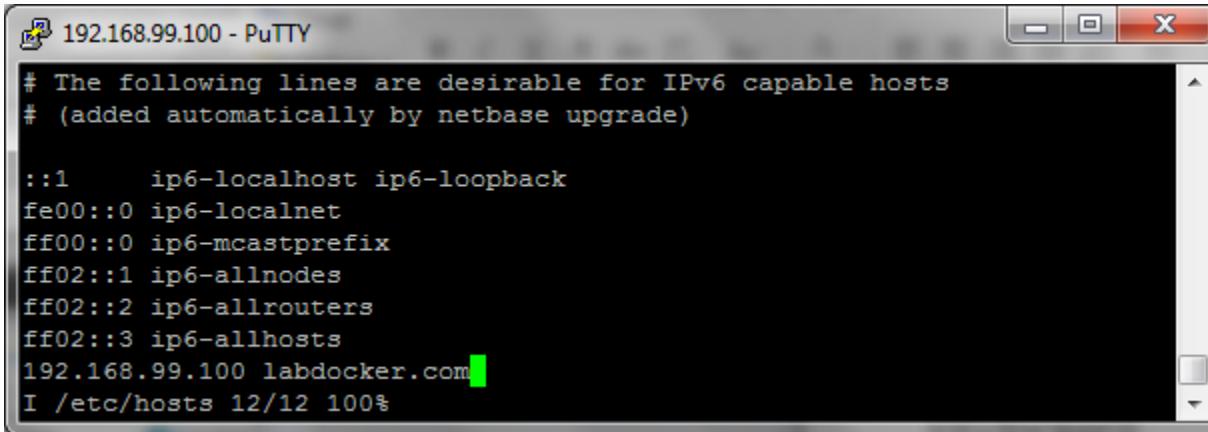
- การสร้าง certificate จาก CA / Self-Signed



The screenshot shows a web browser window with the URL www.selfsignedcertificate.com in the address bar. The page title is "Self-Signed Certificate Generator". Below the title, there is a paragraph explaining that self-signed SSL certificates can be used for temporary servers, particularly for testing and development. A form field labeled "Server name:" contains the value "www.example.com". To the right of the form is a large green button with the text "Generate »". In the top right corner of the browser window, there is an advertisement for "sslstore" featuring a keyhole icon and the text "AFFORDABLE SSL CERTIFICATES" and "ISSUED IN MINUTES*".

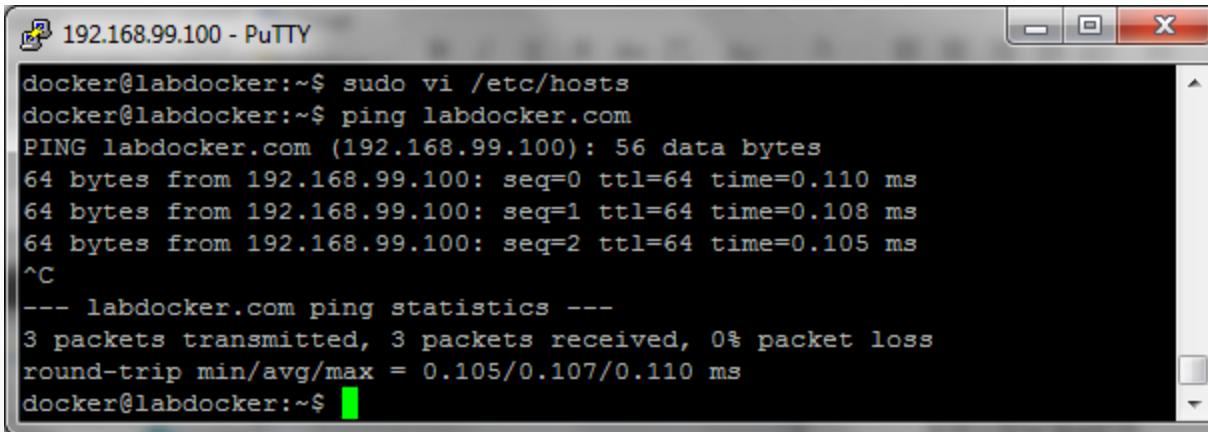
Registry on difference machine

- เพิ่ม domain “labdocker.com” ลงใน /etc/hosts (labdocker, labdocker2)



```
# The following lines are desirable for IPv6 capable hosts
# (added automatically by netbase upgrade)

::1      ip6-localhost ip6-loopback
fe00::0  ip6-localnet
ff00::0  ip6-mcastprefix
ff02::1  ip6-allnodes
ff02::2  ip6-allrouters
ff02::3  ip6-allhosts
192.168.99.100 labdocker.com
I /etc/hosts 12/12 100%
```



```
docker@labdocker:~$ sudo vi /etc/hosts
docker@labdocker:~$ ping labdocker.com
PING labdocker.com (192.168.99.100): 56 data bytes
64 bytes from 192.168.99.100: seq=0 ttl=64 time=0.110 ms
64 bytes from 192.168.99.100: seq=1 ttl=64 time=0.108 ms
64 bytes from 192.168.99.100: seq=2 ttl=64 time=0.105 ms
^C
--- labdocker.com ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 0.105/0.107/0.110 ms
docker@labdocker:~$
```

Registry on difference machine

- สำหรับกรณีเครื่อง ubuntu ที่ไม่ใช้งาน self-certificate ให้ทำการ trust root ca ดังนี้

```
sudo mkdir /usr/local/share/ca-certificates/labdocker.com:5000  
sudo cp labdocker.com.crt /usr/local/share/ca-certificates/labdocker.com:5000  
sudo update-ca-certificates  
sudo service docker restart
```

```
ubuntu@ip-10-0-1-182:~/docker-workshop-112018-cloud/Workshop-2-3-Registry  
instruction.txt labdocker.com.crt labdocker.com.key  
[ubuntu@ip-10-0-1-182:~/docker-workshop-112018-cloud/Workshop-2-3-Registry]$ sudo mkdir /usr/local/share/ca-certificates/labdocker.com:5000  
[ubuntu@ip-10-0-1-182:~/docker-workshop-112018-cloud/Workshop-2-3-Registry]$ sudo cp labdocker.com.crt /usr/local/share/ca-certificates/labdocker.com:5000  
[ubuntu@ip-10-0-1-182:~/docker-workshop-112018-cloud/Workshop-2-3-Registry]$ sudo update-ca-certificates  
Updating certificates in /etc/ssl/certs...  
1 added, 0 removed; done.  
Running hooks in /etc/ca-certificates/update.d...  
done.  
[ubuntu@ip-10-0-1-182:~/docker-workshop-112018-cloud/Workshop-2-3-Registry]$ sudo service docker restart  
ubuntu@ip-10-0-1-182:~/docker-workshop-112018-cloud/Workshop-2-3-Registry$ █
```

Registry on difference machine

- เริ่มใช้งาน registry โดย start container ดังนี้ (labdocker)

```
docker container run -d -p 5000:5000 \
    --restart=always --name registrylab \
    --mount type=bind,source=$(pwd),target=/var/lib/registry \
    --mount type=bind,source=$(pwd),target=/certs \
    --mount type=bind,source=$(pwd),target=/auth \
    -e "REGISTRY_AUTH=hpasswd" \
    -e "REGISTRY_AUTH_HTPASSWD_REALM=Registry Realm" \
    -e REGISTRY_AUTH_HTPASSWD_PATH=/auth/hpasswd \
    -e REGISTRY_STORAGE_DELETE_ENABLED=true \
    -e REGISTRY_HTTP_TLS_CERTIFICATE=/certs/labdocker.com.crt \
    -e REGISTRY_HTTP_TLS_KEY=/certs/labdocker.com.key \
    registry:2.6.2
```

- ดำเนินการ tag image, login และทดสอบ push image docker เข้า registry

```
docker image tag labdocker/alpine:latest labdocker.com:5000/alpine:1.0
docker login -u <username> -p <password>
docker image push labdocker.com:5000/alpine:1.0
```

Registry on difference machine

- ในการตรวจสอบ registry image file สำหรับ registry ที่มีการ enable TLS เรียบง่าย แล้วต้องเพิ่มพารามิเตอร์ “--cacert” เพื่อระบุ certificate สำหรับการใช้งานและเปลี่ยน protocol เป็น https
- การ List รายการ image ที่จัดเก็บไว้บน

```
curl -u <username:password> --cacert /Share_DockerToolbox/labdocker.com.crt -X  
GET https://labdocker.com:5000/v2/_catalog
```

- ตรวจสอบ revision image tag ที่จัดเก็บไว้ในแต่ละ image

```
curl -u <username:password> --cacert /home/docker/labdockercert/labdocker.com.crt -  
X GET https://labdocker.com:5000/v2/alpine/tags/list
```

- สามารถทำการลบ image ที่จัดเก็บไว้โดยอ้างอิงจาก digest

```
curl -u <username:password> --cacert /home/docker/labdockercert/labdocker.com.crt  
-X DELETE https://labdocker.com:5000/v2/alpine/manifests
```

Registry on difference machine

- ดึง image ผ่าน labdocker 2

```
[root@minavgmax:~]# curl -v http://10.0.1.239:5000/_ping -m 1000 -H "Content-Type: application/json" -d '{"method": "GET", "path": "/"}' | jq .[0].body
[ubuntu@ip-10-0-1-239:~/docker-workshop-112018-cloud/Workshop-2-3-Registry]$ docker login labdocker.com:5000
[Username: docker
[Password:
WARNING! Your password will be stored unencrypted in /home/ubuntu/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
[ubuntu@ip-10-0-1-239:~/docker-workshop-112018-cloud/Workshop-2-3-Registry]$ docker image pull labdocker.com:5000/alpine:1.0
1.0: Pulling from alpine
Digest: sha256:02892826401a9d18f0ea01f8a2f35d328ef039db4e1edcc45c630314a0457d5b
Status: Downloaded newer image for labdocker.com:5000/alpine:1.0
ubuntu@ip-10-0-1-239:~/docker-workshop-112018-cloud/Workshop-2-3-Registry$ █
```

- Optional: สำหรับเพิ่มรันเว็บเพื่อตรวจสอบ Container

```
-v `pwd`/auth:/auth \
-e "REGISTRY_AUTH=htpasswd" \
-e "REGISTRY_AUTH_HTPASSWD_REALM=Registry
Realm" \
```

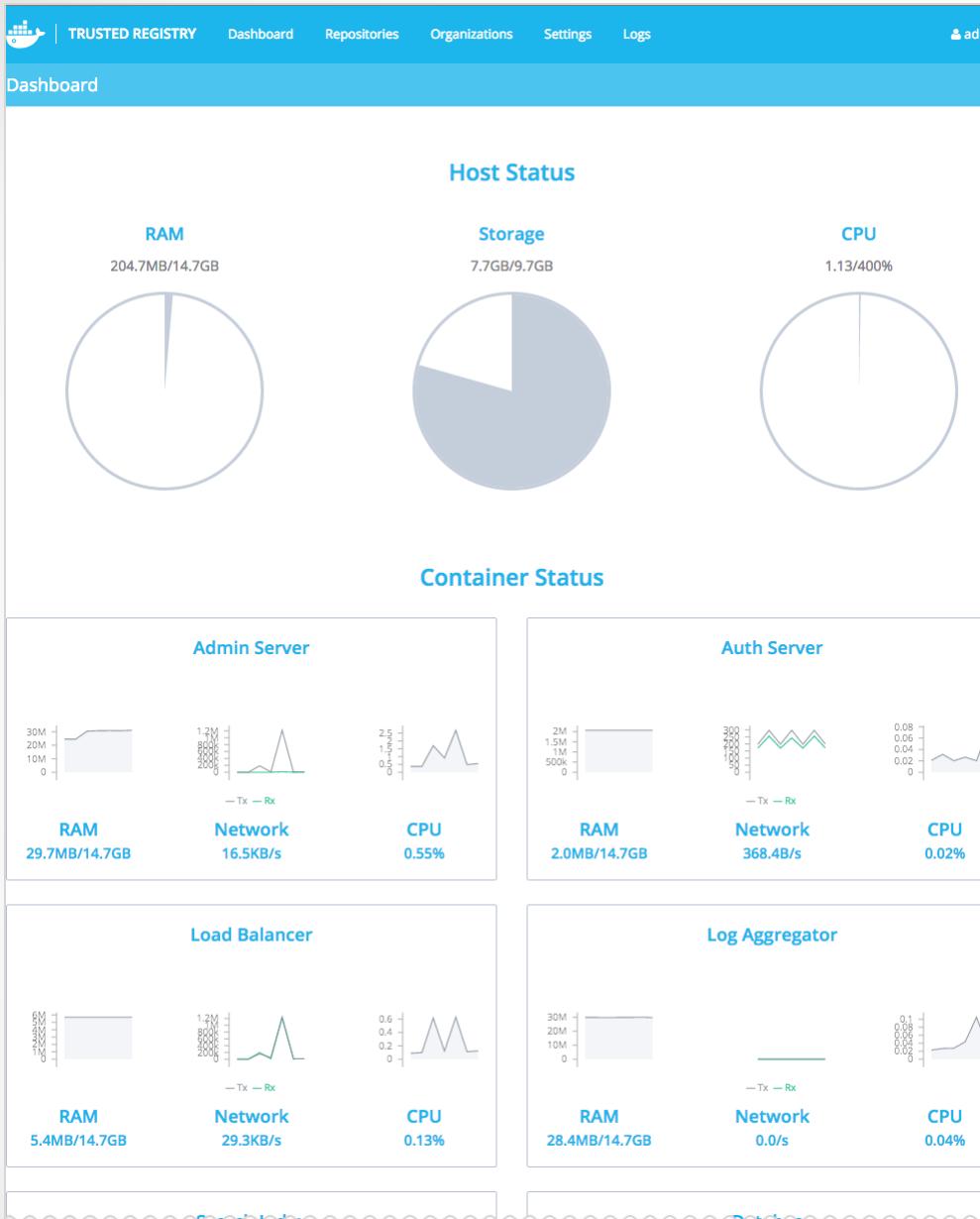
- ทำการลบ container registry เพื่อยกเลิกการใช้งาน

```
docker container stop registry
docker container rm -v registry
```

Ref: <https://docs.docker.com/registry/deploying/>

Docker: The Next-Gen of Virtualization





Docker Trust Registry (DTR)

- Commercial support
- GUI for managing everything
- Integrated LDAP authentication
- GC schedule available

Docker: The Next-Gen of Virtualization



Open Source Registry (Experiment)

- <http://port.us.org>

The screenshot shows the Portus interface for the 'flavio/busybox' repository. The left sidebar has 'Portus' at the top, followed by 'Dashboard' and 'Images'. The main content area shows the repository name 'flavio/busybox' in large blue text. Below it is a table with three rows, each representing a different tag for the repository:

Repository	Tag	Pushed at
flavio/busybox	1.0.0	2015-04-22 14:50:08 UTC
flavio/busybox	2.0.0	2015-04-24 14:12:17 UTC
flavio/busybox	latest	2015-04-24 14:15:34 UTC

Artifact Registry

- Sonatype

The screenshot shows a portion of the Sonatype website's documentation for Nexus Repository Manager 3. At the top, there is a navigation bar with links for Documentation, My Sonatype, Community, Guides, Learn, Support, and Who is Sonatype? Below the navigation bar is a search bar labeled "Search the docs...". A button labeled "Switch to another product ▾" is also present. The main content area has a breadcrumb navigation path: Nexus Repository Manager 3 > Private Registry for Docker. The title of the page is "Private Registry for Docker". A callout box indicates that this feature is "Available in Nexus Repository OSS and Nexus Repository Pro". The page content discusses the revolutionization of application packaging and deployment by Docker containers, mentions the Open Container Initiative, and highlights the support for Docker registries in Nexus Repository Manager Pro and OSS. It also describes how Docker Hub and other public registries like Google Container Registry are being joined. The text further explains the features of Docker registries in Nexus Repository Manager, such as exposing multiple repositories under one URL and launching containers based on those images.

NEXUS REPOSITORY MANAGER 3

> Download
> Release Notes

System Requirements

Upgrade Compatibility - Repository Manager 2 to 3

Repository Manager 2 to 3 Feature Equivalency

Repository Manager Feature Matrix

> Repository Manager Concepts
Supported Formats
> Installation
> Upgrading
> User Interface
> Configuration
> Backup and Restore
Cleanup Policies
> High Availability
Quick Start Guide - Proxying Maven and NPM
Staging
Tagging

Nexus Repository Manager 3 > Private Registry for Docker

Private Registry for Docker

ⓘ Available in Nexus Repository OSS and Nexus Repository Pro

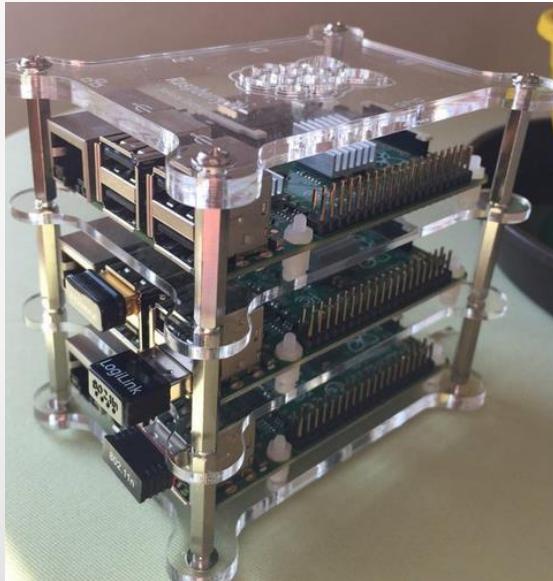
Docker containers and their usage have revolutionized the way applications and the underlying operating system are packaged and deployed to development, testing and production systems. The creation of the [Open Container Initiative](#), and the involvement of a large number of stakeholders, guarantees that the ecosystem of tools around the lightweight containers and their usage will continue to flourish. Docker Hub is the original registry for Docker container images and it is being joined by more and more other publicly available registries such as the [Google Container Registry](#) and others. Nexus Repository Manager Pro and Nexus Repository Manager OSS support Docker registries as the Docker repository format for hosted and proxy repositories. You can expose these repositories to the client-side tools directly or as a repository group, which is a repository that merges and exposes the contents of multiple repositories in one convenient URL. This allows you to reduce time and bandwidth usage for accessing Docker images in a registry as well as share your images within your organization in a hosted repository. Users can then launch containers based on those images, resulting in a completely private Docker registry with all the features available in the repository manager.

Swarm

• • •

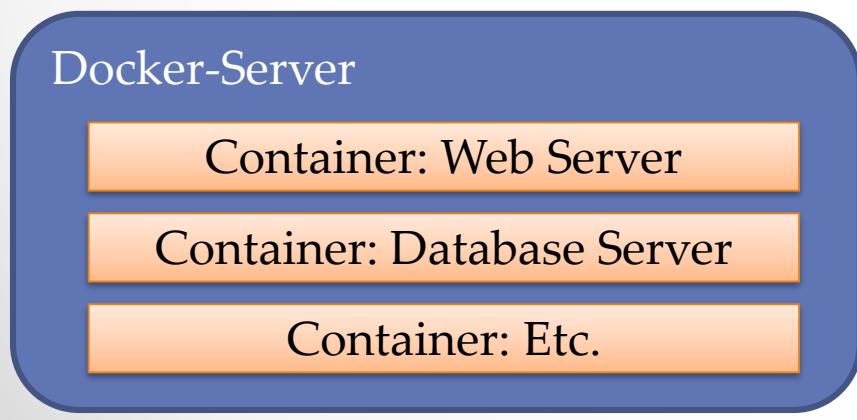
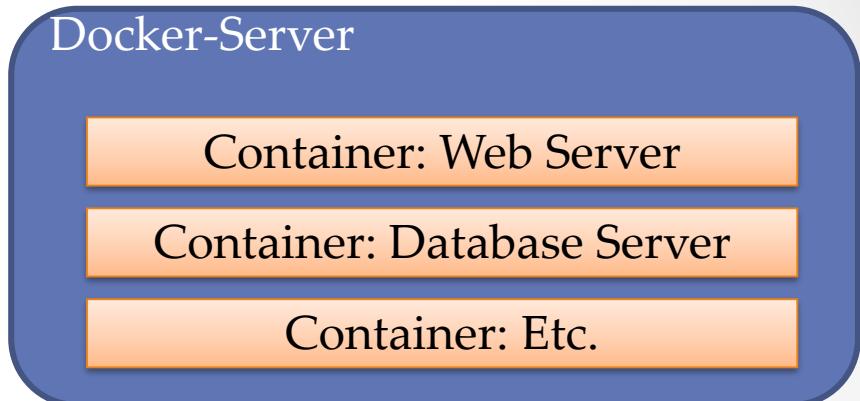
Swarm: Conceptual

- Swarm ถือเป็น native tool ในการจัดการ docker-engine หลายๆเครื่องให้รวมเป็น resource pool เดียวกัน เพื่อให้ง่ายต่อการบริหารจัดการ
- เพิ่มขีดความสามารถในการทำงานร่วมกันของ docker-engine
- Hybrid Swan Cluster (Windows / Linux)
- Fail Over / High Availability (HA)
- Micro Service



Swarm: Conceptual

- Traditional Docker Deployment

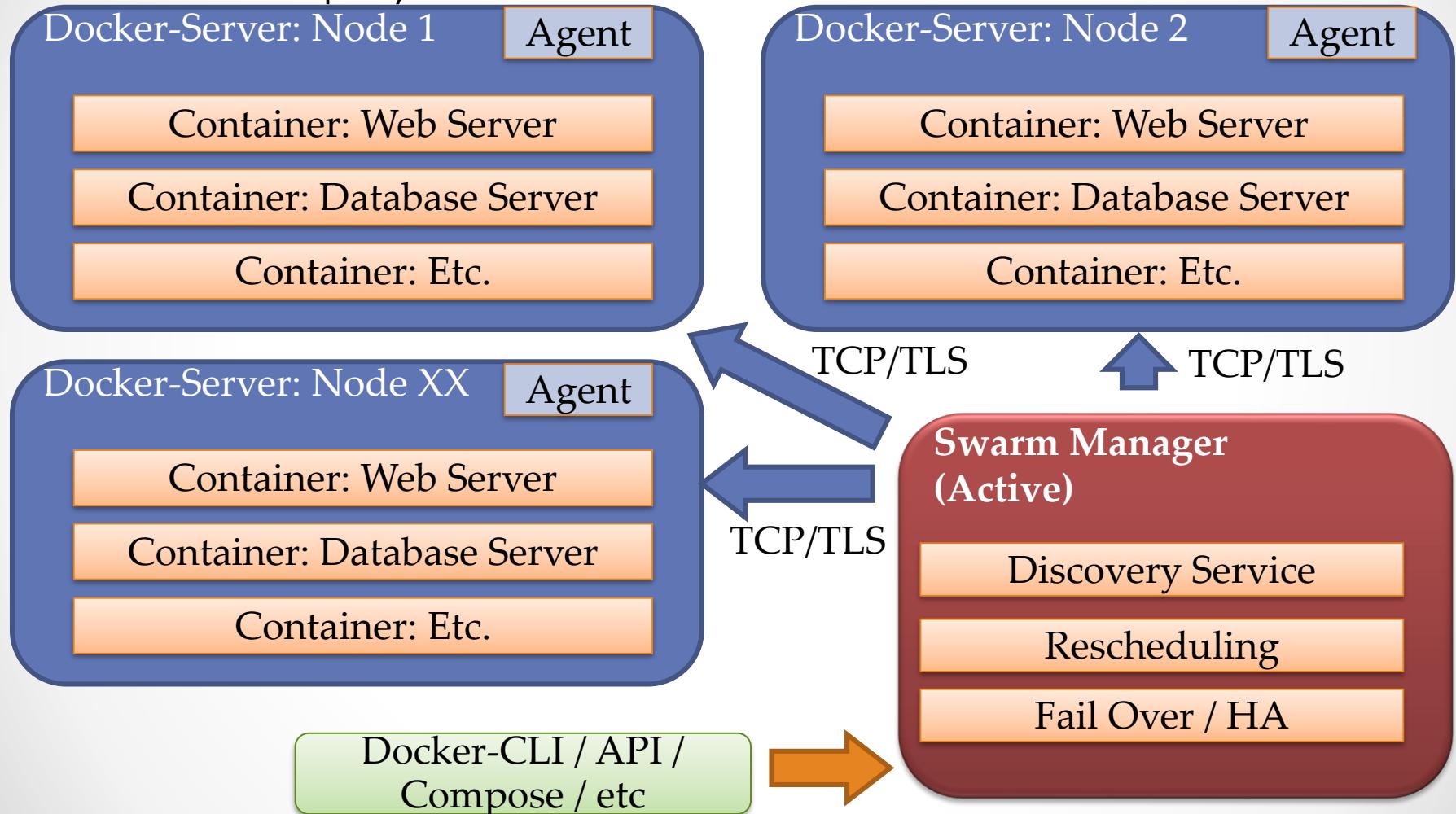


Docker: The Next-Gen of Virtualization



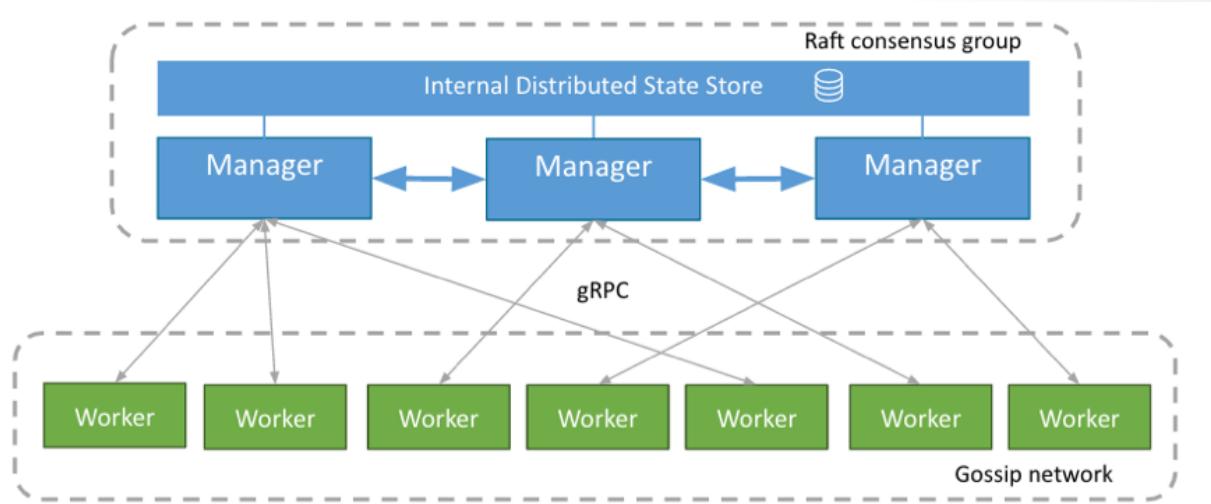
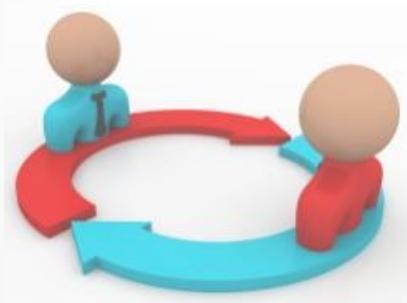
Swarm: Conceptual

- Swarm Deployment



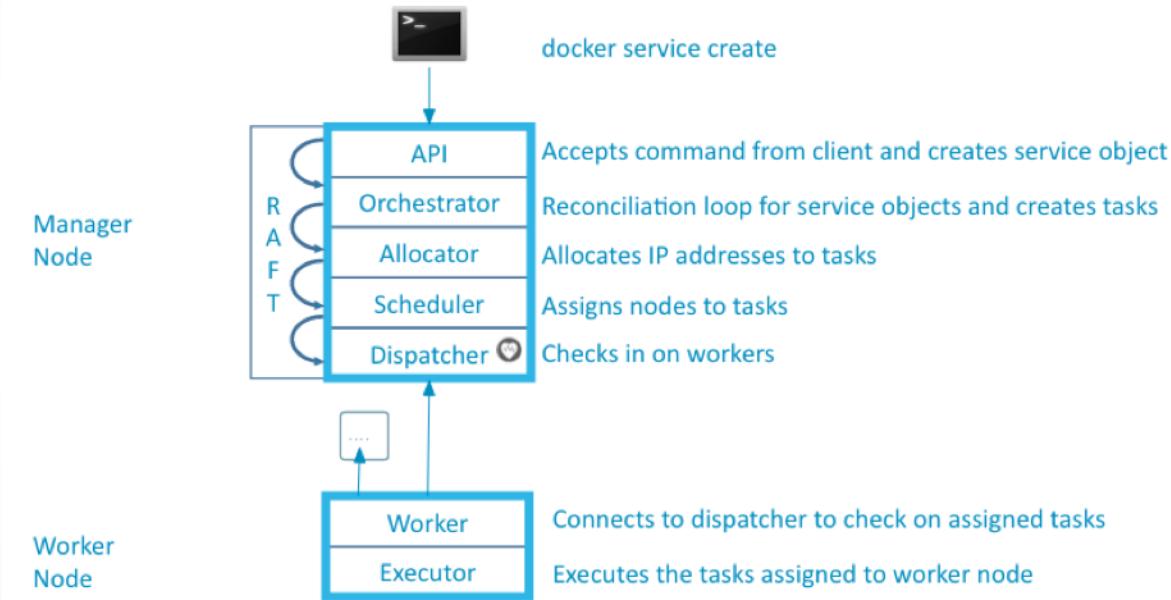
Swarm Mode Architecture

- Swarm Mode (build-in swarm) เป็นฟังก์ชันที่มาพร้อม docker-engine ตั้งแต่แรก และสามารถบริหารจัดการ Swarm ได้ด้วยตัวเองรวมถึง build-in TLS certificate เพื่อใช้ทำงานระหว่าง node ทันที (Security On the Box)
- Swarm Role
 - Manager Node
 - Worker Node

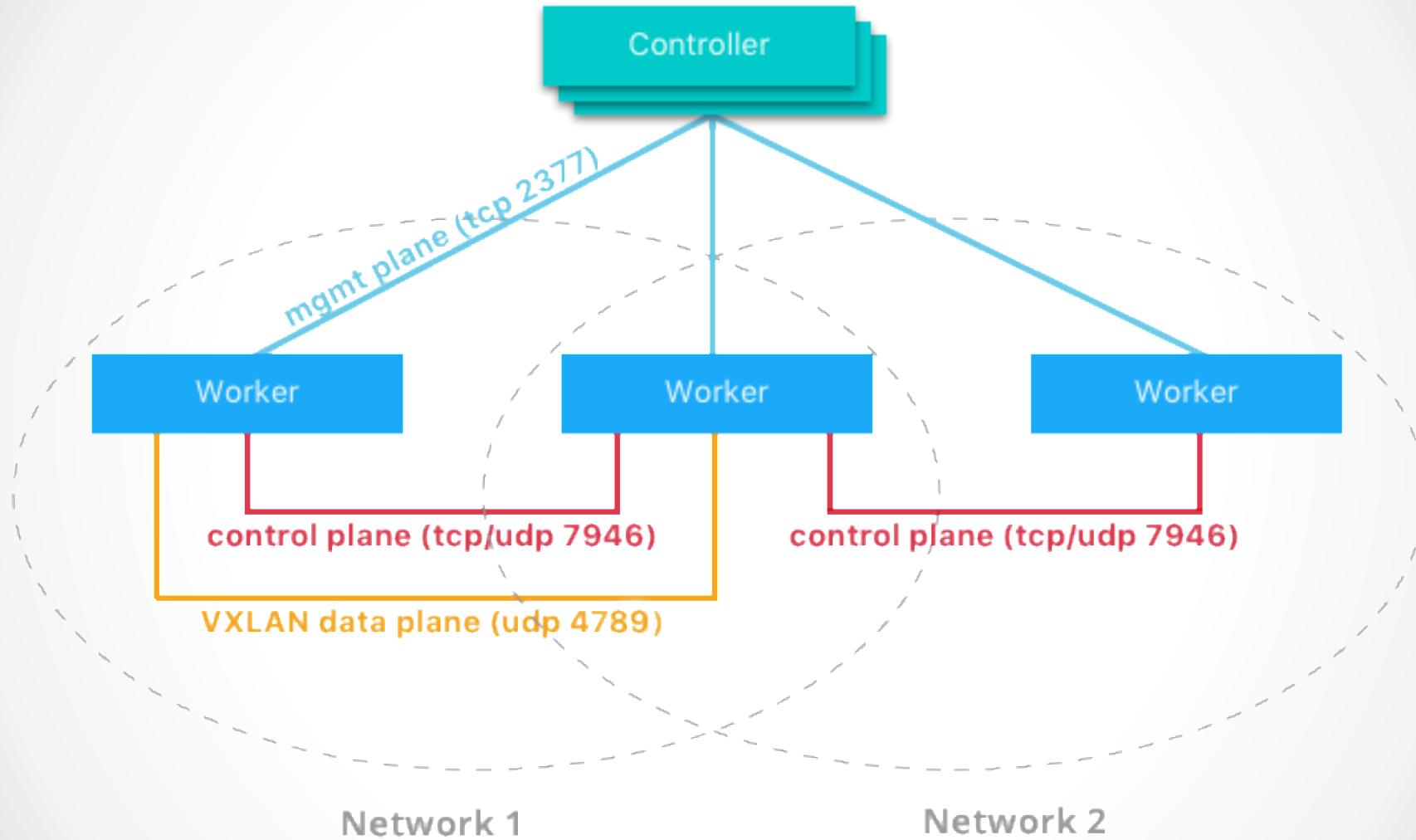


Swarm Mode Architecture

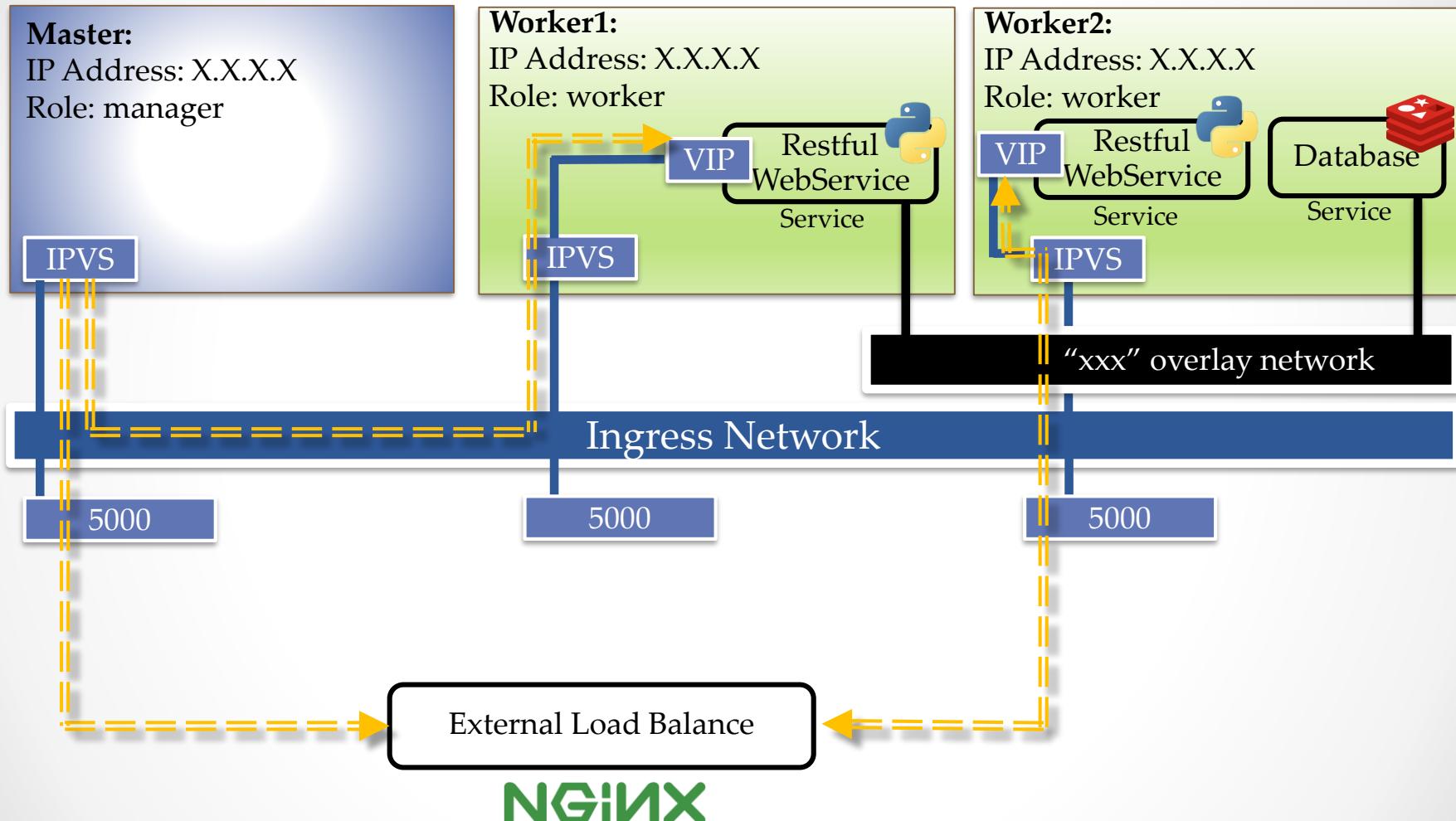
Node Breakdown



Swarm Mode Architecture



Swarm Mode Architecture



Swarm Mode Architecture

<https://blog.docker.com/2016/07/docker-built-in-orchestration-ready-for-production-docker-1-12-goes-ga/>

All
Engineering
Curated
Docker Weekly

The core team also wanted to give a special thanks to one of our external maintainers and Docker Captain, [Chanwit Kaewkasi](#), who through his own undertaking, drove the amazing [DockerSwarm2000](#) initiative which rallied the entire community around scaling an RC of 1.12 with swarm mode to nearly 2.4K nodes and just under 100K containers. This was achieved through our global community, who donated access to their machines in all shapes and sizes from bare metal, to Raspberry Pis, to various clouds to VMs from x86 architectures to ARM-based systems. Through this evaluation using live data, we identified that built-in orchestration in Docker has already—in its first release—doubled Docker's [orchestration scale](#) in just a half a year. While this validates the scalability of the architecture, there is still headroom for greater performance optimization in the future.



Nathan LeClaire @upthecyberpunks
@docker #swarm team forming like
Voltron to inspect the #DockerSwarm2K
results @stevvooe @aluzzardi
@mikegoelzer



Nathan LeClaire
@upthecyberpunks

@chanwit @dongluochen

Jul 30, 2016, 7:56 AM



Swarm Mode Architecture

swarmzilla / swarm3k

Code Issues 0 Pull requests 0 Projects 0 Pulse Graphs

SwarmZilla 3000 Collaborative Project

112 commits 1 branch 0 releases 21 contributors

Branch: master ▾ New pull request Find file Clone or download ▾

chanwit committed on GitHub got 3185 nodes Latest commit 7630dad a day ago

README.md got 3185 nodes a day ago

dashboard2.png Add files via upload 26 days ago

events.png Add files via upload 26 days ago

monitoring.md Update monitoring.md 2 days ago

README.md

Swarm3k - Friday, 28th October 2016 3.00PM UTC

SwarmZilla 3000 Collaborative Project

Swarm Mode Architecture

Name	Company	Number of Nodes Expected to Contribute
@chanwit	Suranaree University	100
@FlorianHeigl	my own boss	10
@jmairehenry	PetalMD	50
@everett_towes	Rackspace	100
@InetCloudArch	Internet Thailand	500
@squeaky_pl	n/a	10
@neverlock	Neverlock	10
@demonware	Demonware	1000
@sujaypillai	Jabil	50
@pilgrimstack	OVH	500
@ajeetsraina	Collabnix	10
@AorJoa	Aiyara Cluster	10
@f_soppelsa	Personal	20
@GroupSprint3r	SPRINT3r	630
@toughHQ	Personal	30
@mmonnaki	N/A	10
@zinuzoid	HotelQuickly	200
@_EthanHunt_	N/A	25
@packethost	Packet	100
@ContainerizeT	ContainerizeThis: The Conference	10
@_pascalandy	FirePress	10
@lucjuggery	TRAXxs	10
@alexellisuk	Personal	10
@svega	Huli	10
@BretFisher	Myself :)	20
@voodootikigod	Emerging Technology Advisors	100
@AlexPostID	Personal	20
@gianarb	ThumpFlow	50
@Rucknar	Personal	10
@lherrerabenitez	Personal	10
@abhisak	Nipa Technology	100
@djalal	NexwayGroup	30



Docker: The Next-Gen of Virtualization



K8S vs Docker what is the difference ?

Topic	K8S	Docker/Swarm
Architecture	Open-system (Base on cluster manager "Borg" for support complex workload)	Swarm: Proprietary of Docker product, "Easy to use", "Extend capability of Docker in cluster"
Operation command	Almost operate by "YAML" file (Declarative Command)	Almost operate by "command" (Imperative Command)
Unit of Work	Pods (Pods >= Container)	Container
How to Identify Work	"Label operation"	Docker: By container name Swarm: By service/stack name
Level of workload management	Service Level: (Simple) Replication Level: (Auto healing) Deployment Level: (Auto healing + Roll Update)	Docker: N/A Swarm: Service Level (Snag with service/stack)
Auto scaling	HPA (Horizontal Pods Scaling) base on CPU	No
Health check	Liveness & Readiness (Multi option to check application health)	Service health only

Swarm Init/Join

- เริ่มการสร้าง swarm ด้วยคำสั่ง docker swarm init [OPTIONS] เพื่อให้ docker เริ่มทำงานใน swarm mode

```
docker swarm init --listen-addr <ip address>:<port>
```

```
docker swarm init --advertise-addr 10.0.1.104:2377 --task-history-limit 2
```

```
[ubuntu@ip-10-0-1-104:~$ docker swarm init --advertise-addr 10.0.1.104:2377 --task-history-limit 2
Swarm initialized: current node (3tnvltyyohbekgygvsb1jjoyv) is now a manager.

To add a worker to this swarm, run the following command:

  docker swarm join --token SWMTKN-1-1qnpobqw0hilpc5i90sz803tlhyi1h4k84uok10y814bvhqf2-0ag405byoqpiqqhoy6r866tv 10.0.1.104:2377

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions.

[ubuntu@ip-10-0-1-104:~$ docker swarm join-token manager
To add a manager to this swarm, run the following command:

  docker swarm join --token SWMTKN-1-1qnpobqw0hilpc5i90sz803tlhyi1h4k84uok10y814bvhqf2-3u9rqxfwhbhw52qwqeeiewybl 10.0.1.104:2377
ubuntu@ip-10-0-1-104:~$ ]
```

Swarm Init/Join

- Option:
 - --cert-expiry duration: กำหนดระยะเวลาในการ expire certificate (default: 2160 hours = 90 Days)
 - --dispatcher-heartbeat duration: กำหนดช่วงเวลาในการ check heartbeat ภายใน swarm (Default: 5 seconds)
 - --external-ca value: กำหนดให้ใช้งาน CA certificate ภายนอกเพื่อทำ trust node ภายใน swarm
 - --force-new-cluster: บังคับการสร้าง swarm cluster ใหม่
 - --advertise-addr value: ระบุ ip address ที่ใช้คุยระหว่าง swarm node (default: 0.0.0.0:2377)
 - --task-history-limit <int>: กำหนดการจัดเก็บ history task ย้อนหลังบน swarm (default: 10)
 - --availability: <"active"/"pause"/"drain">
 - Etc.

Swarm Init/Join

- ทำการ join node เข้าไปยัง swarm cluster ด้วยคำสั่ง

```
docker swarm join --token <token> <ip of swarm manager: port>
```

```
...p_112018 — Terminal MAC Pro — -bash      ...ssh -i k8s_lab ubuntu@13.229.126.99      ...ssh -i k8s_lab ubuntu@13.229.98.3      ...sh -i k8s_lab ubuntu@18.136.196.130 +  
[ubuntu@ip-10-0-1-163:~$ docker swarm join --token SWMTKN-1-1qnpobqw0hilpc5i90sz803t1hyi1h4k84uok10y814bvhyqf2-0ag405byoqpiqqhoy6r866tv 10.0.1.104:2377 ]  
This node joined a swarm as a worker.  
ubuntu@ip-10-0-1-163:~$
```

```
...p_112018 — Terminal MAC Pro — -bash      ...ssh -i k8s_lab ubuntu@13.229.126.99      ...ssh -i k8s_lab ubuntu@13.229.98.3      ...sh -i k8s_lab ubuntu@18.136.196.130 +  
[ubuntu@ip-10-0-1-62:~$ docker swarm join --token SWMTKN-1-1qnpobqw0hilpc5i90sz803t1hyi1h4k84uok10y814bvhyqf2-0ag405byoqpiqqhoy6r866tv 10.0.1.104:2377 ]  
This node joined a swarm as a worker.  
ubuntu@ip-10-0-1-62:~$
```

Swarm Init/Join

- Option:
 - --token <value>: กำหนด token เพื่อใช้ในการ trust swarm cluster ร่วมกัน
 - --listen-addr value: ระบุ ip address ที่ใช้คุยระหว่าง swarm node (default: 0.0.0.0:2377)
 - --advertise-addr value: ระบุ ip address ที่ประกาศให้ผู้อื่นมาคุยในกรณีเป็น swarm-manager
 - --availability: <"active"/"pause"/"drain">
 - Etc.

Swarm Init/Join

- ตรวจสอบ Token ที่ใช้ในการ Join Swarm (Worker/Manager)

```
docker swarm join-token <option> <worker/manager>
```

```
...p_112018 — Terminal MAC Pro — -bash ...ssh -i k8s_lab ubuntu@13.229.126.99 ...ssh -i k8s_lab ubuntu@13.229.98.3 ...h -i k8s_lab ubuntu@13.229.98.3
```

```
ubuntu@ip-10-0-1-104:~$ docker swarm join-token worker
To add a worker to this swarm, run the following command:

  docker swarm join --token SWMTKN-1-1qnpobqw0hilpc5i90sz803tlhyi1h4k84uok10y814bvhyqf2-0ag405byoqpiqgqhoyle866tv 10.0.1.104:2377

ubuntu@ip-10-0-1-104:~$ docker swarm join-token manager
To add a manager to this swarm, run the following command:

  docker swarm join --token SWMTKN-1-1qnpobqw0hilpc5i90sz803tlhyi1h4k84uok10y814bvhyqf2-3u9rqxfwhbh52qwqeeiewybl 10.0.1.104:2377

ubuntu@ip-10-0-1-104:~$
```

- Option:
 - q, --q : แสดงเฉพาะ token เท่านั้น
 - rotate: ทำการเปลี่ยน token ใหม่

Swarm Init/Join

- ตรวจสอบ node ที่ทำงานภายใต้ swarm cluster ด้วยคำสั่ง docker node ls

```
docker node ls
```

```
...rkshop_112018 — Terminal MAC Pro — -bash ...4: ~ — ssh -i k8s_lab ubuntu@13.229.126.99 ...3: ~ — ssh -i k8s_lab ubuntu@13.229.98.3 ... ...~ — ssh -i k8s_lab ubuntu@18.136.199.154
[ubuntu@ip-10-0-1-104:~$ docker node ls
ID           HOSTNAME        STATUS      AVAILABILITY   MANAGER STATUS      ENGINE VERSION
4g6pqogaae5co8zxe8w6pxfd6   ip-10-0-1-62     Ready       Active
3tnvltyohbekgygvsb1jjovy *  ip-10-0-1-104    Ready       Active      Leader
ua0ajfjd4d1mksqzwr80oy2ru  ip-10-0-1-163    Ready       Active
ubuntu@ip-10-0-1-104:~$
```

Swarm Init/Join

- Option:
 - promote <ID>: ทำการ promote node จาก worker กลายเป็น manager
 - demote <ID>: ทำการ demote node จาก manager กลายเป็น worker ปกติ
 - Inspect <ID>: ตรวจสอบค่าคอนฟิกภูเรชั่นของ node
 - ls: แสดง node ทั้งหมดภายใน docker swarm cluster
 - ps <ID>: แสดงงานทั้งหมดที่รันอยู่ภายใน node
 - rm <ID>: ลบ node ออกจาก swarm cluster
 - update <ID>: ทำการ update node information
 - --availability <status>: อัพเดตสถานะของ node (active/pause/drain)
 - --label-add <value>: สร้าง custom label เพื่อใช้อ้างอิงให้กับ node สำหรับการรัน service
 - --label-remove <value>: ลบ custom label
 - --role <role>: role of node (worker/manager)

Workshop 2-4: Swarm Mode

- ทำการตรวจสอบเครื่องใน LAB ที่ได้รับมอบหมายแต่ละกลุ่มซึ่ง

Name: Training_DockerZerotoHero_StudentG1_1

(IP: 10.0.1.X), Role: Master

Name: Training_DockerZerotoHero_StudentG1_2

(IP: 10.0.1.X), Role: Worker1

Name: Training_DockerZerotoHero_StudentG1_3

(IP: 10.0.1.X), Role: Worker2

- เริ่ม initial swarm ที่เครื่อง Master

```
docker swarm init --secret labdocker --listen-addr <private ip  
of Master>:2377
```

Workshop 2-4: Swarm Mode

- ทำการ Join swarm-node1, swarm-node2 เข้าสู่ swarm cluster

```
docker swarm join --token SWMTKN-1-  
1qnpobqw0hilpc5i90sz803tlhyi1h4k84uok10y814bvhqf2-  
0ag405byoqpiqgqhoy6r866tv 10.0.1.104:2377
```

- ตรวจสอบสถานะของ Swarm หลังจาก join node เข้าไปเรียบร้อยแล้ว

```
docker node ls
```

ID	HOSTNAME	STATUS	AVAILABILITY	MANAGER STATUS	ENGINE VERSION
4g6pqogaae5co8zxe8w6pxfd6	ip-10-0-1-62	Ready	Active		18.06.1-ce
3tnvltyohbekgygvsb1jjoyv *	ip-10-0-1-104	Ready	Active	Leader	18.06.1-ce
ua0ajfjd4d1mksqzwr80oy2ru	ip-10-0-1-163	Ready	Active		18.06.1-ce

Swarm Service

- สำหรับการสั่งรัน container บน swarm ได้มีการเปลี่ยนแปลงรูปแบบใหม่ เช่นกัน โดยกำหนดการรัน container ถูกนิยามใหม่ในรูปแบบของคำสั่ง “docker service” เพื่อนิยามแทนการรันกลุ่มของ container แทนคำสั่ง “docker run” โดยเรารสามารถกำหนดค่าพารามิเตอร์ได้เช่นกัน

docker service <action> (create/inspect/tasks/scale/ls/rm/update)

Ex: docker service create -dt --name nodejs \
labdocker/alpineweb:latest node hello.js

```
...018 — Terminal MAC Pro — -bash ...-i k8s_lab ubuntu@13.229.126.99 ...-i k8s_lab ubuntu@13.229.98.3 ...-k8s_lab ubuntu@18.136.196.130 ...
ubuntu@ip-10-0-1-104:~$ docker service create --name nodejs -dt labdocker/alpineweb:latest node hello.js
4i4gu0o15f819jgohbgn994f7
ubuntu@ip-10-0-1-104:~$
```

Swarm Service

```
Options:
  --config config          Specify configurations to expose to the service
  --constraint list         Placement constraints
  --container-label list    Container labels
  --credential-spec credential-spec Credential spec for managed service account (Windows only)
-d, --detach                Exit immediately instead of waiting for the service to converge (default true)
  --dns list               Set custom DNS servers
  --dns-option list        Set DNS options
  --dns-search list        Set custom DNS search domains
  --endpoint-mode string   Endpoint mode (vip or dnsrr) (default "vip")
  --entrypoint command     Overwrite the default ENTRYPOINT of the image
-e, --env list              Set environment variables
  --env-file list          Read in a file of environment variables
  --group list             Set one or more supplementary user groups for the container
  --health-cmd string      Command to run to check health
  --health-interval duration Time between running the check (ms|s|m|h)
  --health-retries int     Consecutive failures needed to report unhealthy
  --health-start-period duration Start period for the container to initialize before counting retries towards unstable (ms|s|m|h)
  --health-timeout duration Maximum time to allow one check to run (ms|s|m|h)
  --help                   Print usage
  --host list              Set one or more custom host-to-IP mappings (host:ip)
  --hostname string         Container hostname
-l, --label list            Service labels
  --limit-cpu decimal       Limit CPUs
  --limit-memory bytes     Limit Memory
  --log-driver string       Logging driver for service
  --log-opt list            Logging driver options
  --mode string             Service mode (replicated or global) (default "replicated")
  --mount mount             Attach a filesystem mount to the service
  --name string             Service name
```

- Reference: https://docs.docker.com/engine/reference/commandline/service_create/

Swarm Service

```
--name string          Service name
--network network       Network attachments
--no-healthcheck        Disable any container-specified HEALTHCHECK
--no-resolve-image      Do not query the registry to resolve image digest and supported platforms
--placement-pref pref   Add a placement preference
-p, --publish port     Publish a port as a node port
-q, --quiet             Suppress progress output
--read-only             Mount the container's root filesystem as read only
--replicas uint         Number of tasks
--reserve-cpu decimal   Reserve CPUs
--reserve-memory bytes  Reserve Memory
--restart-condition string
--restart-delay duration
--restart-max-attempts uint
--restart-window duration
--rollback-delay duration
--rollback-failure-action string
--rollback-max-failure-ratio float
--rollback-monitor duration
--rollback-order string
--rollback-parallelism uint
--secret secret
--stop-grace-period duration
--stop-signal string
-t, --tty               Allocate a pseudo-TTY
--update-delay duration
--update-failure-action string
--update-max-failure-ratio float
--update-monitor duration
--update-order string
--update-parallelism uint
-u, --user string        Username or UID (format: <name|uid>[:<group|gid>])
--with-registry-auth     Send registry authentication details to swarm agents
-w, --workdir string     Working directory inside the container
```

- Reference: https://docs.docker.com/engine/reference/commandline/service_create/

Swarm Service

docker service scale nodejs=5

```
...shop_112018 — Terminal MAC Pro — -bash      ...: ~ — ssh -i k8s_lab ubuntu@13.229.126.99      ...: ~ — ssh -i k8s_lab ubuntu@13.229.98.3 ...      ...— ssh -i k8s_lab ubuntu@18.136.196.130 ...
ubuntu@ip-10-0-1-104:~$ docker service create --name nodejs -dt labdocker/alpineweb:latest node hello.js
4i4gu0o15f819jgohbgn994f7
ubuntu@ip-10-0-1-104:~$ docker service ls
[{"ID": "4i4gu0o15f81", "NAME": "nodejs", "MODE": "replicated", "REPLICAS": "1/1", "IMAGE": "labdocker/alpineweb:latest", "PORTS": ""}]
ubuntu@ip-10-0-1-104:~$ docker service ps nodejs
[{"ID": "h6u6lt7o564f", "NAME": "nodejs.1", "IMAGE": "labdocker/alpineweb:latest", "NODE": "ip-10-0-1-104", "DESIRED STATE": "Running", "CURRENT STATE": "Running about a minute ago", "ERROR": "", "PORTS": ""}]
ubuntu@ip-10-0-1-104:~$ docker service scale nodejs=5
nodejs scaled to 5
overall progress: 5 out of 5 tasks
1/5: running [=====>]
2/5: running [=====>]
3/5: running [=====>]
4/5: running [=====>]
5/5: running [=====>]
verify: Service converged
ubuntu@ip-10-0-1-104:~$ docker service ps nodejs
[{"ID": "h6u6lt7o564f", "NAME": "nodejs.1", "IMAGE": "labdocker/alpineweb:latest", "NODE": "ip-10-0-1-104", "DESIRED STATE": "Running", "CURRENT STATE": "Running 2 minutes ago", "ERROR": "", "PORTS": ""}, {"ID": "r5j8wpaew0tm", "NAME": "nodejs.2", "IMAGE": "labdocker/alpineweb:latest", "NODE": "ip-10-0-1-104", "DESIRED STATE": "Running", "CURRENT STATE": "Running 26 seconds ago", "ERROR": "", "PORTS": ""}, {"ID": "w66ihyd8kbs2", "NAME": "nodejs.3", "IMAGE": "labdocker/alpineweb:latest", "NODE": "ip-10-0-1-163", "DESIRED STATE": "Running", "CURRENT STATE": "Running 16 seconds ago", "ERROR": "", "PORTS": ""}, {"ID": "gjgkorz84g7g", "NAME": "nodejs.4", "IMAGE": "labdocker/alpineweb:latest", "NODE": "ip-10-0-1-62", "DESIRED STATE": "Running", "CURRENT STATE": "Running 16 seconds ago", "ERROR": "", "PORTS": ""}, {"ID": "jubxx7bup73b", "NAME": "nodejs.5", "IMAGE": "labdocker/alpineweb:latest", "NODE": "ip-10-0-1-62", "DESIRED STATE": "Running", "CURRENT STATE": "Running 16 seconds ago", "ERROR": "", "PORTS": ""}]
ubuntu@ip-10-0-1-104:~$
```

Swarm Service

```
docker service update -dt --reserve-cpu 1 --limit-cpu 1 nodejs
```

```
docker service inspect nodejs | more
```

```
...shop_112018 — Terminal MAC Pro — -bash      ...: ~ — ssh -i k8s_lab ubuntu@13.229.126.99      ...: ~ — ssh -i k8s_lab ubuntu@13.229.98.3 ...      ... — ssh

[ubuntu@ip-10-0-1-104:~$ docker service update -dt --reserve-cpu 1 --limit-cpu 1 nodejs
nodejs
[ubuntu@ip-10-0-1-104:~$ docker service inspect nodejs | more
[
  {
    "ID": "4i4gu0o15f819jgohbgn994f7",
    "Version": {
      "Index": 44
    },
    "CreatedAt": "2018-10-21T16:25:07.16972436Z",
    "UpdatedAt": "2018-10-21T16:29:50.119830764Z",
    "Spec": {
      "Name": "nodejs",
      "Labels": {},
      "TaskTemplate": {
        "ContainerSpec": {
          "Image": "labdocker/alpineweb:latest@sha256:a7aa70c78120ed126487aecbf49db1054f1ff131858516c91137c3accbb4a333",
          "Args": [
            "node",
            "hello.js"
          ],
          "Init": false,
          "TTY": true,
          "StopGracePeriod": 10000000000,
          "DNSConfig": {}
        }
      }
    }
  }
]
```

Swarm Service

```
        "Spec": {}  
    },  
    "UpdateStatus": {  
        "State": "updating",  
        "StartedAt": "2018-10-21T16:29:50.119815453Z",  
        "Message": "update in progress"  
    }  
}  
]  
ubuntu@ip-10-0-1-104:~$ █
```

```
...shop_112018 — Terminal MAC Pro — -bash          ...: ~ — ssh -i k8s_lab ubuntu@13.229.126.99          ...: ~ — ssh -i k8s_lab ubuntu@13.229.98.3 ...  
[ubuntu@ip-10-0-1-104:~$ docker service inspect nodejs | grep update  
    "Message": "update completed"  
ubuntu@ip-10-0-1-104:~$ █
```

Swarm Service

docker service ps nodejs

```
...shop_112018 — Terminal MAC Pro — -bash      ...: ~ — ssh -i k8s_lab ubuntu@13.229.126.99      ...: ~ — ssh -i k8s_lab ubuntu@13.229.98.3 ...      ... — ssh -i k8s_lab ubuntu@18.136.196.130 ...
[ubuntu@ip-10-0-1-104:~$ docker service ps nodejs
ID          NAME        IMAGE           NODE        DESIRED STATE   CURRENT STATE        ERROR        PORTS
lchljj6puie2    nodejs.1    labdocker/alpineweb:latest ip-10-0-1-104  Running        Running about a minute ago
h6u6lt7o564f    \_ nodejs.1    labdocker/alpineweb:latest ip-10-0-1-104  Shutdown       Shutdown about a minute ago
ubuntu@ip-10-0-1-104:~$
```

docker service rm nodejs

```
...shop_112018 — Terminal MAC Pro — -bash      ...: ~ — ssh -i k8s_lab ubuntu@13.229.126.99      ...: ~ — ssh -i k8s_lab ubuntu@13.229.98.3 ...      ... — ssh -i k8s_lab ubuntu@18.136.196.130 ...
[ubuntu@ip-10-0-1-104:~$ docker service ps nodejs
ID          NAME        IMAGE           NODE        DESIRED STATE   CURRENT STATE        ERROR        PORTS
lchljj6puie2    nodejs.1    labdocker/alpineweb:latest ip-10-0-1-104  Running        Running 2 minutes ago
h6u6lt7o564f    \_ nodejs.1    labdocker/alpineweb:latest ip-10-0-1-104  Shutdown       Shutdown 2 minutes ago
[ubuntu@ip-10-0-1-104:~$ docker service rm nodejs
nodejs
[ubuntu@ip-10-0-1-104:~$ docker service ps nodejs
no such service: nodejs
ubuntu@ip-10-0-1-104:~$
```

Swarm Service

```
Usage: docker service update [OPTIONS] SERVICE
```

Update a service

Options:

--args command	Service command args
--config-add config	Add or update a config file on a service
--config-rm list	Remove a configuration file
--constraint-add list	Add or update a placement constraint
--constraint-rm list	Remove a constraint
--container-label-add list	Add or update a container label
--container-label-rm list	Remove a container label by its key
--credential-spec credential-spec	Credential spec for managed service account (Windows only)
-d, --detach	Exit immediately instead of waiting for the service to converge (default true)
--dns-add list	Add or update a custom DNS server
--dns-option-add list	Add or update a DNS option
--dns-option-rm list	Remove a DNS option
--dns-rm list	Remove a custom DNS server
--dns-search-add list	Add or update a custom DNS search domain
--dns-search-rm list	Remove a DNS search domain
--endpoint-mode string	Endpoint mode (vip or dnsrr)
--entrypoint command	Overwrite the default ENTRYPOINT of the image
--env-add list	Add or update an environment variable
--env-rm list	Remove an environment variable
--force	Force update even if no changes require it
--group-add list	Add an additional supplementary user group to the container
--group-rm list	Remove a previously added supplementary user group from the container
--health-cmd string	Command to run to check health

- Reference: https://docs.docker.com/engine/reference/commandline/service_update/

Swarm Service

```
--health-interval duration          Time between running the check (ms|s|m|h)
--health-retries int                Consecutive failures needed to report unhealthy
--health-start-period duration     Start period for the container to initialize before counting retries towards unstable (ms|s|m|h)
--health-timeout duration         Maximum time to allow one check to run (ms|s|m|h)
--help                            Print usage
--host-add list                   Add or update a custom host-to-IP mapping (host:ip)
--host-rm list                    Remove a custom host-to-IP mapping (host:ip)
--hostname string                 Container hostname
--image string                     Service image tag
--label-add list                  Add or update a service label
--label-rm list                   Remove a label by its key
--limit-cpu decimal               Limit CPUs
--limit-memory bytes             Limit Memory
--log-driver string              Logging driver for service
--log-opt list                    Logging driver options
--mount-add mount                Add or update a mount on a service
--mount-rm list                  Remove a mount by its target path
--network-add network            Add a network
--network-rm list                Remove a network
--no-healthcheck                Disable any container-specified HEALTHCHECK
--no-resolve-image               Do not query the registry to resolve image digest and supported platforms
--placement-pref-add pref       Add a placement preference
--placement-pref-rm pref        Remove a placement preference
--publish-add port               Add or update a published port
--publish-rm port                Remove a published port by its target port
-q, --quiet                      Suppress progress output
--read-only                      Mount the container's root filesystem as read only
--replicas uint                  Number of tasks
--reserve-cpu decimal            Reserve CPUs
--reserve-memory bytes           Reserve Memory
```

- Reference: https://docs.docker.com/engine/reference/commandline/service_update/

Swarm Service

```
--read-only                                Mount the container's root filesystem as read only
--replicas uint                             Number of tasks
--reserve-cpu decimal                      Reserve CPUs
--reserve-memory bytes                     Reserve Memory
--restart-condition string                 Restart when condition is met ("none"|"on-failure"|"any")
--restart-delay duration                  Delay between restart attempts (ns|us|ms|s|m|h)
--restart-max-attempts uint                Maximum number of restarts before giving up
--restart-window duration                 Window used to evaluate the restart policy (ns|us|ms|s|m|h)
--rollback                                    Rollback to previous specification
--rollback-delay duration                 Delay between task rollbacks (ns|us|ms|s|m|h)
--rollback-failure-action string          Action on rollback failure ("pause"|"continue")
--rollback-max-failure-ratio float        Failure rate to tolerate during a rollback
--rollback-monitor duration               Duration after each task rollback to monitor for failure (ns|us|ms|s|m|h)
--rollback-order string                   Rollback order ("start-first"|"stop-first")
--secret-add secret                      Add or update a secret on a service
--secret-rm list                         Remove a secret
--stop-grace-period duration             Time to wait before force killing a container (ns|us|ms|s|m|h)
--stop-signal string                     Signal to stop the container
-t, --tty                                  Allocate a pseudo-TTY
--update-delay duration                  Delay between updates (ns|us|ms|s|m|h)
--update-failure-action string          Action on update failure ("pause"|"continue"|"rollback")
--update-max-failure-ratio float        Failure rate to tolerate during an update
--update-monitor duration               Duration after each task update to monitor for failure (ns|us|ms|s|m|h)
--update-order string                   Update order ("start-first"|"stop-first")
--update-parallelism uint                Maximum number of tasks updated simultaneously (0 to update all at once)
-u, --user string                        Username or UID (format: <name|uid>[:<group|gid>])
--with-registry-auth                    Send registry authentication details to swarm agents
-w, --workdir string                     Working directory inside the container
```

- Reference: https://docs.docker.com/engine/reference/commandline/service_update/

Workshop 2-4-1: Swarm Service

- สร้าง service nodejs เพื่อเริ่มทำงานบน swarm

```
docker service create -dt --name nodejs \
labdocker/alpineweb:latest node hello.js
```

- ตรวจสอบสถานะของ service หลังจากการสร้าง

```
docker service ls
docker service ps nodejs
```

```
...shop_112018 — Terminal MAC Pro — -bash | ... — ssh -i k8s_lab ubuntu@13.229.126.99 | ...: ~ — ssh -i k8s_lab ubuntu@13.229.98.3 ... | ...— ssh -i k8s_lab ubuntu@18.136.196.130 ...
ubuntu@ip-10-0-1-104:~$ docker service create -dt --name nodejs \
> labdocker/alpineweb:latest node hello.js
sm7d6yzuedh7kpv4o3zv8tyx1
ubuntu@ip-10-0-1-104:~$ 
[ubuntu@ip-10-0-1-104:~$ docker service ls
ID          NAME      MODE      REPLICAS      IMAGE
sm7d6yzuedh7    nodejs     replicated   1/1        labdocker/alpineweb:latest
[ubuntu@ip-10-0-1-104:~$ docker service ps nodejs
ID          NAME      IMAGE      NODE      DESIRED STATE     CURRENT STATE      ERROR      PORTS
pq53791w4q8a    nodejs.1  labdocker/alpineweb:latest  ip-10-0-1-104  Running   Running 7 seconds ago
ubuntu@ip-10-0-1-104:~$
```

Orchestrator Assignment

- Swarm สามารถควบคุมการจ่ายงานให้จาก manager ไปยัง worker ได้หลากหลายเทคนิค เพื่อตอบสนองความต้องการของผู้ใช้งาน
- Docker stack deploy (Compose version 3.4)
 - Replicated (Specific number of container by manual)
 - Global (1 container per node in swarm)
- Assign by default node constrain (--constrain)
 - node.id
 - node.hostname
 - node.role
 - node.labels (user define label)
 - engine.labels
- Assign by service placement preference (--placement-pref)
 - Spread on node.label (user define label)

Orchestrator Assignment

- node constrain
 - Running with constrain with node
 - node.id

```
[ubuntu@ip-10-0-1-104:~$ docker node ls
ID           HOSTNAME   STATUS    AVAILABILITY   MANAGER STATUS      ENGINE VERSION
4g6pqogaae5co8zxe8w6pxfd6   ip-10-0-1-62  Ready     Active          18.06.1-ce
3tnvltyyohbekgygvsb1jjovy * ip-10-0-1-104 Ready     Active          Leader
ua0ajfjd4d1mksqzwr80oy2ru ip-10-0-1-163 Ready     Active          18.06.1-ce
ubuntu@ip-10-0-1-104:~$ docker service create --dt --constraint 'node.id==3tnvltyyohbekgygvsb1jjovy' \
> --name nodejs --replicas=5 labdocker/alpineweb:latest node hello.js
v52nswafowr62xq6ib9rpzu7r
[ubuntu@ip-10-0-1-104:~$ docker service ps nodejs
ID           NAME      IMAGE      NODE      DESIRED STATE   CURRENT STATE      ERROR      PORTS
TS
k1jt8r2sfahw  nodejs.1  labdocker/alpineweb:latest  ip-10-0-1-104  Running        Running less than a second ago
p6rwlezip0i8  nodejs.2  labdocker/alpineweb:latest  ip-10-0-1-104  Running        Running less than a second ago
17z39pfjyb5f  nodejs.3  labdocker/alpineweb:latest  ip-10-0-1-104  Running        Running less than a second ago
oadfa4hqb5q5  nodejs.4  labdocker/alpineweb:latest  ip-10-0-1-104  Running        Running less than a second ago
i2g4d3bkkc70  nodejs.5  labdocker/alpineweb:latest  ip-10-0-1-104  Running        Running less than a second ago
```

Orchestrator Assignment

- Remove constrain/Add constrain
 - node.hostname

```
...orkshop_112018 — Terminal MAC Pro — -bash ...04: ~ — ssh -i k8s_lab ubuntu@13.229.126.99 ...63: ~ — ssh -i k8s_lab ubuntu@13.229.98.3 ... ...~ — ssh -i k8s_lab ubuntu@18.136.196.130 ...+ [ubuntu@ip-10-0-1-104:~$ docker service update --dt --constraint-rm 'node.id==3tnvltyyohbekgvgvsb1jjovy' --constraint-add 'node.hostname!=ip-10-0-1-104' nodejs nodejs
[ubuntu@ip-10-0-1-104:~$ docker service ps nodejs
ID          NAME      IMAGE          NODE      DESIRED STATE     CURRENT STATE      ERROR      PORTS
K1jt8r2sfahw  nodejs.1    labdocker/alpineweb:latest ip-10-0-1-104  Running   Running 3 minutes ago
or0dudbrhex2  nodejs.2    labdocker/alpineweb:latest ip-10-0-1-163  Running   Running 2 seconds ago
p6rwlezix0i8  \_ nodejs.2    labdocker/alpineweb:latest ip-10-0-1-104  Shutdown  Shutdown 3 seconds ago
n9kxw9v7efno  nodejs.3    labdocker/alpineweb:latest ip-10-0-1-62   Ready     Ready 2 seconds ago
17z39pfjyb5f  \_ nodejs.3    labdocker/alpineweb:latest ip-10-0-1-104  Shutdown  Running 2 seconds ago
oadfa4hqb5q5  nodejs.4    labdocker/alpineweb:latest ip-10-0-1-104  Running   Running 3 minutes ago
i2g4d3bkkc70  nodejs.5    labdocker/alpineweb:latest ip-10-0-1-104  Running   Running 3 minutes ago
...orkshop_112018 — Terminal MAC Pro — -bash ...04: ~ — ssh -i k8s_lab ubuntu@13.229.126.99 ...63: ~ — ssh -i k8s_lab ubuntu@13.229.98.3 ... ...~ — ssh -i k8s_lab ubuntu@18.136.196.130 ...
[ubuntu@ip-10-0-1-104:~$ docker service inspect nodejs |grep update
  "Message": "update completed"
[ubuntu@ip-10-0-1-104:~$ docker service ps nodejs
ID          NAME      IMAGE          NODE      DESIRED STATE     CURRENT STATE      ERROR      PORTS
77m2kx4lvs8  nodejs.1    labdocker/alpineweb:latest ip-10-0-1-62   Running   Running 24 seconds ago
K1jt8r2sfahw  \_ nodejs.1    labdocker/alpineweb:latest ip-10-0-1-104  Shutdown  Shutdown 24 seconds ago
or0dudbrhex2  nodejs.2    labdocker/alpineweb:latest ip-10-0-1-163  Running   Running about a minute ago
p6rwlezix0i8  \_ nodejs.2    labdocker/alpineweb:latest ip-10-0-1-104  Shutdown  Shutdown about a minute ago
n9kxw9v7efno  nodejs.3    labdocker/alpineweb:latest ip-10-0-1-62   Running   Running 48 seconds ago
17z39pfjyb5f  \_ nodejs.3    labdocker/alpineweb:latest ip-10-0-1-104  Shutdown  Shutdown 48 seconds ago
p7iek0f77pji  nodejs.4    labdocker/alpineweb:latest ip-10-0-1-163  Running   Running 12 seconds ago
oadfa4hqb5q5  \_ nodejs.4    labdocker/alpineweb:latest ip-10-0-1-104  Shutdown  Shutdown 13 seconds ago
0rcd8pgpe4ex nodejs.5    labdocker/alpineweb:latest ip-10-0-1-163  Running   Running 36 seconds ago
i2g4d3bkkc70  \_ nodejs.5    labdocker/alpineweb:latest ip-10-0-1-104  Shutdown  Shutdown 36 seconds ago
ubuntu@ip-10-0-1-104:~$ ]
```

Workshop 2-4: node constrain

- สร้าง service nodejs เพื่อเริ่มทำงานบน swarm

```
docker service create -dt --constraint  
'node.id==6bei4mbj5pd7yduyhp375b7z4' --name nodejs --  
replicas=5 labdocker/alpineweb:latest node hello.js
```

```
...shop_112018 — Terminal MAC Pro — bash          .ssh — ubuntu@ip-10-0-1-104: ~ — ssh -i k8s_lab ubuntu@13.229.126.99 — 165x16  
...: ~ — ssh -i k8s_lab ubuntu@13.229.126.99      ...: ~ — ssh -i k8s_lab ubuntu@13.229.98.3 ...      ... — ssh -i k8s_lab ubuntu@18.136.196.130 ...  
ubuntu@ip-10-0-1-104:~$ docker node ls  
ID           HOSTNAME   STATUS    AVAILABILITY  MANAGER STATUS   ENGINE VERSION  
4g6pqogaae5co8zxe8w6pxfd6  ip-10-0-1-62  Ready     Active        18.06.1-ce  
3tnvltyyohbekgygvsb1jjovy * ip-10-0-1-104  Ready     Active        Leader  
ua0ajfjd4d1mksqzwr80oy2ru ip-10-0-1-163  Ready     Active        18.06.1-ce  
ubuntu@ip-10-0-1-104:~$ docker service create -dt --constraint 'node.id==3tnvltyyohbekgygvsb1jjovy' \  
> --name nodejs --replicas=5 labdocker/alpineweb:latest node hello.js  
v52nswafowr62xq6ib9rpzu7r  
ubuntu@ip-10-0-1-104:~$ docker service ps nodejs  
ID           NAME      IMAGE      NODE      DESIRED STATE   CURRENT STATE      ERROR      PORTS  
TS  
k1jt8r2sfahw  nodejs.1  labdocker/alpineweb:latest  ip-10-0-1-104  Running    Running less than a second ago  
p6rwlezix0i8  nodejs.2  labdocker/alpineweb:latest  ip-10-0-1-104  Running    Running less than a second ago  
17z39pfjyb5f  nodejs.3  labdocker/alpineweb:latest  ip-10-0-1-104  Running    Running less than a second ago  
oadfa4hqb5q5  nodejs.4  labdocker/alpineweb:latest  ip-10-0-1-104  Running    Running less than a second ago  
i2g4d3bkkc70  nodejs.5  labdocker/alpineweb:latest  ip-10-0-1-104  Running    Running less than a second ago
```

Orchestrator Assignment

- node constrain (custom label)

```
docker node update --label-add 'storage=sas' swarm-mng  
docker node update --label-add 'storage=nvdimm' swarm-node1  
docker node update --label-add 'storage=sata' swarm-node2
```

```
docker node inspect swarm-mng|grep storage  
docker node inspect swarm-node1|grep storage  
docker node inspect swarm-node2|grep storage
```

```
...orkshop_112018 — Terminal MAC Pro — -bash ...04: ~ — ssh -i k8s_lab ubuntu@13.229.126.99 ...63: ~ — ssh -i k8s_lab ubuntu@13.229.98.3 ...  
[ubuntu@ip-10-0-1-104:~$ docker node ls  
ID HOSTNAME STATUS AVAILABILITY MANAGER STATUS ENGINE VERSION  
4g6pqogaae5co8zxe8w6pxfd6 ip-10-0-1-62 Ready Active  
3tnvltyyohbekgygvbs1jjovy * ip-10-0-1-104 Ready Active Leader 18.06.1-ce  
ua0ajfjd4d1mksqzwr80oy2ru ip-10-0-1-163 Ready Active 18.06.1-ce  
[ubuntu@ip-10-0-1-104:~$ docker node update --label-add 'storage=sas' ip-10-0-1-104  
ip-10-0-1-104  
[ubuntu@ip-10-0-1-104:~$ docker node update --label-add 'storage=nvdim' ip-10-0-1-163  
ip-10-0-1-163  
[ubuntu@ip-10-0-1-104:~$ docker node update --label-add 'storage=sata' ip-10-0-1-62  
ip-10-0-1-62  
[ubuntu@ip-10-0-1-104:~$ docker node inspect ip-10-0-1-104|grep storage  
"storage": "sas"  
[ubuntu@ip-10-0-1-104:~$ docker node inspect ip-10-0-1-163|grep storage  
"storage": "nvdim"  
[ubuntu@ip-10-0-1-104:~$ docker node inspect ip-10-0-1-62|grep storage  
"storage": "sata"  
ubuntu@ip-10-0-1-104:~$
```

Orchestrator Assignment

- custom label

```
docker service create --name xxx --add-label 'xxx=xxx'
```

```
docker service create -dt --constraint 'node.labels.storage==sas' --name nodejs --replicas=5 labdocker/alpineweb:latest node hello.js
```

```
...orkshop_112018 — Terminal MAC Pro — bash | ...04: ~ — ssh -i k8s_lab ubuntu@13.229.126.99 | ...63: ~ — ssh -i k8s_lab ubuntu@13.229.98.3 ... | ...~ — ssh -i k8s_lab ubuntu@18.136.112.104

[ubuntu@ip-10-0-1-104:~$ docker service create -dt --constraint 'node.labels.storage==sas' --name nodejs --replicas=5 labdocker/alpineweb:latest node hello.js
pl6f8550yokx23bgm3l2txo91
[ubuntu@ip-10-0-1-104:~$ docker service ps nodejs
ID          NAME        IMAGE          NODE          DESIRED STATE    CURRENT STATE      ERROR          PORTS
vkh4tyggb2z1  nodejs.1  labdocker/alpineweb:latest  ip-10-0-1-104  Running        Running  7 seconds ago
7hko49pz2o3a  nodejs.2  labdocker/alpineweb:latest  ip-10-0-1-104  Running        Running  6 seconds ago
sl1cqvqdnitn  nodejs.3  labdocker/alpineweb:latest  ip-10-0-1-104  Running        Running  6 seconds ago
3axgwaukkmgt  nodejs.4  labdocker/alpineweb:latest  ip-10-0-1-104  Running        Running  7 seconds ago
i2cwhj6qmcbu  nodejs.5  labdocker/alpineweb:latest  ip-10-0-1-104  Running        Running  7 seconds ago
ubuntu@ip-10-0-1-104:~$
```

Workshop 2-4: custom label

- สร้าง docker service โดยระบุ constrain จาก custom label ที่สร้างขึ้น

```
docker service create -dt --constraint  
'node.labels.storage==sas' --name nodejs --replicas=5  
labdocker/alpineweb:latest node hello.js
```

```
...orkshop_112018 — Terminal MAC Pro — bash          ...04: ~ — ssh -i k8s_lab ubuntu@13.229.126.99          ...63: ~ — ssh -i k8s_lab ubuntu@13.229.98.3 ...          ...~ — ssh -i k8s_lab ubuntu@18.136.196.  
ubuntu@ip-10-0-1-104:~$ docker service create -dt  --constraint 'node.labels.storage==sas' --name nodejs --replicas=5 labdocker/alpineweb:latest node hello.js  
p16f8550yokx23bgm3l2txo91  
[ubuntu@ip-10-0-1-104:~$ docker service ps nodejs  
ID           NAME      IMAGE          NODE          DESIRED STATE     CURRENT STATE      ERROR          PORTS  
vkh4tyggb2z1  nodejs.1  labdocker/alpineweb:latest  ip-10-0-1-104  Running        Running 7 seconds ago  
7hko49pz2o3a  nodejs.2  labdocker/alpineweb:latest  ip-10-0-1-104  Running        Running 6 seconds ago  
s11cqvldnitn nodejs.3  labdocker/alpineweb:latest  ip-10-0-1-104  Running        Running 6 seconds ago  
3axgwaukkmg7 nodejs.4  labdocker/alpineweb:latest  ip-10-0-1-104  Running        Running 7 seconds ago  
i2cwhj6qmbcu  nodejs.5  labdocker/alpineweb:latest  ip-10-0-1-104  Running        Running 7 seconds ago  
[ubuntu@ip-10-0-1-104:~$ docker node ls  
ID           HOSTNAME   STATUS          AVAILABILITY    MANAGER STATUS      ENGINE VERSION  
4g6pqogaae5co8zxe8w6pxfd6  ip-10-0-1-62  Ready          Active          Active            18.06.1-ce  
3tnvltyyohbekgygvb1jjovy * ip-10-0-1-104  Ready          Active          Leader           18.06.1-ce  
ua0ajfjd4d1mksqzwr80oy2ru  ip-10-0-1-163  Ready          Active          Active           18.06.1-ce  
[ubuntu@ip-10-0-1-104:~$
```

Orchestrator Assignment

- service placement preference (--placement-pref)

```
docker node update --label-add 'physical=DELLPE820' swarm-mng  
docker node update --label-add 'physical=DELLPE820' swarm-node1  
docker node update --label-add 'physical=HP' swarm-node2
```

```
docker node inspect swarm-mng | grep physical  
docker node inspect swarm-node1 | grep physical  
docker node inspect swarm-node2 | grep physical
```

```
...orkshop_112018 — Terminal MAC Pro — -bash ...04: ~ — ssh -i k8s_lab ubuntu@13.229.126.99 ...63: ~ — ssh -i k8s_lab ubuntu@13.229.98.3 ...  
[ubuntu@ip-10-0-1-104:~$ docker node ls  
ID HOSTNAME STATUS AVAILABILITY MANAGER STATUS ENGINE VERSION  
4g6pqogaae5co8zxe8w6pxfd6 ip-10-0-1-62 Ready Active  
3tnvltyyohbekgygvbs1jjoyv * ip-10-0-1-104 Ready Active Leader 18.06.1-ce  
ua0ajfjd4d1mksqzwr80oy2ru ip-10-0-1-163 Ready Active 18.06.1-ce  
[ubuntu@ip-10-0-1-104:~$ docker node update --label-add 'physical=DELLPE820' ip-10-0-1-104  
ip-10-0-1-104  
[ubuntu@ip-10-0-1-104:~$ docker node update --label-add 'physical=DELLPE820' ip-10-0-1-163  
ip-10-0-1-163  
[ubuntu@ip-10-0-1-104:~$ docker node update --label-add 'physical=HP' ip-10-0-1-62  
ip-10-0-1-62  
[ubuntu@ip-10-0-1-104:~$ docker node inspect ip-10-0-1-104|grep physical  
"physical": "DELLPE820",  
[ubuntu@ip-10-0-1-104:~$ docker node inspect ip-10-0-1-163|grep physical  
"physical": "DELLPE820",  
[ubuntu@ip-10-0-1-104:~$ docker node inspect ip-10-0-1-62|grep physical  
"physical": "HP",  
ubuntu@ip-10-0-1-104:~$
```

Orchestrator Assignment

- service placement preference

```
docker service create -dt --placement-pref 'spread=node.labels.physical' \
--name nodejs --replicas=10 labdocker/alpineweb:latest node hello.js
```

```
...orkshop_112018 — Terminal MAC Pro — bash ...04: ~ — ssh -i k8s_lab ubuntu@13.229.126.99 ...63: ~ — ssh -i k8s_lab ubuntu@13.229.98.3 ... ~ — ssh -i k8s_lab ubuntu@18.136.104.104:~$ docker service create -dt --placement-pref 'spread=node.labels.physical' \
[> --name nodejs --replicas=10 labdocker/alpineweb:latest node hello.js
n36z17qwvkj2euas95qxcxwpt
[ubuntu@ip-10-0-1-104:~$ docker service ps nodejs
ID          NAME      IMAGE          NODE          DESIRED STATE   CURRENT STATE        ERROR          PORTS
zlp6rgo1p7o8  nodejs.1  labdocker/alpineweb:latest  ip-10-0-1-62  Running        Running  2 seconds ago
pyvgye34w7b1  nodejs.2  labdocker/alpineweb:latest  ip-10-0-1-104  Running        Running  2 seconds ago
w4vwxvqflz1x  nodejs.3  labdocker/alpineweb:latest  ip-10-0-1-62  Running        Running  2 seconds ago
w5isypf10p51  nodejs.4  labdocker/alpineweb:latest  ip-10-0-1-62  Running        Running  2 seconds ago
tf2bwqa7bj1q  nodejs.5  labdocker/alpineweb:latest  ip-10-0-1-62  Running        Running  2 seconds ago
6f8tjnlyyrs  nodejs.6  labdocker/alpineweb:latest  ip-10-0-1-163  Running        Running  3 seconds ago
7jidk3fa3opc  nodejs.7  labdocker/alpineweb:latest  ip-10-0-1-62  Running        Running  2 seconds ago
kg2bqew9g4o9  nodejs.8  labdocker/alpineweb:latest  ip-10-0-1-163  Running        Running  2 seconds ago
nlziswaf4p92  nodejs.9  labdocker/alpineweb:latest  ip-10-0-1-163  Running        Running  2 seconds ago
ozm25d536exf  nodejs.10 labdocker/alpineweb:latest  ip-10-0-1-104  Running        Running  2 seconds ago
[ubuntu@ip-10-0-1-104:~$ docker service rm nodejs
nodejs
ubuntu@ip-10-0-1-104:~$
```

Workshop 2-4: Orchestrator

- สร้าง docker service โดยระบุใน swarm กระจายภายใต้ label

```
docker service create -dt --placement-pref  
'spread=node.labels.physical' \  
--name nodejs --replicas=10 labdocker/alpineweb:latest node hello.js
```

```
...orkshop_112018 — Terminal MAC Pro — bash      ...04: ~ — ssh -i k8s_lab ubuntu@13.229.126.99      ...63: ~ — ssh -i k8s_lab ubuntu@13.229.98.3 ...      ... — ssh -i k8s_lab ubuntu@18.136.112.104  
ubuntu@ip-10-0-1-104:~$ docker service create -dt --placement-pref 'spread=node.labels.physical' \  
[> --name nodejs --replicas=10 labdocker/alpineweb:latest node hello.js  
n36z17qwkj2euas95qxcxwpt  
[ubuntu@ip-10-0-1-104:~$ docker service ps nodejs  
ID          NAME        IMAGE          NODE          DESIRED STATE    CURRENT STATE          ERROR          PORTS  
zlp6rgo1p7o8  nodejs.1  labdocker/alpineweb:latest  ip-10-0-1-62  Running        Running  2 seconds ago  
pyvgye34w7b1  nodejs.2  labdocker/alpineweb:latest  ip-10-0-1-104  Running        Running  2 seconds ago  
w4vwvxvqflz1x nodejs.3  labdocker/alpineweb:latest  ip-10-0-1-62  Running        Running  2 seconds ago  
w5isypf10p51  nodejs.4  labdocker/alpineweb:latest  ip-10-0-1-62  Running        Running  2 seconds ago  
tf2bwqa7bj1q  nodejs.5  labdocker/alpineweb:latest  ip-10-0-1-62  Running        Running  2 seconds ago  
6f8tjnlyyrs   nodejs.6  labdocker/alpineweb:latest  ip-10-0-1-163  Running        Running  3 seconds ago  
7jidk3fa3opc   nodejs.7  labdocker/alpineweb:latest  ip-10-0-1-62  Running        Running  2 seconds ago  
kg2bqew9g4o9   nodejs.8  labdocker/alpineweb:latest  ip-10-0-1-163  Running        Running  2 seconds ago  
nlziswaf4p92   nodejs.9  labdocker/alpineweb:latest  ip-10-0-1-163  Running        Running  2 seconds ago  
ozm25d536exf  nodejs.10 labdocker/alpineweb:latest  ip-10-0-1-104  Running        Running  2 seconds ago  
[ubuntu@ip-10-0-1-104:~$ docker service rm nodejs  
nodejs  
ubuntu@ip-10-0-1-104:~$
```

Config and Secret

- Make secret data and configuration great again !
- การทำงานของ microservice ที่รวมกันเป็น application stack จะมีส่วนของข้อมูลคอนฟิกต่างๆ ที่จำเป็นต้องใช้งานในระบบ เช่น
 - Root password of database
 - Environment variable
 - Custom variable
 - Path of mount volume data
 - Etc
- docker config จะใช้เพื่อจัดเก็บข้อมูลคอนฟิกต่างๆ ของ container
- docker secret ใช้จัดเก็บข้อมูลที่เป็นความลับโดยการเข้ารหัสข้อมูล

Config and Secret

- config can add/update with service anytime
 - Linux container: /<config name>
 - Windows container: c:\<config name>

```
echo "config value" | docker config create <config name> -
```

```
docker config create <config name> <config file>
```

```
docker config rm <config name>
```

- Apply config to service

```
docker service <create/update> \  
--config source=<config name>,target=<path to map config>
```

```
docker service <create/update> \  
--config-rm <config name>
```

Config and Secret

- secret is all same as configure except that secret is encrypt all data (encode base64)
 - Linux container: /run/secrets/<secret name>
 - Windows container: c:\ProgramData\Docker\Secret

```
echo "config value" | docker secret create <secret name> -
```

```
docker secret create <secret name> <secret file>
```

```
docker secret rm <secret name>
```

- Apply config to service

```
docker service <create/update> --secret <secret name>
```

```
docker service <create/update> \  
--secret source=<secret name>,target=<target path>
```

Workshop 2-4: Config and Secret

- สร้าง nginx service เพื่อรับการให้บริการ https (TLS 1.2) ผ่าน config and secret

```
docker config create nginx.conf /Share_DockerToolbox/nginx.conf
```

```
docker secret create labdocker.com.crt \
/Share_DockerToolbox/labdocker.com.crt
```

```
docker secret create labdocker.com.key \
/Share_DockerToolbox/labdocker.com.key
```

```
...orkshop_112018 — Terminal MAC Pro — bash ...04: ~ — ssh -i k8s_lab ubuntu@13.229.126.99 ...63: ~ — ssh -i k8s_lab ubuntu@13.229.98.3 ...

[ubuntu@ip-10-0-1-104:~$ docker config create nginx.conf ~/docker_workshop_112018/Workshop-2-4-Swarm/nginx.conf
bni17w6tmbks39l1ri3ryvvxz
[ubuntu@ip-10-0-1-104:~$ docker config ls
ID                  NAME          CREATED             UPDATED
bni17w6tmbks39l1ri3ryvvxz  nginx.conf    3 seconds ago    3 seconds ago
ubuntu@ip-10-0-1-104:~$ docker secret create labdocker.com.crt ~/docker_workshop_112018/Workshop-2-4-Swarm/labdocker.com.crt
pt2dhhjiqroqd7gb6ig9bv62n
ubuntu@ip-10-0-1-104:~$ docker secret create labdocker.com.key ~/docker_workshop_112018/Workshop-2-4-Swarm/labdocker.com.key
nu11fusm2zbl815l12s8pe6nb
[ubuntu@ip-10-0-1-104:~$ docker secret ls
ID                  NAME          DRIVER             CREATED             UPDATED
pt2dhhjiqroqd7gb6ig9bv62n  labdocker.com.crt   Less than a second ago  Less than a second ago
nu11fusm2zbl815l12s8pe6nb  labdocker.com.key    Less than a second ago  Less than a second ago
ubuntu@ip-10-0-1-104:~$ docker service create -dt \
```

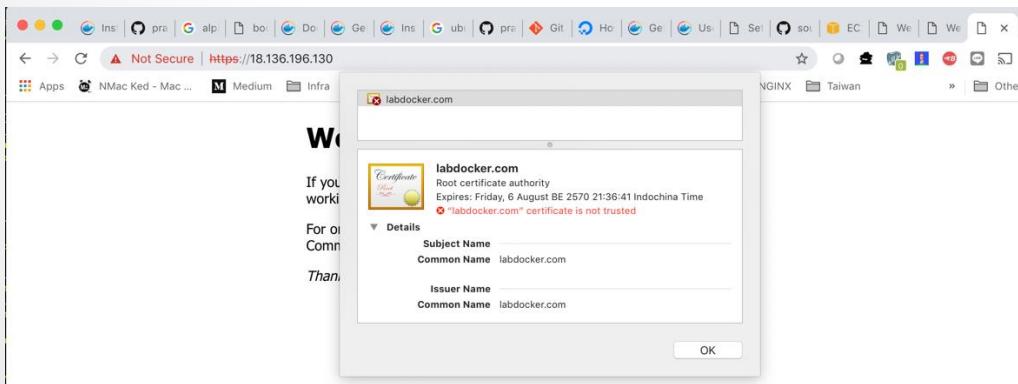
Workshop 2-4: Config and Secret

- nginx.conf

```
server {  
    listen 443 ssl;  
    server_name localhost;  
    ssl_certificate      /run/secrets/labdocker.com.crt;  
    ssl_certificate_key  /run/secrets/labdocker.com.key;  
    location / {  
        root /usr/share/nginx/html;  
        index index.html index.htm;  
    }  
}
```

Workshop 2-4: Config and Secret

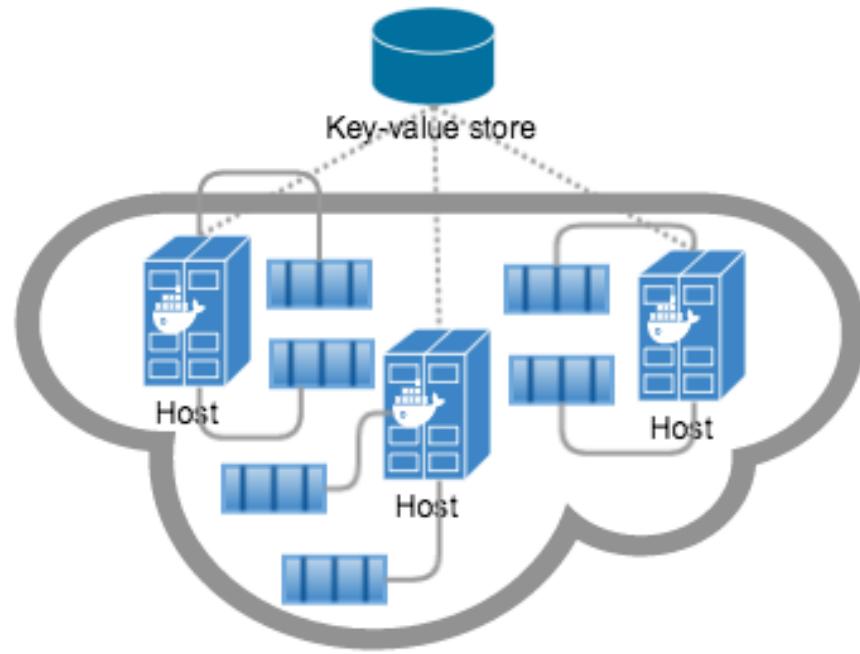
```
docker service -dt create \
--name nginx \
--secret labdocker.com.key \
--secret labdocker.com.crt \
--config source=nginx.conf,target=/etc/nginx/nginx.conf \
-p 443:443 labdocker/nginx:latest
```



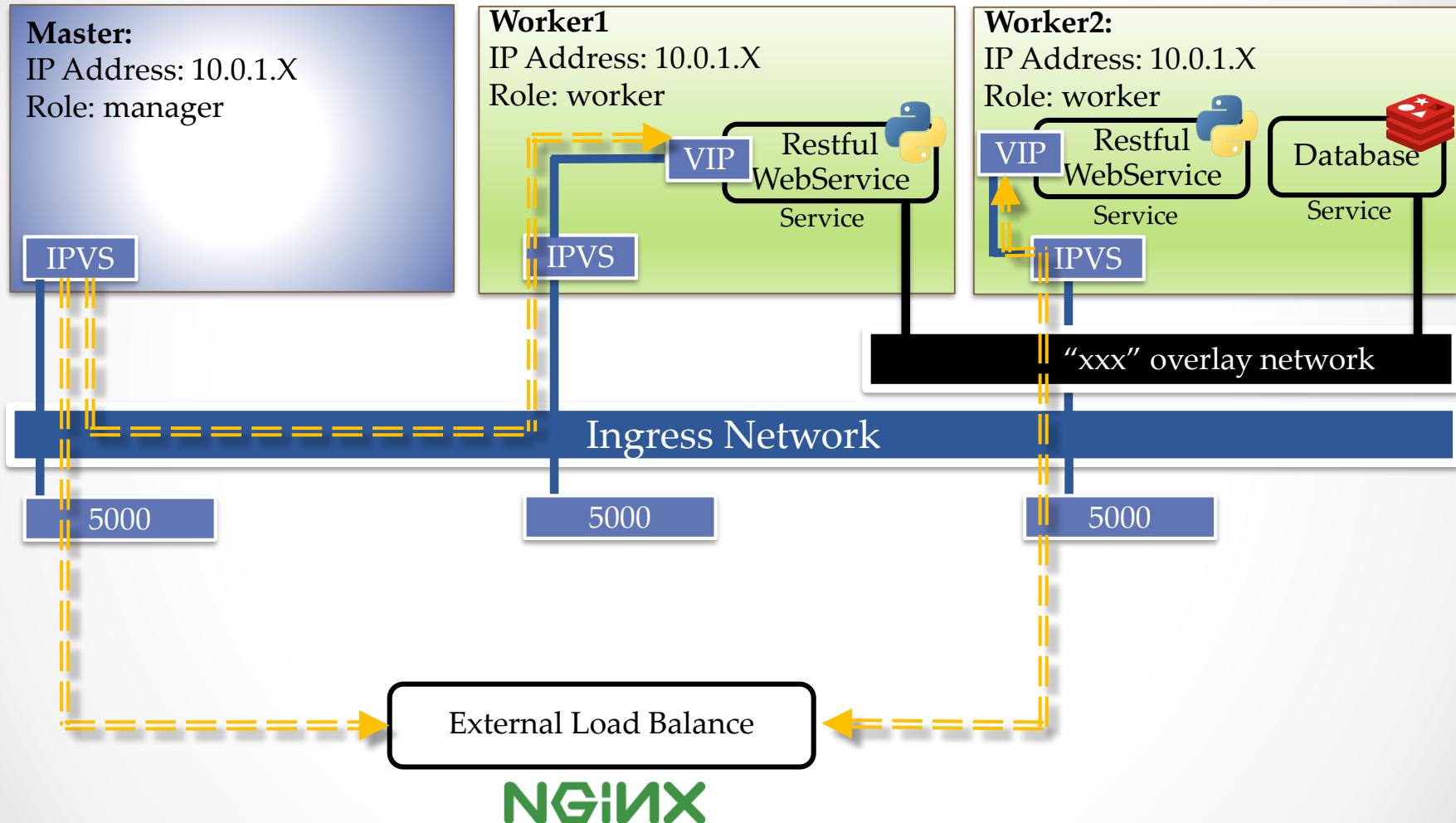
```
praparns-MacBook-Pro:docker_workshop_112018 praparn$ curl -k https://18.136.196.130
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
body {
    width: 35em;
    margin: 0 auto;
    font-family: Tahoma, Verdana, Arial, sans-serif;
}
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>
<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>
<p><em>Thank you for using nginx.</em></p>
</body>
</html>
praparns-MacBook-Pro:docker_workshop_112018 praparn$
```

Overlay and Ingress Network

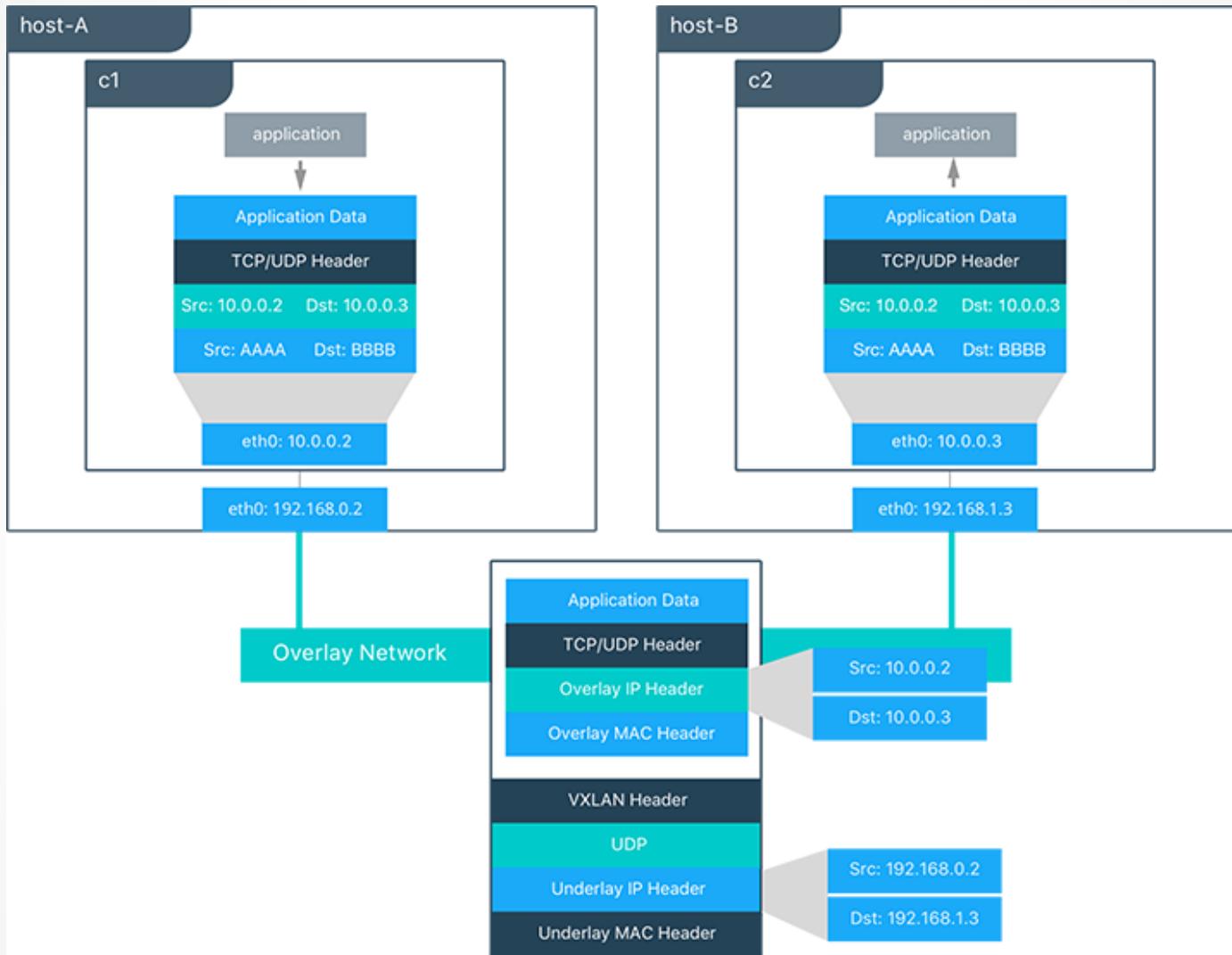
- เพื่อให้ทุกๆ node ภายใน swarm cluster สามารถมองเห็นกันได้ docker ได้เตรียมระบบ network แบบ overlay
- การทำงานของ Overlay จะอาศัยกลไกของ key value store เป็นหลักในการตรวจสอบ (Discovery / Network status / IP Address etc)
- Default swarm จะสร้าง overlay network ชื่อ “ingress” เพื่อรับการใช้งาน service



Overlay and Ingress Network



Overlay and Ingress Network



Workshop 2-4: Overlay and Ingress

- สร้าง overlay network บน swarm manager

```
docker network create --driver overlay --  
subnet=192.168.100.0/24 swarmnet
```

```
...orkshop_112018 — Terminal MAC Pro — -bash ...04: ~ — ssh -i k8s_lab ubuntu@13.229.126.99 ...63: ~ — ssh -  
[ubuntu@ip-10-0-1-104:~$ docker network create --driver overlay --subnet=192.168.100.0/24 swarmnet  
lwxhyfrildn83atwsqxdb9ior  
[ubuntu@ip-10-0-1-104:~$ docker network ls  
NETWORK ID      NAME      DRIVER      SCOPE  
cdb776996466    bridge    bridge      local  
7f907ae7a4c9    docker_gwbridge    bridge      local  
d44f5d624dd1    host      host      local  
hd63tliqe10s    ingress    overlay      swarm  
a1ef67647608    none      null      local  
lwxhyfrildn8    swarmnet    overlay      swarm  
ubuntu@ip-10-0-1-104:~$
```

Workshop 2-4: Overlay and Ingress

- สร้าง new service เพื่อใช้งาน swarm network

```
docker service -dt create --name nodejs --replicas=2 --  
network-add swarmnet -p 3000:3000  
labdocker/alpineweb:latest node hello.js
```

```
[rootshop_112018 — Terminal MAC Pro — bash] ...04: ~ — ssh -i k8s_lab ubuntu@13.229.126.99 ...63: ~ — ssh -i k8s_lab ubuntu@13.229.98.3 ... ...~ — ssh -i k8s_lab ubuntu@18.136.198.154 ...  
ubuntu@ip-10-0-1-104:~$ docker service create -dt --name nodejs \  
> --replicas=2 --network swarmnet -p 3000:3000 \  
[> labdocker/alpineweb:latest node hello.js  
4zw4z04vcxgu0k819q1ms1ar6  
[ubuntu@ip-10-0-1-104:~$ docker service ps nodejs  
ID NAME IMAGE NODE DESIRED STATE CURRENT STATE ERROR PORTS  
t1022pjovfjs nodejs.1 labdocker/alpineweb:latest ip-10-0-1-104 Running Preparing 1 second ago  
51tldbd8z00y nodejs.2 labdocker/alpineweb:latest ip-10-0-1-163 Running Preparing 1 second ago  
ubuntu@ip-10-0-1-104:~$  
  
[ubuntu@ip-10-0-1-104:~$ docker ps  
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES  
bfc3f7a2e264 labdocker/alpineweb:latest "node hello.js node ..." About a minute ago Up About a minute 3000/tcp nodejs.1.t1022pjovfjsrxgtfficvr9mf  
[ubuntu@ip-10-0-1-104:~$ docker inspect nodejs.1.t1022pjovfjsrxgtfficvr9mf|grep IPAddress  
"SecondaryIPAddresses": null,  
"IPAddress": "",  
"IPAddress": "10.255.0.8",  
"IPAddress": "192.168.100.6",  
ubuntu@ip-10-0-1-104:~$  
  
[ubuntu@ip-10-0-1-163:~$ docker ps  
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES  
cf75d3a7074b labdocker/alpineweb:latest "node hello.js node ..." 2 minutes ago Up 2 minutes 3000/tcp nodejs.2.51tldbd8z00yaq761ac6ozpy0  
[ubuntu@ip-10-0-1-163:~$ docker inspect nodejs.2.51tldbd8z00yaq761ac6ozpy0|grep IPAddress  
"SecondaryIPAddresses": null,  
"IPAddress": "",  
"IPAddress": "10.255.0.9",  
"IPAddress": "192.168.100.7",  
ubuntu@ip-10-0-1-163:~$
```

Workshop 2-4: Overlay and Ingress

- ตรวจสอบ IP Address ของ service และทดสอบ Ping ข้าม node

```
docker service inspect nodejs |more
```

```
...orkshop_112018 — Terminal MAC Pro — -bash ...04: ~ — ssh -i k8s_lab ubuntu@13.229.126.99 ...63: ~ — ssh -i k8s_lab ubuntu@13.229.98.3 ... ~ — s
ubuntu@ip-10-0-1-104:~$ docker service inspect nodejs |more
[{"Service": {
    "ID": "4zw4z04vcxgu0k819q1msiar6",
    "Version": {
        "Index": 233
    },
    "CreatedAt": "2018-10-21T17:12:30.830220563Z",
    "UpdatedAt": "2018-10-21T17:12:30.831796908Z",
    "Spec": {
        "Name": "nodejs",
        "Labels": {},
        "TaskTemplate": {
            "ContainerSpec": {
                "Image": "labdocker/alpineweb:latest@sha256:a7aa70c78120ed126487aecbf49db1054f1ff131858516c91137c3accbb4a333",
                "Args": [
                    "node",
                    "hello.js"
                ],
                "Init": false,
                "TTY": true,
                "StopGracePeriod": 10000000000,
                "DNSConfig": {},
                "Isolation": "default"
            }
        }
    }
}}, {"Service": {
    "ID": "4zw4z04vcxgu0k819q1msiar6",
    "Version": {
        "Index": 233
    },
    "CreatedAt": "2018-10-21T17:12:30.830220563Z",
    "UpdatedAt": "2018-10-21T17:12:30.831796908Z",
    "Spec": {
        "Name": "nodejs",
        "Labels": {},
        "TaskTemplate": {
            "ContainerSpec": {
                "Image": "labdocker/alpineweb:latest@sha256:a7aa70c78120ed126487aecbf49db1054f1ff131858516c91137c3accbb4a333",
                "Args": [
                    "node",
                    "hello.js"
                ],
                "Init": false,
                "TTY": true,
                "StopGracePeriod": 10000000000,
                "DNSConfig": {},
                "Isolation": "default"
            }
        }
    }
}}, {"Service": {
    "ID": "4zw4z04vcxgu0k819q1msiar6",
    "Version": {
        "Index": 233
    },
    "CreatedAt": "2018-10-21T17:12:30.830220563Z",
    "UpdatedAt": "2018-10-21T17:12:30.831796908Z",
    "Spec": {
        "Name": "nodejs",
        "Labels": {},
        "TaskTemplate": {
            "ContainerSpec": {
                "Image": "labdocker/alpineweb:latest@sha256:a7aa70c78120ed126487aecbf49db1054f1ff131858516c91137c3accbb4a333",
                "Args": [
                    "node",
                    "hello.js"
                ],
                "Init": false,
                "TTY": true,
                "StopGracePeriod": 10000000000,
                "DNSConfig": {},
                "Isolation": "default"
            }
        }
    }
}}]
```

```
...orkshop_112018 — Terminal MAC Pro — -bash ...04: ~ — ssh -i k8s_lab ubuntu@13.229.126.99 ...63: ~ — ssh -i k8s_lab ubuntu@13.229.98.3 ...
}
],
"VirtualIPs": [
    {
        "NetworkID": "hd63tliqe10s27mu9w4ilckui",
        "Addr": "10.255.0.7/16"
    },
    {
        "NetworkID": "lwxyhyfrildn83atwsqxdb9ior",
        "Addr": "192.168.100.5/24"
    }
]
}
]
ubuntu@ip-10-0-1-104:~$
```

Workshop 2-4: Overlay and Ingress

- ตรวจสอบ IP Address ของ service และทดสอบ Ping ข้าม node

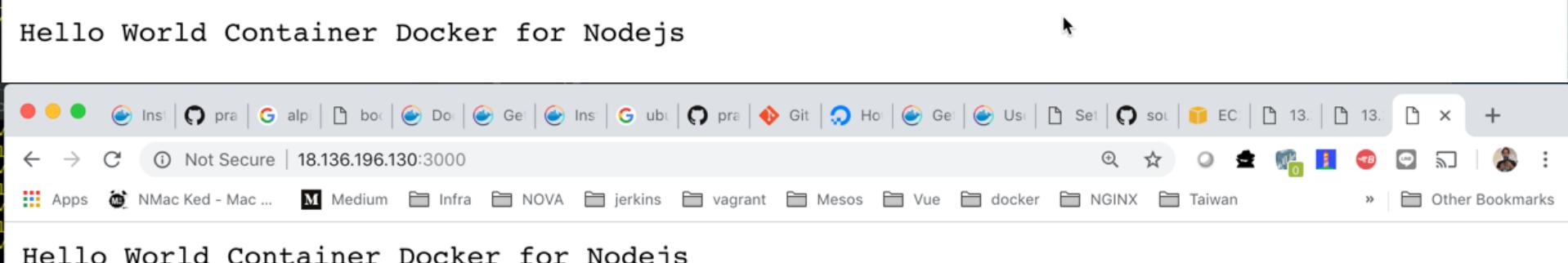
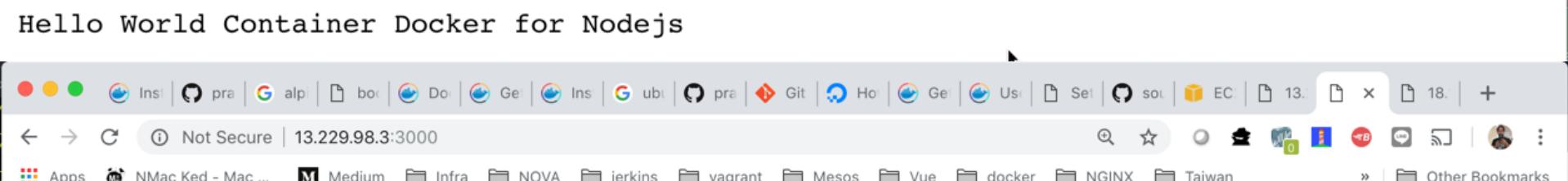
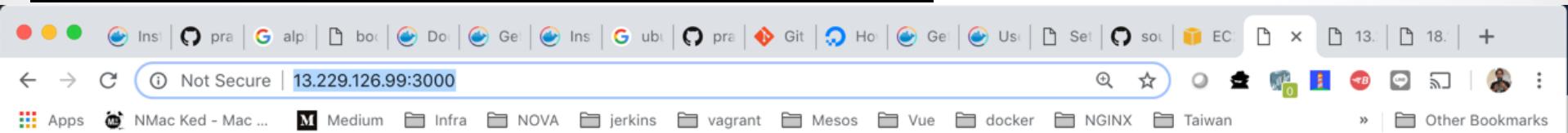
```
[ubuntu@ip-10-0-1-104:~$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                 NAMES
bfc3f7a2e264        labdocker/alpineweb:latest "node hello.js node ..."   4 minutes ago    Up 4 minutes          3000/tcp
[ubuntu@ip-10-0-1-104:~$ docker exec -it nodejs.1.t1022pjovfjsrxgtfficvr9mf ping 192.168.100.5
PING 192.168.100.5 (192.168.100.5): 56 data bytes
64 bytes from 192.168.100.5: seq=0 ttl=64 time=0.116 ms
64 bytes from 192.168.100.5: seq=1 ttl=64 time=0.076 ms
^C
--- 192.168.100.5 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 0.076/0.096/0.116 ms
[ubuntu@ip-10-0-1-104:~$ docker exec -it nodejs.1.t1022pjovfjsrxgtfficvr9mf ping 192.168.100.7
PING 192.168.100.7 (192.168.100.7): 56 data bytes
64 bytes from 192.168.100.7: seq=0 ttl=64 time=1.114 ms
64 bytes from 192.168.100.7: seq=1 ttl=64 time=0.226 ms
^C
--- 192.168.100.7 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 0.226/0.670/1.114 ms
ubuntu@ip-10-0-1-104:~$ 

[ubuntu@ip-10-0-1-163:~$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                 NAMES
cf75d3a7074b        labdocker/alpineweb:latest "node hello.js node ..."   7 minutes ago    Up 7 minutes          3000/tcp
[ubuntu@ip-10-0-1-163:~$ docker exec -it nodejs.2.51t1dbd8z00yaq761ac6ozpy0 ping 192.168.100.5
PING 192.168.100.5 (192.168.100.5): 56 data bytes
64 bytes from 192.168.100.5: seq=0 ttl=64 time=0.100 ms
64 bytes from 192.168.100.5: seq=1 ttl=64 time=0.062 ms
^C
--- 192.168.100.5 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 0.062/0.081/0.100 ms
[ubuntu@ip-10-0-1-163:~$ docker exec -it nodejs.2.51t1dbd8z00yaq761ac6ozpy0 ping 192.168.100.6
PING 192.168.100.6 (192.168.100.6): 56 data bytes
64 bytes from 192.168.100.6: seq=0 ttl=64 time=0.244 ms
64 bytes from 192.168.100.6: seq=1 ttl=64 time=0.237 ms
^C
--- 192.168.100.6 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 0.237/0.240/0.244 ms
ubuntu@ip-10-0-1-163:~$ 
```

Workshop 2-4: Overlay and Ingress

- เรียกใช้บริการผ่าน http port ด้วย browser/curl

```
praparns-MacBook-Pro:docker_workshop_112018 praparn$ curl http://18.136.196.130:3000
Hello World Container Docker for Nodejs
praparns-MacBook-Pro:docker_workshop_112018 praparn$ curl http://13.229.126.99:3000
Hello World Container Docker for Nodejs
praparns-MacBook-Pro:docker_workshop_112018 praparn$ curl http://13.229.98.3:3000
Hello World Container Docker for Nodejs
praparns-MacBook-Pro:docker_workshop_112018 praparn$
```



HA Manager Role

- Add redundancy swarm manager in swarm cluster
 - ทำการ change role node จาก worker → manager

```
docker node update --role manager <node id>
```

```
...orkshop_112018 — Terminal MAC Pro — -bash ...04: ~ — ssh -i k8s_lab ubuntu@13.229.126.99 ...63: ~ — ssh -i k8s_lab ubuntu@13.229.98.3

[ubuntu@ip-10-0-1-104:~$ docker node ls
ID                      HOSTNAME        STATUS        AVAILABILITY      MANAGER STATUS      ENGINE VERSION
4g6pqogaae5co8zxe8w6pxfd6  ip-10-0-1-62  Ready       Active
3tnvltyyohbekgygvsb1jjovy * ip-10-0-1-104 Ready       Active
ua0ajfjd4d1mksqzwr80oy2ru ip-10-0-1-163 Ready       Active
[ubuntu@ip-10-0-1-104:~$ docker node update --role manager ip-10-0-1-163
ip-10-0-1-163
[ubuntu@ip-10-0-1-104:~$ docker node ls
ID                      HOSTNAME        STATUS        AVAILABILITY      MANAGER STATUS      ENGINE VERSION
4g6pqogaae5co8zxe8w6pxfd6  ip-10-0-1-62  Ready       Active
3tnvltyyohbekgygvsb1jjovy * ip-10-0-1-104 Ready       Active
ua0ajfjd4d1mksqzwr80oy2ru ip-10-0-1-163 Ready       Active
[ubuntu@ip-10-0-1-104:~$ docker node update --role manager ip-10-0-1-62
ip-10-0-1-62
[ubuntu@ip-10-0-1-104:~$ docker node ls
ID                      HOSTNAME        STATUS        AVAILABILITY      MANAGER STATUS      ENGINE VERSION
4g6pqogaae5co8zxe8w6pxfd6  ip-10-0-1-62  Ready       Active
3tnvltyyohbekgygvsb1jjovy * ip-10-0-1-104 Ready       Active
ua0ajfjd4d1mksqzwr80oy2ru ip-10-0-1-163 Ready       Active
ubuntu@ip-10-0-1-104:~$
```

HA Manager Role

- Add redundancy swarm manager in swarm cluster
 - Restart major manager

```
[ubuntu@ip-10-0-1-104:~$ docker node ls
ID          HOSTNAME  STATUS  AVAILABILITY  MANAGER STATUS  ENGINE VERSION
4g6pqogaae5co8zxe8w6pxfd6  ip-10-0-1-62  Ready   Active        Reachable      18.06.1-ce
3tnvltyyohbekgygvsb1jjovy * ip-10-0-1-104  Ready   Active        Leader         18.06.1-ce
ua0ajfjd4d1mksqzwr80oy2ru  ip-10-0-1-163  Ready   Active        Reachable      18.06.1-ce
[ubuntu@ip-10-0-1-104:~$ sudo shutdown -r now
Connection to 13.229.224.202 closed by remote host.
Connection to 13.229.224.202 closed.
praparns-MacBook-Pro:~ ssh praparn$
```

- Other will take role for manager within 5 min.

```
[ubuntu@ip-10-0-1-163:~$ docker node ls
ID          HOSTNAME  STATUS  AVAILABILITY  MANAGER STATUS  ENGINE VERSION
4g6pqogaae5co8zxe8w6pxfd6  ip-10-0-1-62  Ready   Active        Reachable      18.06.1-ce
3tnvltyyohbekgygvsb1jjovy  ip-10-0-1-104  Ready   Active        Leader         18.06.1-ce
ua0ajfjd4d1mksqzwr80oy2ru * ip-10-0-1-163  Ready   Active        Reachable      18.06.1-ce
[ubuntu@ip-10-0-1-163:~$ docker node ls
ID          HOSTNAME  STATUS  AVAILABILITY  MANAGER STATUS  ENGINE VERSION
4g6pqogaae5co8zxe8w6pxfd6  ip-10-0-1-62  Ready   Active        Reachable      18.06.1-ce
3tnvltyyohbekgygvsb1jjovy  ip-10-0-1-104  Unknown  Active        Unreachable    18.06.1-ce
ua0ajfjd4d1mksqzwr80oy2ru * ip-10-0-1-163  Ready   Active        Leader         18.06.1-ce
ubuntu@ip-10-0-1-163:~$
```

HA Manager Role

- Remove
 - Update node to worker2 and set “Drain” to node

```
[ubuntu@ip-10-0-1-104:~$ docker node ls
ID          HOSTNAME        STATUS      AVAILABILITY  MANAGER STATUS      ENGINE VERSION
4g6pqogaae5co8zxe8w6pxfd6  ip-10-0-1-62  Ready       Active        Reachable           18.06.1-ce
3tnvltyyohbekgygvbs1jjovy * ip-10-0-1-104  Ready       Active        Reachable           18.06.1-ce
ua0ajfjd4d1mksqzwr80oy2ru  ip-10-0-1-163  Ready       Active        Leader              18.06.1-ce
[ubuntu@ip-10-0-1-104:~$ docker node update --role worker ip-10-0-1-62
ip-10-0-1-62
[ubuntu@ip-10-0-1-104:~$ docker node update --availability drain ip-10-0-1-62
ip-10-0-1-62
[ubuntu@ip-10-0-1-104:~$ docker node ls
ID          HOSTNAME        STATUS      AVAILABILITY  MANAGER STATUS      ENGINE VERSION
4g6pqogaae5co8zxe8w6pxfd6  ip-10-0-1-62  Ready       Drain         Reachable           18.06.1-ce
3tnvltyyohbekgygvbs1jjovy * ip-10-0-1-104  Ready       Active        Reachable           18.06.1-ce
ua0ajfjd4d1mksqzwr80oy2ru  ip-10-0-1-163  Ready       Active        Leader              18.06.1-ce
ubuntu@ip-10-0-1-104:~$ ]
```

- Leave swarm on node “labdocker”

```
[...orkshop_112018 — Terminal MAC Pro — -bash  ...4: ~
[ubuntu@ip-10-0-1-62:~$ docker swarm leave
Node left the swarm.
ubuntu@ip-10-0-1-62:~$ ]
```

```
[ubuntu@ip-10-0-1-104:~$ docker node ls
ID          HOSTNAME        STATUS      AVAILABILITY  MANAGER STATUS      ENGINE VERSION
4g6pqogaae5co8zxe8w6pxfd6  ip-10-0-1-62  Down        Drain        Reachable           18.06.1-ce
3tnvltyyohbekgygvbs1jjovy * ip-10-0-1-104  Ready       Active        Leader              18.06.1-ce
ua0ajfjd4d1mksqzwr80oy2ru  ip-10-0-1-163  Ready       Active        Leader              18.06.1-ce
[ubuntu@ip-10-0-1-104:~$ docker node rm ip-10-0-1-62
ip-10-0-1-62
[ubuntu@ip-10-0-1-104:~$ docker node ls
ID          HOSTNAME        STATUS      AVAILABILITY  MANAGER STATUS      ENGINE VERSION
3tnvltyyohbekgygvbs1jjovy * ip-10-0-1-104  Ready       Active        Reachable           18.06.1-ce
ua0ajfjd4d1mksqzwr80oy2ru  ip-10-0-1-163  Ready       Active        Leader              18.06.1-ce
ubuntu@ip-10-0-1-104:~$ ]
```

HA Manager Role

- Add redundancy swarm manager in swarm cluster
 - ทำการ join node เข้าไปเป็น manager swarm cluster ด้วยคำสั่ง

```
docker swarm join --ca-hash <hash> <ip of swarm manager:  
port>
```

```
...orkshop_112018 — Terminal MAC Pro — -bash ...4: ~ — ssh -i k8s_lab ubuntu@13.229.224.202 ...63: ~ — ssh -i k8s_lab ubuntu@13.229.98.3 ...~ — ssh -i k8s_lab ubuntu@18.136.196.13  
[ubuntu@ip-10-0-1-62:~$ docker swarm leave  
Node left the swarm.  
[ubuntu@ip-10-0-1-62:~$ docker swarm join --token SwMTKN-1-1qnpobqw0hilpc5i90sz803tlhyi1h4k84uok10y814bvhqf2-3u9rqxfwhbhw52qwqeeiewybl 10.0.1.104:2377  
This node joined a swarm as a manager.  
[ubuntu@ip-10-0-1-62:~$ docker node ls  
ID HOSTNAME STATUS AVAILABILITY MANAGER STATUS ENGINE VERSION  
mydkcrgwkurxyjjb1kf14odl7 * ip-10-0-1-62 Ready Active Reachable 18.06.1-ce  
3tnvltyyohbekgygvbs1jjoyv Ready Active Reachable 18.06.1-ce  
ua0ajfjd4d1mksqzwr80oy2ru ip-10-0-1-163 Ready Active Leader 18.06.1-ce  
ubuntu@ip-10-0-1-62:~$
```

- ตรวจสอบค่า token เพื่อ join swarm

```
[ubuntu@ip-10-0-1-62:~$ docker swarm join-token manager  
To add a manager to this swarm, run the following command:  
  
    docker swarm join --token SwMTKN-1-1qnpobqw0hilpc5i90sz803tlhyi1h4k84uok10y814bvhqf2-3u9rqxfwhbhw52qwqeeiewybl 10.0.1.62:2377  
ubuntu@ip-10-0-1-62:~$
```

Workshop 2-4: HA Manager Role

- ทำการ join new manager เข้าสู่ swarm cluster/update/drain

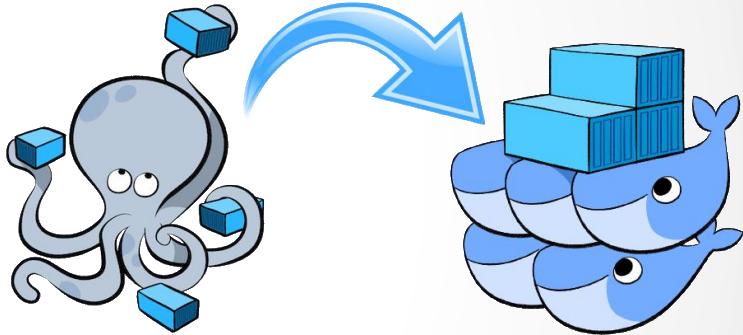
```
[ubuntu@ip-10-0-1-104:~$ docker node ls
ID                      HOSTNAME        STATUS        AVAILABILITY  MANAGER STATUS      ENGINE VERSION
4g6pqogaae5co8zxe8w6pxfd6  ip-10-0-1-62  Ready       Active        Reachable      18.06.1-ce
3tnvltyyohbekgygvsb1jjovy *  ip-10-0-1-104  Ready       Active        Leader        18.06.1-ce
ua0ajfjd4d1mksqzwr80oy2ru  ip-10-0-1-163  Ready       Active        Reachable      18.06.1-ce
ubuntu@ip-10-0-1-104:~$ ]
```

```
[ubuntu@ip-10-0-1-104:~$ docker node ls
ID                      HOSTNAME        STATUS        AVAILABILITY  MANAGER STATUS      ENGINE VERSION
4g6pqogaae5co8zxe8w6pxfd6  ip-10-0-1-62  Ready       Active        Reachable      18.06.1-ce
3tnvltyyohbekgygvsb1jjovy *  ip-10-0-1-104  Ready       Active        Reachable      18.06.1-ce
ua0ajfjd4d1mksqzwr80oy2ru  ip-10-0-1-163  Ready       Active        Leader        18.06.1-ce
[ubuntu@ip-10-0-1-104:~$ docker node update --role worker ip-10-0-1-62
ip-10-0-1-62
[ubuntu@ip-10-0-1-104:~$ docker node update --availability drain ip-10-0-1-62
ip-10-0-1-62
```

```
[ubuntu@ip-10-0-1-104:~$ docker node ls
ID                      HOSTNAME        STATUS        AVAILABILITY  MANAGER STATUS      ENGINE VERSION
4g6pqogaae5co8zxe8w6pxfd6  ip-10-0-1-62  Ready       Drain        Reachable      18.06.1-ce
3tnvltyyohbekgygvsb1jjovy *  ip-10-0-1-104  Ready       Active        Leader        18.06.1-ce
ua0ajfjd4d1mksqzwr80oy2ru  ip-10-0-1-163  Ready       Active        Leader        18.06.1-ce
ubuntu@ip-10-0-1-104:~$ ]
```

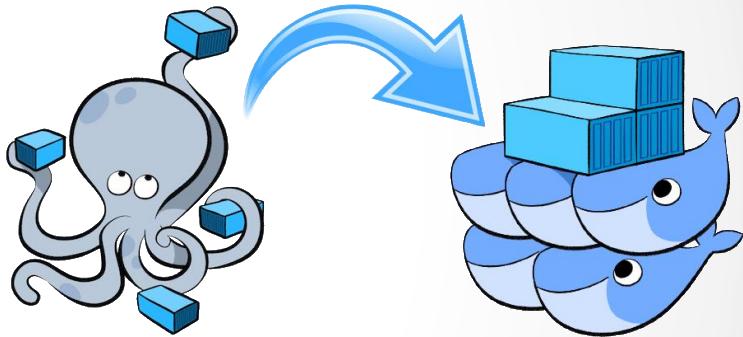
Docker Stack Deploy (Compose)

- Compose syntax: (docker stack deploy)
 - services:
 - XXX (service name): (Current Version 3.3)
 - image: <image name>
 - **deploy: <SWARM>**
 - mode
 - global
 - replicated
 - resource:
 - limits:
 - cpus: 'X.X'
 - memory: XXXM
 - reservations:
 - cpus: 'X.X'
 - memory: 2=XXM
 - restart_policy:
 - condition: on-failure/any
 - delay: XXs
 - max_attempts: X
 - windows: XXs
 - update_config:
 - parallelism: X
 - delay: XXs
 - failure_action: Xms
 - max_failure_ratio: X



Docker Stack Deploy (Compose)

- Not support for docker stack deploy
 - build
 - cgroup_parent
 - container_name
 - devices
 - dns
 - dns_search
 - tmpfs
 - external_links
 - links
 - network_mode
 - security_opt
 - stop_signal
 - sysctls
 - userns_mode



Docker Stack Deploy (Compose)

- Compose syntax: (docker stack deploy)

```
docker stack deploy -c <compose file> <stack name>
```

```
docker stack ls
```

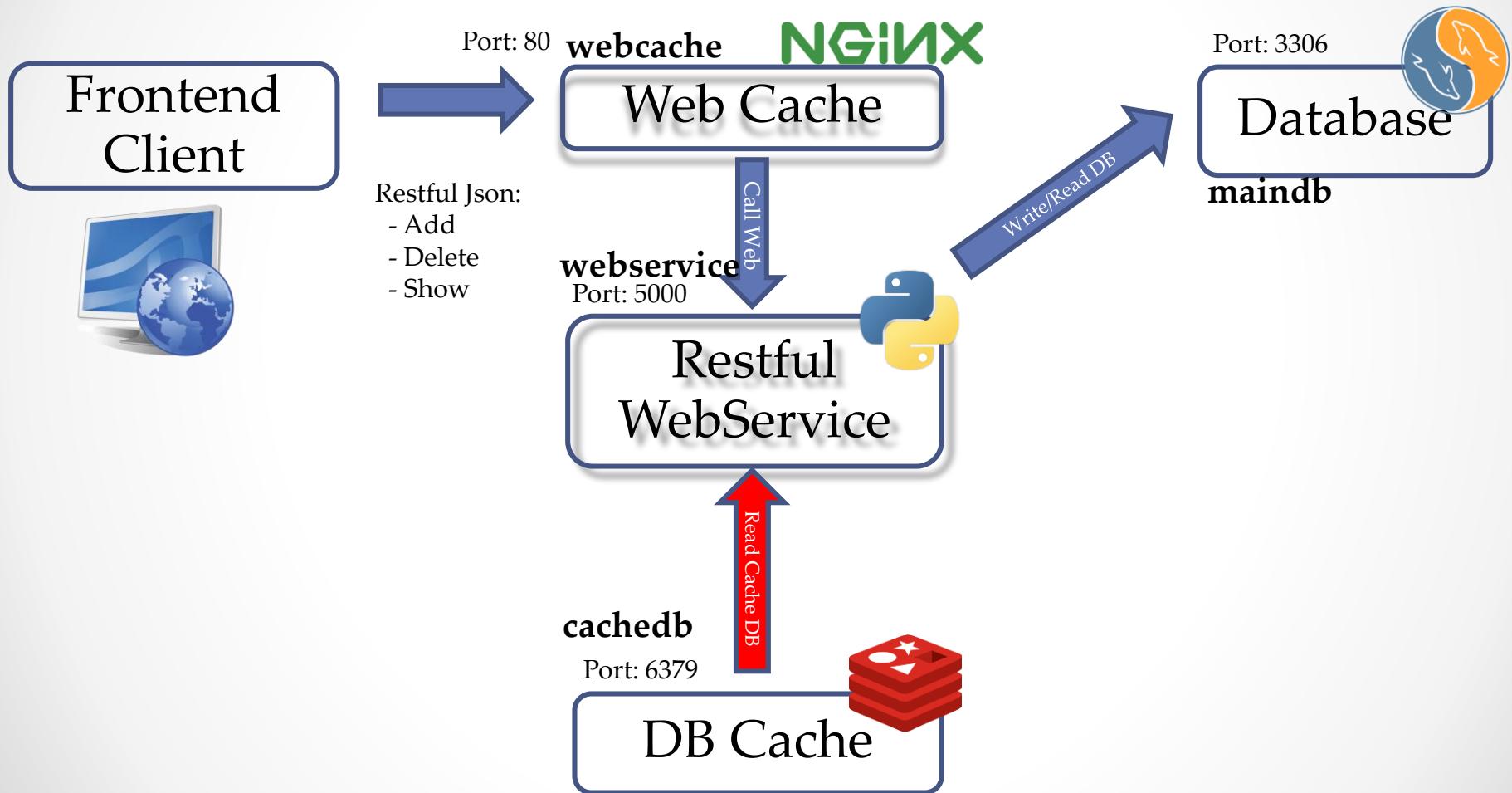
```
docker stack ps <stack name>
```

```
docker stack services <stack name>
```

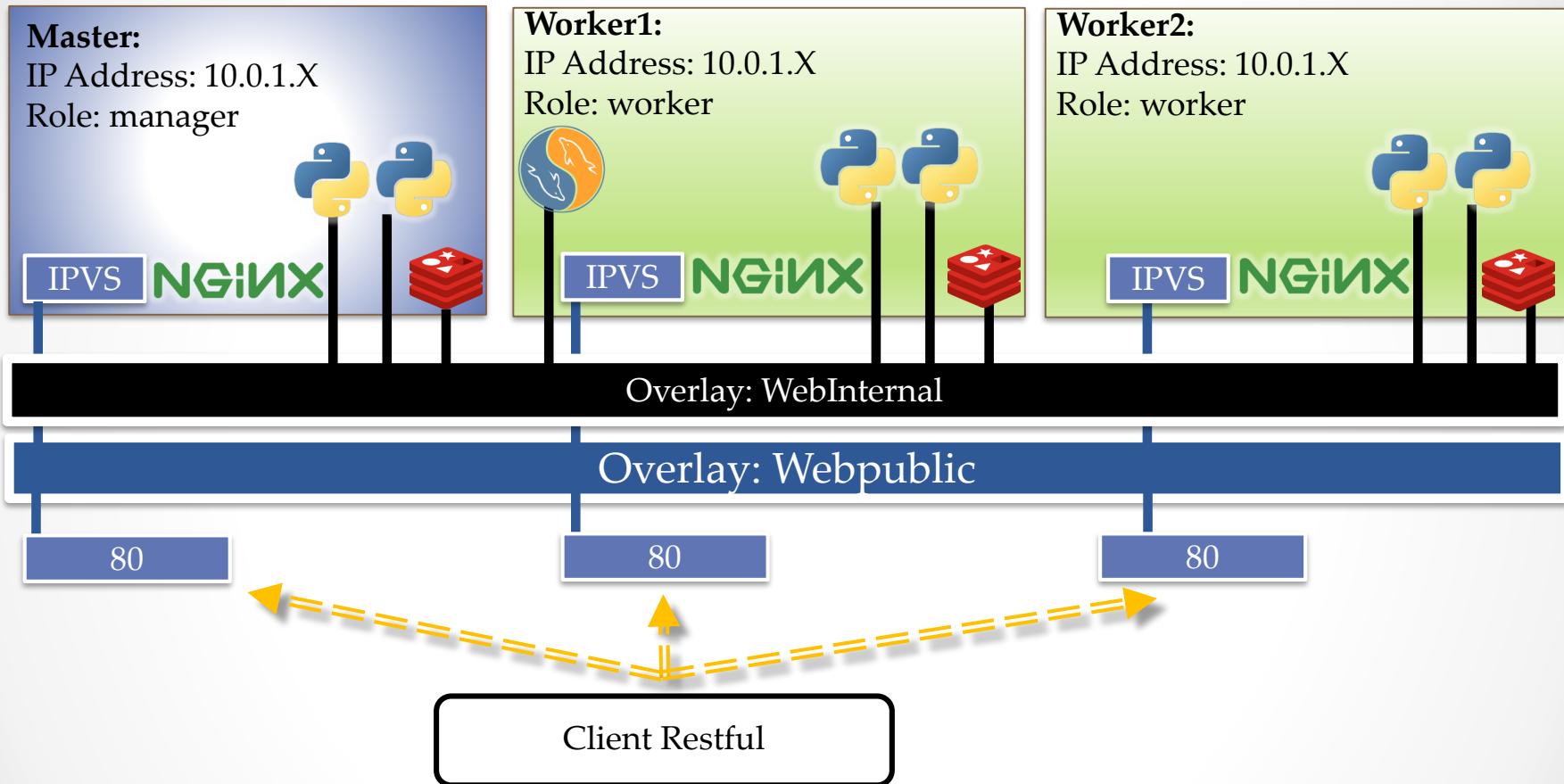
```
docker stack rm <stack name>
```

Workshop 2-4: Swarm Compose

- Optimize for WorkLoad



Workshop 2-4: Swarm Compose



Workshop 2-4: Swarm Compose

```
1  version: '3.3'
2  services:
3    webcache:
4      image: labdocker/cluster:webcache
5      container_name: nginx
6      deploy:
7        mode: global
8        update_config:
9          parallelism: 1
10       delay: 10s
11       restart_policy:
12         condition: on-failure
13         delay: 30s
14         max_attempts: 5
15         window: 60s
16         endpoint_mode: vip
17       depends_on:
18         - webservice
19         - cachedb
20         - maindb
21     networks:
22       webpublic:
23         aliases:
24           - webcache
25       webinternal:
26         aliases:
27           - webcache
28     ports:
29       - "80:80"
30
```

```
31   webservice:
32     image: labdocker/cluster:webservice
33     container_name: webservice
34     deploy:
35       mode: replicated
36       replicas: 6
37       update_config:
38         parallelism: 2
39         delay: 10s
40       restart_policy:
41         condition: on-failure
42         delay: 30s
43         max_attempts: 3
44         window: 60s
45         endpoint_mode: vip
46     depends_on:
47       - cachedb
48       - maindb
49   networks:
50     webinternal:
51       aliases:
52         - webservice
53     ports:
54       - "5000:5000"
```

Workshop 2-4: Swarm Compose

```
58 maindb:
59   image: labdocker/mariadb:latest
60   container_name: maindb
61   deploy:
62     mode: replicated
63     replicas: 1
64     endpoint_mode: vip
65   environment:
66     - MYSQL_ROOT_PASSWORD=password
67   networks:
68     webinternal:
69       aliases:
70         - maindb
71
72 cachedb:
73   image: labdocker/redis:latest
74   container_name: cachedb
75   deploy:
76     mode: global
77     endpoint_mode: vip
78   networks:
79     webinternal:
80       aliases:
81         - cachedb
82
83
84
85
86
87
88
89
90
91
92
93
```

Workshop 2-4: Swarm Compose

```
ubuntu@ip-10-0-1-104:~/docker_workshop_112018/Workshop-2-4-Swarm/python_restfulset$ docker stack deploy -c docker-compose_swarm.yml webservice
Ignoring deprecated options:

container_name: Setting the container name is not supported.

Creating network webservice_webpublic
Creating network webservice_webinternal
Creating service webservice_webcache
Creating service webservice_webservice
Creating service webservice_maindb
Creating service webservice_cachedb
ubuntu@ip-10-0-1-104:~/docker_workshop_112018/Workshop-2-4-Swarm/python_restfulset$ █
```

Workshop 2-4: Swarm Compose

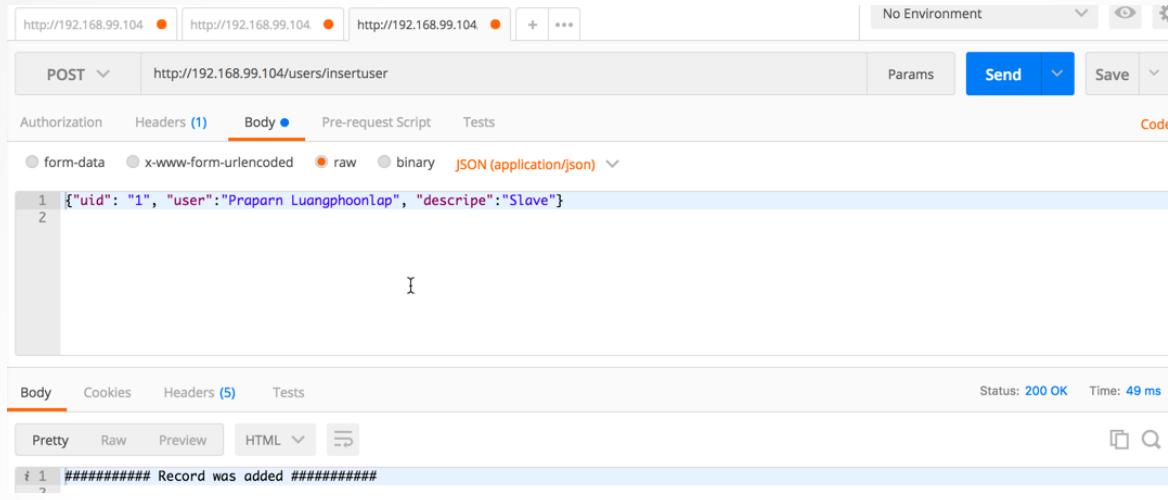
ID	NAME	IMAGE	NODE	DESIRED STATE	CURRENT STATE
	PORTS				
mcj0ux4x25uv o ol223n2c2htp o fj0qu565u8mt o kw8oprgn8b3v ago "task: non-zero exit (1)" a4foun5hzks3 ago "task: non-zero exit (1)" horm2noczrvt ago "task: non-zero exit (1)" vhutisinb7np ago "task: non-zero exit (1)" ktjzx8thiq6k ago "task: non-zero exit (1)" 7siaoo4hnvmh ago "task: non-zero exit (1)" axdh6t2ua971 e ago t8jc8aopm6wu e ago 32d2fao3cjcu e ago mtbhxlkuwjg7 o nw2jky8drn42 ago "task: non-zero exit (1)" ivhofu5u9kxe	webservice_webcache.wwzx60rvl5res39v0lxir2gfh webservice_webcache.0k95omm28xfef22thummpz3uf webservice_webcache.q814dhi3bz14sze4zmjpiy3qa webservice_webcache.wwzx60rvl5res39v0lxir2gfh webservice_webcache.0k95omm28xfef22thummpz3uf webservice_webcache.q814dhi3bz14sze4zmjpiy3qa webservice_webcache.wwzx60rvl5res39v0lxir2gfh webservice_webcache.0k95omm28xfef22thummpz3uf webservice_cachedb.wwzx60rvl5res39v0lxir2gfh webservice_cachedb.q814dhi3bz14sze4zmjpiy3qa webservice_cachedb.0k95omm28xfef22thummpz3uf webservice_webservice.1 _ webservice_webservice.1 webservice_maindb.1	labdocker/cluster:webcache labdocker/cluster:webcache labdocker/cluster:webcache labdocker/cluster:webcache labdocker/cluster:webcache labdocker/cluster:webcache labdocker/cluster:webcache labdocker/cluster:webcache labdocker/redis:latest labdocker/redis:latest labdocker/redis:latest labdocker/cluster:webservice labdocker/cluster:webservice labdocker/mysql:latest	swarm-node2 swarm-mng swarm-node1 swarm-node2 swarm-mng swarm-node1 swarm-node2 swarm-mng swarm-node2 swarm-node1 swarm-mng swarm-node1 swarm-node2 swarm-mng swarm-node1 swarm-node2 swarm-mng swarm-node1 swarm-node2	Running Running Running Shutdown Shutdown Shutdown Shutdown Shutdown Running Running Running Running Running Running Running Running Running Running Running	Running 46 seconds ag Running 49 seconds ag Running 50 seconds ag Failed about a minute Failed about a minute Running about a minut Running about a minut

Workshop 2-4: Swarm Compose

```
praparns-MacBook-Pro:~ praparn$ docker-machine ls
NAME      ACTIVE   DRIVER    STATE     URL
labdocker - virtualbox Running  tcp://192.168.99.103:2376          SWARM   DOCKER    ERRORS
v1.15/version: x509: certificate is valid for 192.168.99.100, not 192.168.99.103
swarm-mng - virtualbox Running  tcp://192.168.99.104:2376          v17.06.0-ce
swarm-node1 - virtualbox Running  tcp://192.168.99.105:2376          v17.06.0-ce
swarm-node2 - virtualbox Running  tcp://192.168.99.106:2376          v17.06.0-ce
praparns-MacBook-Pro:~ praparn$ export Server_IP=192.168.99.104
praparns-MacBook-Pro:~ praparn$ curl http://$Server_IP:$Server_Port/
<H1> Welcome Page from Container Python Lab </H1>Checkpoint Date/Time: Sun Aug 13 17:27:58 2017
```

The screenshot shows the Postman application interface. At the top, there's a header bar with a URL input field containing "http://192.168.99.104", a red circular button, and other icons. Below the header is a toolbar with "GET", "http://192.168.99.104", "Params", "Send", and "Save" buttons. Underneath the toolbar, there are tabs for "Authorization", "Headers", "Body", "Pre-request Script", and "Tests". The "Body" tab is currently selected. In the body section, there's a dropdown for "Type" set to "No Auth". Below the type dropdown, there are tabs for "Body", "Cookies", "Headers (6)", and "Tests". The "Headers" tab is selected. To the right of the headers, it says "Status: 200 OK" and "Time: 24 ms". At the bottom of the body section, there are buttons for "Pretty", "Raw", "Preview", "HTML", and a copy icon. The preview area contains the response body: "*i 1* <H1> Welcome Page from Container Python Lab </H1>Checkpoint Date/Time: Sun Aug 13 17:29:12 2017".

Workshop 2-4: Swarm Compose



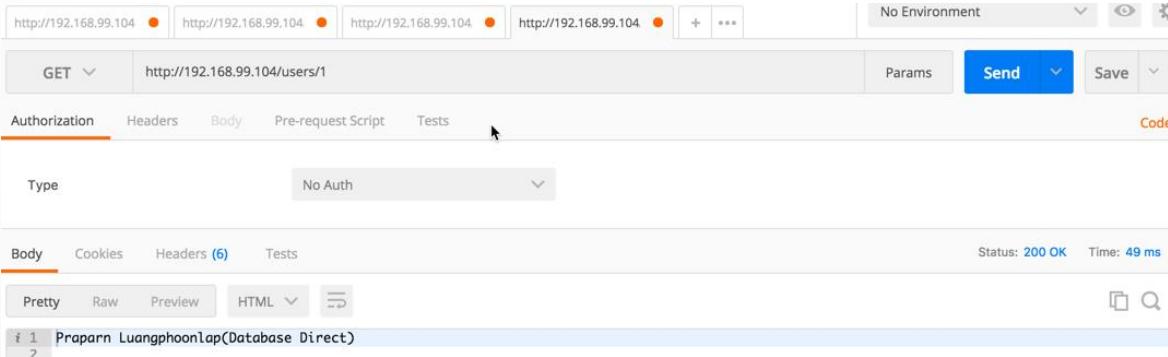
POST <http://192.168.99.104/users/insertuser>

Body (1)

```
{ "uid": "1", "user": "Praparn Luangphoonlap", "desribe": "Slave"}
```

Status: 200 OK Time: 49 ms

Record was added #####



GET <http://192.168.99.104/users/1>

Body (6)

```
Praparn Luangphoonlap(Database Direct)
```

Status: 200 OK Time: 49 ms

Workshop 2-4: Swarm Compose

The screenshot shows a Postman interface with the following details:

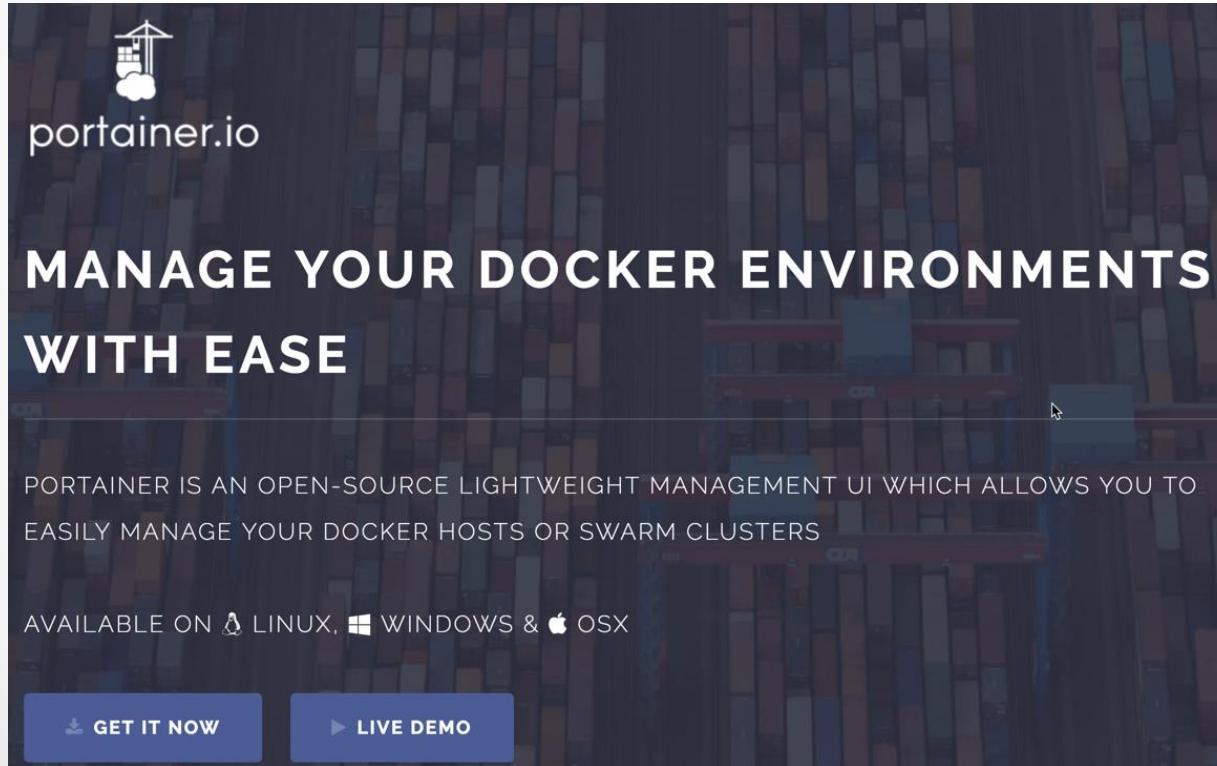
- URL:** http://192.168.99.104/users/removeuser/1
- Type:** No Auth
- Status:** 200 OK
- Time:** 33 ms
- Response Body:** ##### Record was deleted (Both Database Cache) #####

Portainer for Docker

• • •

portainer for Docker

- portainer.io เป็น open source management ที่สามารถใช้บริหารจัดการ docker-machine ผ่านหน้า GUI web ได้อย่างมีประสิทธิภาพแบบ On premise
- การใช้งาน portainer จะใช้การบริหารจัดการผ่านหน้าเว็บ (`x.x.x.x:9000`)



portainer for Docker



Please create the initial administrator user.

Username

Password

Confirm password ✓

✓ The password must be at least 8 characters long

[Create user](#)



Connect Portainer to the Docker environment you want to manage.

⚡ Local Manage the Docker environment where Portainer is running

⚡ Remote Manage a remote Docker environment

⚠ This feature is not yet available for native Docker Windows containers.
Please ensure that you have started the Portainer container with the following Docker flag `-v "/var/run/docker.sock:/var/run/docker.sock"` in order to connect to the local Docker environment.

[Connect](#)

Docker: The Next-Gen of Virtualization



portainer for Docker

portainer.io

ACTIVE ENDPOINT: local

ENDPOINT ACTIONS: Dashboard, App Templates, Containers, Images, Networks, Volumes, Events, Engine

PORTAINER SETTINGS: User management, Endpoints, Registries, Settings

Home Dashboard

Node info

Name	labdocker
Docker version	18.01.0-ce
CPU	4
Memory	1 GB

Containers: 1 running, 0 stopped

Images: 31

Volumes: 5

Networks: 4

2.4 GB

Container list

Containers

Help support portainer | admin | my account | log out

Containers

Start Stop Kill Restart Pause Resume Remove + Add container

Name	State	Quick actions	Stack	Image	IP Address	Published Ports	Ownership
vigilant_pare	running	Containers	-	portainer/portainer	172.17.0.2	9000:9000	public

Items per page: 10

Docker: The Next-Gen of Virtualization



portainer for Docker

portainer.io

ACTIVE ENDPOINT
local

ENDPOINT ACTIONS

Dashboard

App Templates

Containers

Images

Networks

Volumes

Events

Engine

PORAINER SETTINGS

User management

Endpoints

Registries

Settings

Create container
Containers > Add container

Name: nginx

Image configuration
Name: labdocker/nginx:latest Registry: DockerHub

Always pull the image:

Ports configuration
Publish all exposed ports:

Port mapping: map additional port
host: 80 → container: 80 TCP UDP

Access control
Enable access control:

Administrators
I want to restrict the management of this resource to administrators only

Restricted
I want to restrict the management of this resource to a set of users and/or teams

Actions

Advanced container settings

Help support portainer [my account](#) [log out](#)

admin

Docker: The Next-Gen of Virtualization



portainer for Docker

192.168.99.100

NMac Ked - Mac OS... M Medium jenkins vagrant Mesos Vue docker NGINX Taiwan Kubernetes PWA_Progressive_W... MYSQL_Cluster

Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org. Commercial support is available at nginx.com.

Thank you for using nginx.

Container list

Containers

Help support portainer admin
[my account](#) [log out](#)

Containers

Search Settings

<input type="checkbox"/> Name	State <small>Filter</small>	Quick actions	Stack	Image	IP Address	Published Ports	Ownership
<input type="checkbox"/> nginx	running	 	-	labdocker/nginx:latest	172.17.0.3	 80:80	 administrators
<input type="checkbox"/> vigilant_pare	running	 	-	portainer/portainer	172.17.0.2	 9000:9000	 public

Items per page 10

Docker: The Next-Gen of Virtualization



Shipyard for Docker

portainer.io

ACTIVE ENDPOINT local

ENDPOINT ACTIONS

- Dashboard
- App Templates
- Containers
- Images
- Networks
- Volumes
- Events
- Engine

PORAINER SETTINGS

- User management
- Endpoints
- Registries
- Settings

Container statistics
Containers > nginx > Stats

About statistics

This view displays real-time statistics about the container nginx as well as a list of the running processes inside this container.

Refresh rate 5s

Memory usage

CPU usage

Network usage

Processes

Q Search

portainer.io

ACTIVE ENDPOINT local

ENDPOINT ACTIONS

- Dashboard
- App Templates
- Containers
- Images
- Networks
- Volumes
- Events
- Engine

PORAINER SETTINGS

Container console
Containers > nginx > Console

Console

Exec into container as default user using command sh

Disconnect

```
/usr/sbin # ls
add-shell    arping      chroot      deluser     fakeidentd   ftpd       killall5   nanddump   nginx      rdate      readprofile  sendmail   udhcpd
addgroup    brctl      cron        dnsd       fbset       httpd      loadfont   nandwrite  ntpd       rdev       remove-shell  setfont   setlogcons
adduser     chpasswd    delgroup    ether-wake  fdformat   inetd      lspci     nbd-client powertop  readahead  rfkill      setlogons
/usr/sbin #
```

Docker: The Next-Gen of Virtualization



Recapture for Day 2

- Dockerfile and Build
- Compose
- Registry
- Swarm Mode
 - Conceptual of Swarm
 - Swarm Mode Architecture
 - Swarm init/join cluster system
 - Swarm service
 - Orchestrator Assignment
 - Config and Secret
 - Network Overlay and Ingress
 - HA Manager Role
 - Docker Stack Deploy (Compose Swarm Mode)
- portainer for Docker
- Q&A



Q&A

By Praparn Luengphoonlap
Email: eva10409@gmail.com

Docker: The Next-Gen of Virtualization

