



Docker:Zero to Hero (Day 1)

By Praparn Luengphoonlap
Email: eva10409@gmail.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

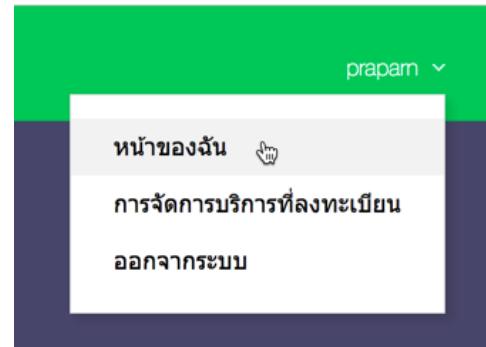
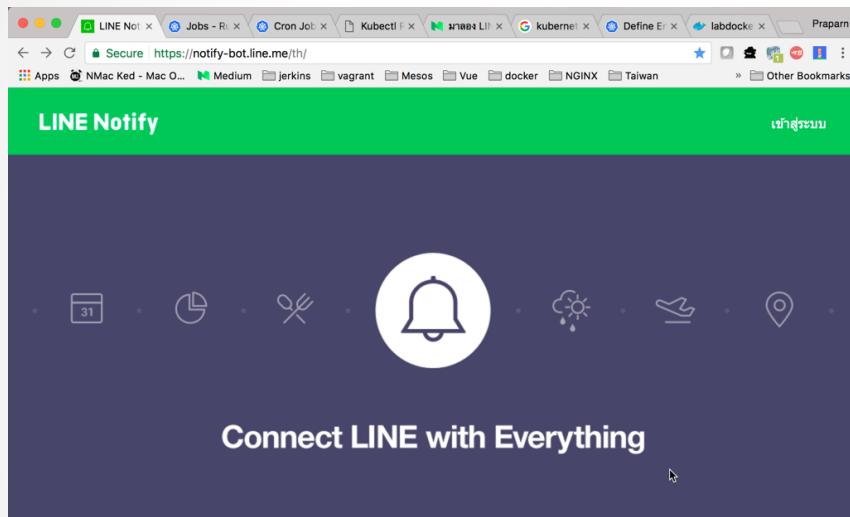
- Dockerfile and Build
- Compose
- Registry
- Swarm Mode
 - Conceptual of Swarm
 - Swarm Mode Architecture
 - Swarm init/join cluster system
 - Swarm service
 - Orchestrator Assignment
 - Config and Secret
 - Network Overlay and Ingress
 - HA Manager Role
 - Docker Stack Deploy (Compose Swarm Mode)
-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



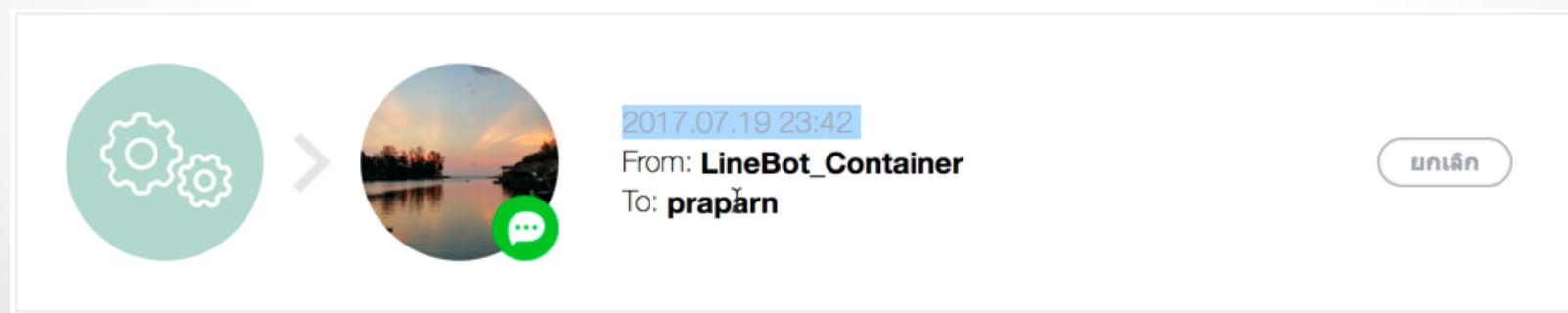
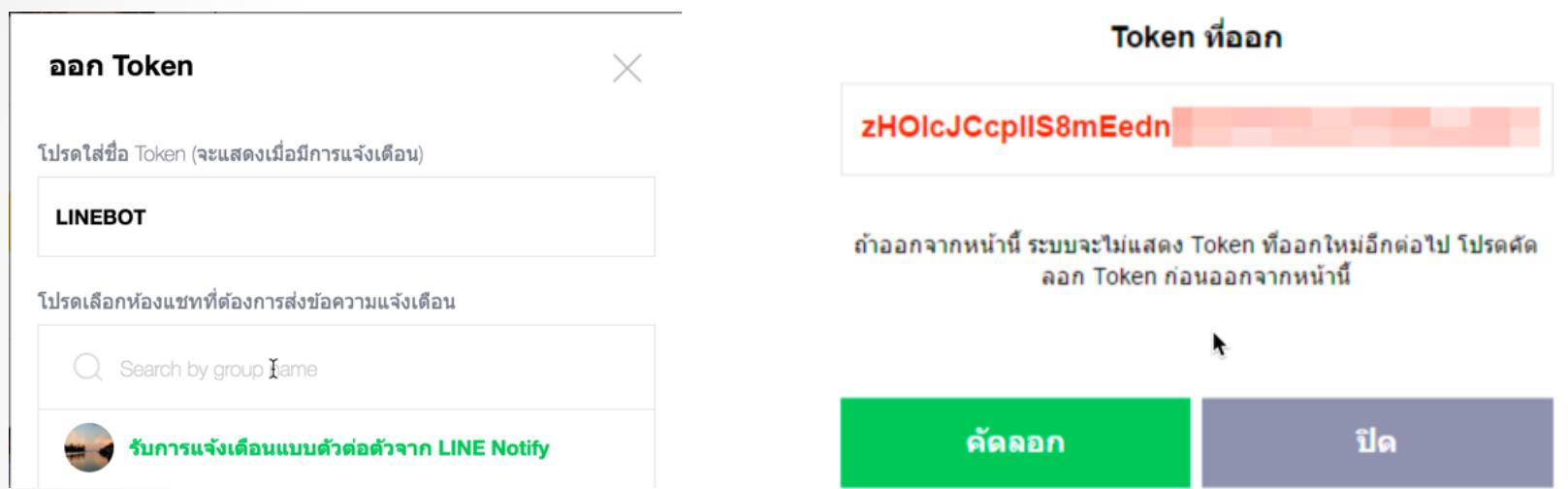
ออก Access Token (สำหรับผู้พัฒนา)

เมื่อใช้ Access Token แบบบุคคล จะสามารถตั้งค่าการแจ้งเตือนได้โดยไม่ต้องลงทะเบียนกับเว็บเซอร์วิส



Prerequisite

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



Lab Resource

- Repository for lab

The screenshot shows a web-based Docker registry interface. At the top, there is a dark header bar with a ship icon, 'Explore', 'Help', a search bar containing 'labdocker', and 'Sign up / Log in' buttons. Below the header, the page title 'Repositories (7)' is displayed in blue. A dropdown menu above the repository list is set to 'All'. The main content area displays three repository entries, each represented by a blue ship icon:

Repository	Status	Stars	Pulls	Details
labdocker/alpineweb	public	0	31	DETAILS
labdocker/nginx	public	0	16	DETAILS
labdocker/alpine	public	0	7	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://goo.gl/jHomH1>
 - Download on GitHub
 - git@github.com:paparn/docker_workshop_112017.git

The screenshot shows a GitHub repository page for the user 'paparn' named 'dockerworkshop'. The repository has 15 commits, 1 branch, 0 releases, and 1 contributor. The latest commit is 'Update compose version 1.8.0' by 'paparn' 10 days ago. The commits are listed below:

Commit	Message	Date
Workshop_1-1_Download_Install	First commit	19 days ago
Workshop_1-2_Access_PoolImage	First commit	19 days ago
Workshop_1-3_Create_PushImage	First commit	19 days ago
Workshop_1-4_Run_Container	First commit	19 days ago
Workshop_1-5_CPU_Memory_IO	First commit	19 days ago
Workshop_1-6_Network	First commit	19 days ago
Workshop_1-7_Volume	First commit	19 days ago
Workshop_1-8_Inspect_Logs	First commit	19 days ago
Workshop_2-1_DockerFile	20160726214615	15 days ago
Workshop_2-2_Compose	Update compose version 1.8.0	10 days ago
Workshop_2-3_Registry	20160726214615	15 days ago

Workshop 1-1: Download & Install



The screenshot shows the Docker website at <https://www.docker.com/products/docker-toolbox>. The page features the Docker logo and navigation links for Docs, Support, Training, Tech Blog, Blog, Docker Hub, Why Docker?, Products, Partners, Community, Company, Careers, and Open Source. A prominent section titled "Docker Toolbox" includes a cartoon illustration of a red toolbox overflowing with various Docker-related icons like a whale, a brain, and a magnifying glass. Below the illustration, a text block states: "The Docker Toolbox is an installer to quickly and easily install and setup a Docker environment on your computer." Two download buttons are provided: one for Mac (labeled "Download") and one for Windows (labeled "Download").

- <https://www.docker.com/products/docker-toolbox>
- Install the software
 - Windows : Windows_Install_Docker.pdf
 - Mac OS: Mac_Install_Docker.pdf

Workshop 1-1: Download & Install

The screenshot shows a web browser window with the URL labs.play-with-docker.com. The page displays a "Welcome!" message and a CAPTCHA verification. Below this, there is a large Docker logo featuring a blue whale carrying a stack of shipping containers. To the right of the logo, the word "docker" is written in a large, lowercase, sans-serif font.

On the left side of the browser window, there is a sidebar with the following items:

- Waiting for labs.play-with-docker.com...
- Kubernetes_Training.pptx (Cancelled)

The main content area of the browser shows a terminal session titled "eb5f5443_node1". The session details are as follows:

IP	10.0.50.3
Memory	2.19% (89.52MiB / 3.996GiB)
CPU	0.19%

Below the session details is a "DELETE" button. The terminal session output is as follows:

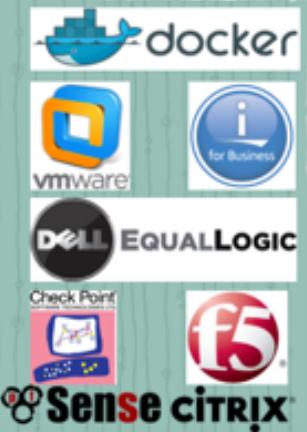
```
#####
# WARNING!!!!
# This is a sandbox environment. Using personal credentials
# is HIGHLY! discouraged. Any consequences of doing so are
# completely the user's responsibilites.
#
# The PWD team.
#####
[node1] (local) root@10.0.50.3 ~
$ docker ps
CONTAINER ID        IMAGE               COMMAND
NAMES
[node1] (local) root@10.0.50.3 ~
$
```

Docker: The Next-Gen of Virtualization

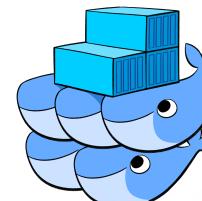




Senior Infrastructure / System Engineer
Compute, Network & Storage System (CNS)
Infrastructure Enterprise Equipment
Virtual Machine for Open System in Prd
Core Application on IBM AS/400 System
Instructor / Consult for Docker's Solution



The logos displayed include Docker, VMware, i for Business, DELL EQUALLOGIC, Check Point, f5, Sense CITRIX, and Citrix.



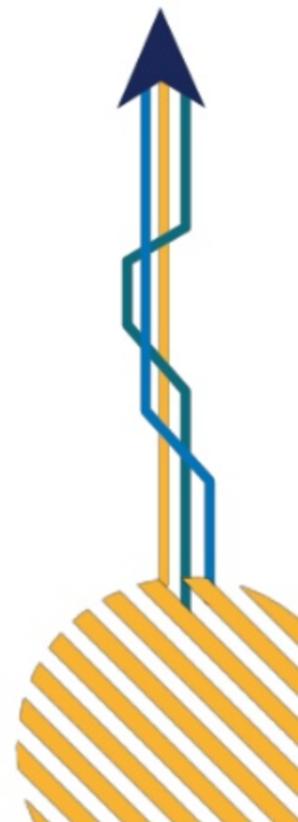
Present by: Praparn L. (eva10409@gmail.com)



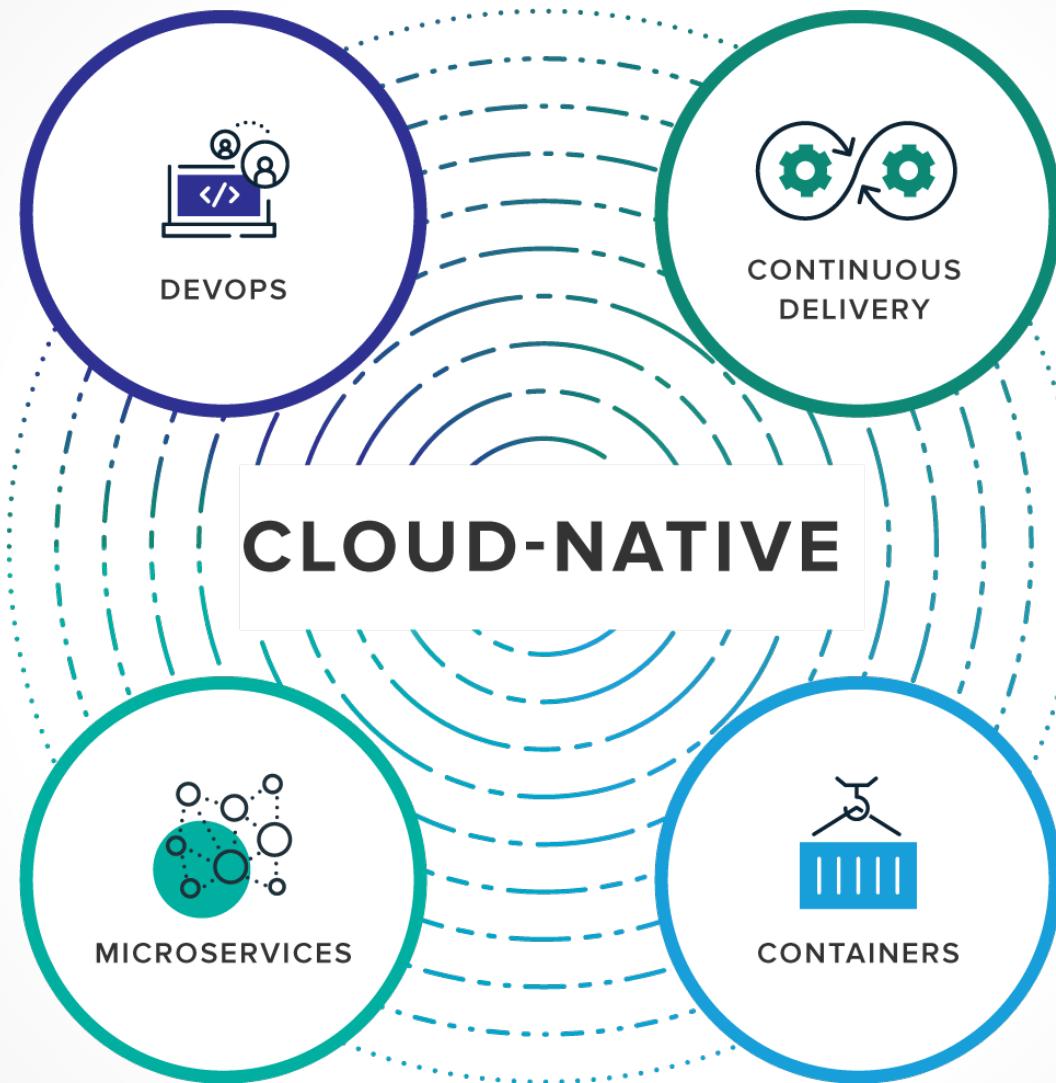
Landscape of the world now



Imagine how the
world should work



Landscape of the world now



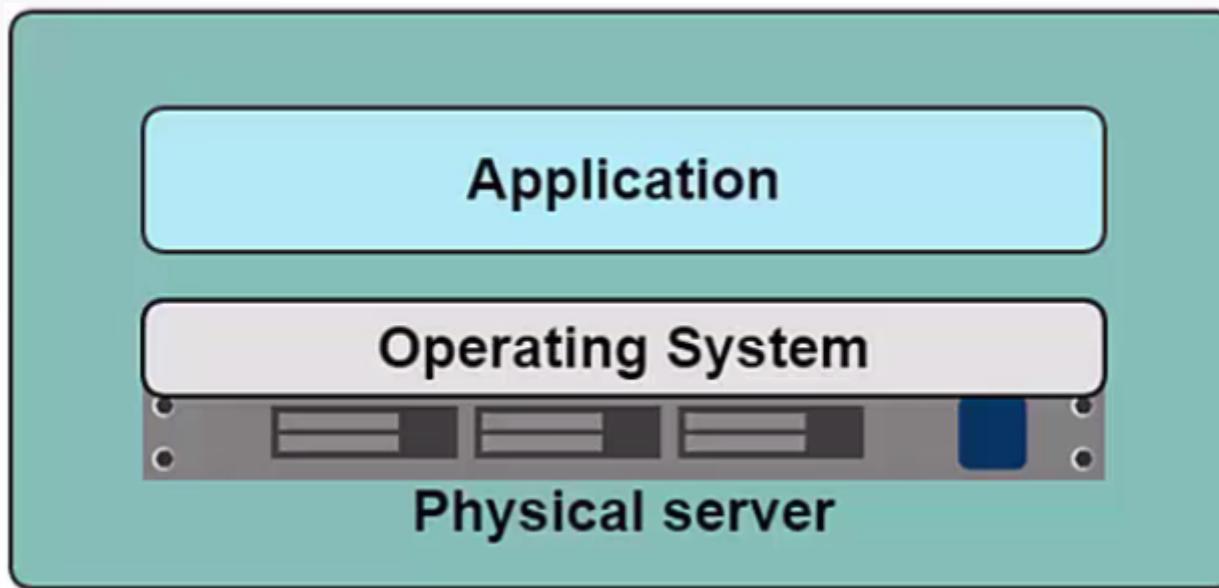
Kubernetes: Production Workload Orchestration



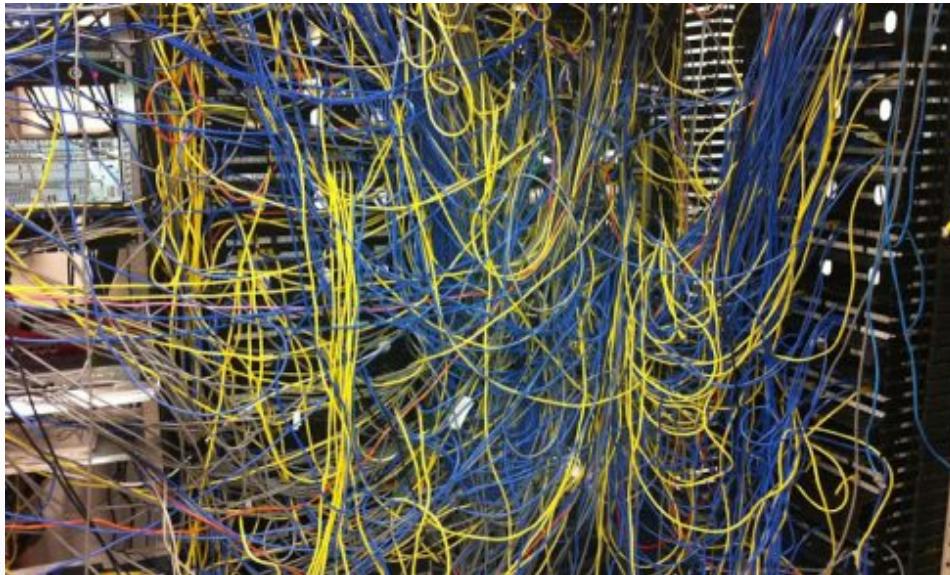
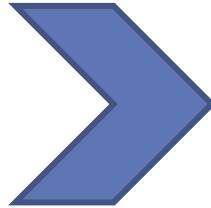
Docker Principle

• • •

What is docker ?



Existing Technology

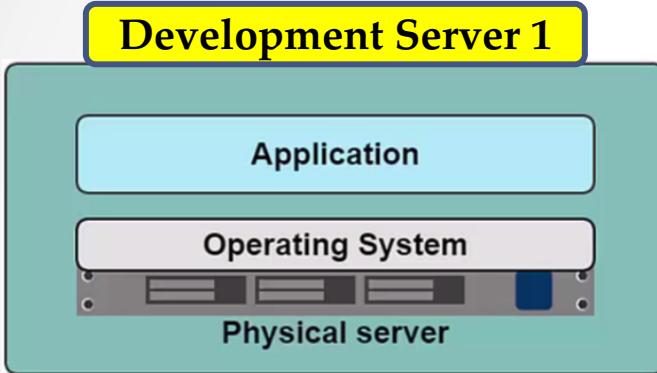


Docker: The Next-Gen of Virtualization

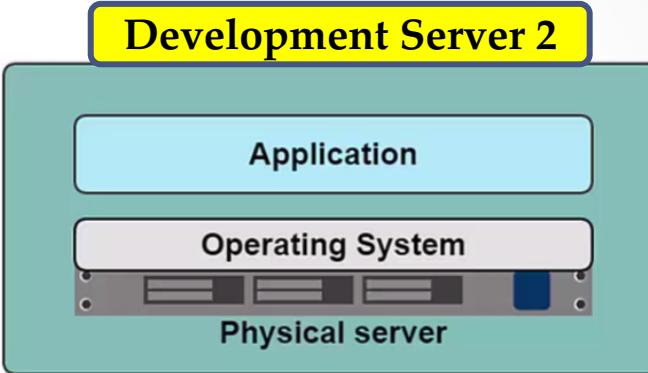


Existing Technology

- 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)

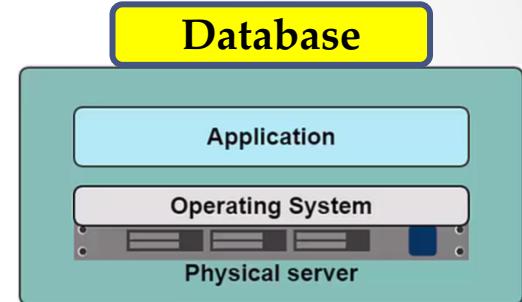
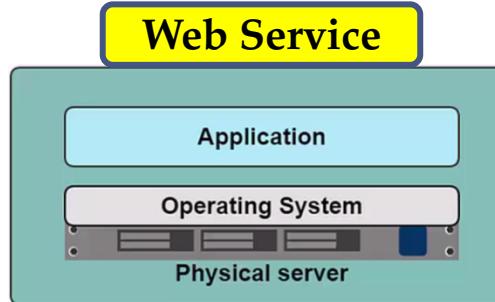
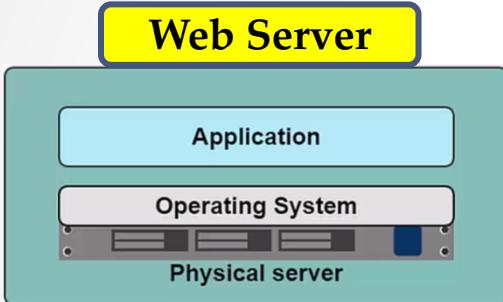


- IIS 8.0
- .Net Framework 3.5
- ASP.NET
- Etc

- 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

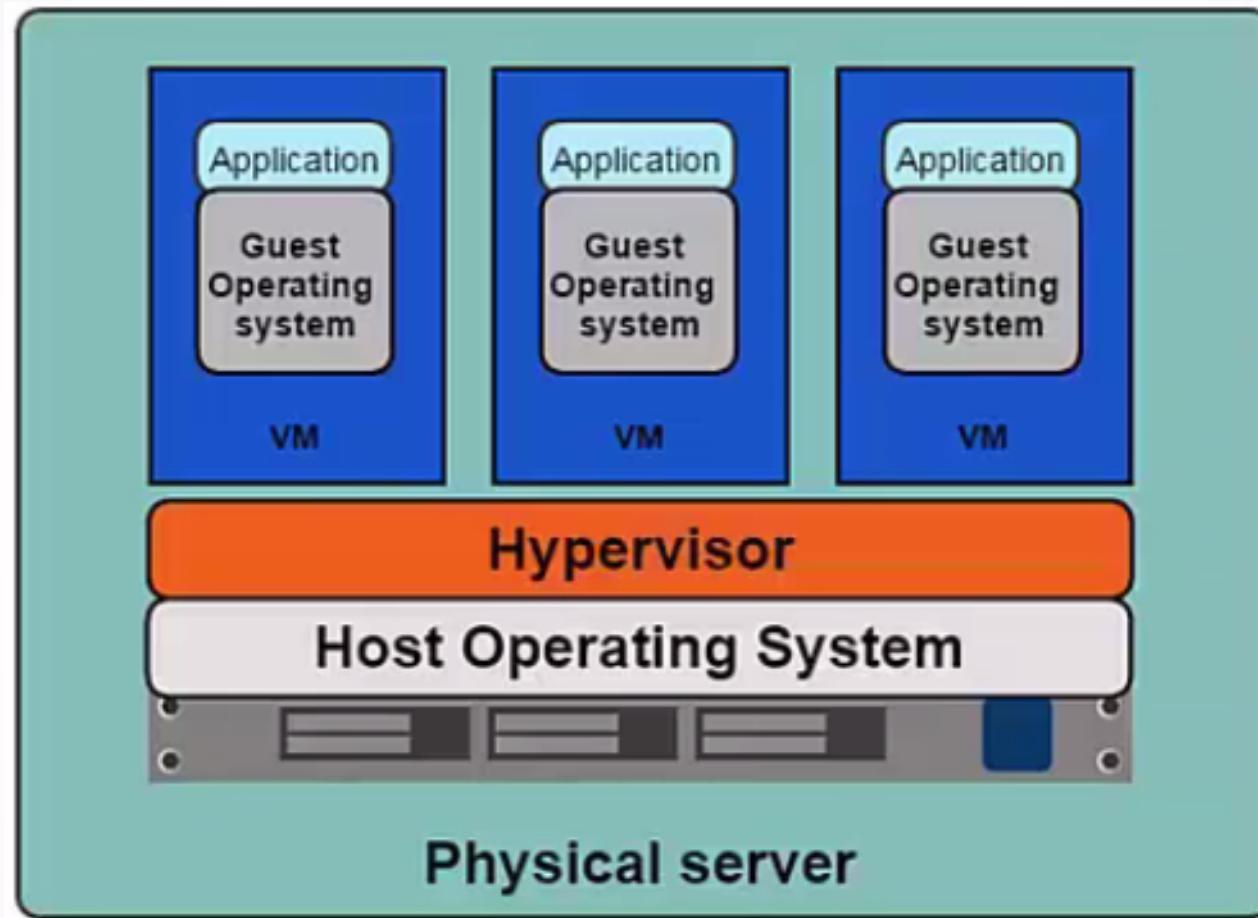
- Day 2: Application 2: Need to implement
 - Need PHP 7.0 ?
 - MariaDB 10.1.14 (Need search feature on 10.1)
- Problem ?
 - Possible to upgrade PHP to 7.0 ? / How to test existing application ?
 - What effect to MariaDB upgrade ?

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)

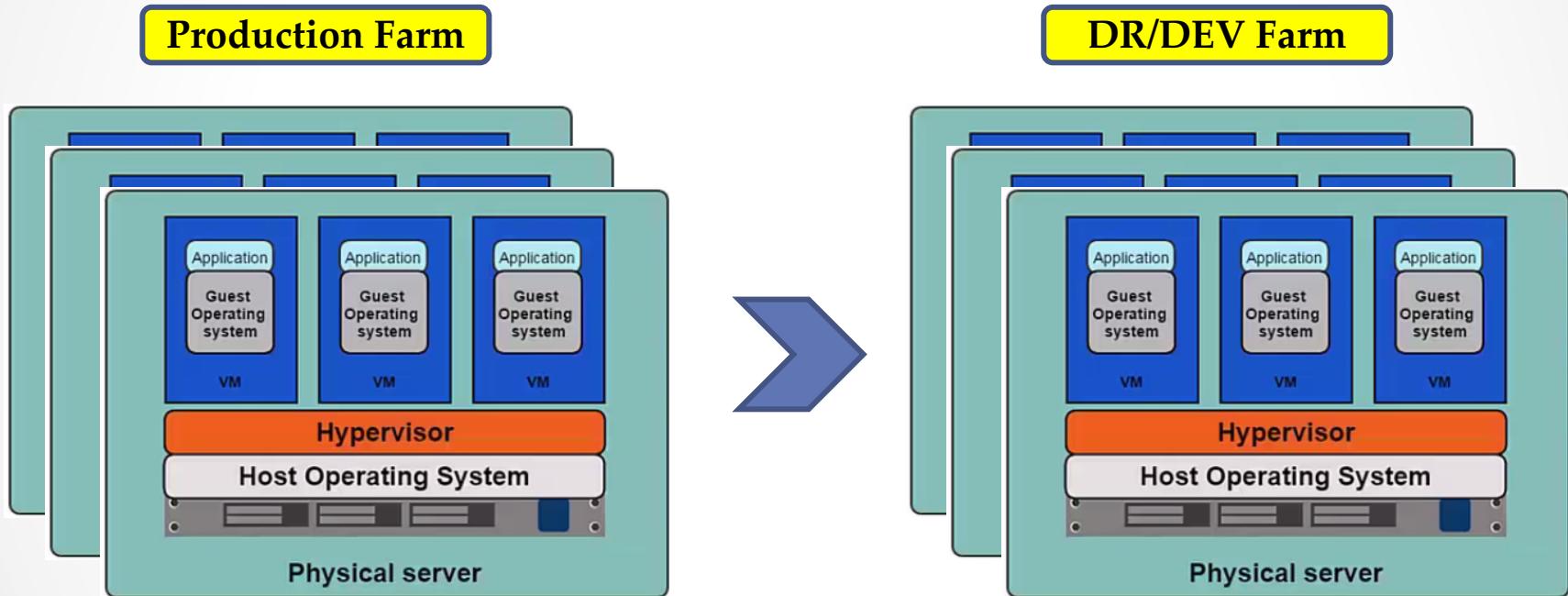


What is docker ?



Existing Technology

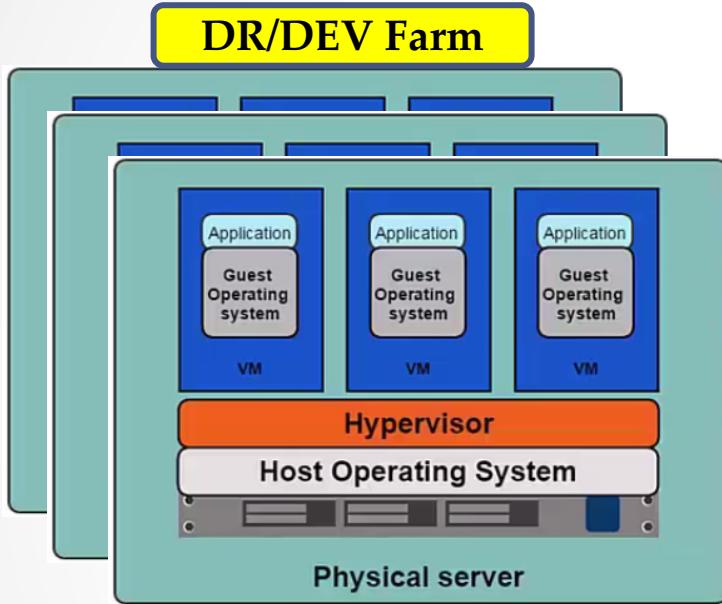
- 1 Physical server : 1 – N VMWare (&OS)
- Virtualize Hardware (CPU, Memory, Disk, Network etc)



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

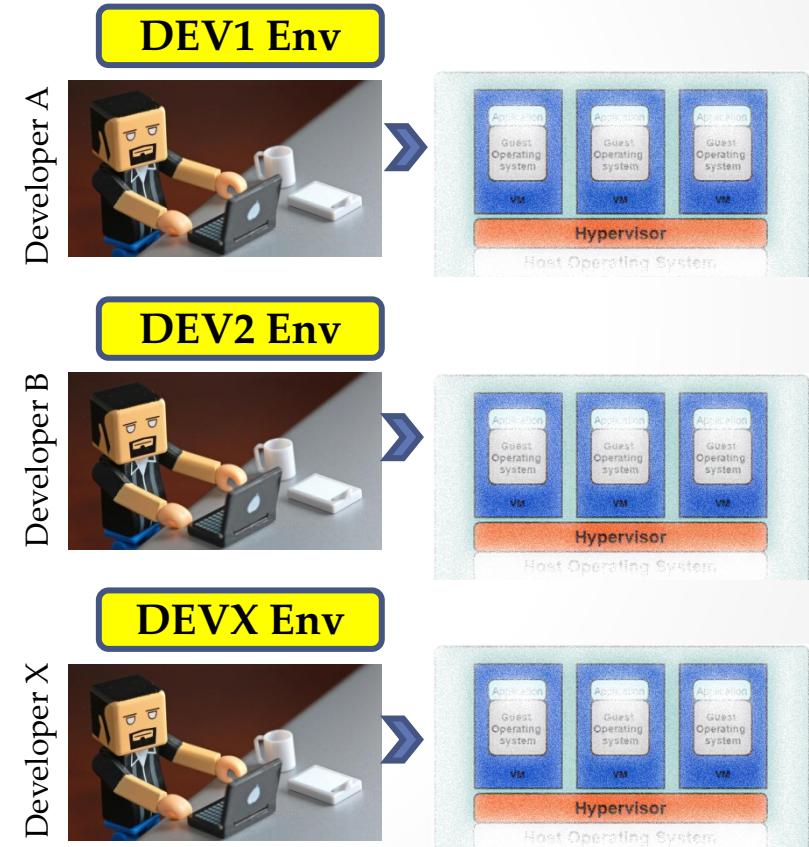
Existing Technology

- 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



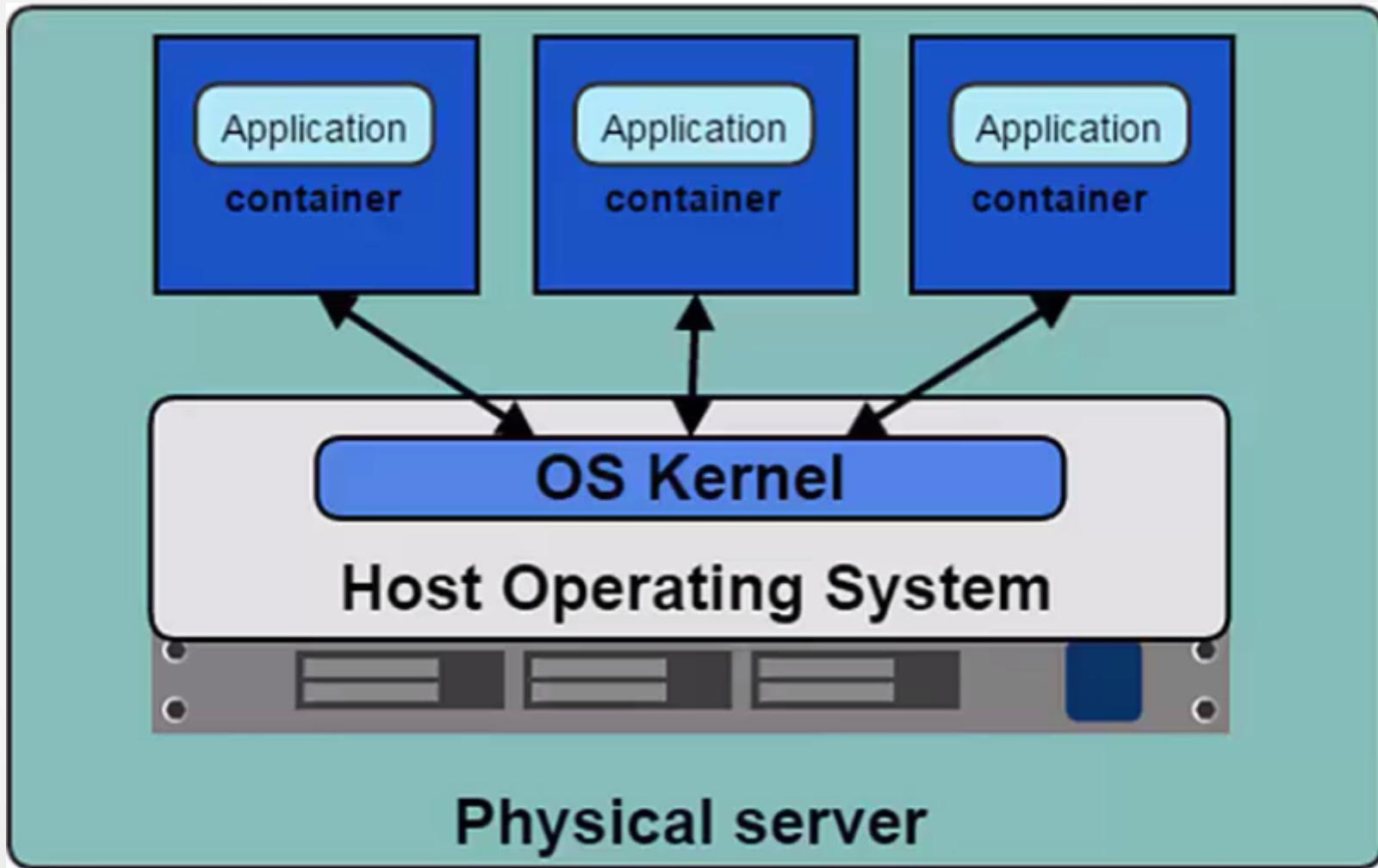
- Apache 2.20 Web Server
- PHP 5.5 Engine
- Laravel 4.1 Framework

- IIS 8
- .Net FrameWork 3.5

- MariaDB 5.1

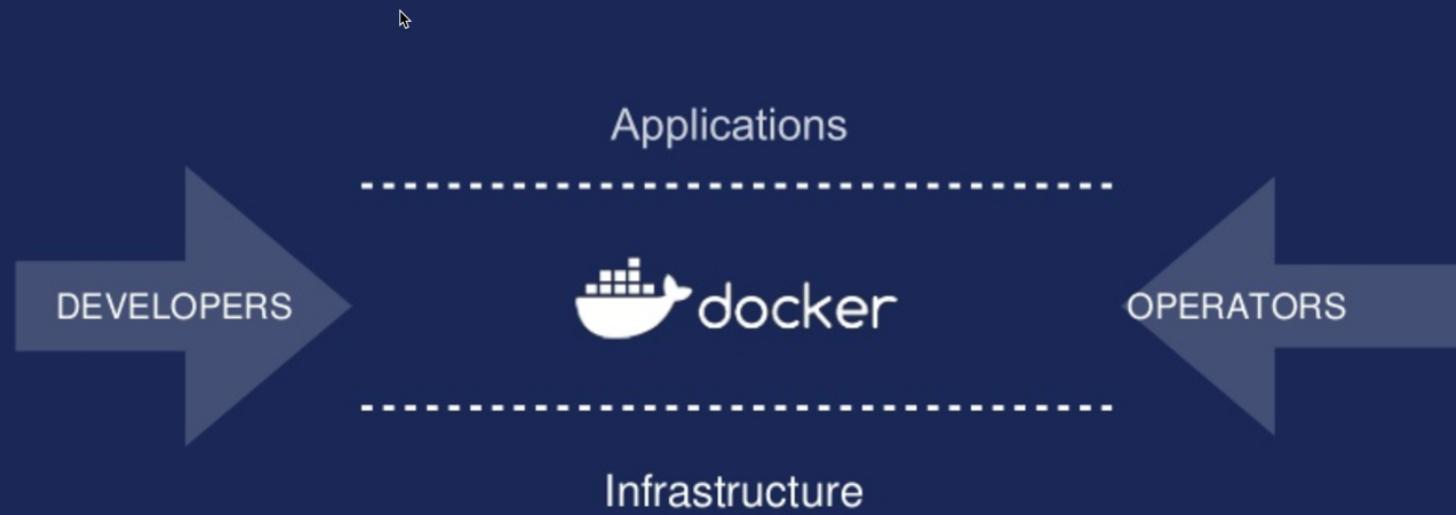
- Day 2: Application 2: Need to implement
 - Need PHP 7.0 ?
 - MariaDB 10.1.14 (Need search feature on 10.1)
- So... The problem still exist.

What is docker ?



What is docker ?

The Docker Platform in a nutshell



What is docker ?

Core Principles of the Docker Platform

INDEPENDENCE



OPENNESS



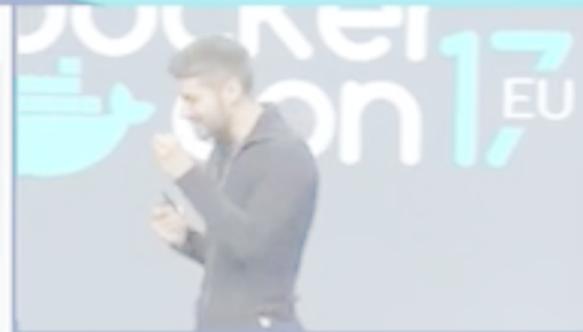
SIMPLICITY



 dockercon EU

What is docker ?

WE ARE
ONE BIG
COMMUNITY



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 (1.8.3) (Now)
- รองรับการติดตั้งบนบนลินุกซ์ 64 บิต (kernel 3.1.0) (Official)
 - Ubuntu
 - Debian
 - Red Hat Enterprise Linux
 - CentOS
 - Fedora
 - Microsoft Windows Server 2016
 - Suse Linux Enterprise Server
 - Raspberry PI

What is docker ?

Desktop

Platform	Docker CE x86_64	Docker CE ARM	Docker EE
Docker for Mac (macOS)	✓		
Docker for Windows (Microsoft Windows 10)	✓		

Cloud

Platform	Docker CE x86_64	Docker CE ARM	Docker EE
Amazon Web Services	✓		✓
Microsoft Azure	✓		✓

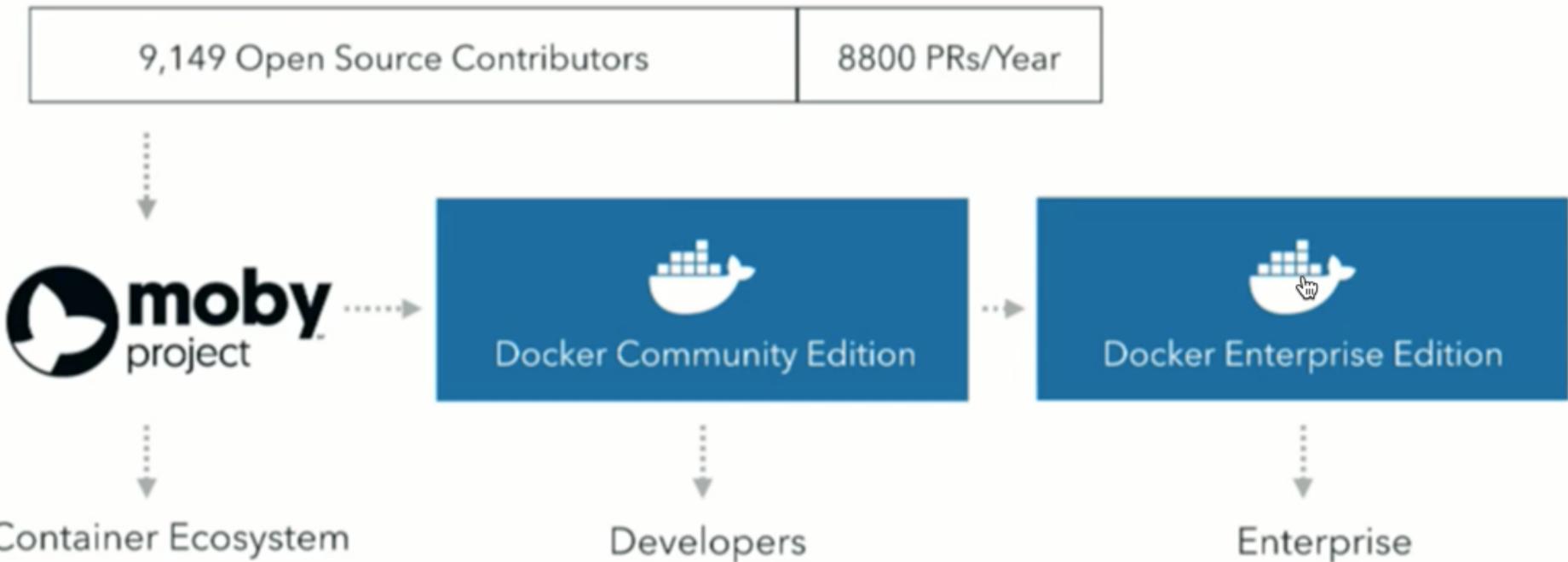
See also [Docker Cloud](#) for setup instructions for Digital Ocean, Packet, SoftLink, or Bring Your Own Cloud.

Server

Platform	Docker CE x86_64	Docker CE ARM	Docker CE System Z (s390x)	Docker EE
CentOS	✓			✓
Debian	✓	✓		
Fedora	✓			
Microsoft Windows Server 2016				✓
Oracle Linux				✓
Red Hat Enterprise Linux				✓
SUSE Linux Enterprise Server				✓
Ubuntu	✓	✓	✓	✓

What is docker ?

The Docker Innovation Model



What is docker ?



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

Enterprise Edition (EE)

- CaaS enabled platform subscription (integrated container orchestration, management and security)
- Enterprise class support
- Quarterly releases, supported for one year each with backported patches and hotfixes.
- Certified Infrastructure, Plugins, Containers

Community Edition (CE)

- Free Docker platform for "do it yourself" dev and ops
- Monthly Edge release with latest features for developers
- Quarterly release with maintenance for ops

Lifecycle

Squaring the circle: Faster releases and better stability



Docker EE Availability

From Docker



AUTHORIZED RESELLER

docker

OEM: Direct L2 / L2 Support Included



Cloud Marketplaces

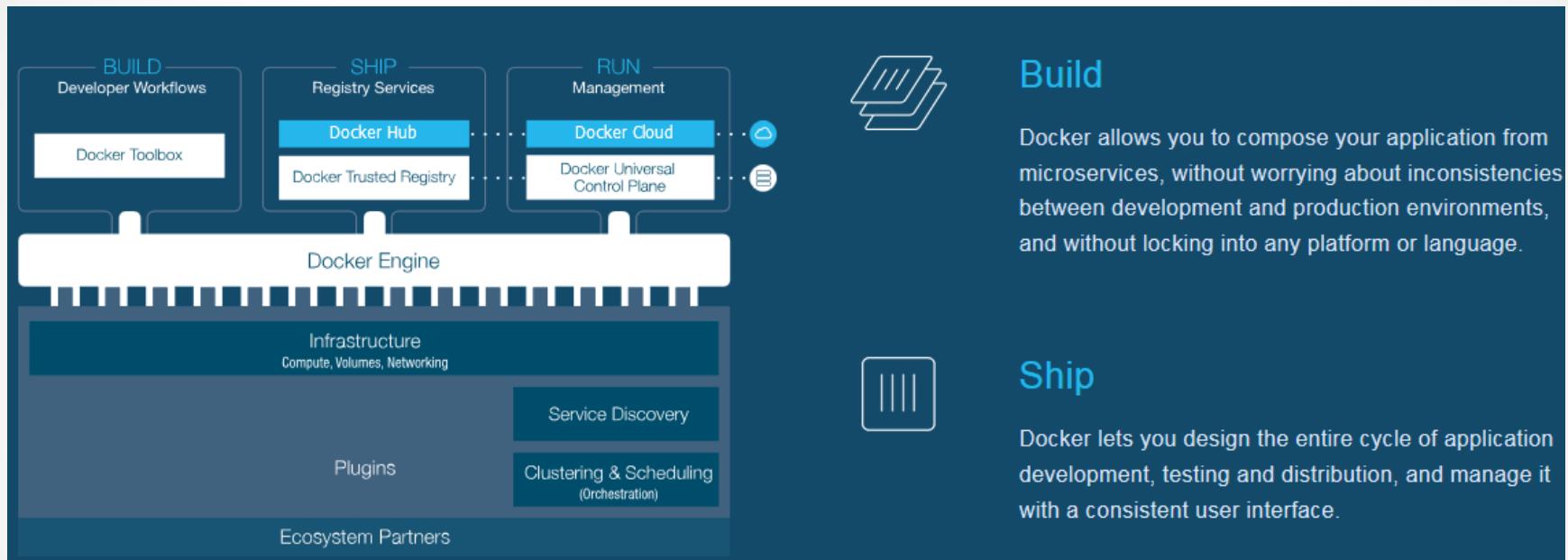


Ref: <https://blog.docker.com/2017/03/docker-online-meetup-recap-docker-enterprise-edition-ee-community-edition-ce/>

Docker: The Next-Gen of Virtualization



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

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 ^[11]	Yes	Yes ^[10]
Virtuozzo	Linux, Windows	Proprietary	2000 ^[14]	Yes	Yes	Yes	Yes ^[11]	Yes	Yes	Yes ^[11]	Partial ^[12]	Yes	Yes
Solaris Containers (Zones)	illumos (OpenSolaris), Solaris	CDDL, Proprietary	2004	Yes	Yes (ZFS)	Yes	Partial ^[13]	Yes	Yes	Yes ^{[13][17][18]}	Partial ^[13]	Partial ^[13]	Yes ^[8]
FreeBSD jail	FreeBSD	BSD License	2000 ^[20]	Yes	Yes (ZFS)	Yes ^[14]	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	?
WPARs	AIX	Proprietary	2007	Yes	No	Yes	Yes	Yes	Yes	Yes ^[24]	No	Yes ^[25]	?
HP-UX Containers (SRP) ^[9]	HPUX	Proprietary	2007	Yes	No	Partial ^[15]	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
		Proprietary	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



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 frame.

Solomon Hykes

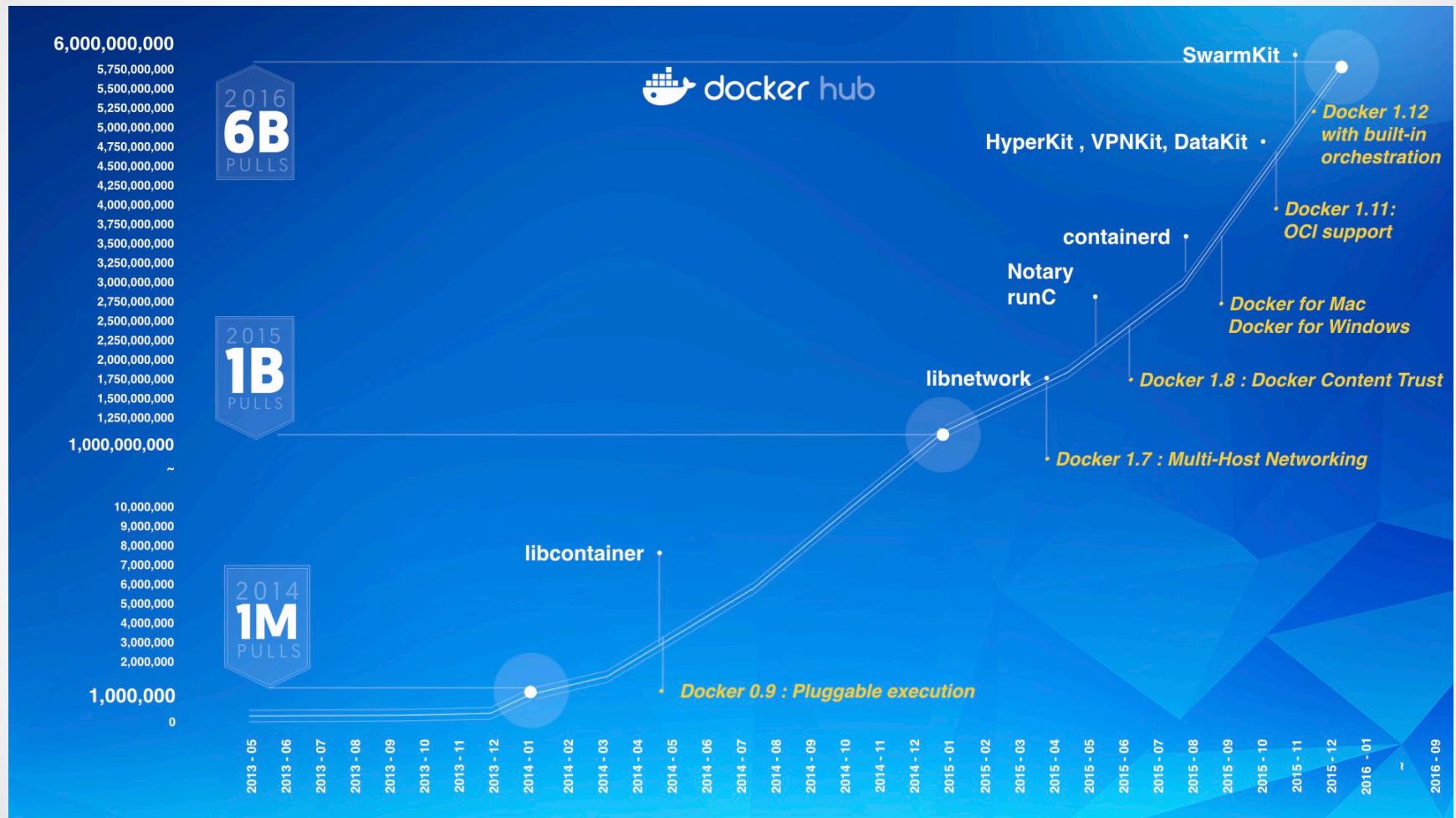
CTO and Founder

 dockercon¹⁷ EU

Docker: The Next-Gen of Virtualization



Growth of Docker (2016 stats)

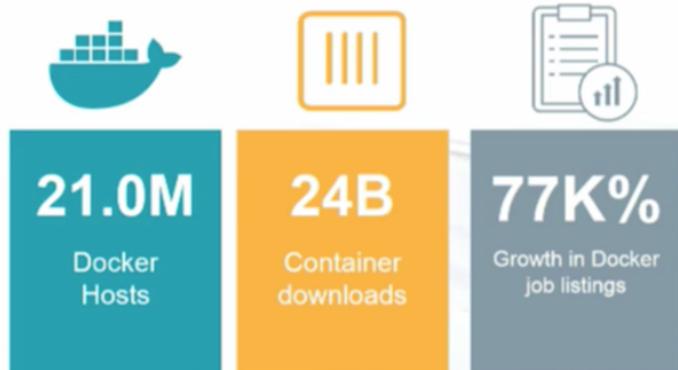


Docker: The Next-Gen of Virtualization



Growth of Docker (Now)

Docker Momentum

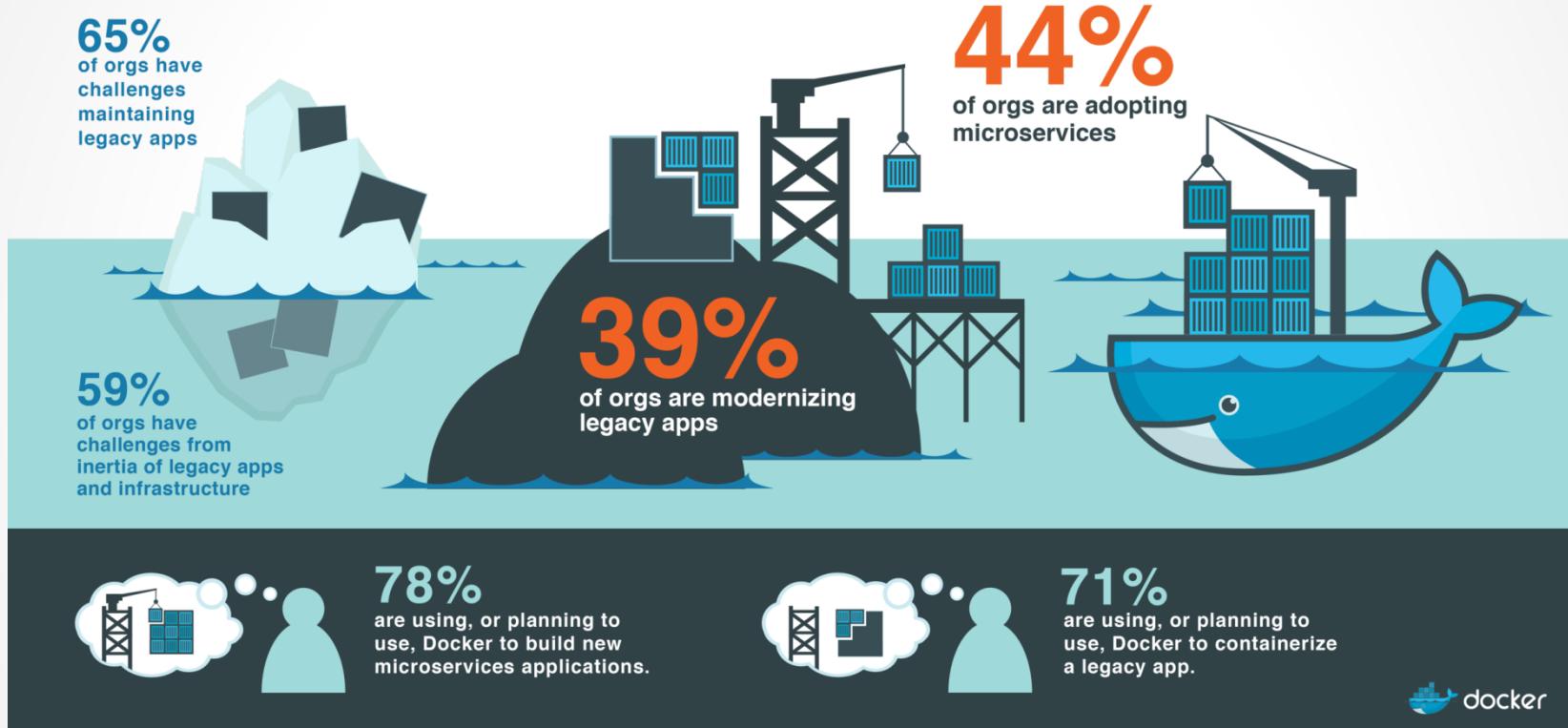


Industry Standards



docker
con 17 EU

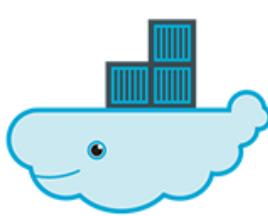
Trend of Modernize Application



Cloud Strategy Survey

80%

say Docker is part
of cloud strategy



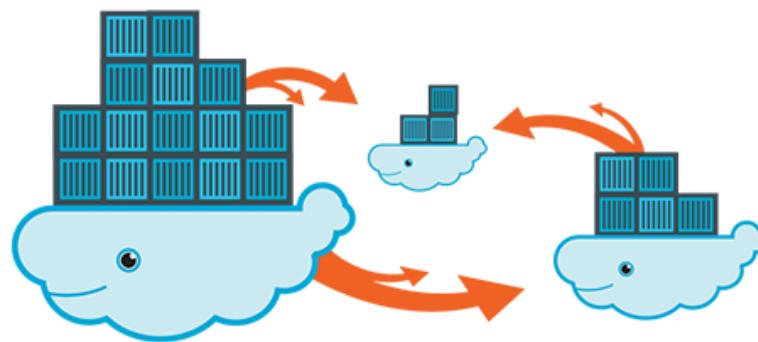
60%

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 ?

Breaking the
Pattern:
Docker
Enterprise
and
Microservices



Who use docker now ?

Visa Today

- Live in production for 6 months
- 100K transactions per day
- 10X app scalability
- Multiple clusters across multiple regions

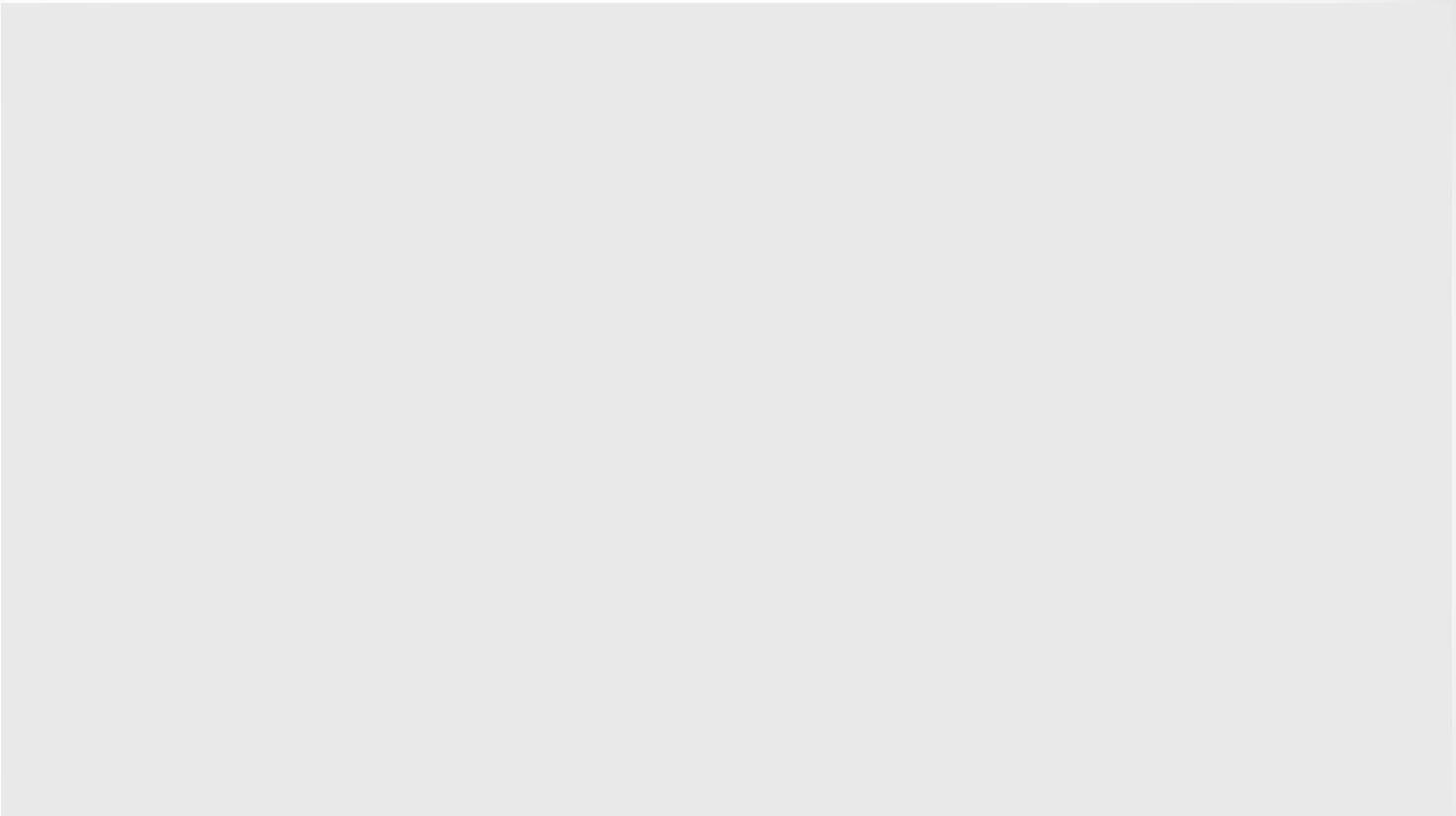


Who use docker now ?

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

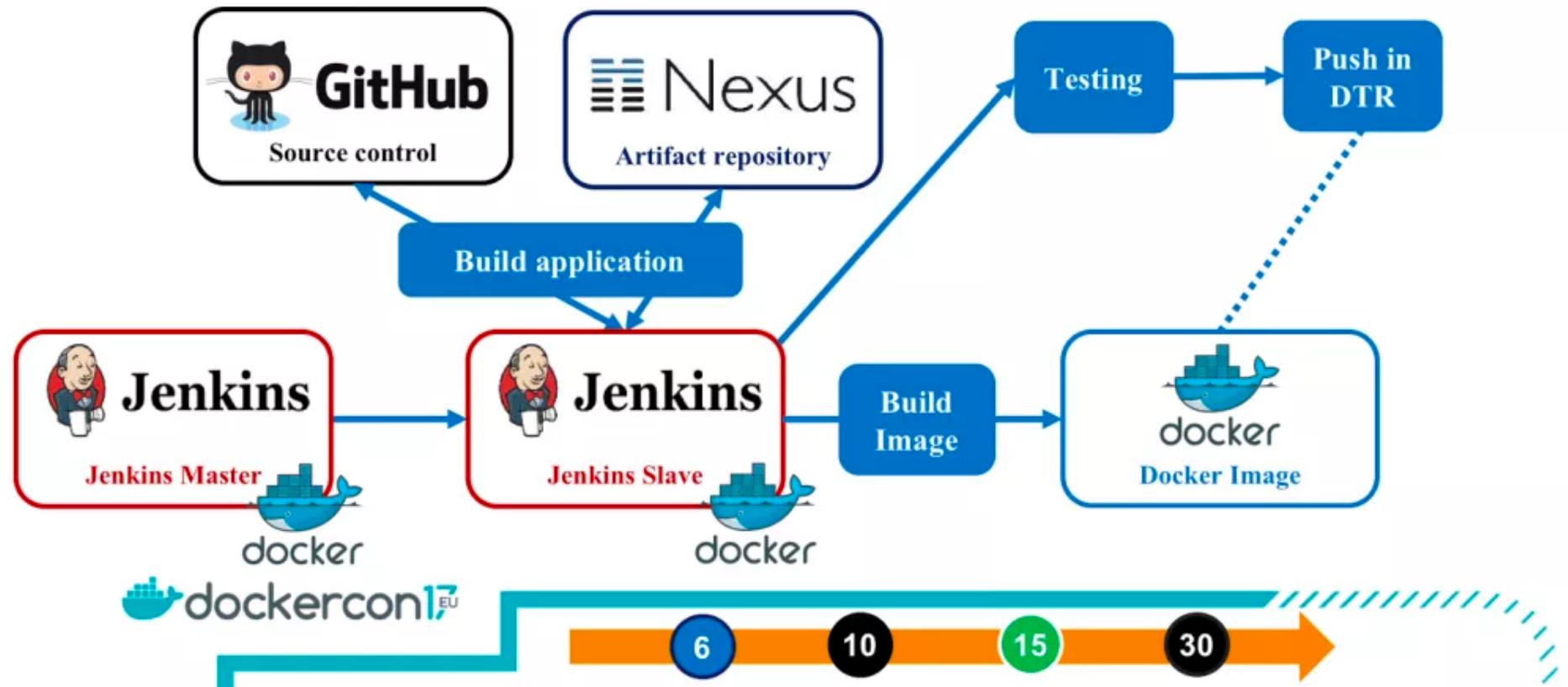
Workshop Automate Software Development



Who use docker now ?



Level 1 - Build

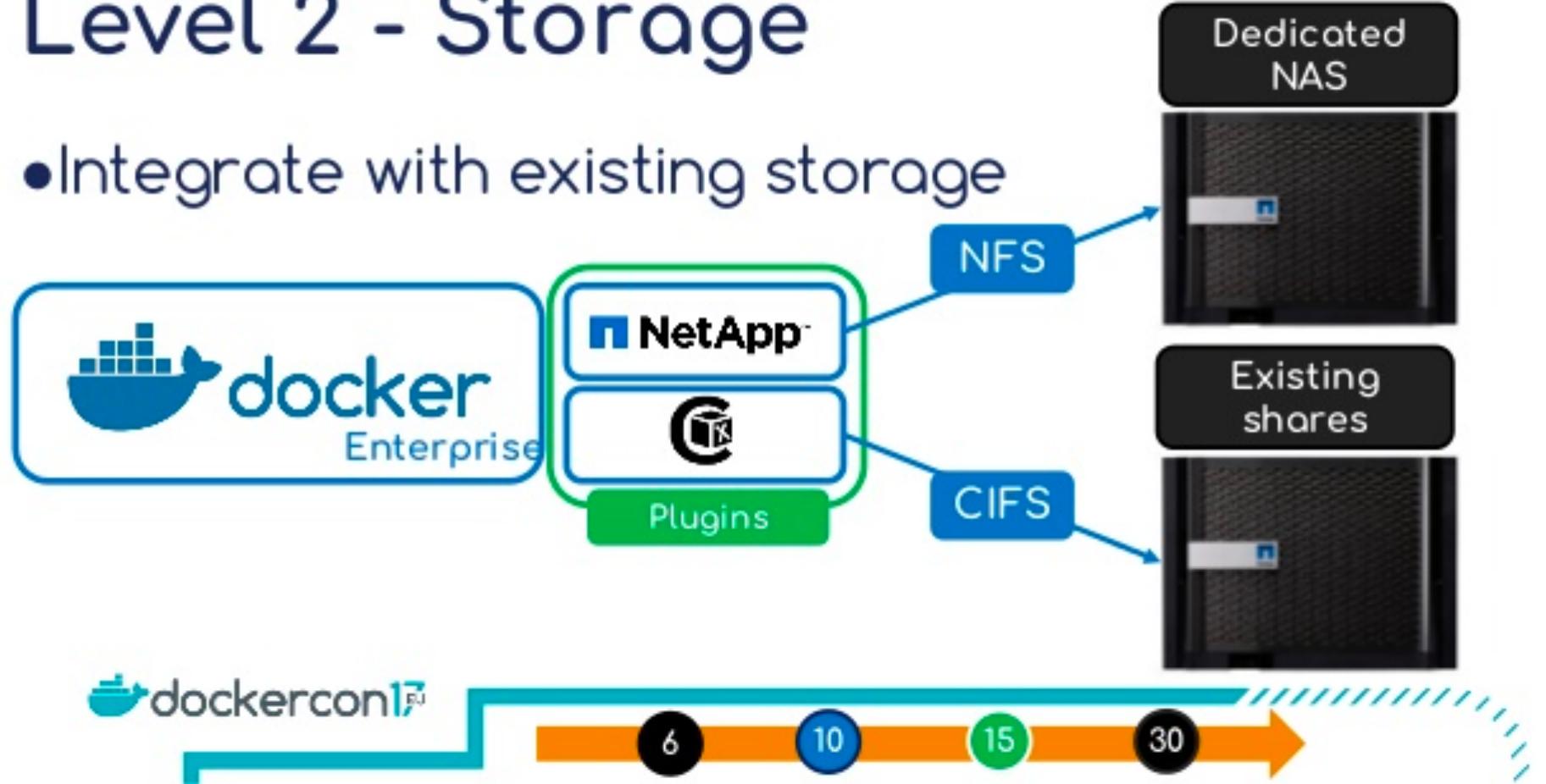


Who use docker now ?

SOCIETE
GENERALE

Level 2 - Storage

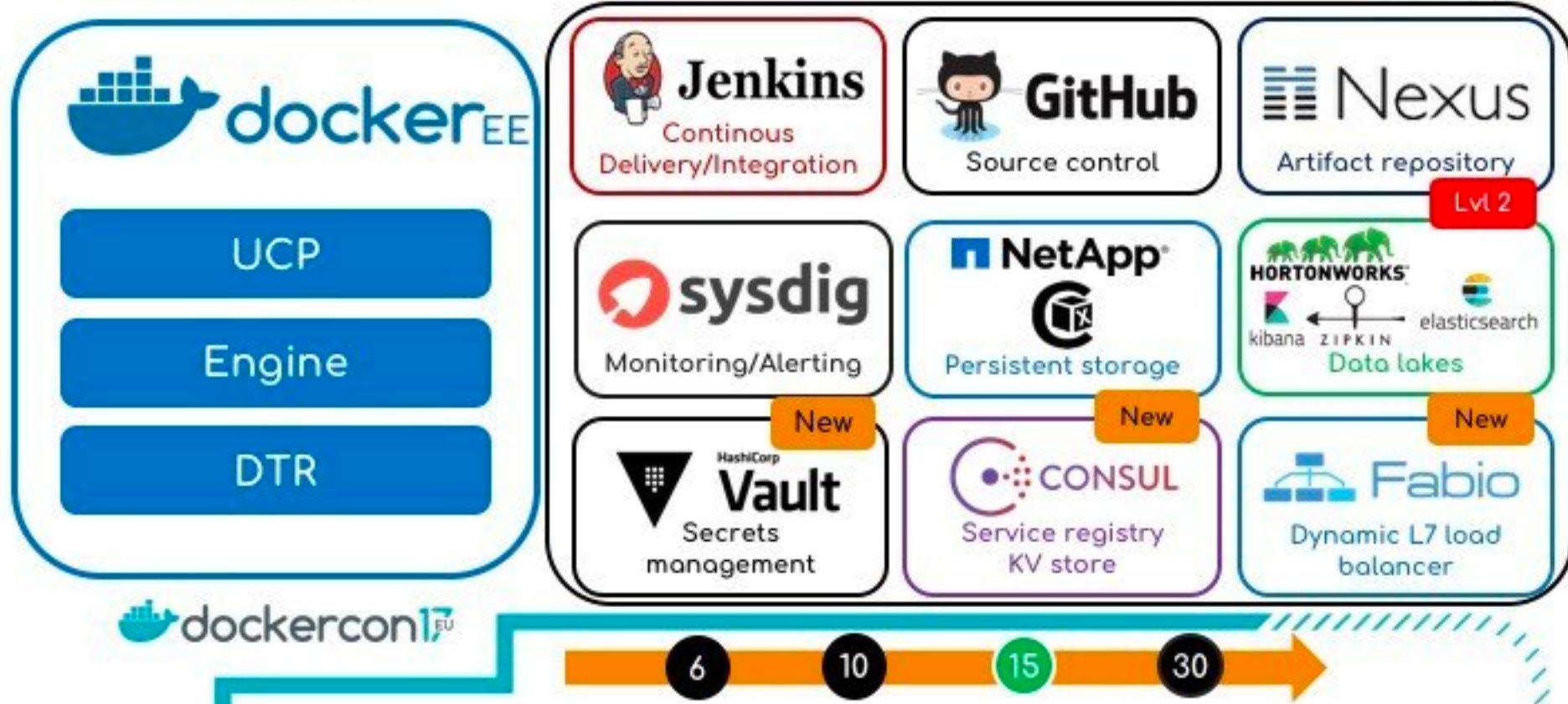
- Integrate with existing storage



Who use docker now ?

Level 3

SOCIETE
GENERALE



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

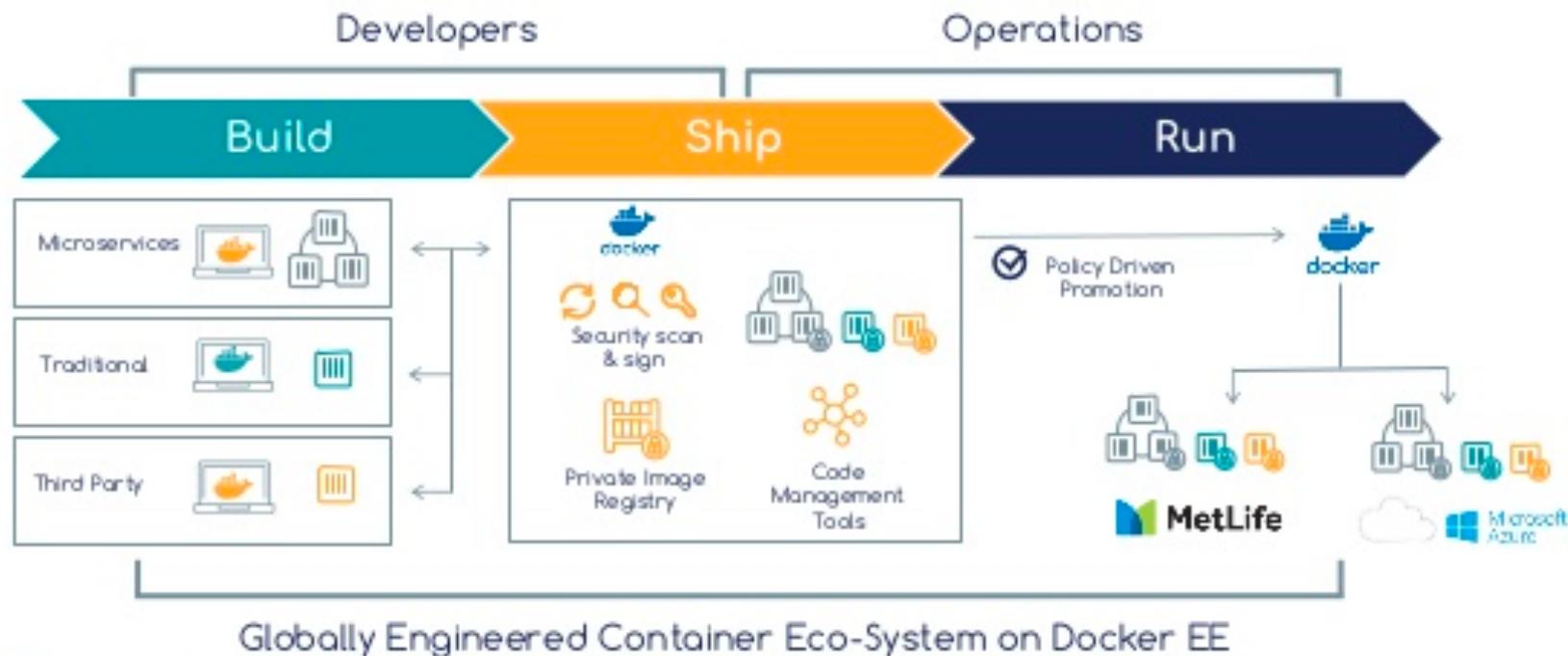
**11:15 AM - Docker 0 to 60 in 5 Months:
How a Traditional Fortune 40 Company
Turns on a Dime**

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

=

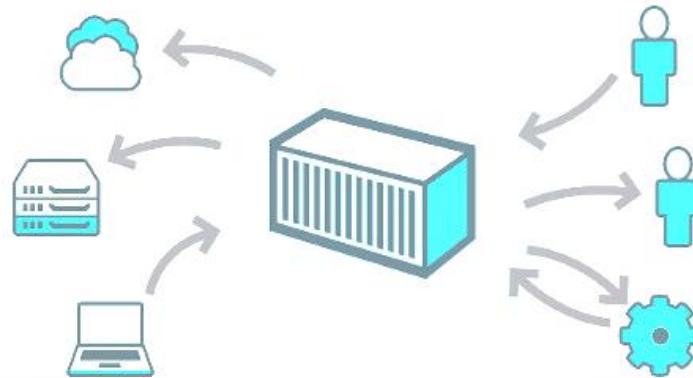
-66%

Cost Reduction

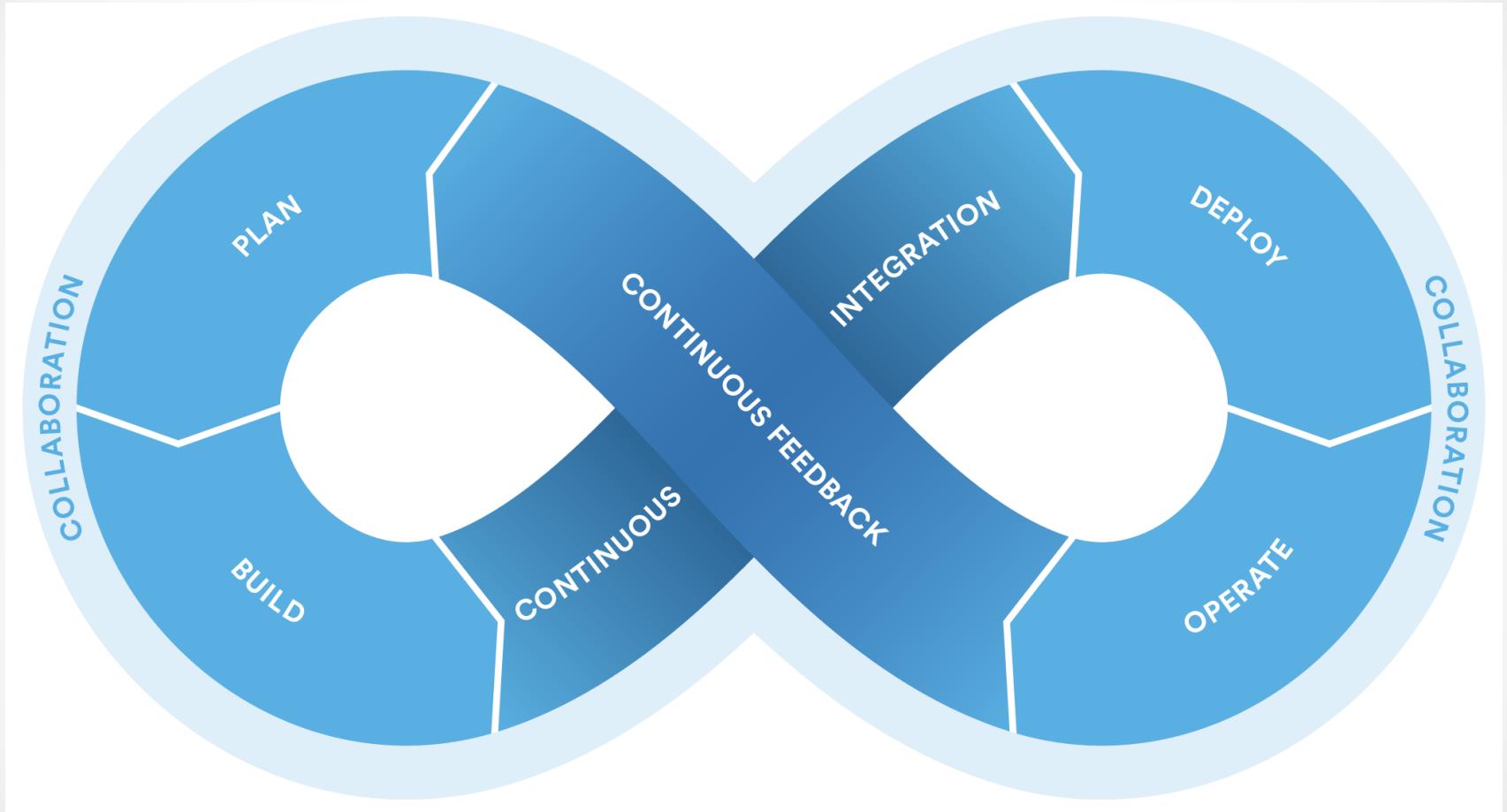


Container Life Cycle

- 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)

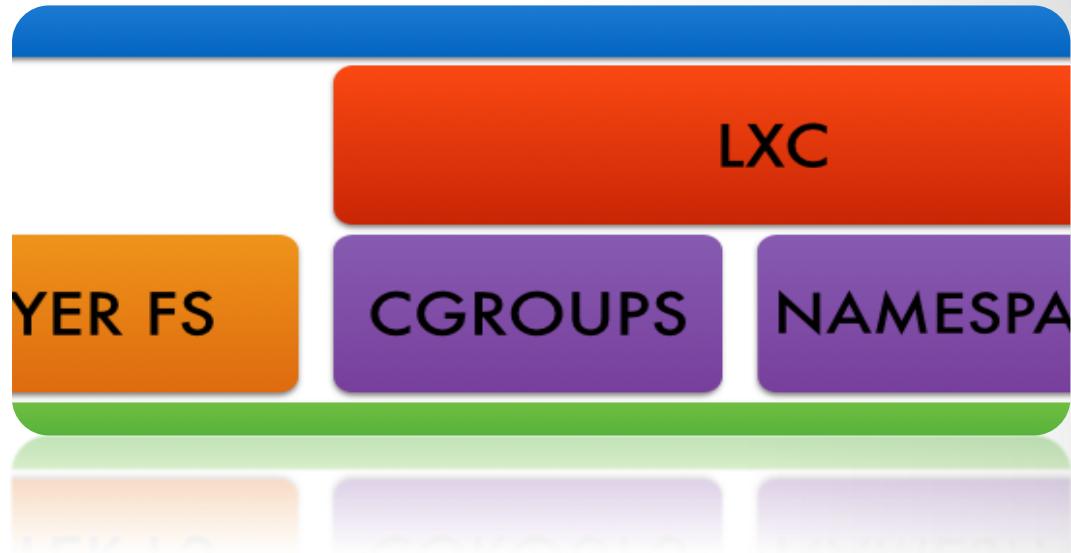


Container Life Cycle



Docker Architecture

- Docker Engine
 - Docker Client
(CLI & GUI)
 - Docker Daemon
- Docker Hub
 - Cloud service
 - Private Registry
- Docker images
- Docker containers



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 config)
- สามารถรันโปรแกรมได้บนเครื่องแม่ข่ายทุกๆระบบปฏิบัติการฯ ที่ติดตั้ง docker ได้
- สามารถ scale-out ได้ง่ายในอนาคต
- World open for docker !!!

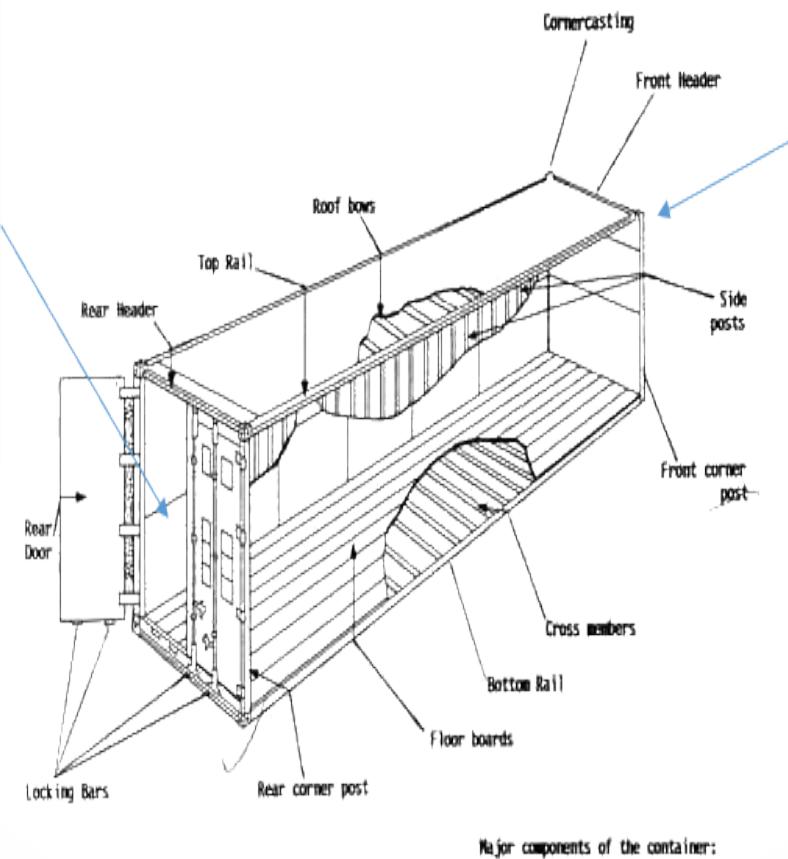
Separate of Concern

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

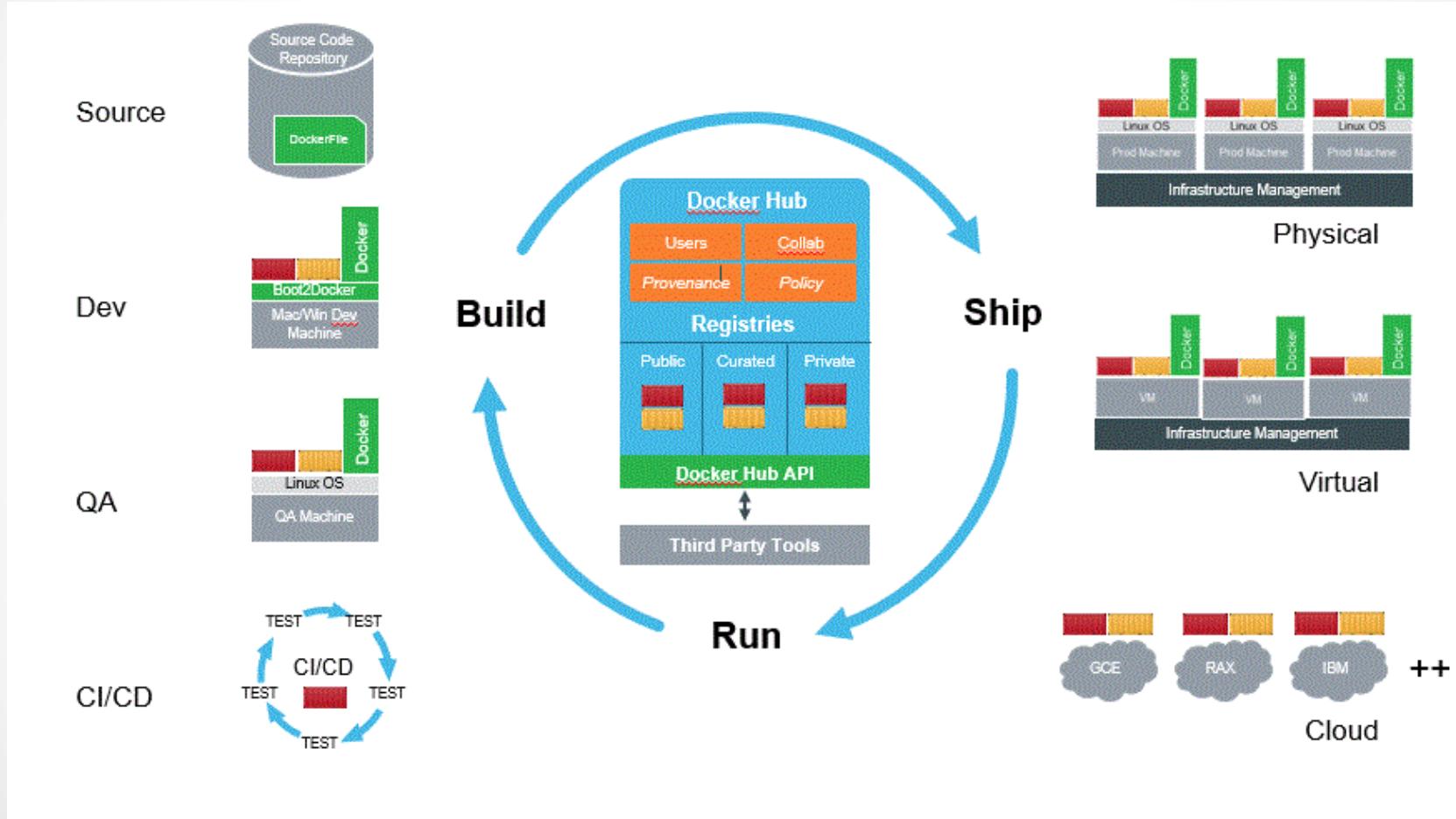
Oscar the Ops Guy

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

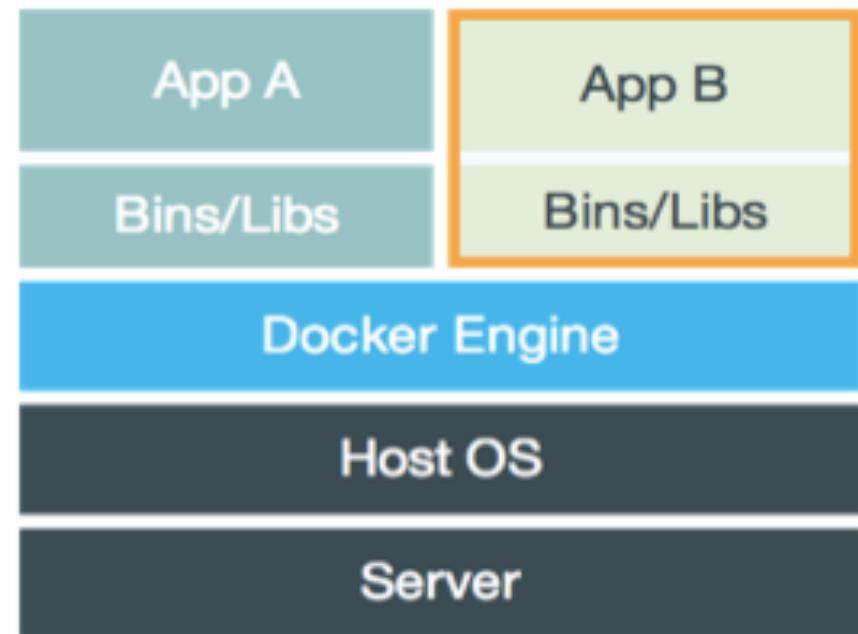
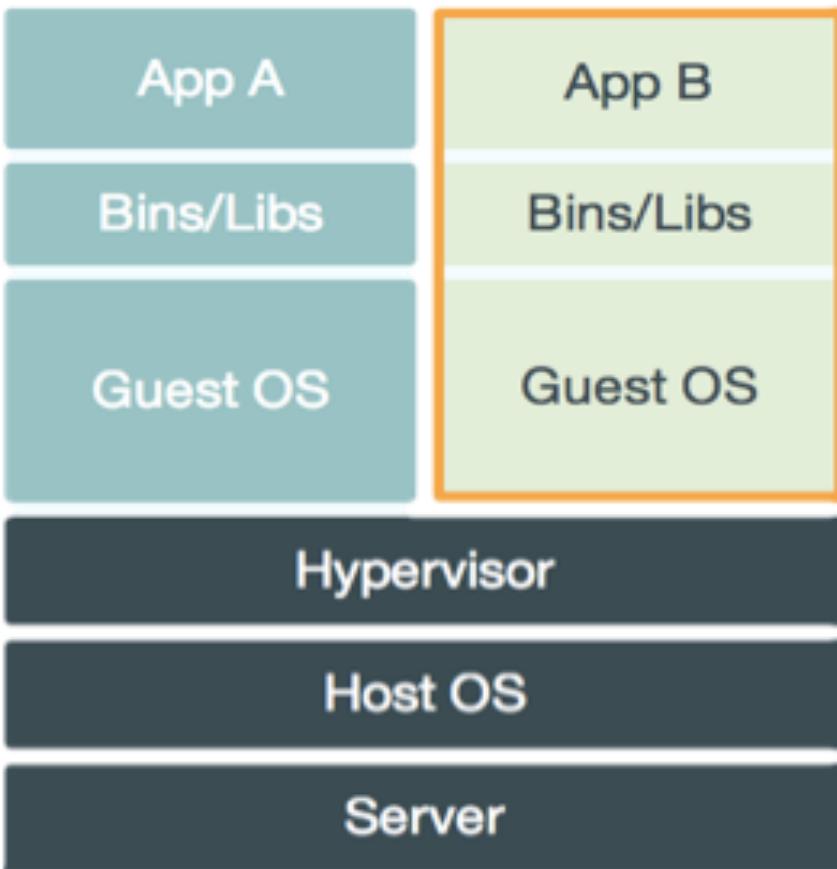


Benefit of docker for DevOps

- Build-Ship-Run



Docker vs VMWare



Docker: The Next-Gen of Virtualization



Docker vs VMWare



Docker: The Next-Gen of Virtualization



Docker Version

```
docker@labdocker:~$ docker version
Client:
Version: 18.01.0-ce
API version: 1.35
Go version: go1.9.2
Git commit: 03596f5
Built: Wed Jan 10 20:05:20 2018
OS/Arch: linux/amd64
Experimental: false
Orchestrator: swarm

Server:
Engine:
Version: 18.01.0-ce
API version: 1.35 (minimum version 1.12)
Go version: go1.9.2
Git commit: 03596f5
Built: Wed Jan 10 20:13:12 2018
OS/Arch: linux/amd64
Experimental: false
docker@labdocker:~$
```

```
docker@labdocker:~$ docker system info
Containers: 0
Running: 0
Paused: 0
Stopped: 0
Images: 9
Server Version: 18.01.0-ce
Storage Driver: aufs
Root Dir: /mnt/sda1/var/lib/docker/aufs
Backing Filesystem: extfs
Dirs: 43
Dirperm1 Supported: true
Logging Driver: json-file
Cgroup Driver: cgroupfs
Plugins:
Volume: local
Network: bridge host 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: 89623f28b87a6004d4b785663257362d1658a729
runc version: b2567b37d7b75eb4cf325b77297b140ea686ce8f
init version: 949e6fa
Security Options:
seccomp
Profile: default
Kernel Version: 4.4.111-boot2docker
Operating System: Boot2Docker 18.01.0-ce (TCL 8.2.1); HEAD : 0bb7bbd - Thu Jan 11 16:32:39 UTC 2018
OSType: linux
Architecture: x86_64
CPUs: 1
Total Memory: 995.9MiB
Name: labdocker
ID: YS3D:ZHW5:0BP6:UG3W:DIA3:R7NZ:DVUQ:6GHZ:P626:ZQZH:W7N5:W2ZY
Docker Root Dir: /mnt/sda1/var/lib/docker
Debug Mode (client): false
Debug Mode (server): true
File Descriptors: 19
Goroutines: 34
System Time: 2018-01-20T03:40:06.696661116Z
EventsListeners: 0
Username: labdockerthailand
Registry: https://index.docker.io/v1/
Labels:
provider=virtualbox
Experimental: false
Insecure Registries:
127.0.0.0/8
Live Restore Enabled: false
docker@labdocker:~$
```

What's New

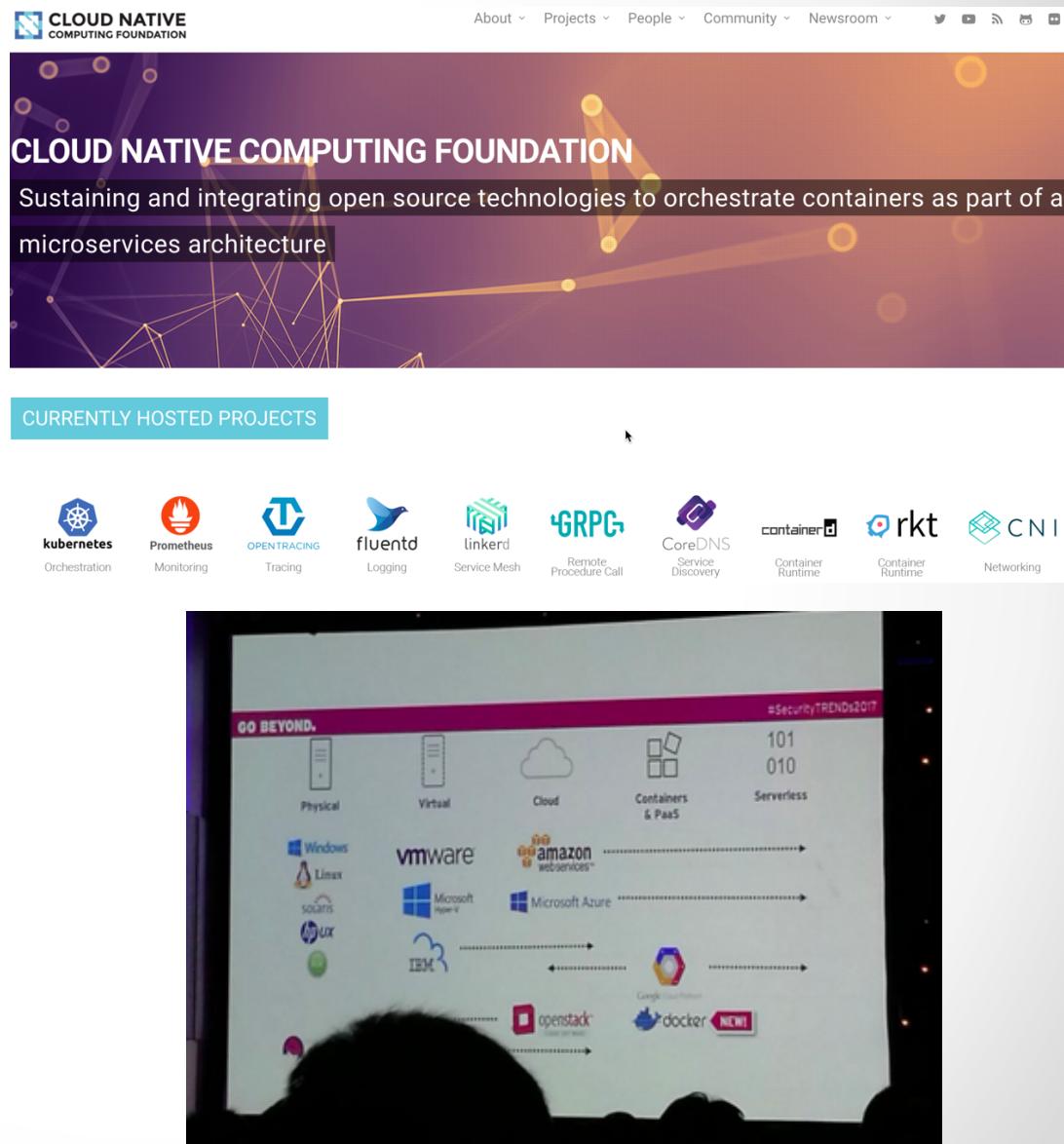


Docker's History of contribution to the OCI

Docker has lead the development of OCI from the initial commit to the donation of runc and Docker V2 Image format as a base for the image format specification.

2014	2015	2016	2017
● FEBRUARY 18 First commit of libcontainer	● APRIL Support for V2 of content addressable Docker Registry API released - groundwork for the OCI Image Specification	● JANUARY Formation of the OCI TOB - 2 Docker maintainers - Michael Crosby and Diogo Monica join effort	● MARCH Docker donates containerd to the CNCF
● FEBRUARY 20 nsinit is developed - the inspiration for runC	● MAY 1 Docker moves to donate its image format and runtime	● APRIL Schema2 of Docker's V2 Image Format support	● APRIL Docker announces project Moby
● MAR 7 Docker 0.9 ships with libcontainer as the default runtime	● MAY/ JUNE Docker works with inaugural participants and Linux Foundation to donate container format/runtime	● JULY 1.0 of OCI runtime and image format	● DECEMBER Docker spins out containerd
● JUNE 22 Docker announces donation of base container format and runtime, runc*, the cornerstone for the OCI			
● DECEMBER 17 Docker announces containerd, a daemon to manage runc			

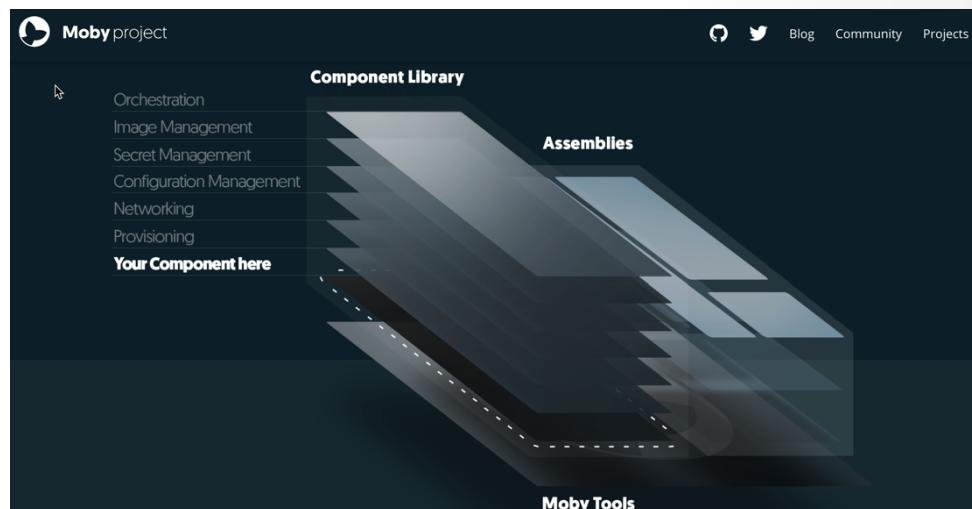
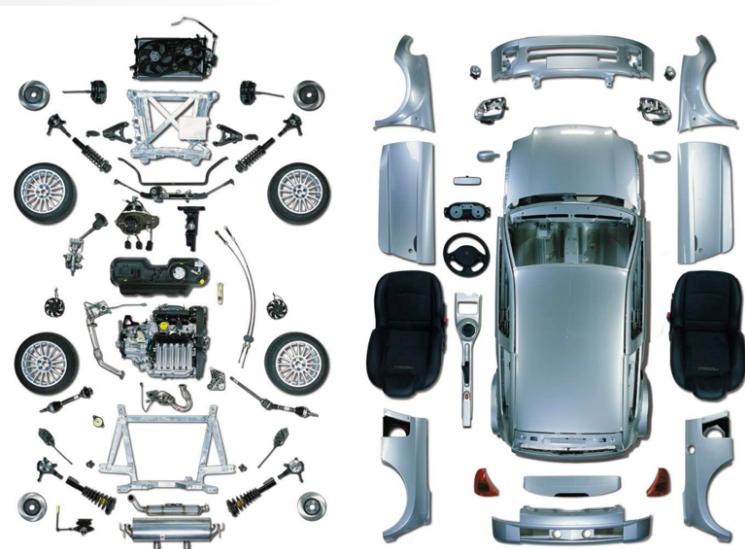
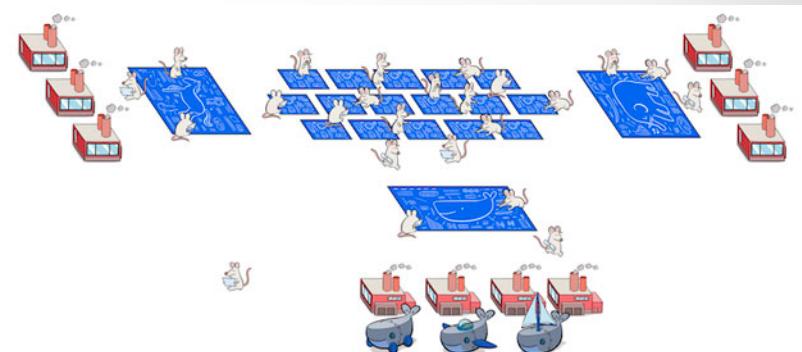
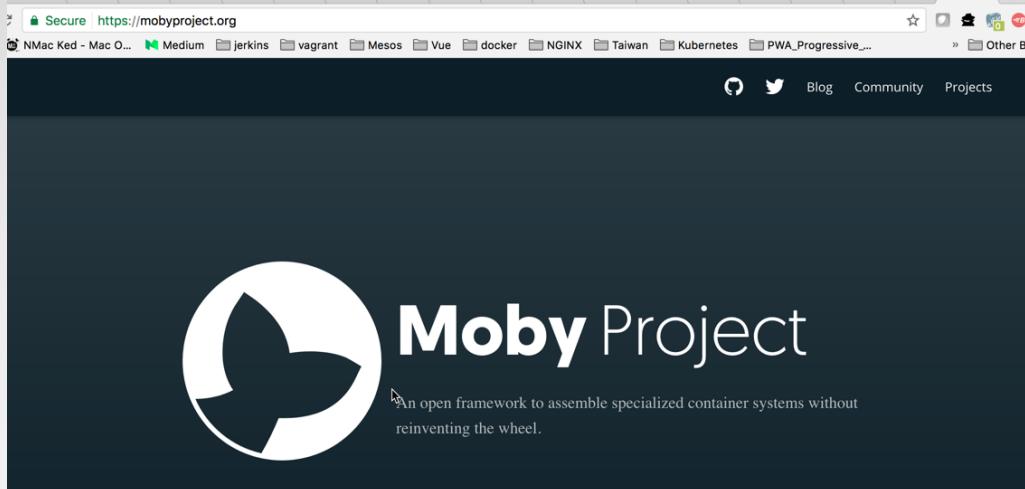
Docker: The Next-Gen of Virtualization



The Cloud Native Computing Foundation (CNCF) homepage features a purple background with abstract geometric shapes. The header includes the CNCF logo and navigation links for About, Projects, People, Community, and Newsroom. Below the header, the text "Sustaining and integrating open source technologies to orchestrate containers as part of a microservices architecture" is displayed. A section titled "CURRENTLY HOSTED PROJECTS" lists various projects with their logos: Kubernetes (Orchestration), Prometheus (Monitoring), OpenTracing (Tracing), fluentd (Logging), linkerd (Service Mesh), gRPC (Remote Procedure Call), CoreDNS (Service Discovery), containerd (Container Runtime), orkt (Container Runtime), and CNI (Networking). At the bottom, a large image shows a person's silhouette against a screen displaying a "GO BEYOND" slide with various cloud and container-related icons and statistics like "101 010" and "Serverless".



What's New



Docker: The Next-Gen of Virtualization



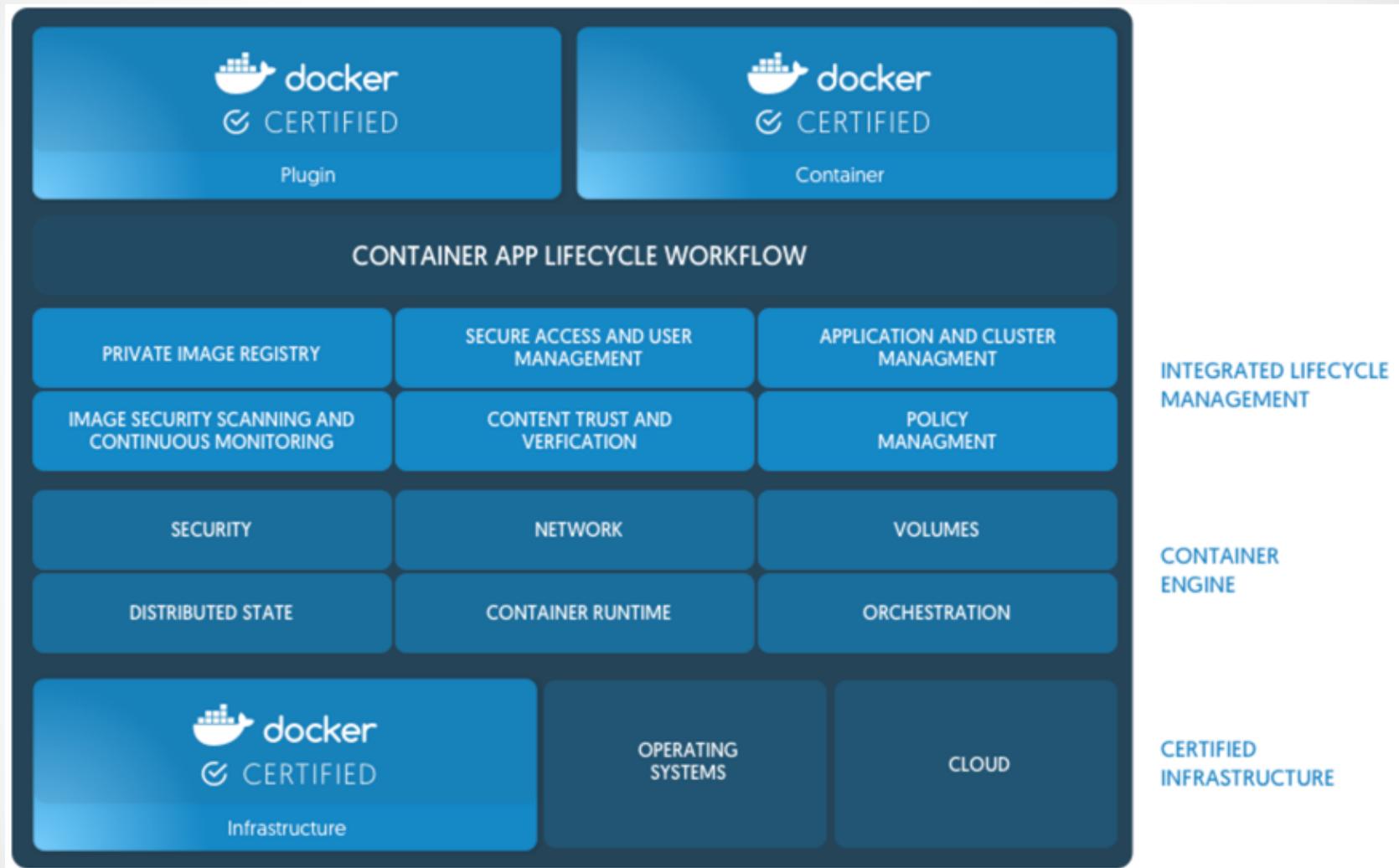
What's New

	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 New



What's News

17.06.0-ce (2017-06-28)

Note: Docker 17.06.0 has an issue in the image builder causing a change in the behavior of the `ADD` instruction of Dockerfile when referencing a remote `.tar.gz` file. The issue will be fixed in Docker 17.06.1.

Note: Starting with Docker CE 17.06, Ubuntu packages are also available for IBM z Systems using the s390x architecture.

Note: Docker 17.06 by default disables communication with legacy (v1) registries. If you require interaction with registries that have not yet migrated to the v2 protocol, set the `--disable-legacy-registry=false` daemon option. Interaction with v1 registries will be removed in Docker 17.12.

Server

Platform	Docker CE x86_64	Docker CE ARM	Docker CE System Z (s390x)	Docker EE
CentOS	✓			✓
Debian	✓	✓		
Fedora	✓			
Microsoft Windows Server 2016				✓
Oracle Linux				✓
Red Hat Enterprise Linux			✓	✓
SUSE Linux Enterprise Server				✓
Ubuntu	✓	✓	✓	✓

What's News

17.09.0-ce (2017-09-26)

Builder

- Add `--chown` flag to `ADD/COPY` commands in Dockerfile [moby/moby#34263](#)
- Fix cloning unneeded files while building from git repositories [moby/moby#33704](#)

Client

- Allow extension fields in the v3.4 version of the compose format [docker/cli#452](#)
- Make compose file allow to specify names for non-external volume [docker/cli#306](#)
- Support `--compose-file` – as stdin [docker/cli#347](#)
- Support `start_period` for healthcheck in Docker Compose [docker/cli#475](#)
- Add support for `stop-signal` in docker stack commands [docker/cli#388](#)
- Add support for update order in compose deployments [docker/cli#360](#)
- Add ulimits to unsupported compose fields [docker/cli#482](#)
- Add `--format` to `docker-search` [docker/cli#440](#)
- Show images digests when `{{.Digest}}` is in format [docker/cli#439](#)
- Print output of `docker stack rm` on `stdout` instead of `stderr` [docker/cli#491](#)
- Fix `docker history --format {{json .}}`' printing human-readable timestamps instead of ISO8601 when `--human=true` [docker/cli#438](#)
- Fix idempotence of `docker stack deploy` when secrets or configs are used [docker/cli#509](#)
- Fix presentation of random host ports [docker/cli#404](#)
- Fix redundant service restarts when service created with multiple secrets [moby/moby#34746](#)

What's News

Secure | <https://docs.microsoft.com/en-us/sql/linux/quickstart-install-connect-docker>

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

 Microsoft Technologies Documentation Resources

Docs Windows Microsoft Azure Visual Studio Office More

Docs / SQL / SQL Server on Linux

Filter

About SQL Server on Linux

- > Overview
- Quickstarts
 - Install & Connect - Red Hat
 - Install & Connect - SUSE
 - Install & Connect - Ubuntu
- Run & Connect - Docker**

> Concepts

> Samples

> Resources

Download PDF

Run the SQL Server 2017 container image with Docker

2017-7-17 • 7 min to read • Contributors 

In this quick start tutorial, you use Docker to pull and run the SQL Server 2017 RC1 container image, [mssql-server-linux](#). Then connect with **sqlcmd** to create your first database and run queries.

This image consists of SQL Server running on Linux based on Ubuntu 16.04. It can be used with the Docker Engine 1.8+ on Linux or on Docker for Mac/Windows.

 **Note**

This quick start specifically focuses on using the mssql-server-linux image. The Windows image is not covered, but you can learn more about it on the [mssql-server-windows Docker Hub page](#).

Prerequisites

- Docker Engine 1.8+ on any supported Linux distribution or Docker for Mac/Windows.
- Minimum of 4 GB of disk space
- Minimum of 4 GB of RAM
- [System requirements for SQL Server on Linux](#).

What's News

Screenshot of the Docker Hub interface showing the repository `microsoft/mssql-server-linux`.

The repository is a PUBLIC REPOSITORY. The last push was 21 minutes ago.

Tags available:

Tag Name	Compressed Size	Last Updated
rc2	482 MB	21 minutes ago
latest	482 MB	21 minutes ago
rc1	478 MB	16 days ago
ctp2-0	451 MB	2 months ago
ctp2-1	462 MB	2 months ago
ctp1-4	382 MB	5 months ago
ctp1-3	370 MB	6 months ago
ctp1-2	367 MB	6 months ago
ctp1-1	340 MB	6 months ago

Docker: The Next-Gen of Virtualization



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		✓	✓

What's News

Docker: Now powered by Swarm and Kubernetes

GENERALLY AVAILABLE
Q1 2018

Beta signup is open!
www.docker.com/kubernetes



Docker: The Next-Gen of Virtualization

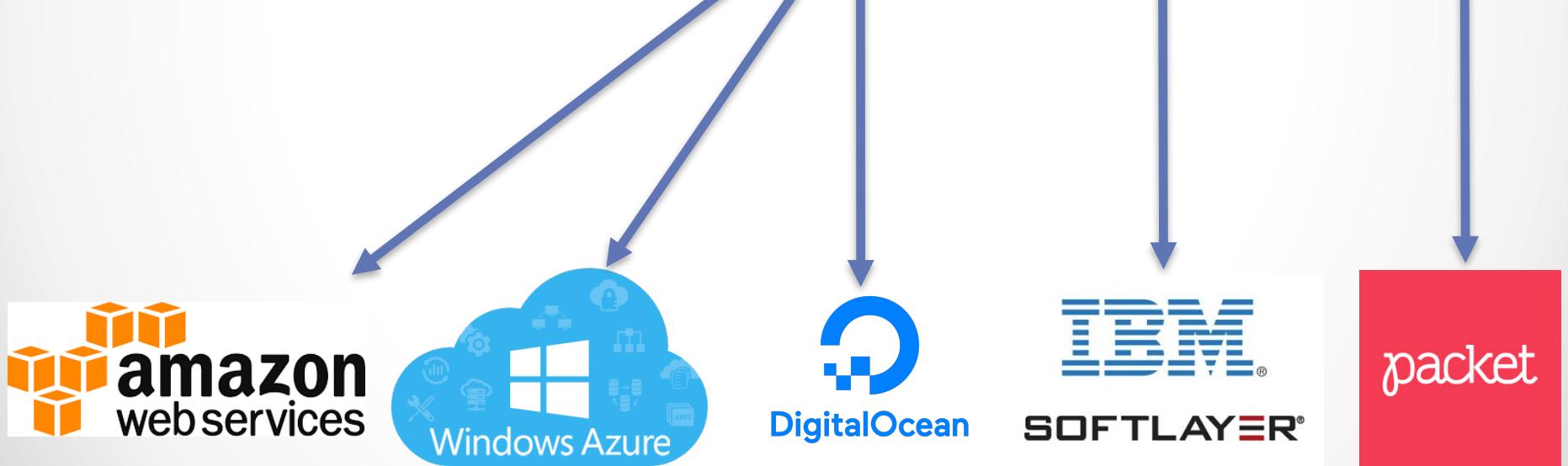


Cloud Support Docker



The screenshot shows the Docker Cloud Cloud Settings page. On the left, a sidebar lists options: General, Cloud providers, Source providers, Notifications, Default Privacy, Billing, Plan, Coupon Codes, and Quotas. The main area displays a user profile for "labdocker" (Member since Feb 08, 2016). Below this is a section titled "Cloud providers" listing various cloud services with their status and tier information:

Provider	Action	Status	Tier
Amazon Web Services	Add new credentials	Green	Free Tier
Digital Ocean	Add new credentials	Green	\$20 Code
Microsoft Azure	Add new credentials	Green	Global Admin
SoftLayer	Add new credentials	Green	Free trial
Packet	Add new credentials	Green	\$25 code



Docker: The Next-Gen of Virtualization



Docker Machine

• • •

Platform of Docker

- Docker Toolbox (Legacy) 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

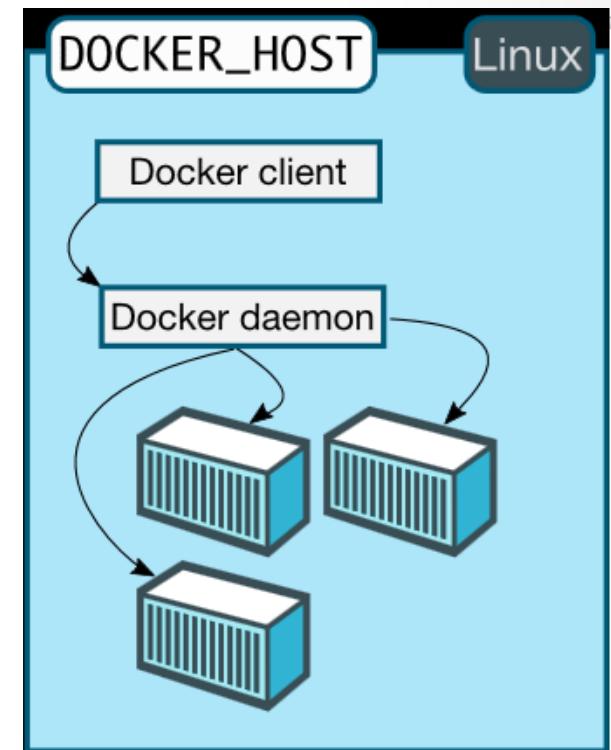
- Pros and Cons.
- Docker Toolbox
 - Pros
 - Compatible for All (Windows 7, 8, 10, MAC OS X)
 - Limit result for operate
 - Suitable for simulate multiple node or swarm
 - Cons
 - Multiple step for transfer source code and debug
 - Overhead for VirtualBox

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

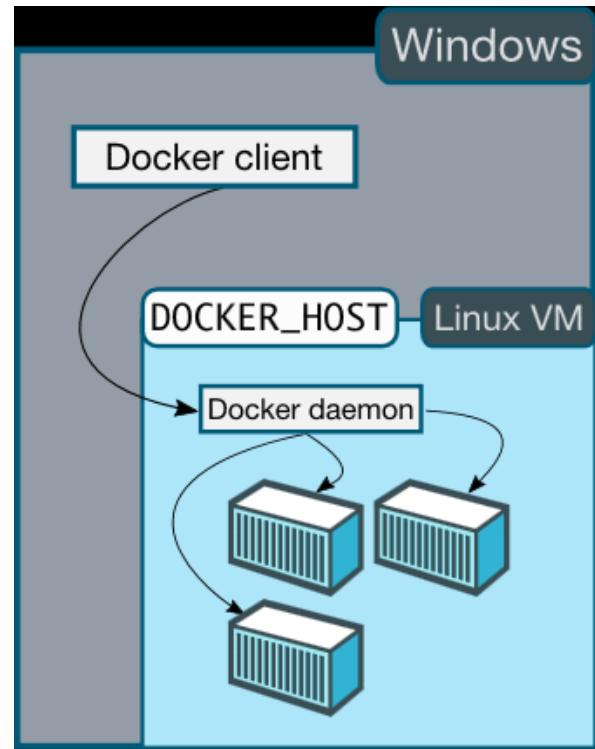
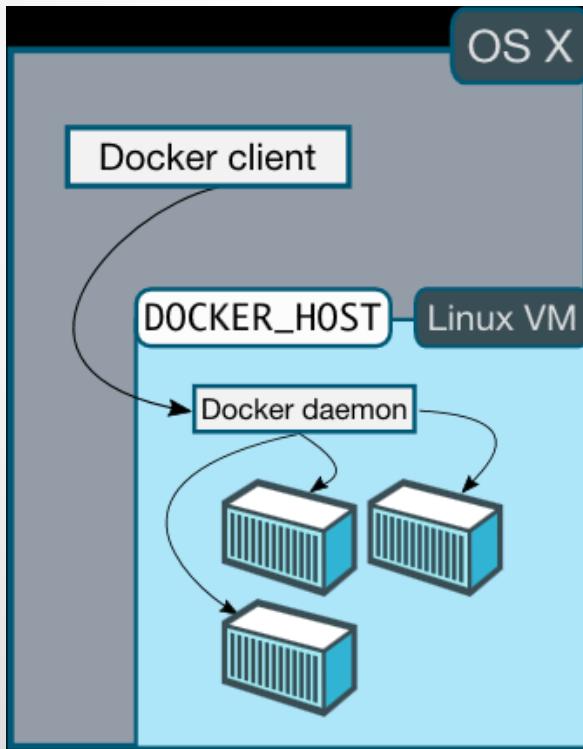
Platform of Docker

- **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 โดยเฉพาะดังนั้นมีการสั่งรันจึงเกิดการ remote run ไปยังเครื่อง virtual machine (docker-machine)



Docker Architecture

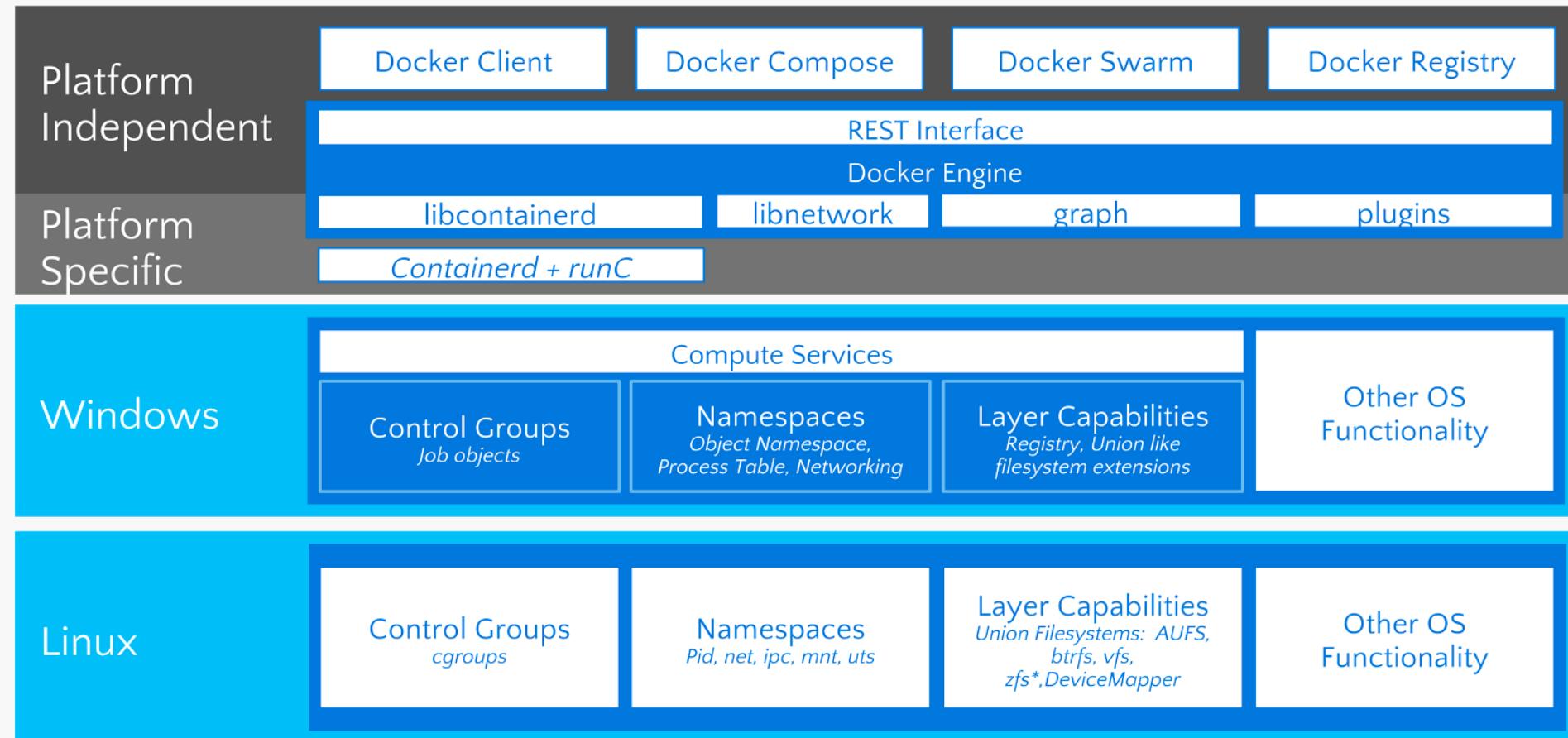


Image2Docker (Prototype)

<https://github.com/docker/communitytools-image2docker-win>

Medium jenkins vagrant Mesos Vue docker NGINX Taiwan Kubernetes PWA_Progressive_W... MYSQL_Cluster GeneralKB Nodejs

README.md

Image2Docker

Image2Docker is a PowerShell module which ports existing Windows application workloads to Docker. It supports multiple application types, but the initial focus is on IIS and ASP.NET apps. You can use Image2Docker to extract ASP.NET websites from a VM - or from the local machine or a remote machine. Then so you can run your existing apps in Docker containers on Windows, with no application changes.

Documentation

- IIS and ASP.NET

Introduction

This project aims to provide a framework to simplify the creation of Dockerfiles for Windows Docker images, based upon analysis of existing Windows machines.

Microsoft Windows 10 and Windows Server 2016 introduce new capabilities for containerizing applications. There are two types of container formats supported on the Microsoft Windows platform:

- Hyper-V Containers - Containers with a dedicated kernel and stronger isolation from other containers
- Windows Server Containers - application isolation using process and namespace isolation, and a shared kernel with the container host

Docker-machine CLI

- ตรวจสอบสถานะ docker-machine

```
docker-machine ls
```

- สั่งเริ่มการทำงานของ docker-machine

```
docker-machine start labdocker
```

- สั่งหยุดการทำงานของ docker-machine

```
docker-machine stop labdocker
```

- สั่งสร้าง docker-machine เครื่องใหม่

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

Workshop 1-2: Pull Image

- เชื่อมต่อ กับ virtual machine (Boot2Docker)
 - Windows
 - Via command prompt: ➔ docker-machine ssh labdocker
 - Via SSH ➔ Connect to ip address of virtual machine
 - Mac OS
 - Via terminal : ➔ docker-machine ssh labdocker
 - Via SSH ➔ ssh docker@<ip address>

Workshop 1-2: Pull Image

ทำการ download image ubuntu ลงมาที่เครื่อง boot2docker โดยใช้คำสั่งดังนี้

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

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

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

Compare with Ubuntu image

```
docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
labdocker/alpineweb	latest	20f37a9ebc28	17 months ago	24.5MB
labdocker/ubuntu	latest	3876b81b5a81	18 months ago	188MB
labdocker/alpine	latest	14f89d0e6257	18 months ago	4.79MB

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/>

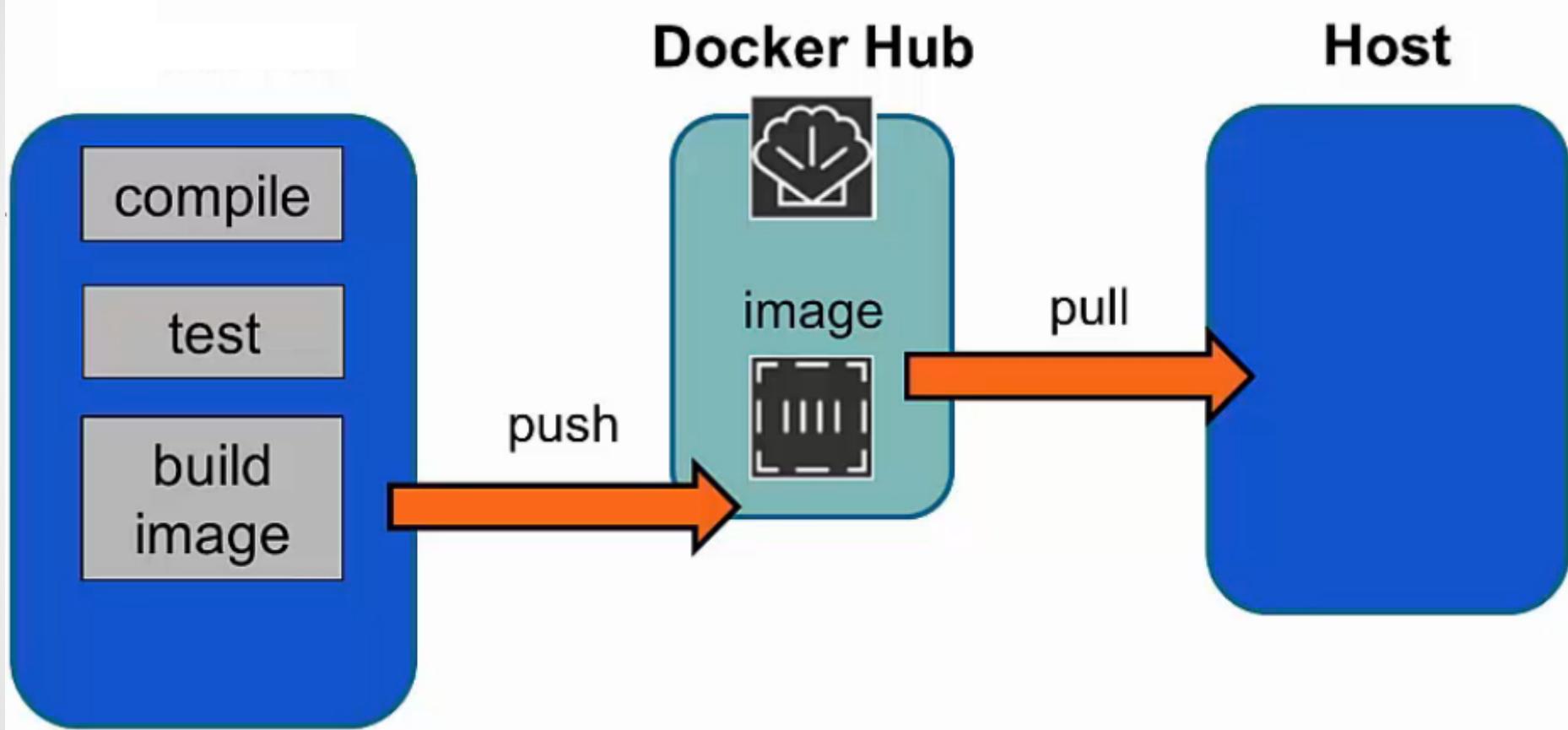
Docker: The Next-Gen of Virtualization



Image, Repository and Container

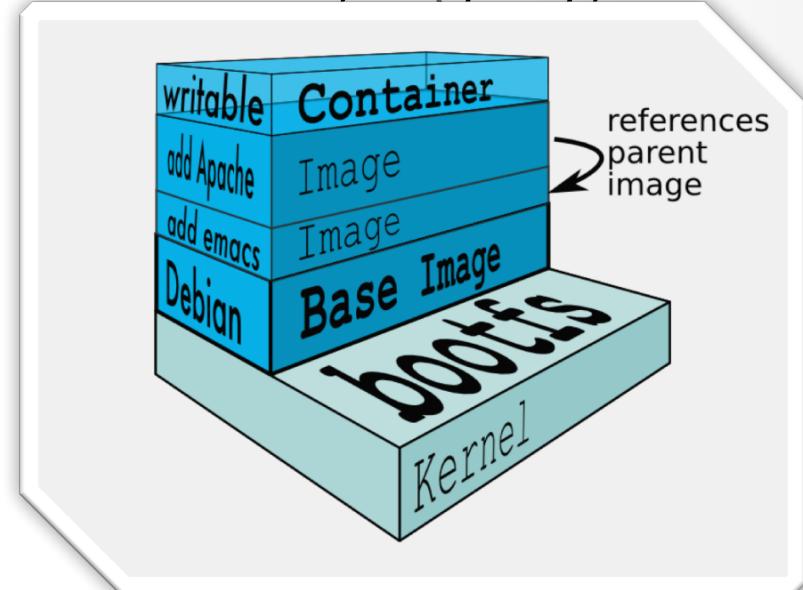
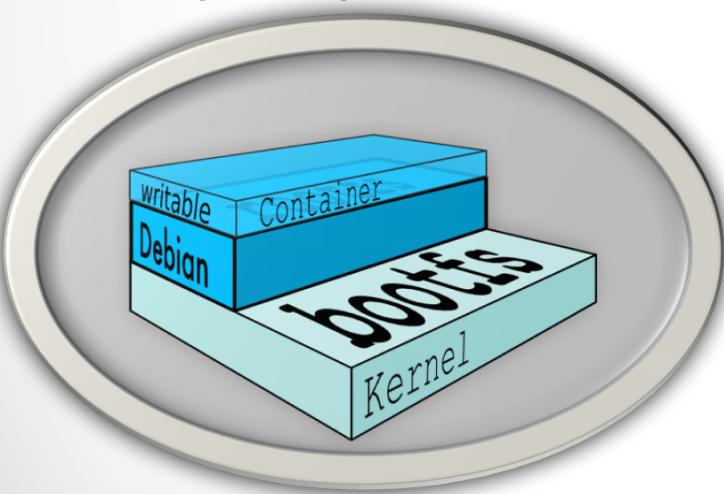
• • •

Docker Deployment



Docker Image

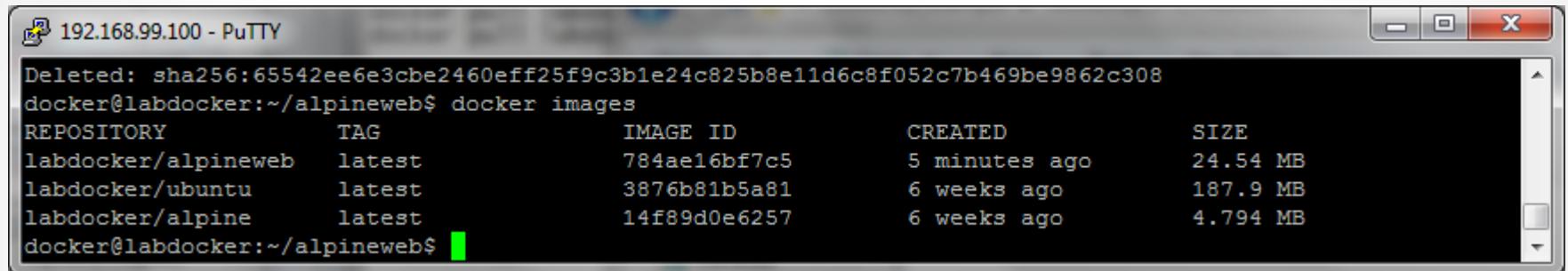
- Image คือ template ที่ถูกสร้างขึ้นเพื่อเตรียมใช้ในการรัน container
- เป็นไฟล์ที่อ่านได้อย่างเดียว
- ถูกสร้างโดยผู้ใช้งานเอง หรือผู้อื่น
- จัดเก็บไว้ใน repository (hub.docker.com, registry, trust registry)



Docker Image

- เรียกดู image ที่อยู่ภายในเครื่อง

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



192.168.99.100 - PuTTY

```
Deleted: sha256:65542ee6e3cbe2460eff25f9c3b1e24c825b8e11d6c8f052c7b469be9862c308
docker@labdocker:~/alpineweb$ docker images
REPOSITORY          TAG        IMAGE ID      CREATED       SIZE
labdocker/alpineweb latest      784ae16bf7c5  5 minutes ago  24.54 MB
labdocker/ubuntu    latest      3876b81b5a81  6 weeks ago   187.9 MB
labdocker/alpine    latest      14f89d0e6257  6 weeks ago   4.794 MB
docker@labdocker:~/alpineweb$
```

- 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) ต่างๆรวมไปที่ศูนย์กลางเพื่อให้ docker engine บนเครื่องต่างๆมาเรียก image ไปใช้ทำงาน
- รองรับการเก็บ version ของ image อย่างเป็นระบบ
- สามารถให้ข้อมูลหรือคุณมีอ่อนแหน่งานในการใช้งานกับผู้ 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 displaying the Docker Hub homepage at <https://hub.docker.com>. The page has a dark blue header with a search bar containing 'Search' and a navigation bar with links for 'Explore' and 'Help'. A 'Log In' button is also visible. The main content area features a large call-to-action button with the text 'Build, Ship, & Run Any App, Anywhere' in white. Below this, a sub-headline reads 'Dev-test pipeline automation, 100,000+ free apps, public and private registries'. To the right, there's a 'New to Docker?' section with a 'Create your free Docker ID to get started.' link, followed by three input 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

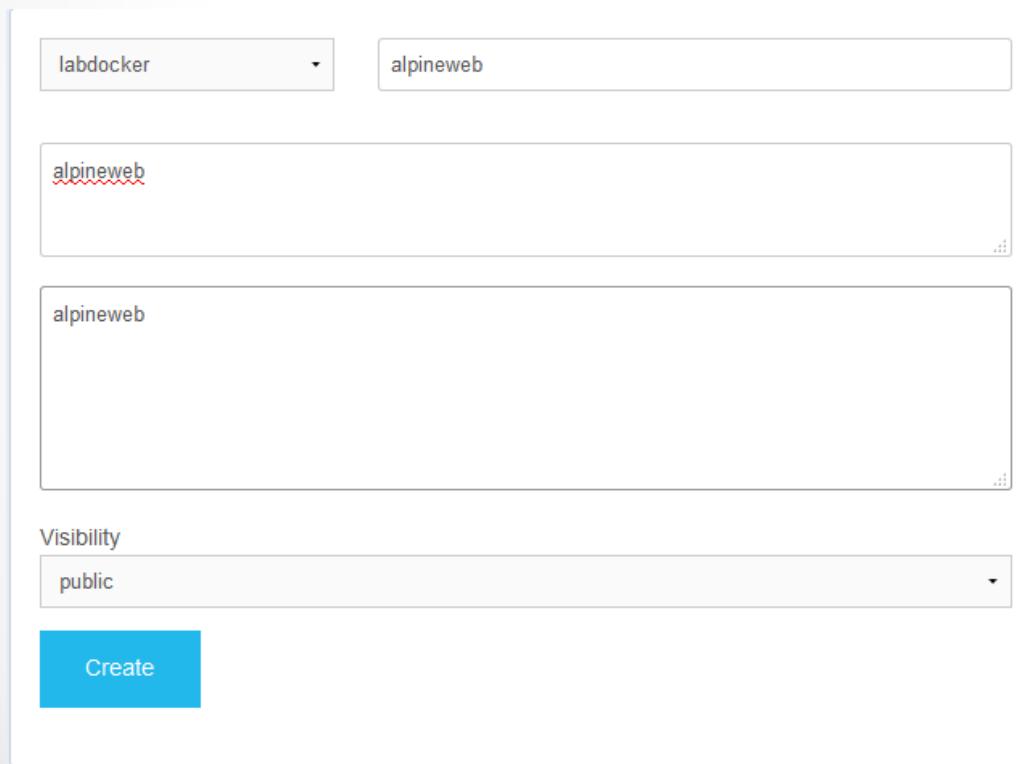
- Logoff ออกจาก Repository

```
docker logoff
```

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

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



Workshop 1-3: Build & Push Image

- เข้าใช้งาน boot2docker และตรวจสอบ 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. In the center is the title "The Docker Store" and a subtitle "Find Trusted and Enterprise Ready Containers, Plugins, and Docker Editions". Below this is a search bar with placeholder text "Search for great content (e.g. mysql)". At the bottom, there are navigation links for "Docker EE", "Docker CE", "Containers", and "Plugins". A cursor arrow is visible near the bottom center.

Docker: The Next-Gen of Virtualization



Store.docker.com

The screenshot shows the Docker Store interface. At the top, there's a navigation bar with the Docker logo, a search bar, and links for Explore, Publish, and Feedback. A user profile icon for 'paparn' is also present. The main title 'The Docker Store' is centered above a subtitle 'Find Trusted and Enterprise Ready Containers, Plugins, and Docker Editions'. Below this, there are tabs for Docker EE, Docker CE, Containers (which is selected), and Plugins. On the left, there are filters for Type (Store selected, Community (Docker Hub) available), Docker Certified (Docker Certified checked), and Categories (Analytics, Application Frameworks, Application Infrastructure, Application Services). The main content area displays 1 - 10 of 26 available images under the 'Databases' category. It lists two items: 'Neo4j Community' by Neo Technology, Inc., and 'Oracle Database Enterprise Edition' by Oracle. Both items have a brief description and a 'View Details' button.

Docker store

Explore Publish Feedback

paparn

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

TYPE

Store

Community (Docker Hub)

DOCKER CERTIFIED

Docker Certified

CATEGORIES

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](#)

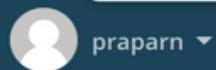
Docker: The Next-Gen of Virtualization



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

Docker: The Next-Gen of Virtualization



Cloud.docker.com

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

Docker Cloud

The official cloud service for continuously delivering Docker applications.

New to Docker?

Create your free Docker ID to get started.

Choose your Docker ID

Email address

Choose a password

Sign up

By signing up, you agree to Docker's [Terms of Service](#). [Privacy Policy](#).

Docker: The Next-Gen of Virtualization



Cloud.docker.com

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

PRIVATE REPOS

Starts at \$7/month

■ All accounts include 1 free private repo

☛ Unlimited public repos

☛ Automated builds

☛ Parallel builds

☛ **Docker Security Scanning**

☛ Automated tests

[Try it out free](#)

NODES - STANDARD MODE

\$15/node/mo

■ All accounts include 1 free managed node

■ All nodes provisioned with Docker Engine 1.11

☛ Manage unlimited Stacks, Services and Containers

☛ Logging, Scaling, Notifications and more

☛ [Bring your own node](#)

☛ Only available in Standard Mode

[Try it out free](#)

Cloud.docker.com

The screenshot shows the Docker Cloud Settings interface. On the left, there's a sidebar with sections for BUILD (Repositories), APPLICATIONS (Stacks, Services, Containers), and INFRASTRUCTURE (Node Clusters). A 'Swarm Mode' toggle switch is also present. The main area is titled 'Cloud Settings' and features a 'BETA' indicator. It includes a user profile section for 'labdocker' (Member since Feb 08, 2016) and a 'Cloud providers' section listing several cloud services with their status and access options.

Cloud providers

Provider	Action	Status	Role	
Amazon Web Services	Add new credentials			Free Tier
Digital Ocean	Add new credentials			\$20 Code
Microsoft Azure	Add new credentials			Global Admin
SoftLayer	Add new credentials			Free trial
Packet	Add new credentials			\$25 code

Status.docker.com

 docker status What is Docker? Product Get Docker ▾ Docs Community

Docker System Status

Current system status information.

Docker System Status

All Systems Operational

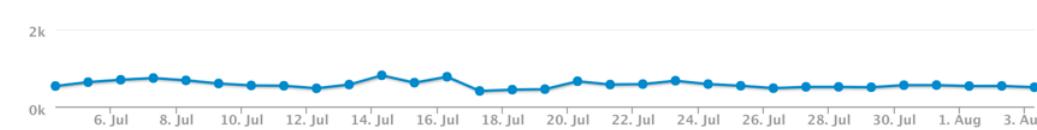
Metrics

SI IHSRIE

Today Week Month

Our system status page is a real-time view of the performance and uptime

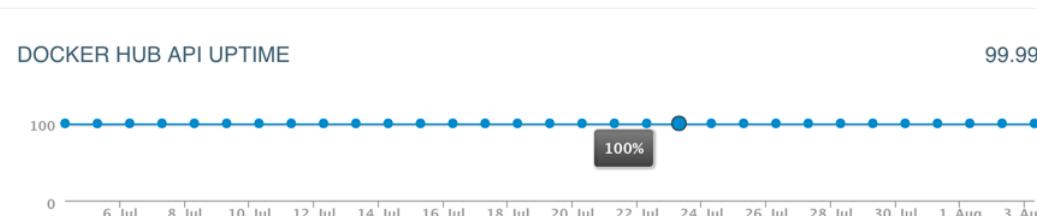
DOCKER HUB API RESPONSE TIME 594.42MS



2k
0k

6. Jul 8. Jul 10. Jul 12. Jul 14. Jul 16. Jul 18. Jul 20. Jul 22. Jul 24. Jul 26. Jul 28. Jul 30. Jul 1. Aug 3. Aug

DOCKER HUB API UPTIME 99.99%



100
0

6. Jul 8. Jul 10. Jul 12. Jul 14. Jul 16. Jul 18. Jul 20. Jul 22. Jul 24. Jul 26. Jul 28. Jul 30. Jul 1. Aug 3. Aug

100%

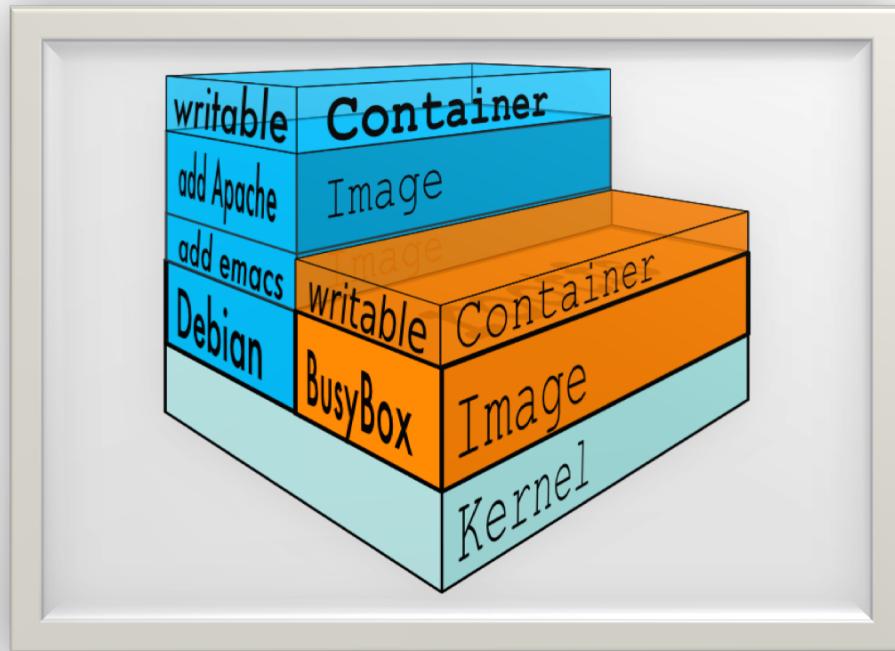
<https://status.docker.com/>

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 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



ธนาคารแห่งประเทศไทย

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

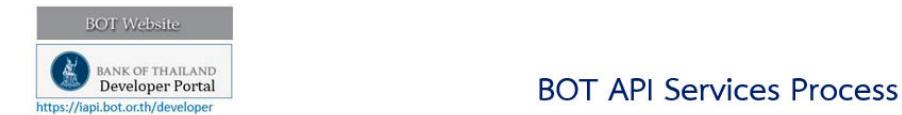
api-key : xk3h6ash-ahw5l3mch6hl3-2hhg6l4mng2346l34l6

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



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":"timeout=3","x-ratelimit-limit": "1000", "x-ratelimit-imagelimit": "50", "x-ratelimit-remaining": "998", "x-ratelimit-imageRemaining": "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="U9G1L457H60CugT7VmBaEacbHV9RX0Py5005cYaGsm";
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
- สั่งรัน container แบบ Daemon โดยตั้งชื่อว่า linebot เพื่อทำการดึงข้อมูลจาก API ของธนาคารแห่งประเทศไทยอุปกรณ์แล้วส่ง 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
- สั่งรัน 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

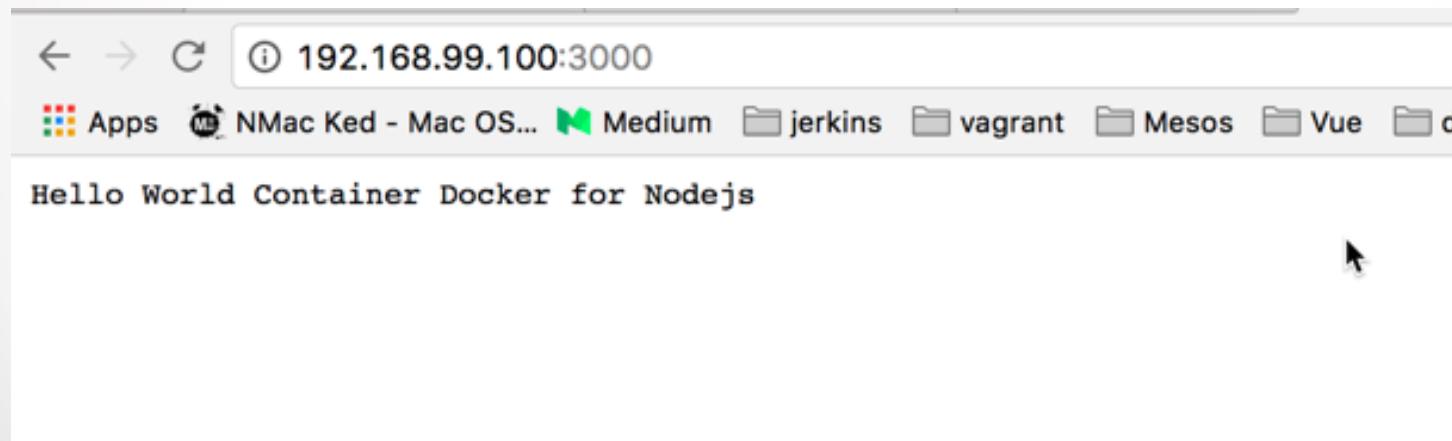
```
docker container exec -i -t nodejs sh
```

```
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
- สั่งรัน 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
```

- สั่งรัน 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 yellow feet, above the word "cAdvisor". Below the logo is a navigation bar with a single item: "/". Underneath the bar are several sections: "Docker Containers", "Subcontainers", and "Isolation". The "Isolation" section contains a blue header bar with the text "CPU" and "Shares 1024 shares".

Workshop 1-5: CPU Configure

- Download busybox เพื่อใช้ในการทดสอบ

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

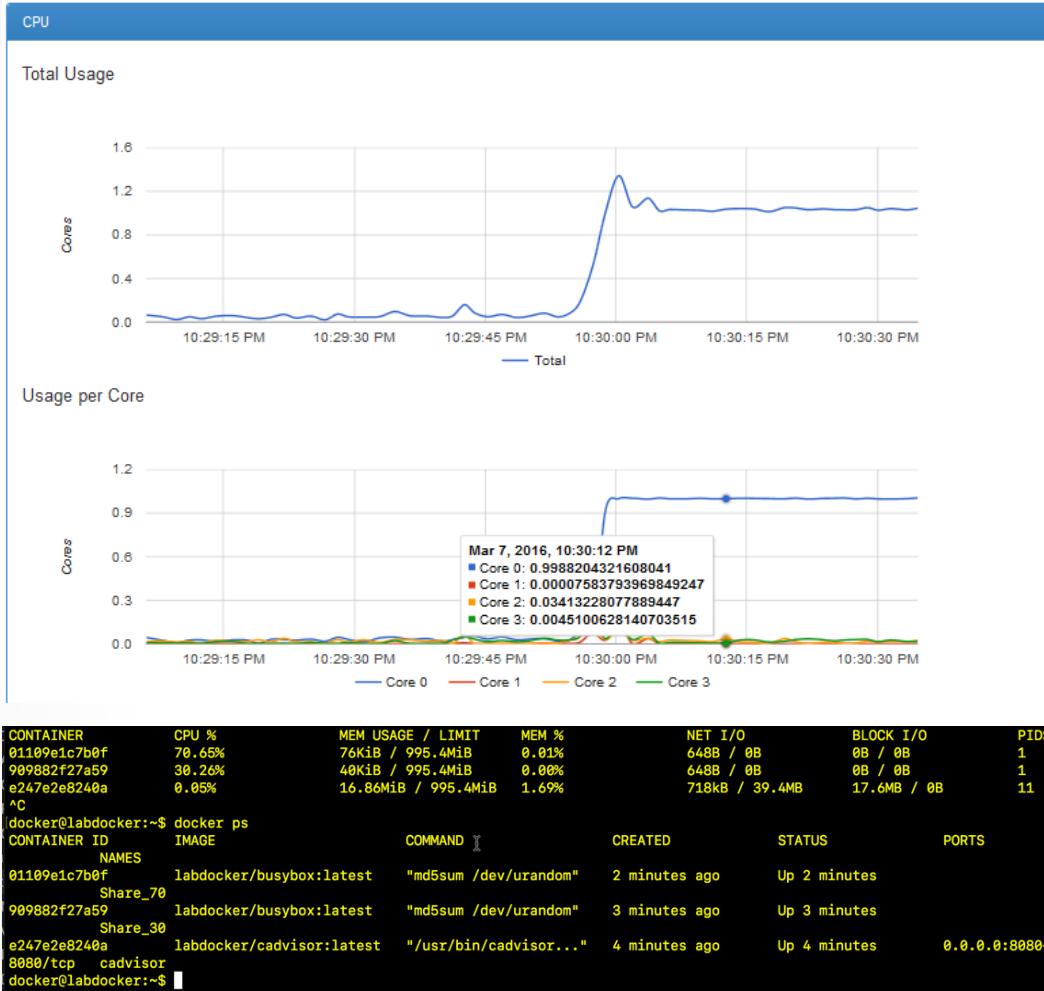
- Scenario 1 (Single CPU: 70/30)

```
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



Workshop 1-5: CPU Configure

- Download busybox เพื่อใช้ในการทดสอบ

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

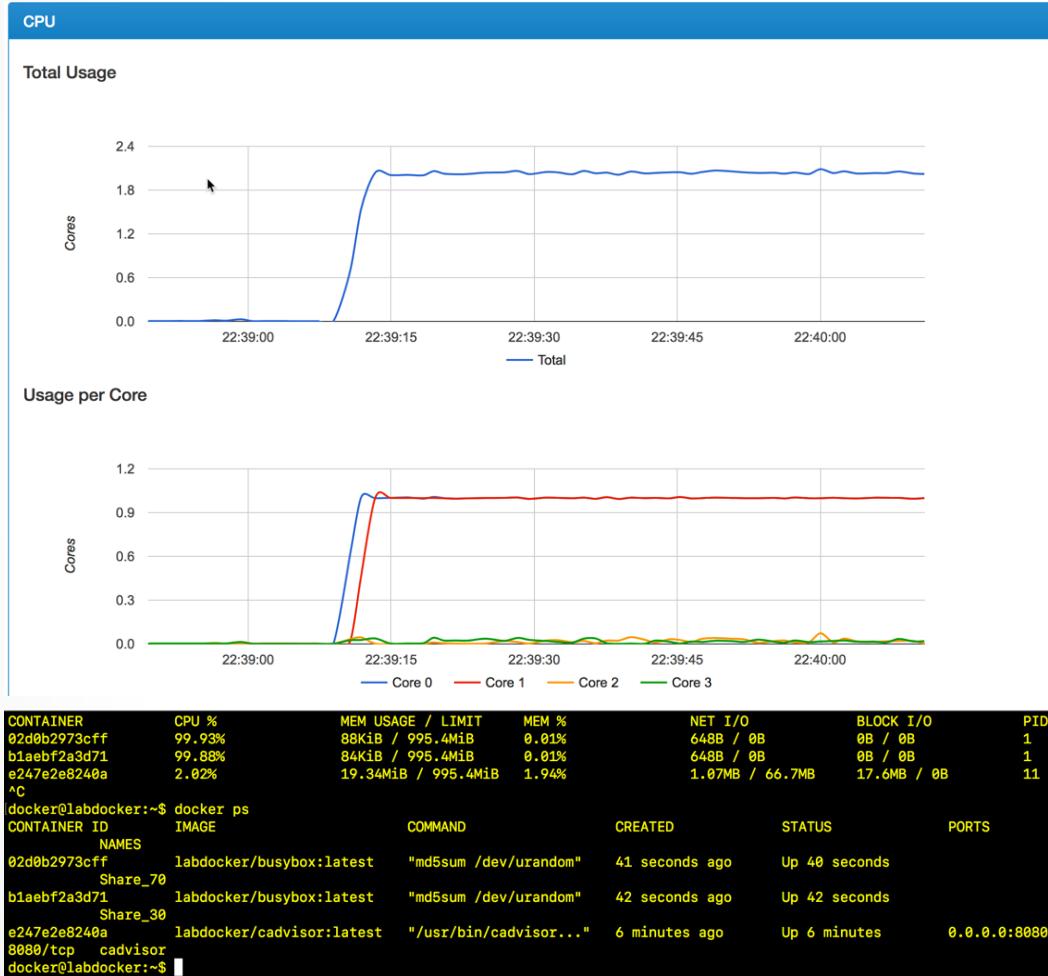
- Scenario 2 (Multiple CPU: 70/30)

```
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
```

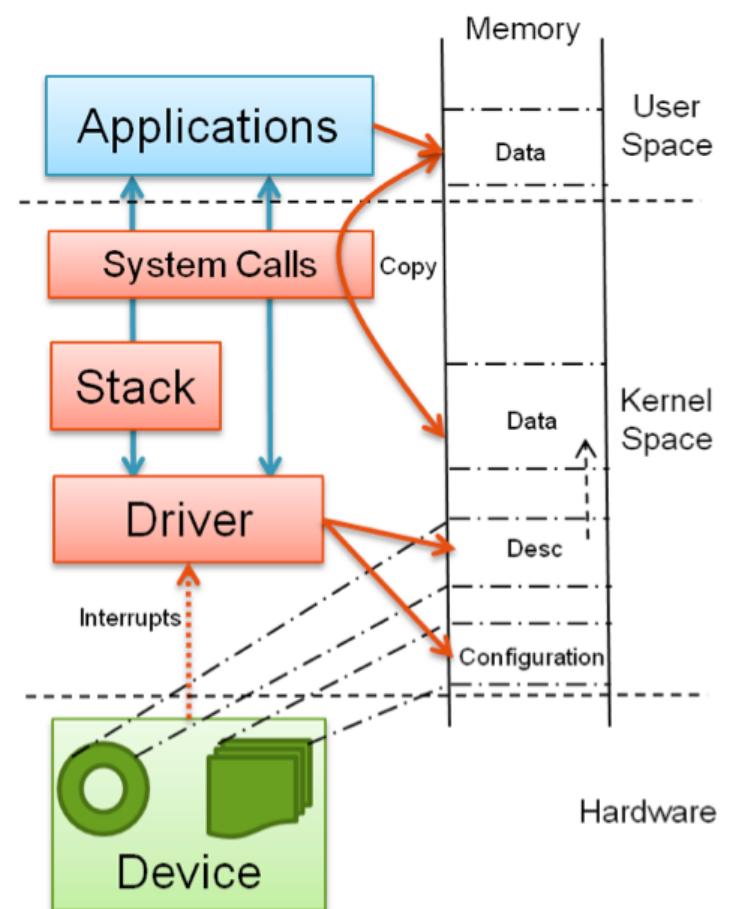
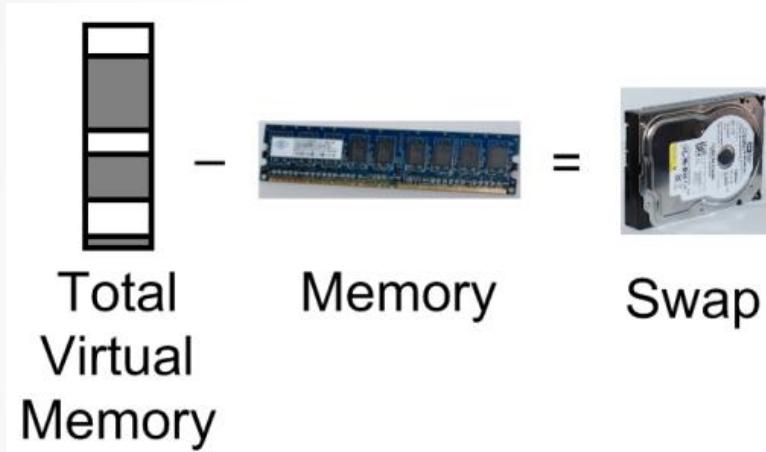
Workshop 1-5: CPU Configure

- 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 และกำหนดเป็น "--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 เท่าๆ กัน (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 → IOPS สูง / Transfer ต่ำ
- Database server → IOPS 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

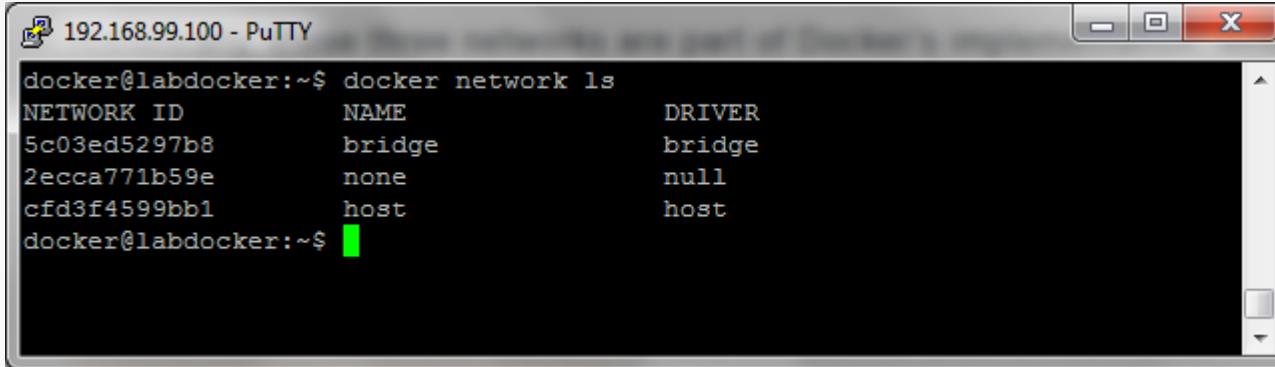
- Raw disk performance: disk iops x unit of disk
 - **Ex:** SAS 128GB (15K) x 6 units => $175 \times 6 = 1060$ iops
- Raid disk performance:
 - $((\text{raw iops} \times \% \text{write}) / \text{penalty}) + (\text{raw} \times \% \text{read})$
 - Normal situation (write 30%, read 70%)
 - **Ex:** Raid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

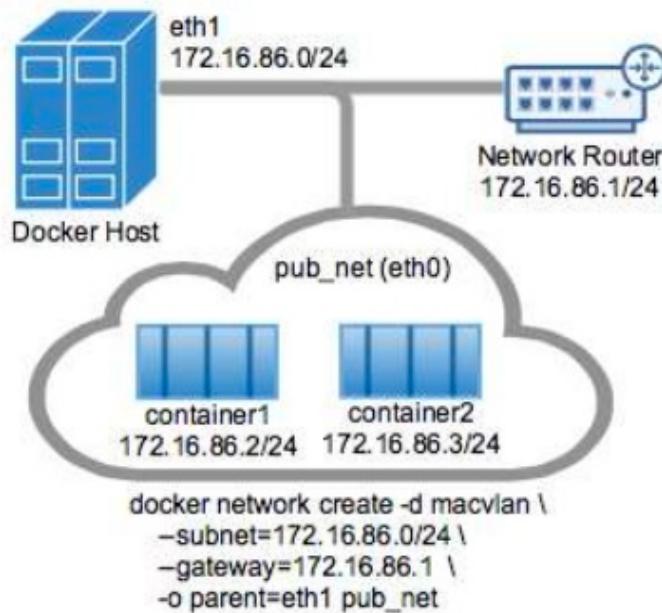


```
192.168.99.100 - Putty
docker@labdocker:~$ docker network ls
NETWORK ID      NAME      DRIVER
5c03ed5297b8    bridge    bridge
2ecca771b59e    none     null
cf3f4599bb1     host     host
docker@labdocker:~$
```

- **bridge** 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

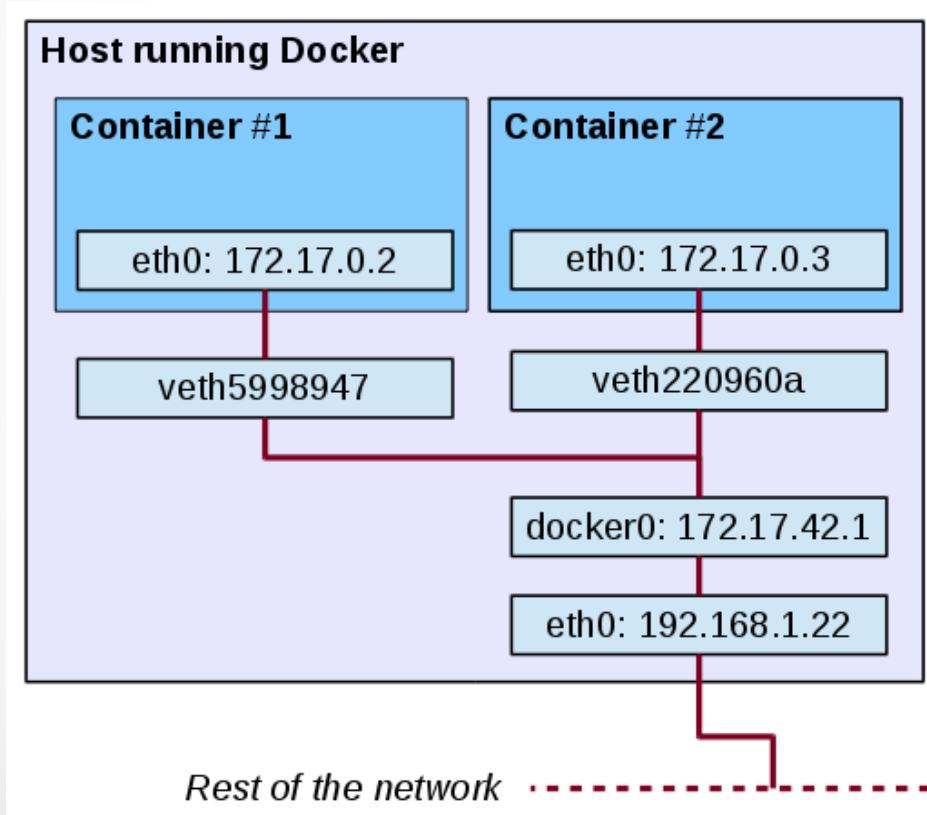
- Additional network type
- **MACVLAN**



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

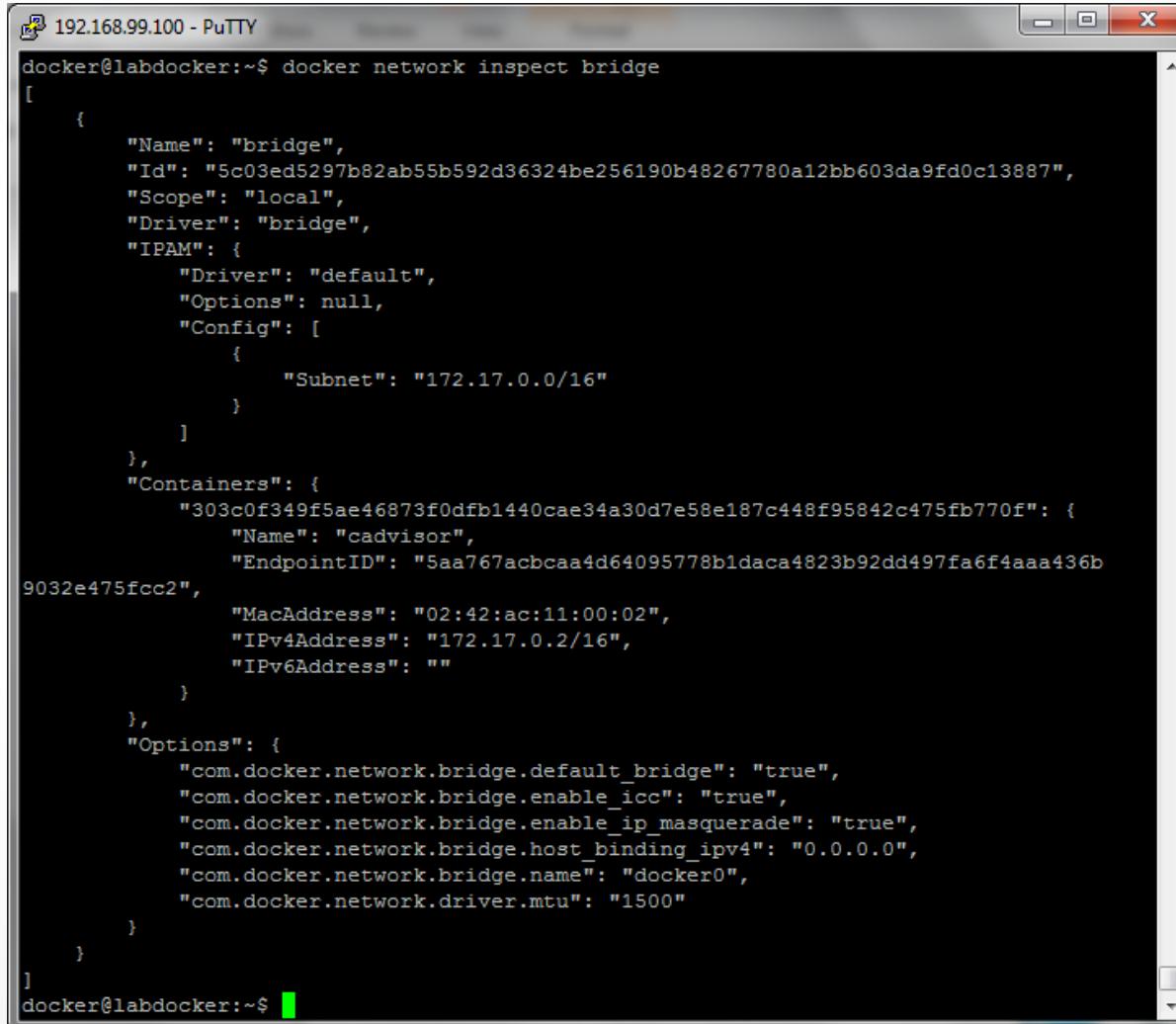
Network

- 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" <new in 1.11>
--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)

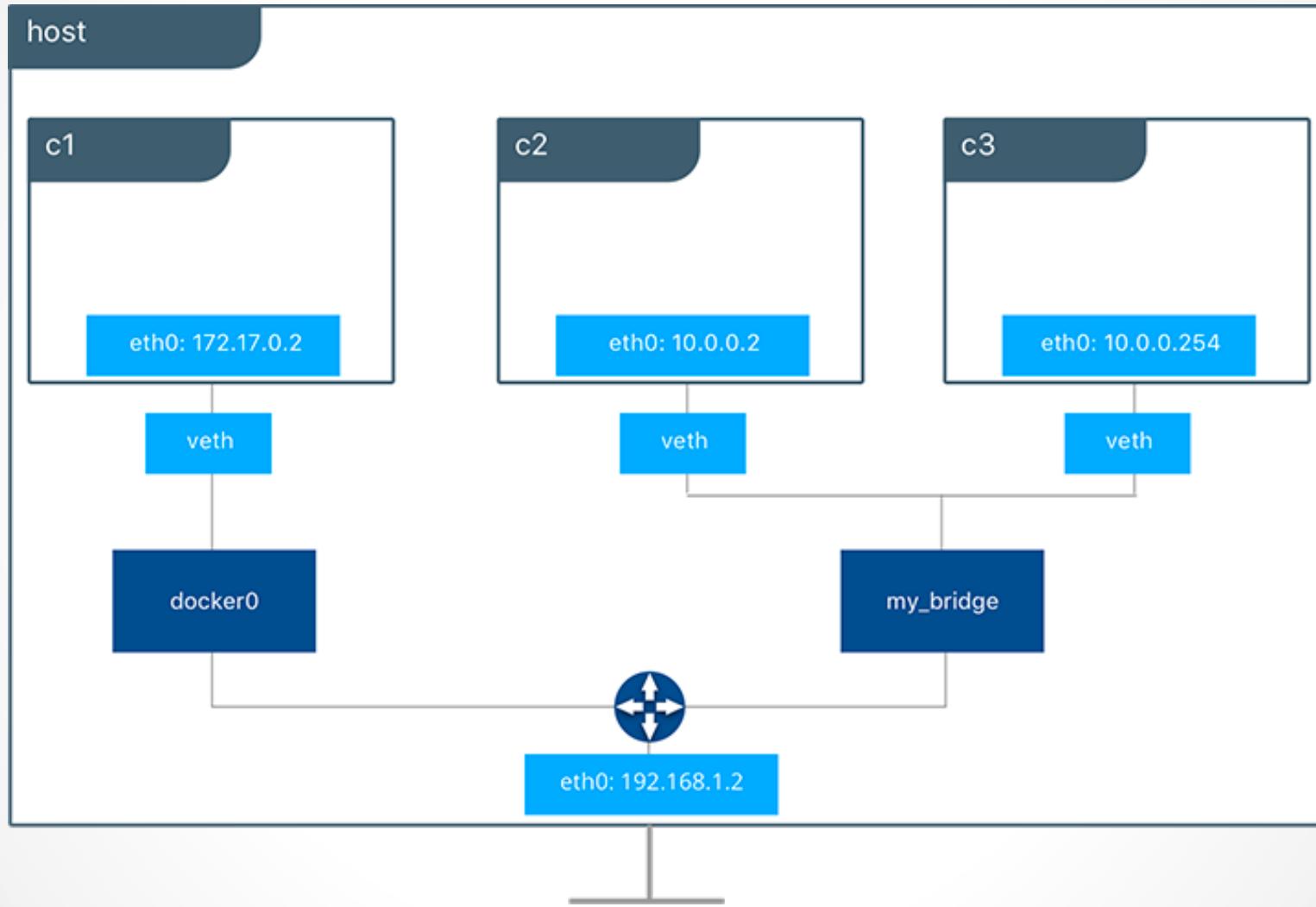
```
docker network create --driver <default/null/host> name
```

```
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)

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

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

- 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

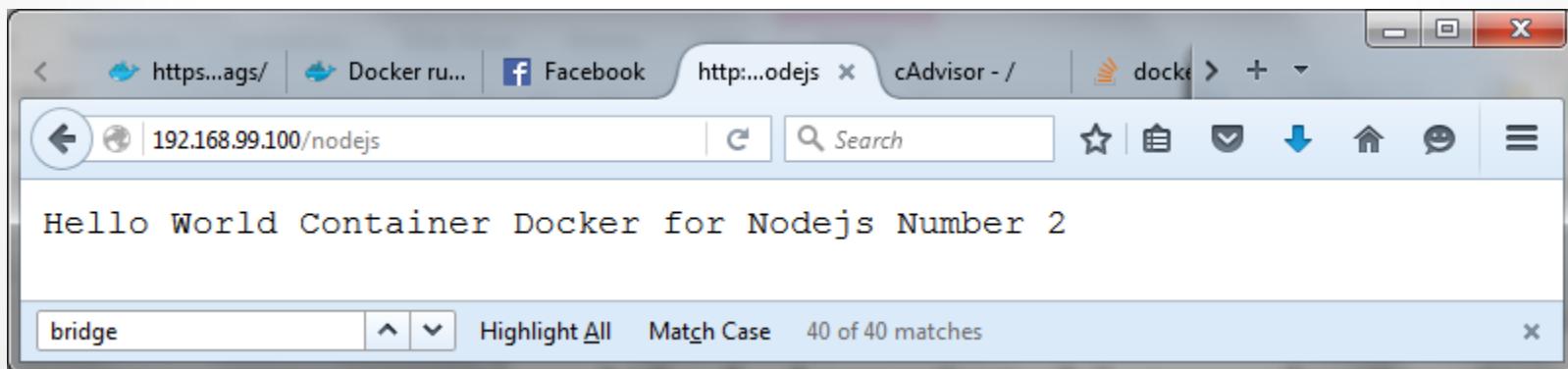
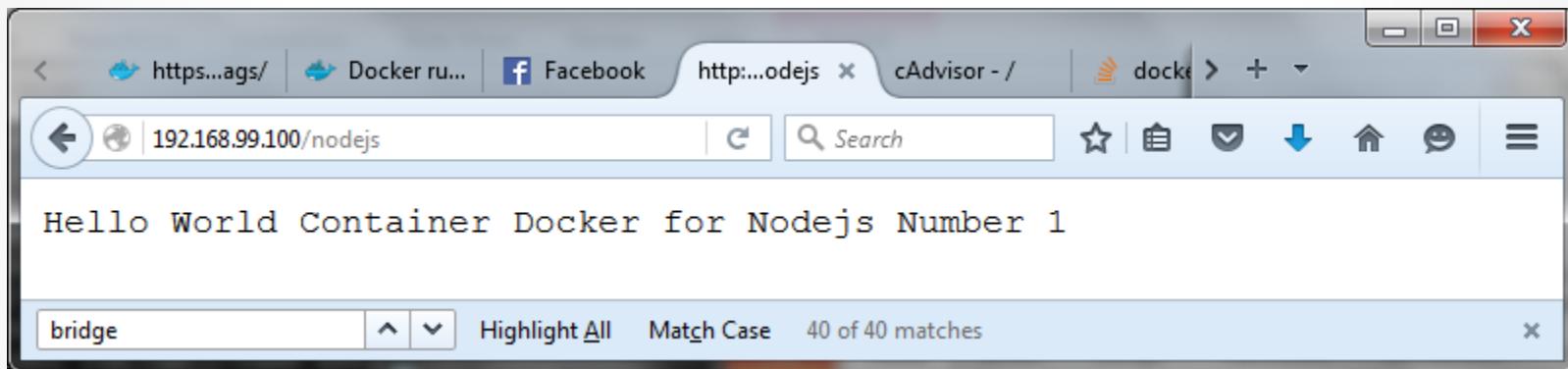
Workshop 1-6: Network Configure

- 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 #
```

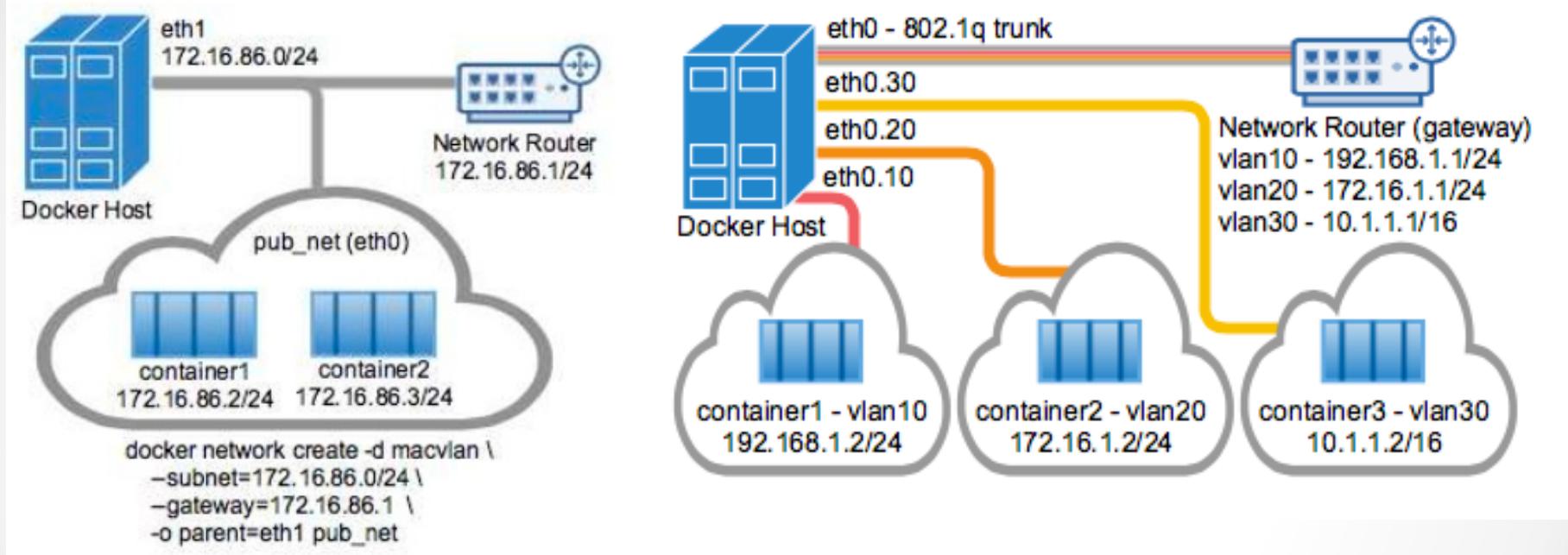
- nginx.conf

```
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)



Workshop 1-6: Network Configure

- Part 2: Reverse Proxy Network (MACVLAN)
- 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
 - 192.168.99.193 – 192.168.99.206

<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.

Workshop 1-6: Network Configure

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

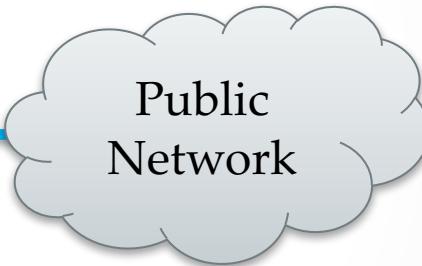
NGINX:

IP Address: 192.168.99.195

DNS Name: nginx

Service Port: 8080

Network Card: Eth1
IP Address: 192.168.99.100



Workshop 1-6: Network Configure

- 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
- ในการมีต้องการทำ 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

Workshop 1-6: Network Configure

- 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

Workshop 1-6: Network Configure

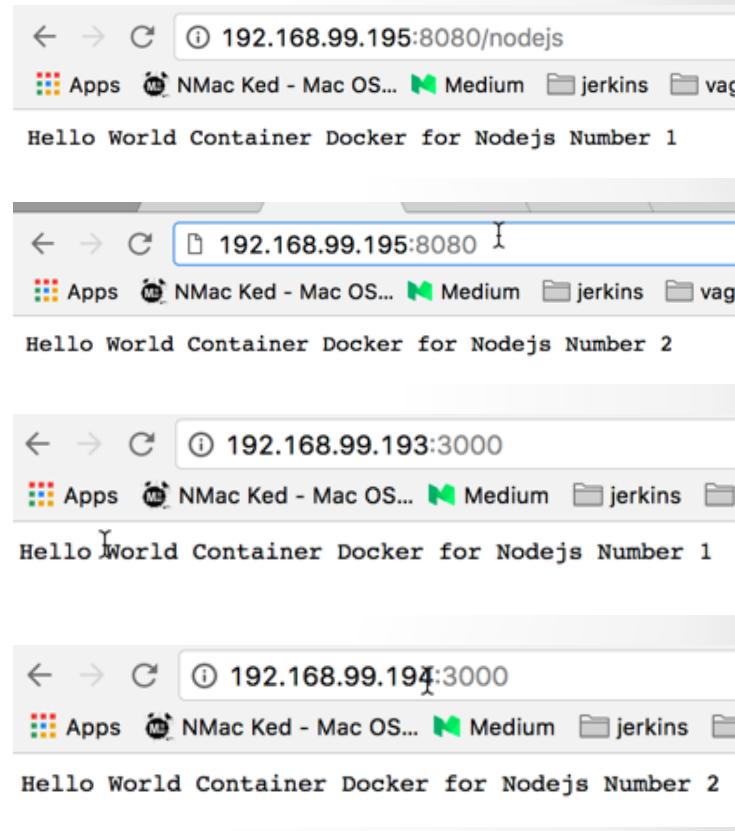
- Internal test:

```
[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
```

Workshop 1-6: Network Configure

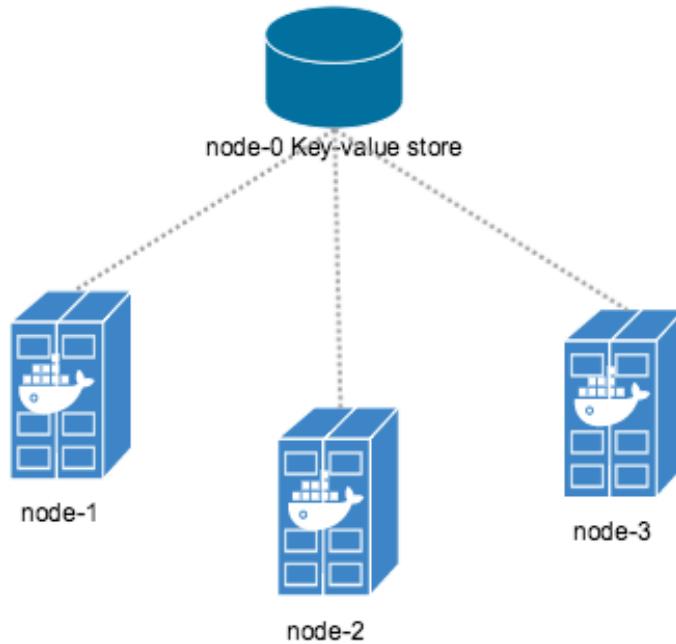
- External Test:

```
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

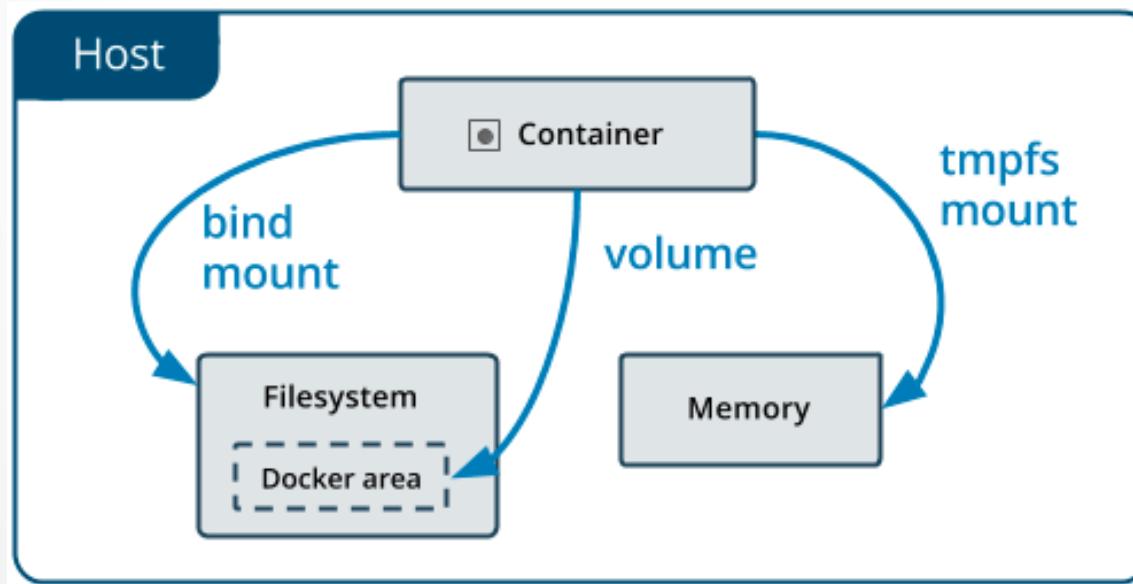
• • •

Volume

- ตามปกติแล้ว docker จะเก็บข้อมูลทุกอย่างเอาไว้ภายใน container
 - /var/log
 - /sys/data
 - /etc/nagios/
 - /etc/mysql
 - Etc.
- สำหรับการใช้งาน container ใน production environment docker ได้สร้าง tool สำหรับอำนวยความสะดวกในการใช้งานข้อมูลร่วมกันในหลายๆกรณี 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

- การใช้ 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 ทิ้งจาก 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

Link	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

- การ backup volume จาก 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
```

- การ restore volume จาก 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-persistence space โดย tmpfs ไม่สามารถ Share ระหว่าง container และไม่สามารถใช้งานบน Windows ได้

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

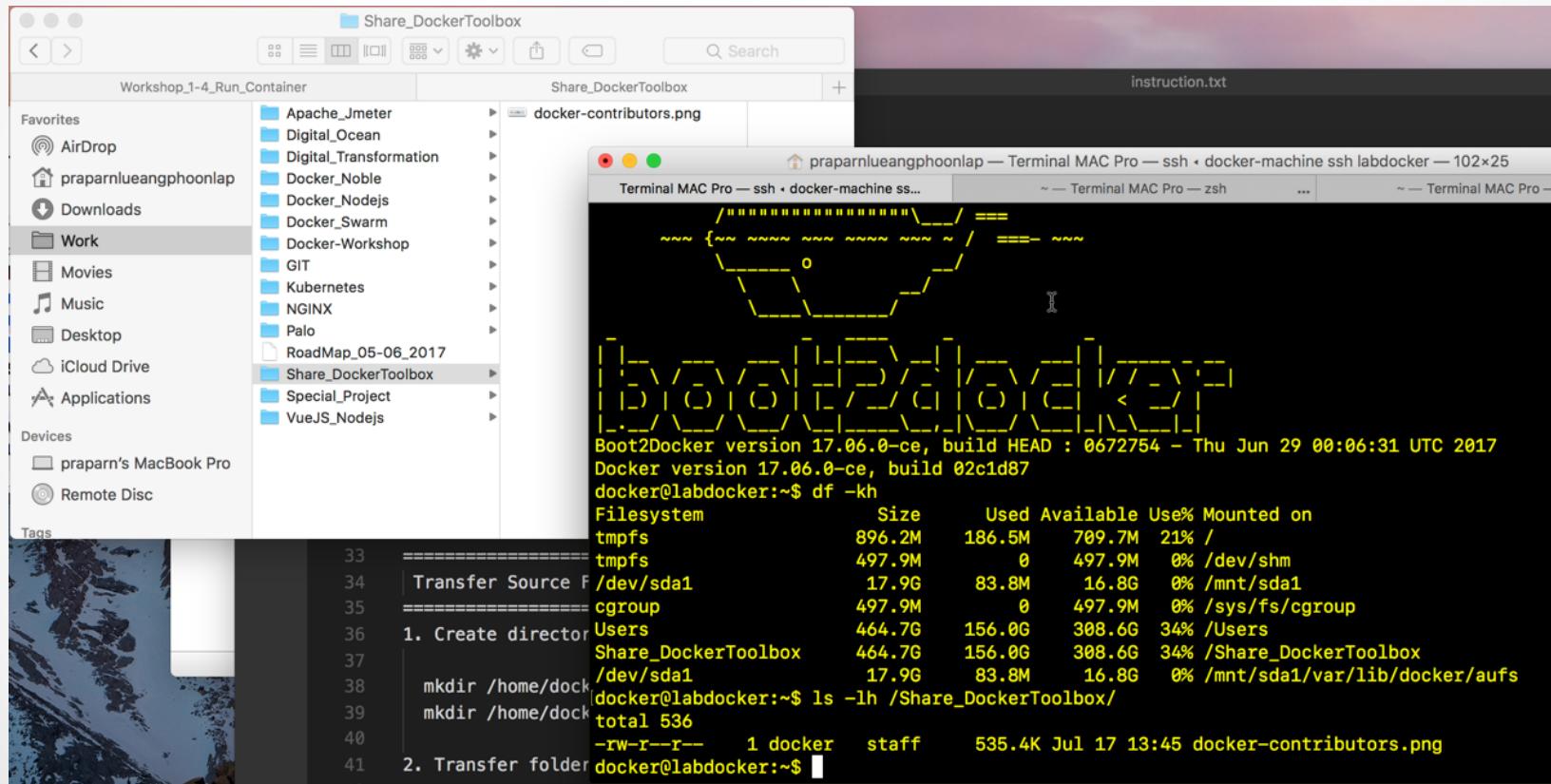
Ex: docker volume create --driver local --opt type=tmpfs \ --opt device=tmpfs --opt o=size=100m,uid=1000 tmptest

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

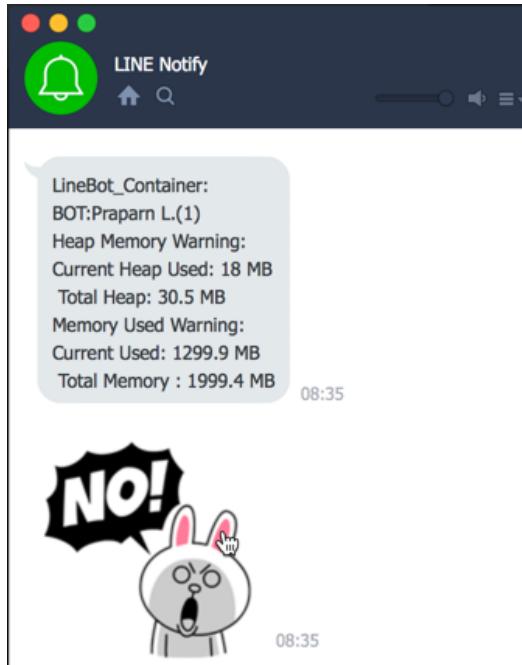
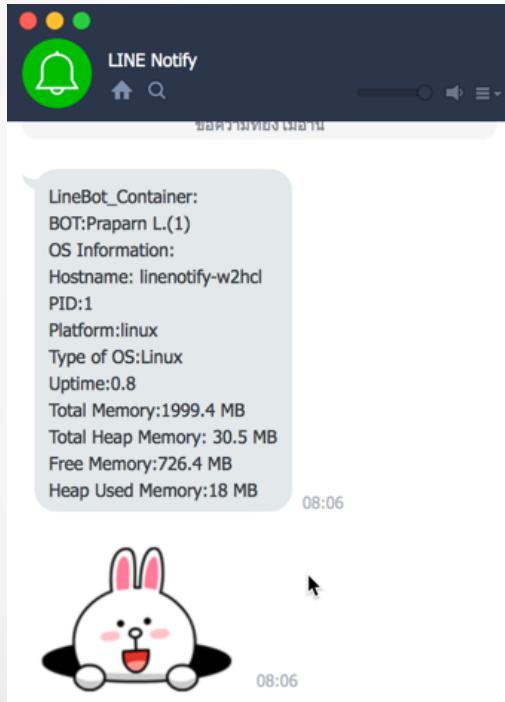
Workshop 1-7: Share Volume

- Before Workshop
- Following manual "create sharefile for dockertoolbox" to initial share volume



Workshop 1-7: Share Volume

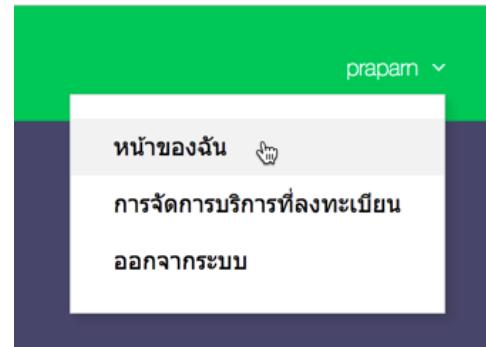
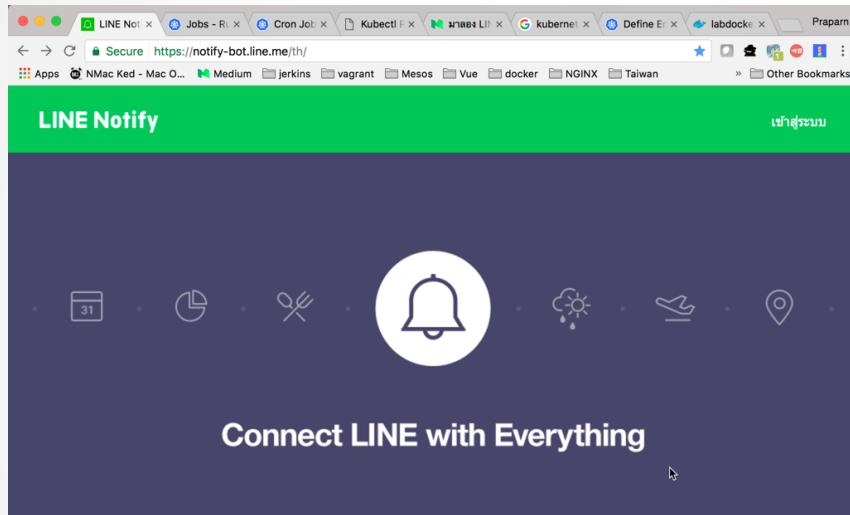
- Part 1: Host Path debug program
- Purpose: ทำการ Map volume เพื่อนำ source code เข้าไปทำการ debug/run บน container
- Example on WorkShop: "Monitor and LINE notify container"



LINE

Workshop 1-7: Share Volume

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



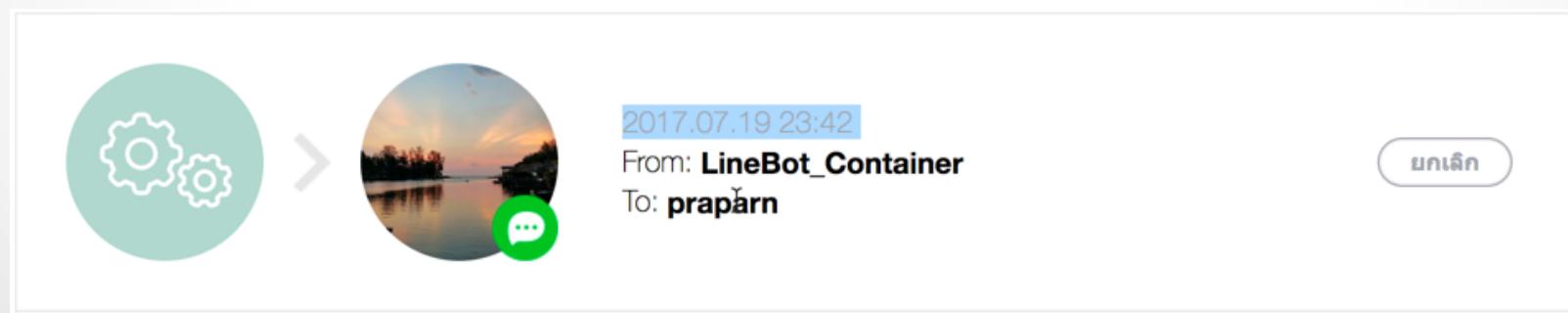
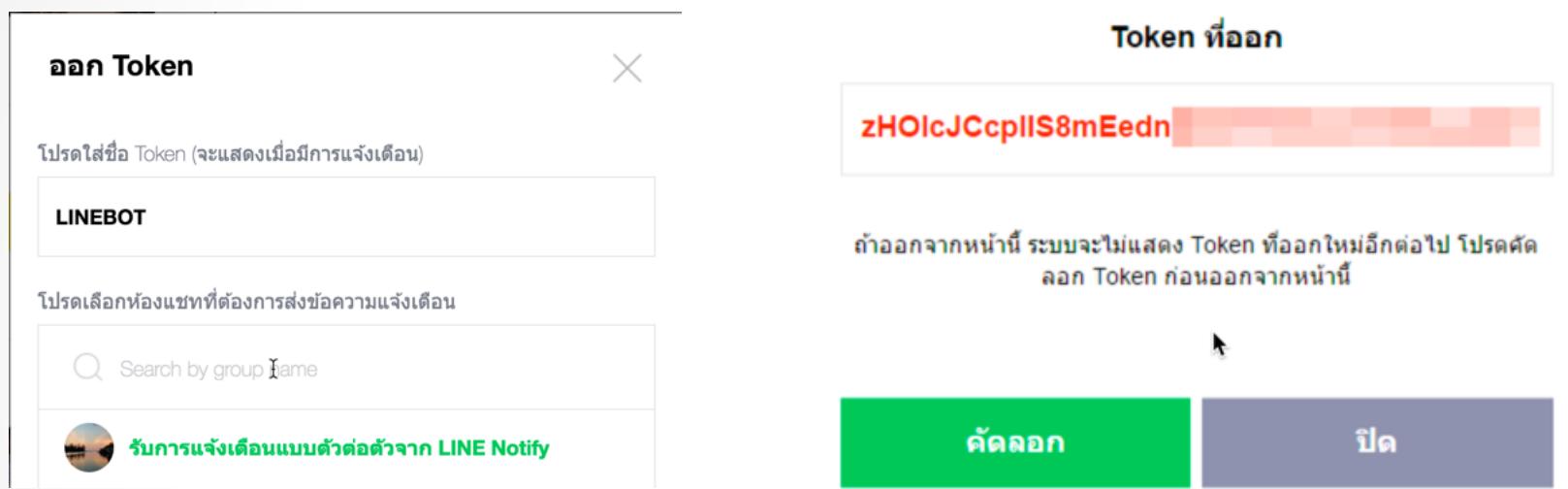
ออก Access Token (สำหรับผู้พัฒนา)

เมื่อใช้ Access Token แบบบุคคล จะสามารถตั้งค่าการแจ้งเตือนได้โดยไม่ต้องลงทะเบียนกับเว็บเซอร์วิส



Workshop 1-7: Share Volume

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



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.jsnull{"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

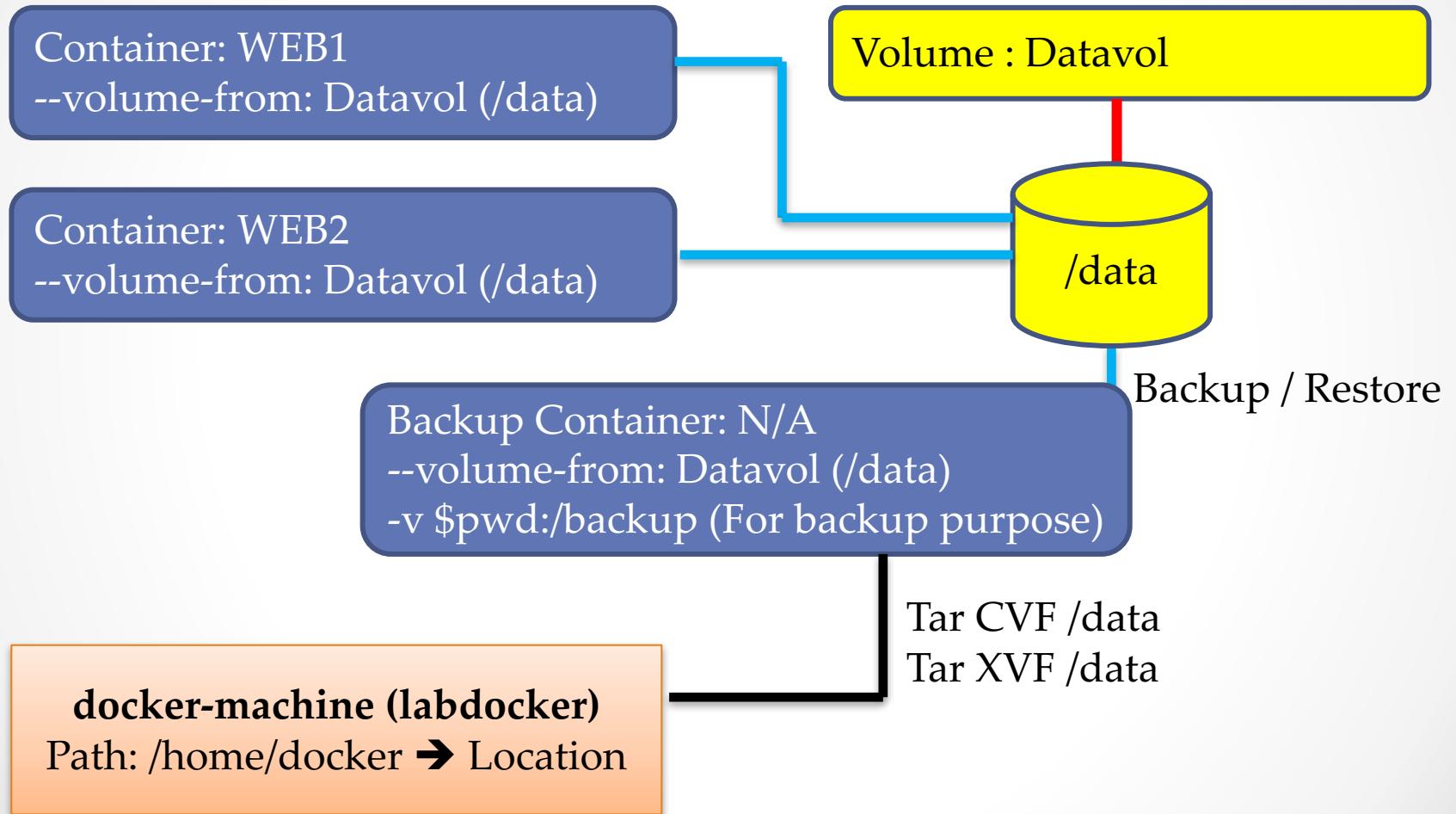
11:30

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 กำหนดให้ใส่วันและเวลาในล็อกไฟล์
 - since= <unix timestamps> กำหนดเวลาเริ่มตรวจสอบล็อกไฟล์ หรือ กำหนด yyyy-mm-dd
 - tail = "all" / number of line

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

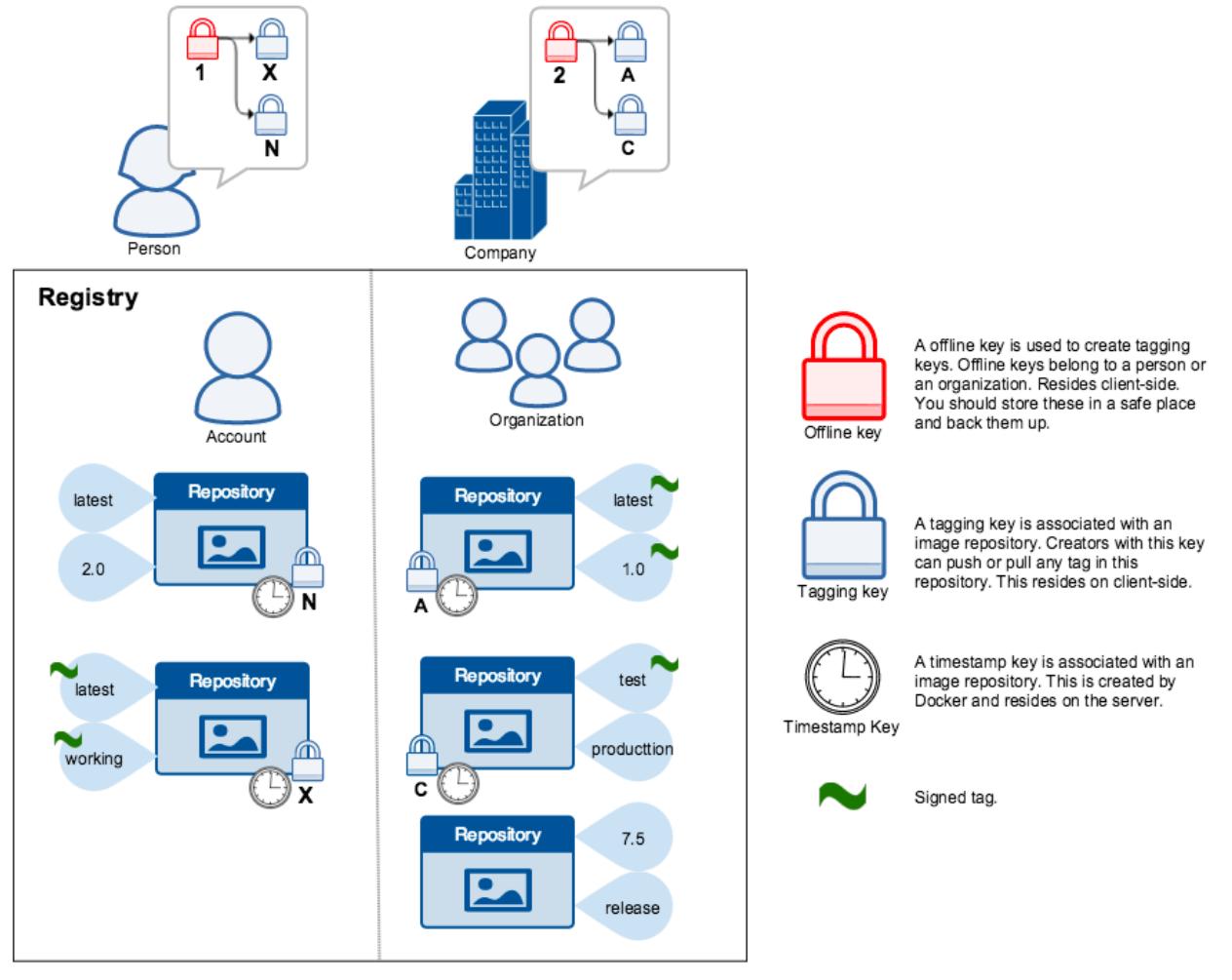
- Secure enhancement by limit capability of container inside
 - Normally user will be “root” inside container
 - Almost container job (web server, restful server, database server) don’t need high privilege as “root” for operate
 - Some activity should be prohibit on container
 - Mount disk operation
 - 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) ในการส่ง 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

- Part 1: 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: sha256:65242e8220a341cec40628caaee77eb4acd2fc252329aa853526fde15a4a1d85 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

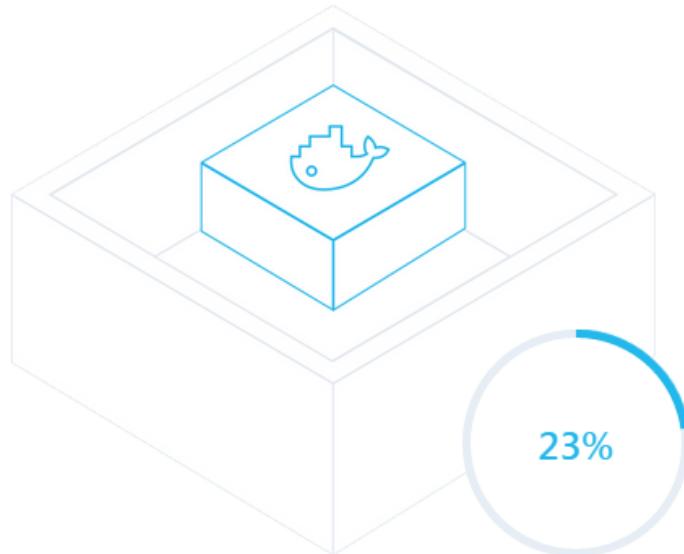
• • •

Docker: The Next-Gen of Virtualization



Kitematic

- GUI tool for operate docker

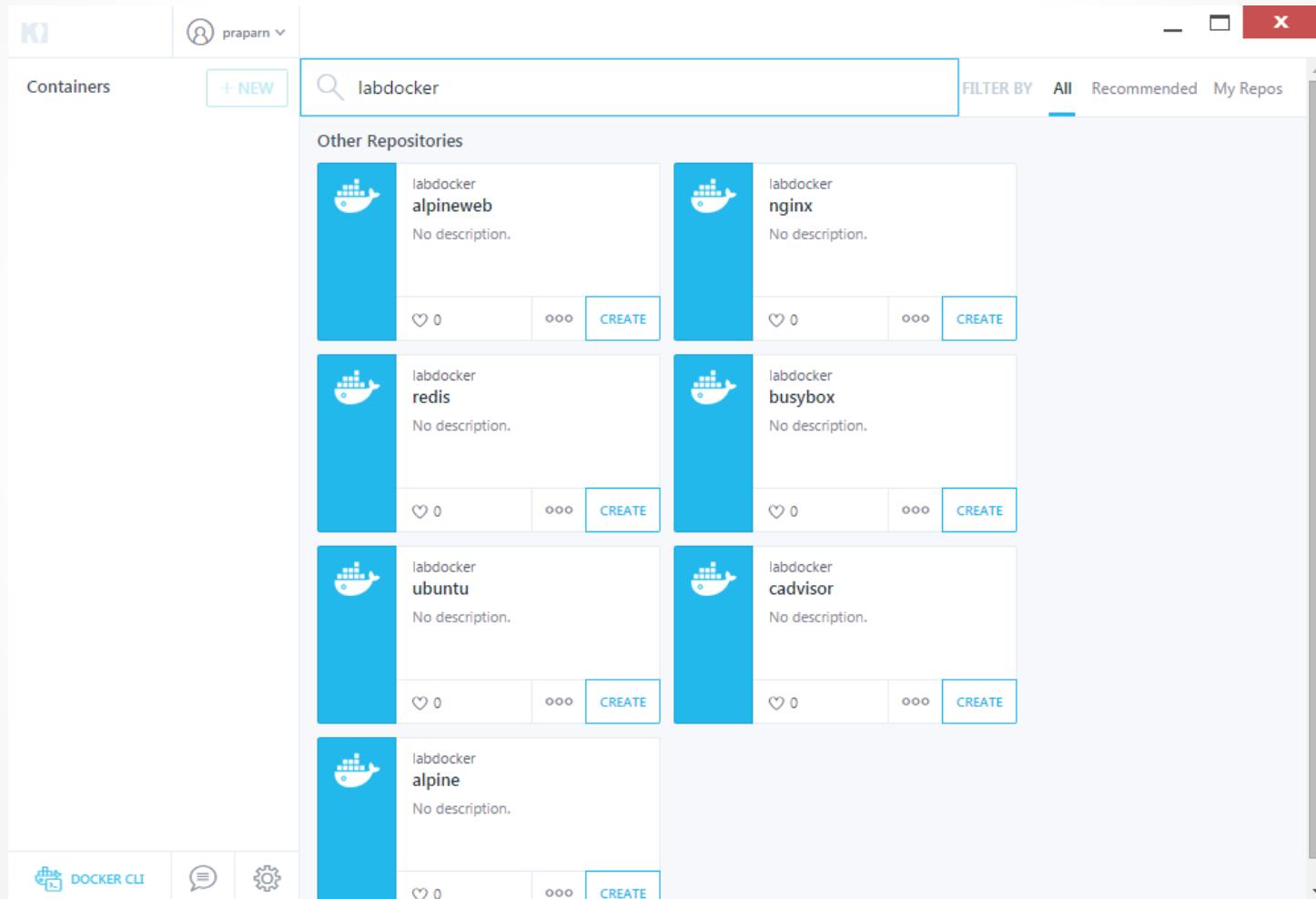


Starting Docker VM

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



Docker: The Next-Gen of Virtualization



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)
- Shipyard for Docker
- Q&A

Dockerfile and Build

• • •

Dockerfile

- docker file คือการคำสั่งที่ใช้สร้าง image ขึ้นเพื่อใช้งานโดยเฉพาะ 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 เท่าที่จำเป็นต้องใช้งานบน 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 โดยเราสามารถ ignore cache ด้วย option “--no-cache=true”
 - ตามปกติ docker จะเปรียบเทียบ instruction ที่กำหนดไว้กับ image layer ที่
 - Consider: instruction, base image

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)

Docker-machine: labdocker
Path: /Share_DockerToolbox

/nodejs

dockerfile



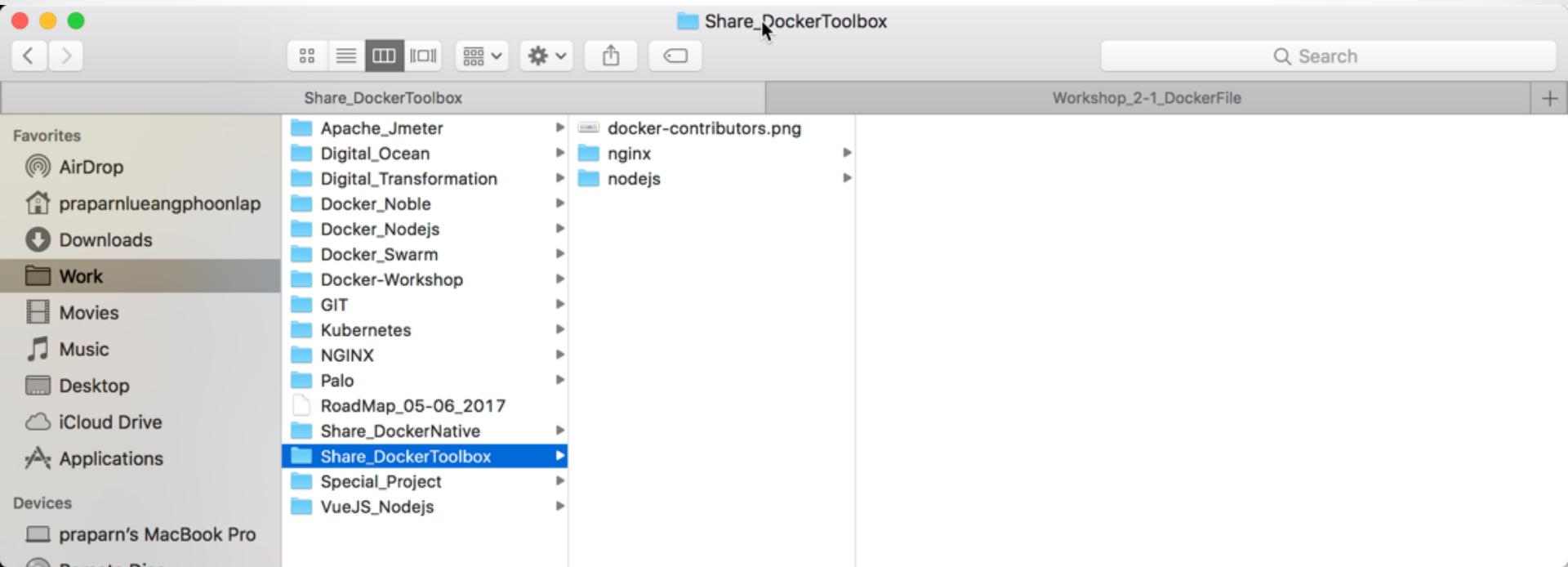
/nginx

dockerfile



Workshop 2-1: DockerFile

- **NODEJS container**
- copy folder “nodejs”, “nginx” ไปวางไว้ที่ /Share_DockerToolbox



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

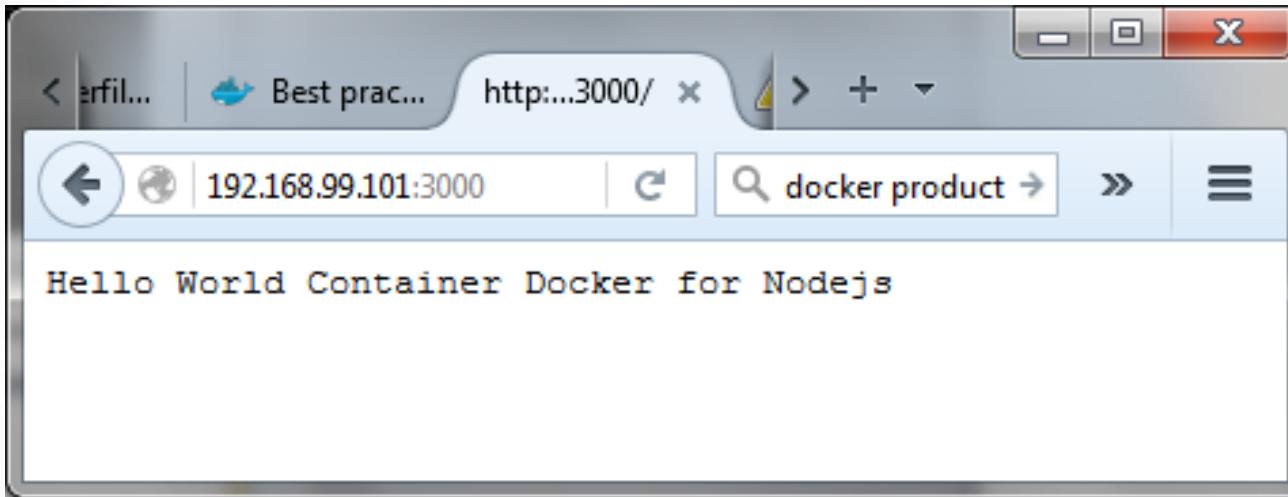
- Output

```
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

- **NGINX container**
- แก้ไข configue nginx.conf เพื่อใช้ทำ reverse proxy

```
upstream nodejs_web {  
    server 192.168.99.131:3000;  
}  
  
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;  
    }  
}
```

Workshop 2-1: DockerFile

- ตรวจสอบ dockerfile ตามตัวอย่างด้านล่าง

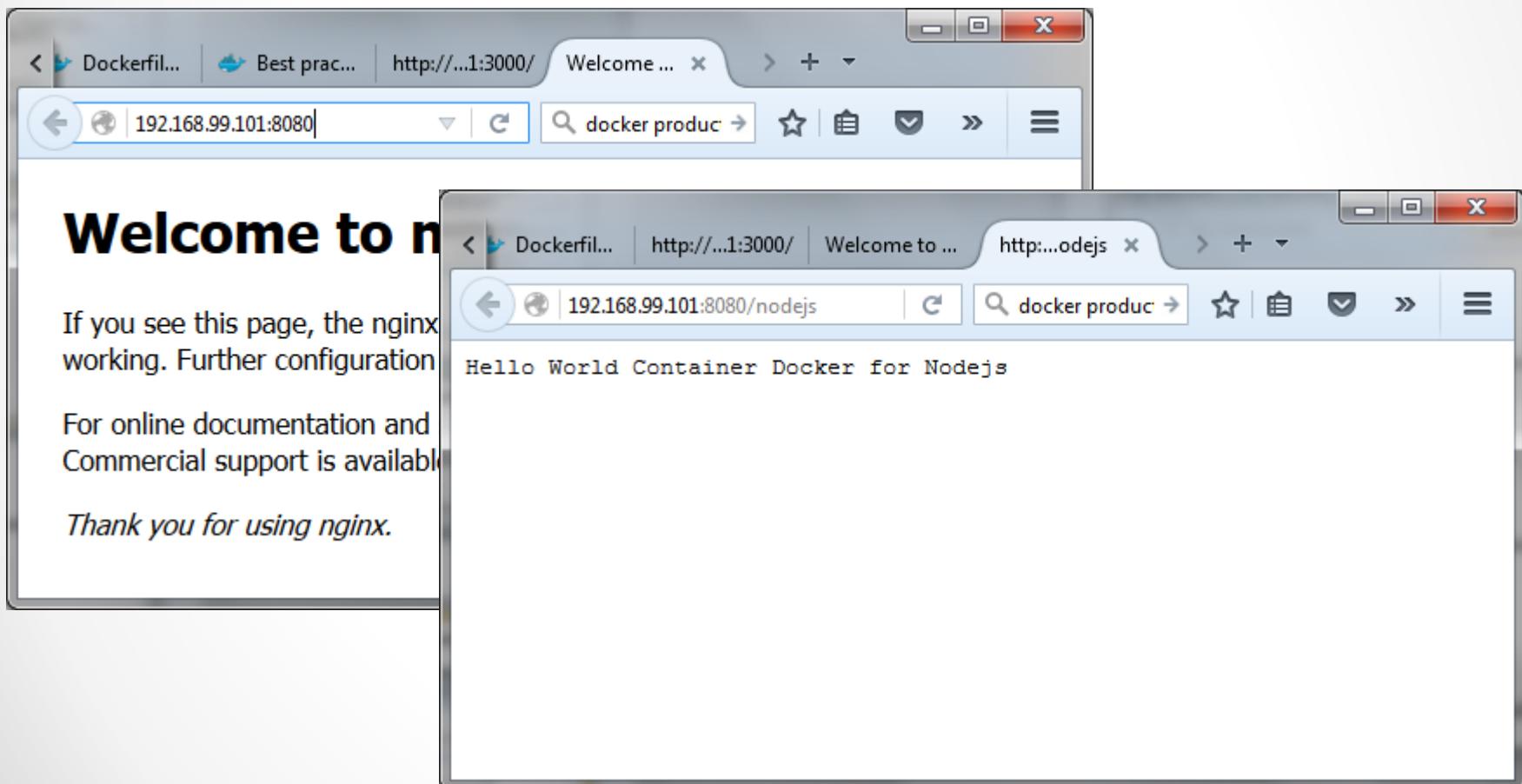
```
FROM labdocker/alpine:latest
MAINTAINER Praparn Lueangphoonlap (eva10409@gmail.com)
LABEL Description="NodeJS/NGINX Build Container" Version="1.0"
RUN apk update && \
    apk add nginx
COPY nginx.conf /etc/nginx/nginx.conf
WORKDIR /usr/sbin
ENTRYPOINT ["nginx","-c","/etc/nginx/nginx.conf"]
EXPOSE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



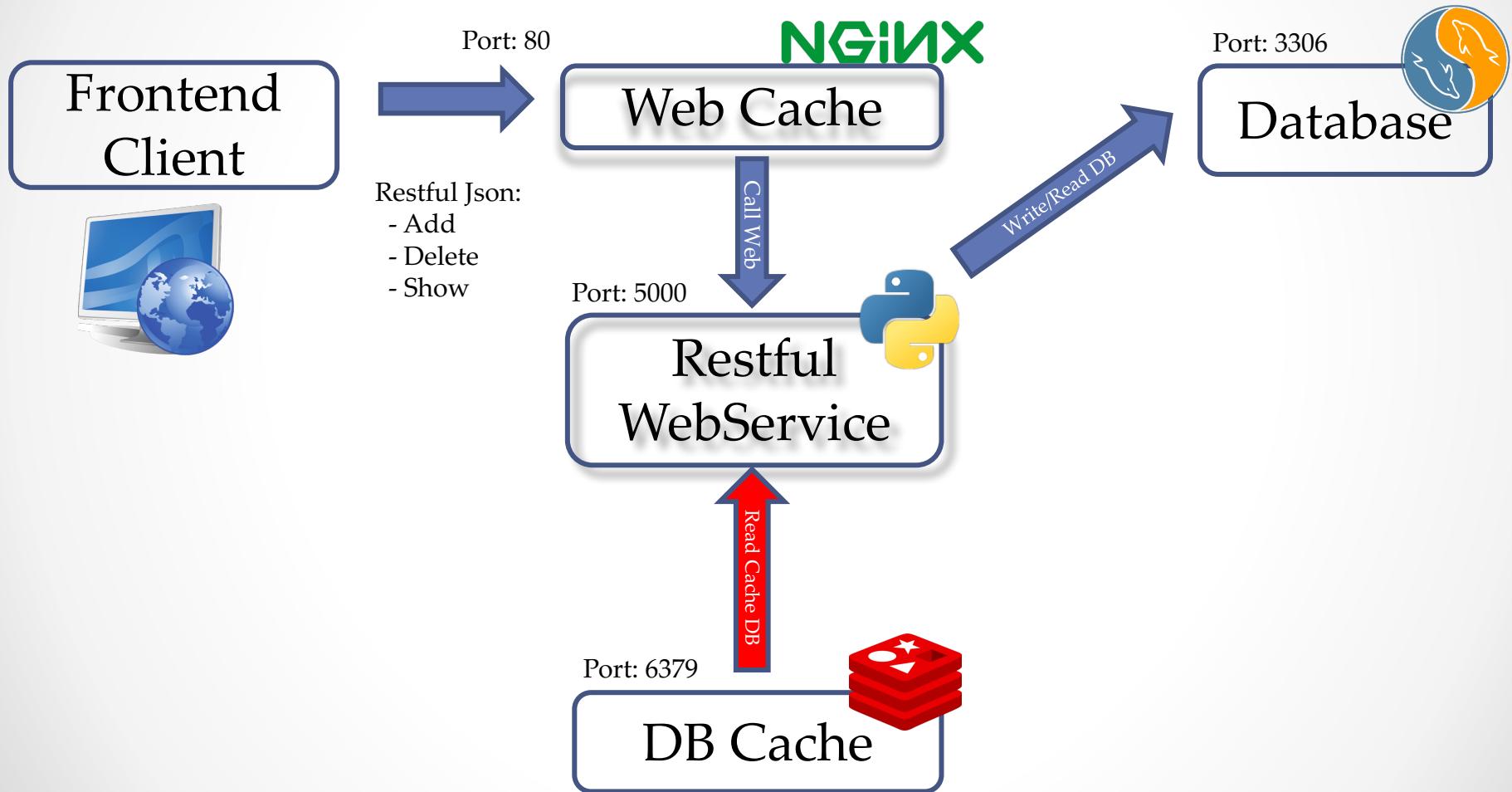
Workshop 2-1: DockerFile

- 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),
        DESRIPE 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, DESRIPE) VALUES (%s,%s,%s)", (req_json['uid'], req_json['user'], req_json['desripe']))
    #curl -i -H "Content-Type: application/json" -X POST -d '{"uid": "1", "user":"Praparn Luangphoonlap", "desripe":"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;  
    [REDACTED]  
    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
```



Mysql Database



Redis Key-Value Database

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 \ /Share_DockerToolbox/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 80
```

```
docker build -t labdocker/webcache:1.0 \
-f dockerfile_nginx /Share_DockerToolbox/python_restfulset/
```

Workshop 2-1: DockerFile

- Test run container

```
~ — Terminal MAC Pro — ssh + docker-machine ssh labdocker ~ — Terminal MAC Pro — zsh ~ — Terminal MAC Pro — -bash
docker@labdocker:~$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS              NAMES
13f776fdf3c0        labdocker/webcache:1.0   "nginx -c /etc/nginx..."   21 seconds ago    Up 21 seconds      0.0.0.0:80->80/tcp   webcache
42e238466523        labdocker/webservice:1.0  "python main.py"       26 seconds ago    Up 26 seconds      0.0.0.0:5000->5000/tcp  webservice
5781e3b3da50        labdocker/redis:latest    "docker-entrypoint..."  36 seconds ago    Up 35 seconds      6379/tcp           cachedb
0258e5719d3f        labdocker/mysql:latest   "docker-entrypoint..."  About a minute ago Up About a minute  3306/tcp           maindb
docker@labdocker:~$
```

```
praparnlueangphoonlap — Terminal MAC Pro — zsh — 148x24
~ — Terminal MAC Pro — ssh + docker-machine ssh labdocker ~ — Terminal MAC Pro — zsh
praparns-MacBook-Pro% export Server_IP=192.168.99.100
praparns-MacBook-Pro% export Server_Port=80
praparns-MacBook-Pro% curl http://$Server_IP:$Server_Port/
<H1> Welcome Page from Container Python Lab </H1>Checkpoint Date/Time: Sun Aug  6 16:04:37 2017
praparns-MacBook-Pro%
```

Welcome Page from Container Python Lab

Checkpoint Date/Time: Sun Aug 6 16:05:01 2017

Workshop 2-1: DockerFile

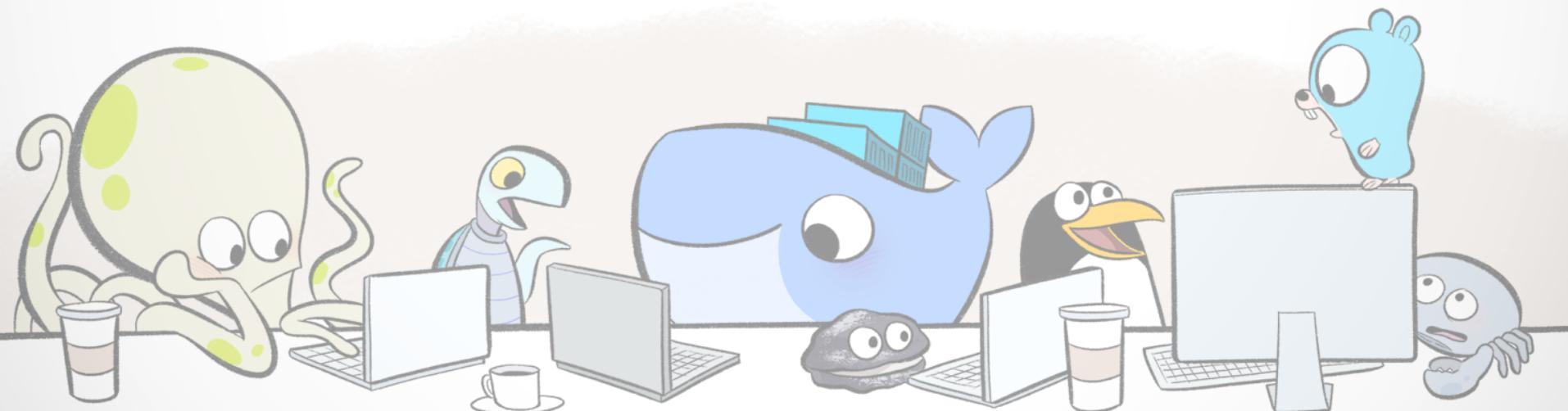
- Init / Insert Data

```
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 Wongsaisawan", "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

The screenshot shows the Postman interface with a successful API call. The URL is `http://192.168.99.103`. The response status is `200 OK` and the time taken is `20 ms`. The response body contains the text: `<H1> Welcome Page from Container Python Lab </H1>Checkpoint Date/Time: Sun Aug 13 14:02:51 2017`.

The screenshot shows the Postman interface with a successful API call. The URL is `http://192.168.99.103/init`. The response status is `200 OK` and the time taken is `30 ms`. The response body contains the text: `##### 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 with 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 these are buttons for '+', '...', 'No Environment' (with a dropdown arrow), 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'. There are buttons for 'Params', 'Send' (highlighted with a blue background), and 'Save'. Under the 'Body' tab, which is selected (indicated by an orange dot), the content type is set to 'JSON (application/json)'. Below this, there are radio buttons for 'form-data', 'x-www-form-urlencoded', 'raw' (which is selected and highlighted in orange), and 'binary'. The raw content is a JSON object:

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

. In the bottom section, under the 'Body' tab, there are tabs for 'Cookies', 'Headers (5)', and 'Tests'. On the right, the status is shown as 'Status: 200 OK' and 'Time: 25 ms'. Below this, there are buttons for 'Pretty', 'Raw', 'Preview', and 'HTML' (with a dropdown arrow). The preview area shows the response:

```
i 1 ##### Record was added #####
2
```

. To the right of the preview are icons for copy (square with arrows) 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 GET request to `http://192.168.99.103/users/1`. The top instance shows a response status of 200 OK with a time of 22 ms. The response body contains the JSON object `{ "id": 1, "name": "Praparn Luangphoonlap", "username": "Database Direct"}`. The bottom instance shows a response status of 200 OK with a time of 24 ms. The response body contains the JSON object `{ "id": 1, "name": "Praparn Luangphoonlap", "username": "Database Cache"}`.

Workshop 2-1: DockerFile

- Case for POSTMAN

The screenshot shows two separate API requests made using the Postman application.

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

- Authorization:** No Auth
- Body:** Pretty
- Response:** Status: 200 OK, Time: 22 ms
- Body Content:** `i 1 ##### Record was deleted #####
2`

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

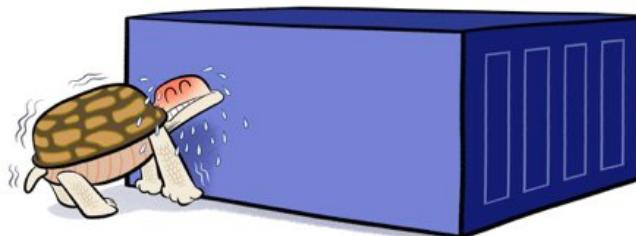
- Authorization:** No Auth
- Body:** Pretty
- Response:** Status: 200 OK, Time: 25 ms
- Body Content:** `i 1 ##### Record was deleted (Both Database Cache) #####
2`

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

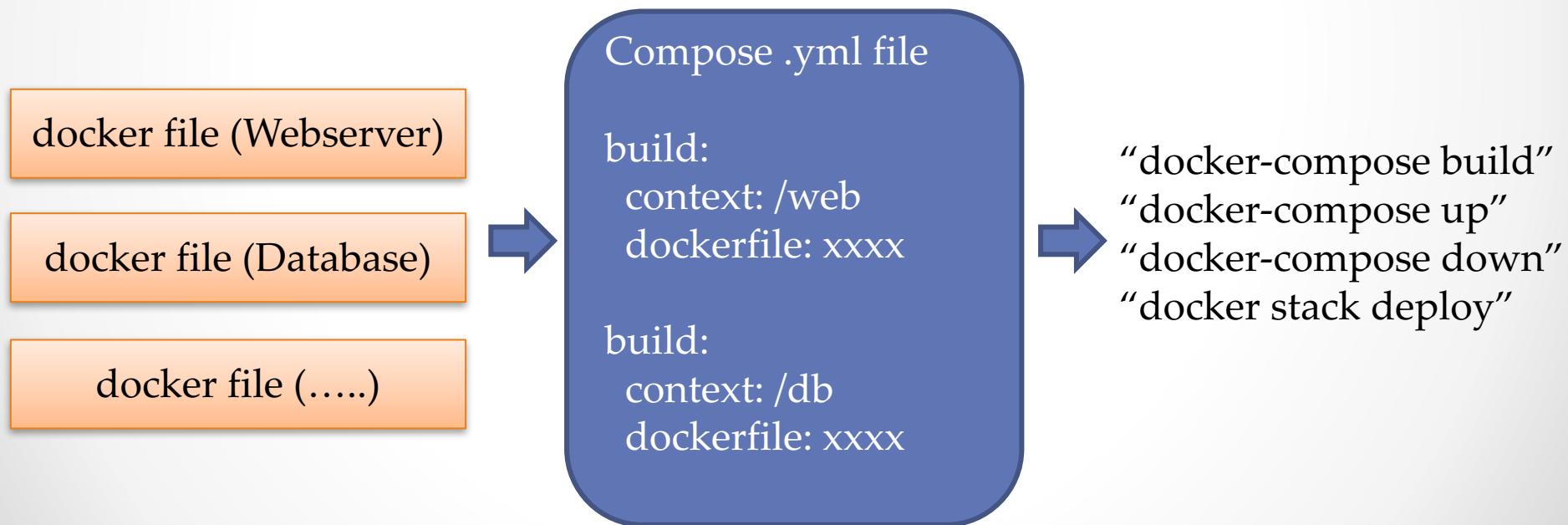
• • •

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
- เหมาะสมสำหรับการสร้าง 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
3.4	17.09.0+
3.3	17.06.0+
3.2	17.04.0+
3.1	1.13.1+
3.0	1.13.0+
2.3	17.06.0+ 
2.2	1.13.0+
2.1	1.12.0+
2.0	1.10.0+
1.0	1.9.1.+

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



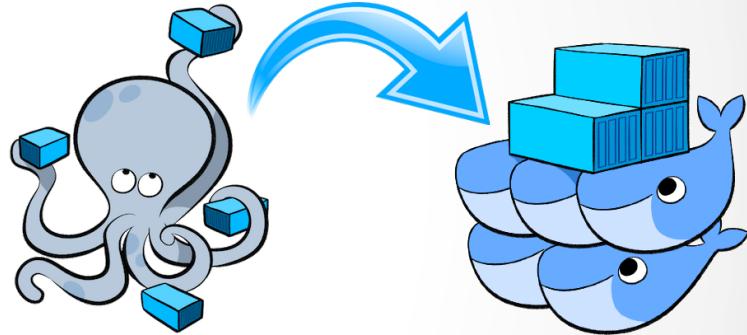
Ref:<https://docs.docker.com/compose/compose-file/#command>

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
 - tmpfs
 - external_links
 - links
 - network_mode
 - security_opt
 - stop_signal
 - sysctls
 - userns_mode



Workshop 2-2: Compose

- Install compose (Boot2docker)

```
curl -L https://github.com/docker/compose/releases/download/1.17.0/docker-compose-`uname -s`-`uname -m` > /home/docker/docker-compose
```

```
sudo cp /home/docker/docker-compose /usr/local/bin/docker-compose
```

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

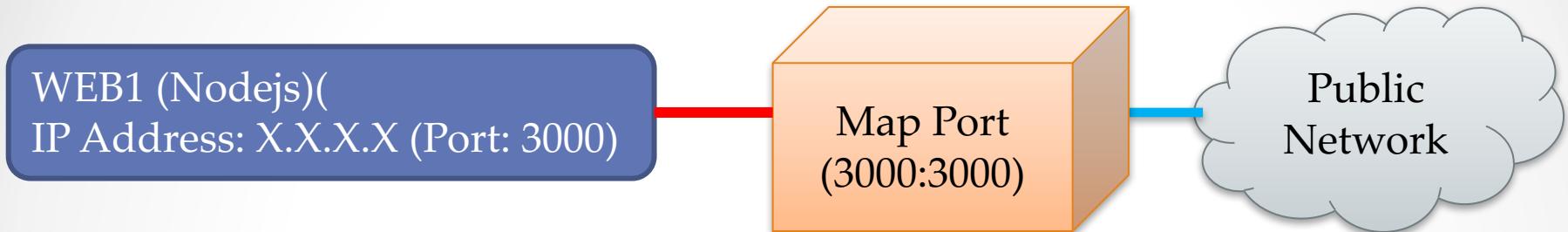
- Install ubuntu & redhat

```
curl -L https://github.com/docker/compose/releases/download/1.17.0/docker-compose-`uname -s`-`uname -m` > /usr/local/bin/docker-compose
```

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

Workshop 2-2: Compose

- Part1: componenodejs



Path: /Share_DockerToolbox

docker-compose.yml

/web

dockerfile



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 Address: 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: /Share_DockerToolbox

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 ใหม่มาจากรегистรี

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

- ทำการลบ container registry เพื่อยกเลิกการใช้งาน

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

Workshop 2-3: Registry

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

Docker-machine: labdocker

IP Address: 192.168.99.100

“labdocker.com”= 192.168.99.100

Container: Registry: 5000
(Repository)

Reference for
Registry



Private Key (*.key)
(labdocker.com)



Public Key (*.crt)
(labdocker.com)

/home/docker/labdockercert

Docker-machine: labdocker2

IP Address: 192.168.99.102

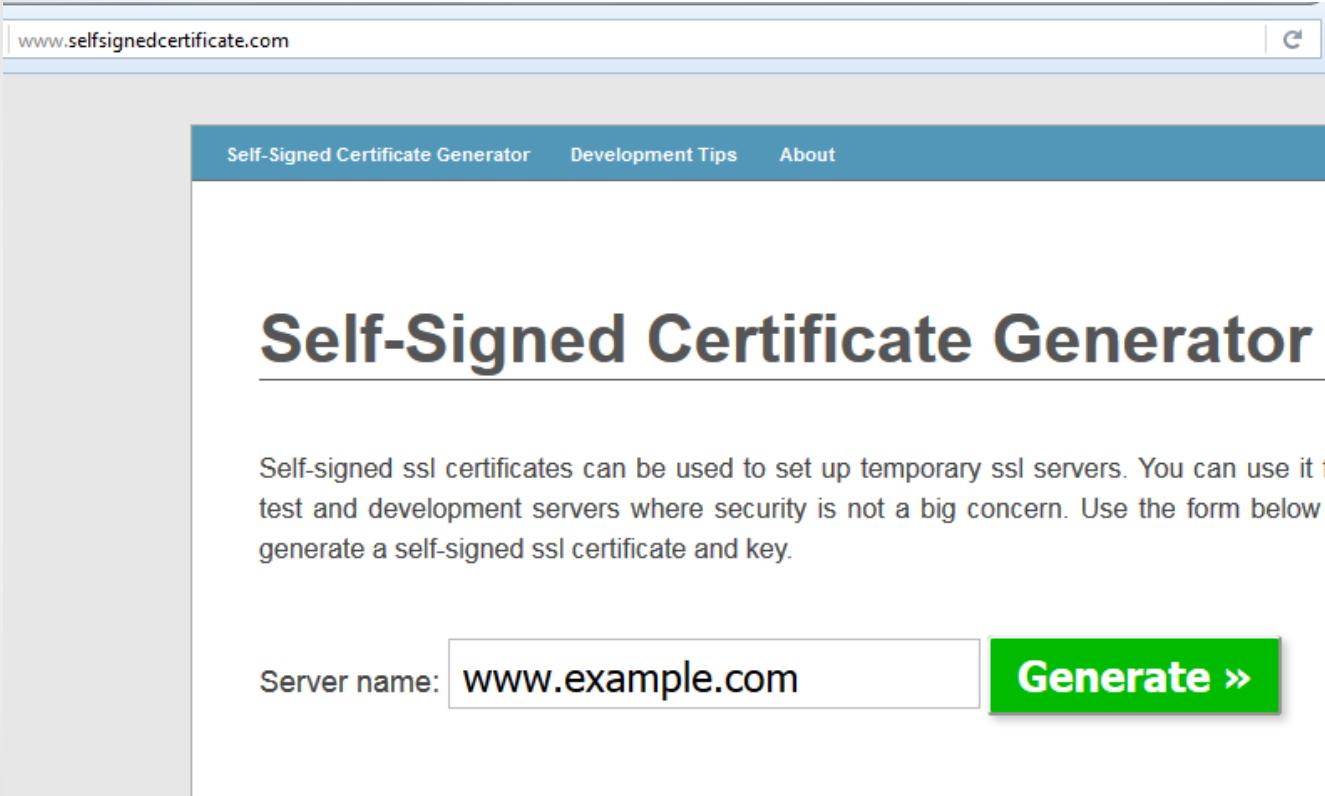
“labdocker.com”= 192.168.99.100

Certificate Authority (CA)
/ Self-Certificate

Generate Key

Workshop 2-3: Registry

- 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 contains the server name "www.example.com", and a large green button labeled "Generate »" is positioned next to it.



The sslstore logo features a stylized keyhole icon above the word "sslstore" in a green and blue sans-serif font.



An advertisement for sslstore. It includes the text "AFFORDABLE SSL CERTIFICATES" in white on a blue background, and "ISSUED IN MINUTES*" in green on a white background.

Workshop 2-3: Registry

- สร้าง docker-machine ในมือกหนึ่งเครื่องชื่อ “labdocker2”

```
docker-machine create --driver virtualbox  
labdocker2
```

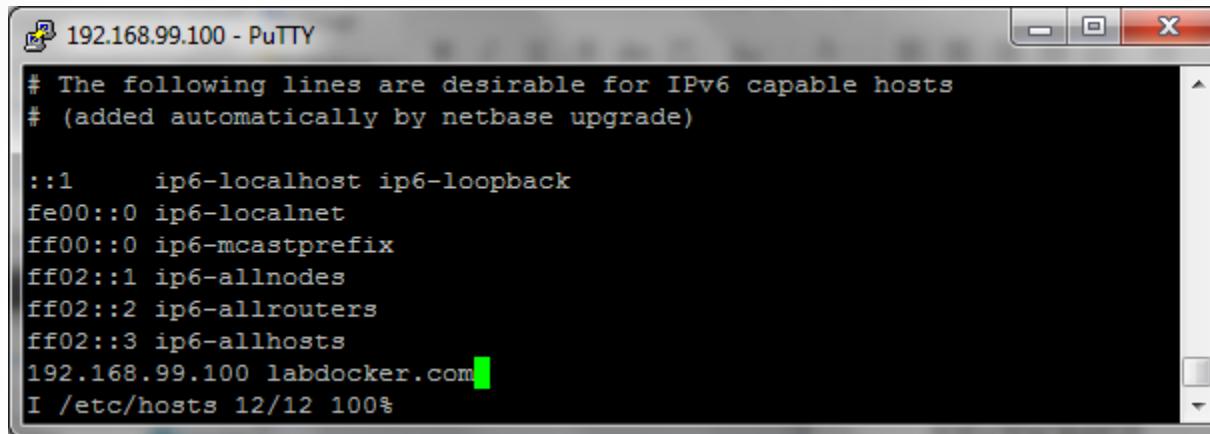
- ตรวจสอบ IP Address และ ssh เข้าสู่ docker-machine “labdocker2”

```
docker-machine ls
```

```
C:\> Administrator: C:\Windows\system32\cmd.exe  
C:\>Users\praparn.COM3099.000>docker-machine start labdocker2  
(labdocker2) OUT | Starting VM...  
Started machines may have new IP addresses. You may need to re-run the `docker-m  
achine env` command.  
C:\>Users\praparn.COM3099.000>docker-machine ls  
NAME ACTIVE DRIVER STATE URL SWARM  
labdocker - virtualbox Running tcp://192.168.99.100:2376  
labdocker2 - virtualbox Running tcp://192.168.99.102:2376
```

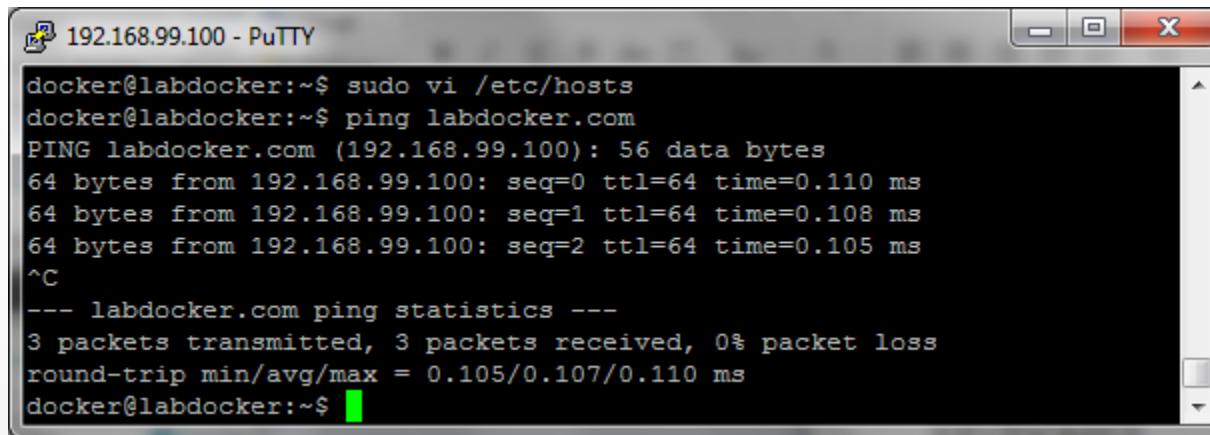
Workshop 2-3: Registry

- 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:~$
```

Workshop 2-3: Registry

- ในกรณี self-certificate เครื่อง docker-machine จำเป็นต้อง trust root ca โดย copy *.crt ไว้ในพาร์ทิชัน /etc/docker/certs.d/<domain name>

```
sudo mkdir /etc/docker/certs.d  
sudo mkdir /etc/docker/certs.d/labdocker.com:5000  
sudo cp /Share_DockerToolbox/labdocker.com.crt  
/etc/docker/certs.d/labdocker.com:5000
```

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

```
sudo cp /Share_DockerToolbox/labdocker.com.cert /usr/local/share/ca-certificates  
sudo update-ca-certificates  
sudo service docker restart
```

Workshop 2-3: Registry

- เริ่มใช้งาน registry โดย start container ดังนี้ (labdocker)

```
docker container run -d -p 5000:5000 \
    --restart=always --name registrylab \
    --mount type=bind,source=/home/docker,target=/var/lib/registry \
    --mount type=bind,source=/Share_DockerToolbox,target=/certs \
    --mount type=bind,source=/Share_DockerToolbox,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
```

Workshop 2-3: Registry

- ในการตรวจสอบ registry image file สำหรับ registry ที่มีการ enable TLS เรียนรู้อยแล้วต้องเพิ่มพารามิเตอร์ “--cacert” เพื่อบรรบ 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
```

Workshop 2-3: Registry

- ดึง image ผ่าน labdocker 2

```
docker image pull labdocker.com:5000/alpine:1.0
```

- 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



Workshop 2-3: Registry

- ดึง image ผ่าน labdocker 2

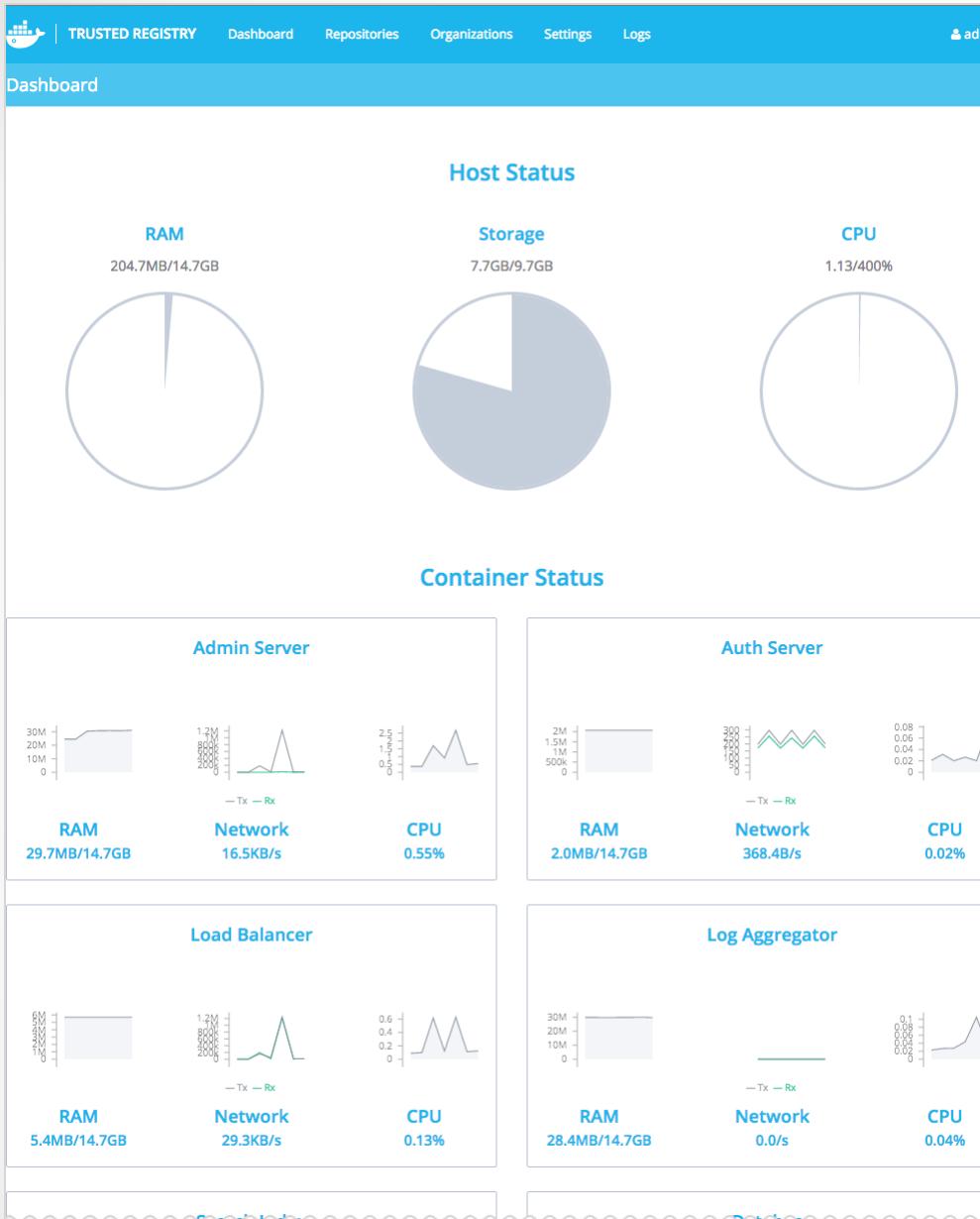
```
docker image pull labdocker.com:5000/alpine:1.0
```

- ทำการลบ container registry เพื่อยกเลิกการใช้งาน

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

- Optional: สำหรับเพิ่มการ authenticate login ก่อนทำการ push registry

```
-v `pwd`/auth:/auth \  
-e "REGISTRY_AUTH=htpasswd" \  
-e "REGISTRY_AUTH_HTPASSWD_REALM=Registry  
Realm" \  
Ref: https://docs.docker.com/registry/deploying/
```



Docker: The Next-Gen of Virtualization



Trust Registry

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

Open Source Registry (Experiment)

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

The screenshot shows the Portus interface. The left sidebar has a dark background with white icons and text: 'Portus' at the top, followed by 'Dashboard' and 'Images'. The main area has a light gray background. At the top, it says 'fl avio/busybox'. Below that is a table with three rows. The columns are 'Repository', 'Tag', and 'Pushed at'. The first row shows 'fl avio/busybox' with '1.0.0' and '2015-04-22 14:50:08 UTC'. The second row shows 'fl avio/busybox' with '2.0.0' and '2015-04-24 14:12:17 UTC'. The third row shows 'fl avio/busybox' with 'latest' and '2015-04-24 14:15:34 UTC'. A user profile for 'fl avio' is visible in the top right corner.

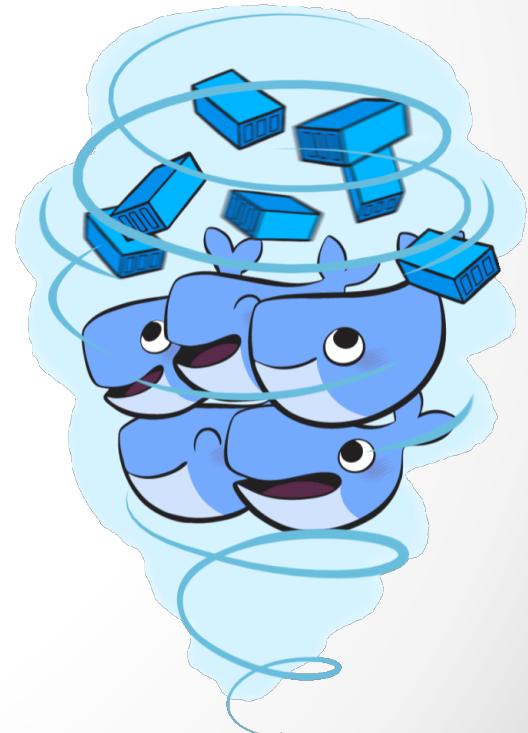
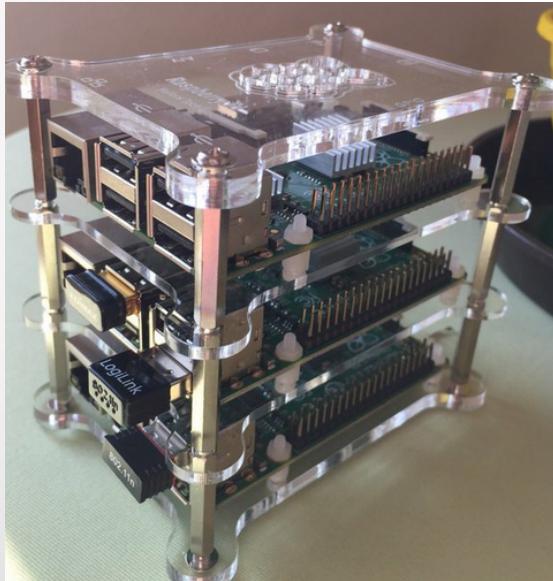
Repository	Tag	Pushed at
fl avio/busybox	1.0.0	2015-04-22 14:50:08 UTC
fl avio/busybox	2.0.0	2015-04-24 14:12:17 UTC
fl avio/busybox	latest	2015-04-24 14:15:34 UTC

Swarm

• • •

Swarm: Conceptual

- Swarm ถือเป็น native tool ในการจัดการ docker-engine หลายเครื่อง ให้รวมเป็น 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-engine: Node 1

Container: Web Server

Container: Database Server

Container: Etc.

Docker-engine: Node 2

Container: Web Server

Container: Database Server

Container: Etc.

Docker-CLI / API /
Compose / etc

Docker-engine: Node XX

Container: Web Server

Container: Database Server

Container: Etc.

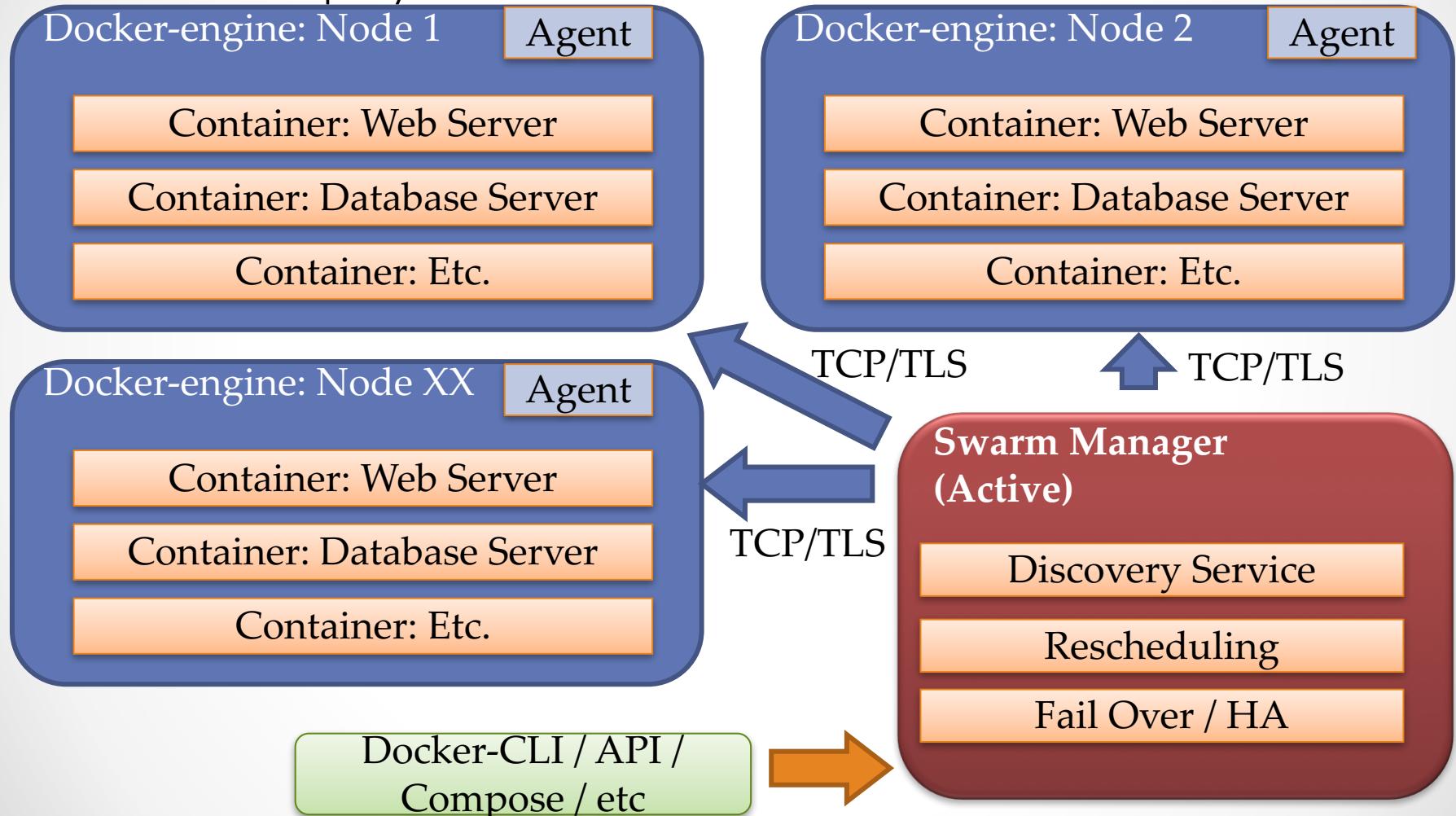
Docker-CLI / API /
Compose / etc

Docker: The Next-Gen of Virtualization



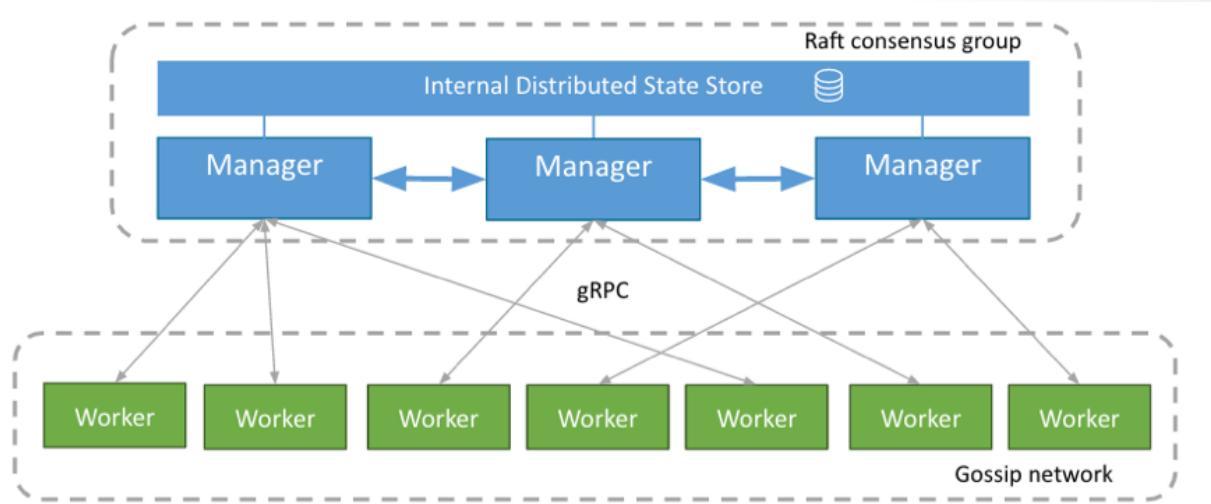
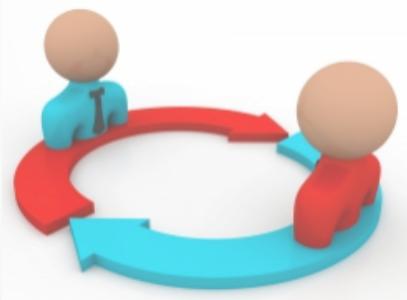
Swarm: Conceptual

- Swarm Deployment



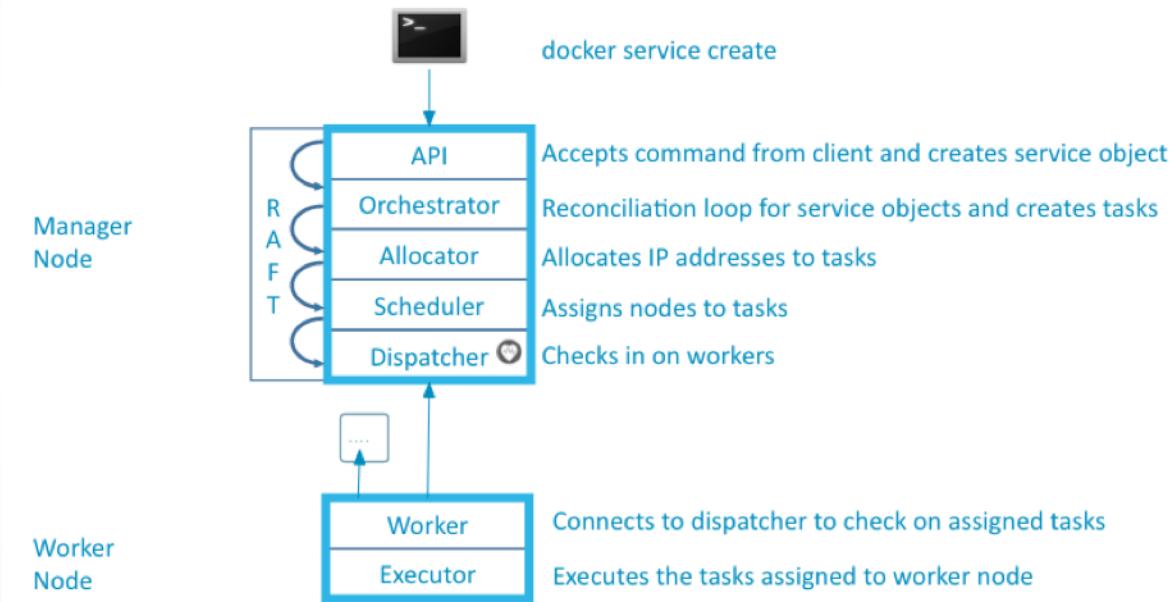
Swarm Mode Architecture

- 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

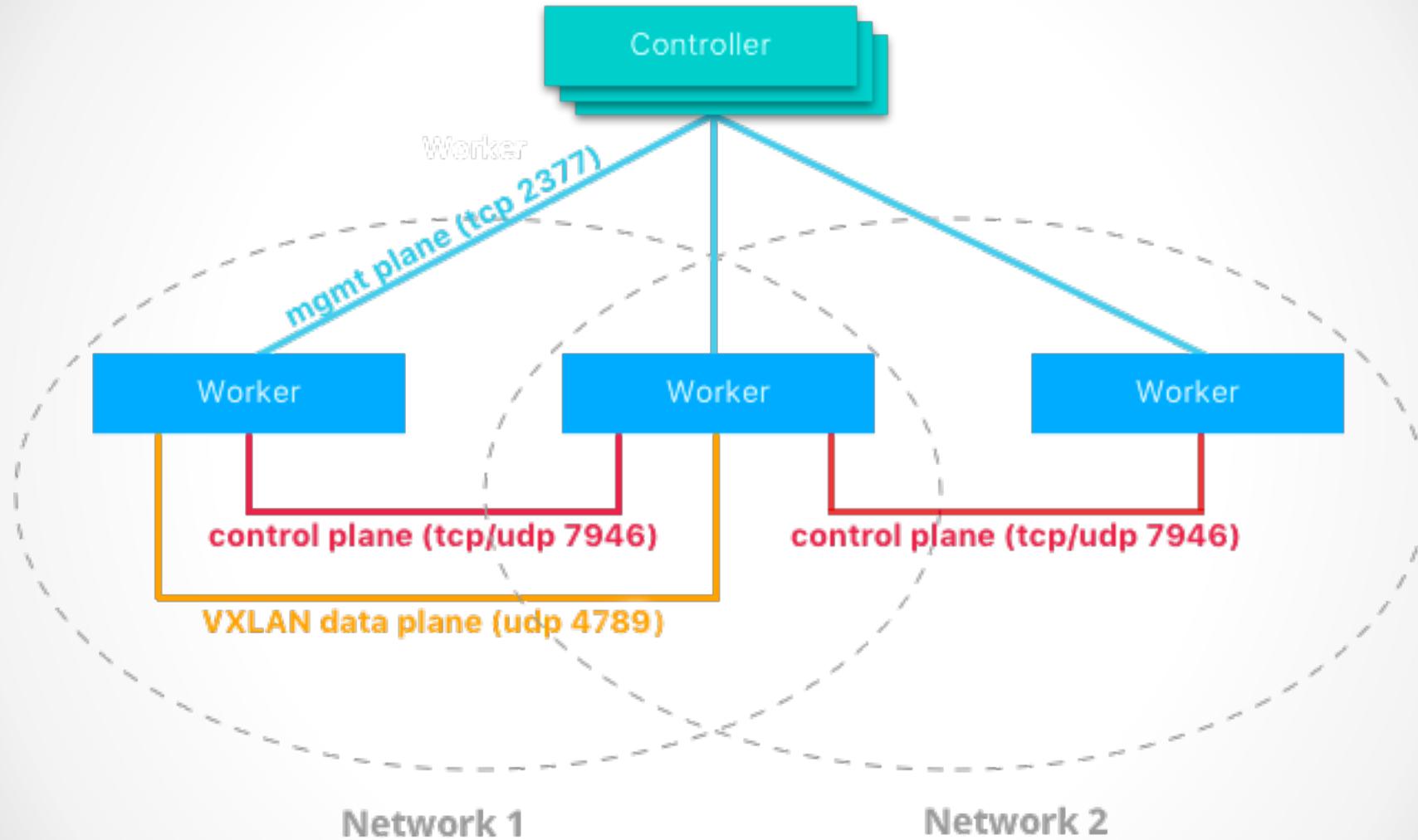


Swarm Mode Architecture

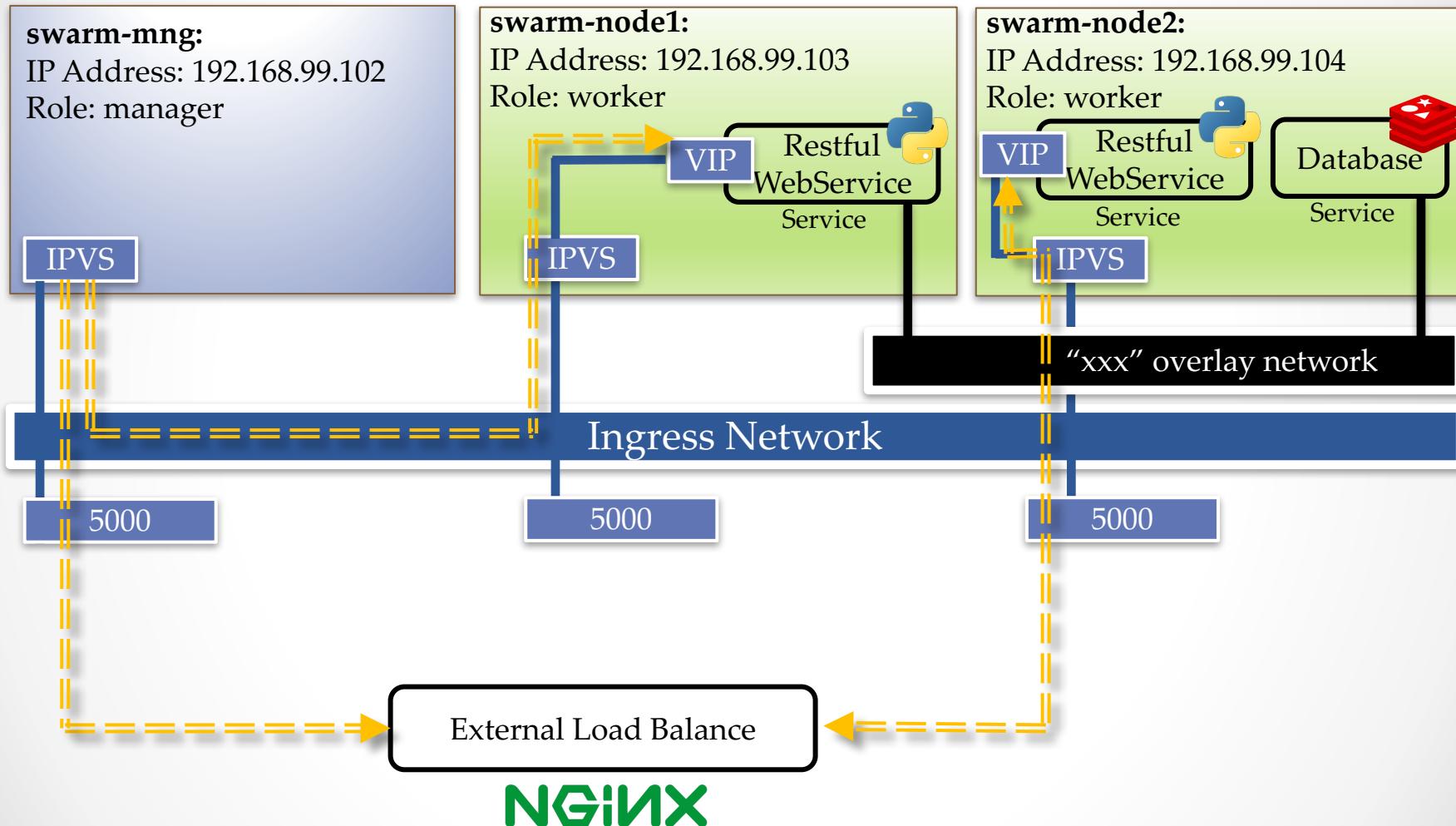
Node Breakdown



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 [DockerSwarm](#) 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
dashboard2.png	Add files via upload
events.png	Add files via upload
monitoring.md	Update monitoring.md
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_toews	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
@mrnonaki	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 192.168.99.102:2377
```

```
docker@swarm-mng:~$ docker swarm init --advertise-addr 192.168.99.102:2377
Swarm initialized: current node (j65b873flobaqui845vq40m6e) is now a manager.

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

  docker swarm join --token SWMTKN-1-36b4m3gpm3rd7szs82p6hsyg4cml5cq86vc3l4sfzezei1iz6m-5makre25xpxb8y8i85vyedjnj 192.168.99.102:2377

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

docker@swarm-mng:~$ docker swarm join-token manager
To add a manager to this swarm, run the following command:

  docker swarm join --token SWMTKN-1-36b4m3gpm3rd7szs82p6hsyg4cml5cq86vc3l4sfzezei1iz6m-bzcryc1kj16ftmttkxoppodjz 192.168.99.102:2377
```

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>
```

```
Boot2Docker version 17.06.0-ce, build HEAD : 0672754 - Thu Jun 29 00:06:31 UTC 2017
Docker version 17.06.0-ce, build 02c1d87
docker@swarm-node1:~$ docker swarm join \
> --token SWMTKN-1-36b4m3gpm3rd7szs82p6hsyg4cml5cq86vc3l4sfzezei1iz6m-5makre25xpxb8y8i85vyedjn \
|> 192.168.99.102:2377
This node joined a swarm as a worker.
docker@swarm-node1:~$ █
```

```
Boot2Docker version 17.06.0-ce, build HEAD : 0672754 - Thu Jun 29 00:06:31 UTC 2017
Docker version 17.06.0-ce, build 02c1d87
docker@swarm-node2:~$ docker swarm join \
> --token SWMTKN-1-36b4m3gpm3rd7szs82p6hsyg4cml5cq86vc3l4sfzezei1iz6m-5makre25xpxb8y8i85vyedjn \
|> 192.168.99.102:2377
This node joined a swarm as a worker.
docker@swarm-node2:~$ █
```

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 ที่ประกาศให้ผู้อื่นมาคุยในกรณี เป็น swarm-manager
 - --availability: <"active"/"pause"/"drain">
 - Etc.

Swarm Init/Join

- ตรวจสอบ Token ที่ใช้ในการ Join Swarm (Worker/Manager)

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

```
docker@swarm-mng:~$ docker swarm join-token worker
To add a worker to this swarm, run the following command:

  docker swarm join --token SWMTKN-1-36b4m3gpm3rd7szs82p6hsyg4cml5cq86vc3l4sfzezei1iz6m-5makre25xpxb8y8i85vyedjnj 192.168.99.102:2377

docker@swarm-mng:~$ docker swarm join-token manager
To add a manager to this swarm, run the following command:

  docker swarm join --token SWMTKN-1-36b4m3gpm3rd7szs82p6hsyg4cml5cq86vc3l4sfzezei1iz6m-bzcryc1kj16ftmttkxoppodjz 192.168.99.102:2377

docker@swarm-mng:~$ █
```

- 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
```

```
[docker@swarm-mng:~$ docker node ls
ID                      HOSTNAME        STATUS  AVAILABILITY  MANAGER STATUS
9d691mgn29newn26phadsavny  swarm-node1  Ready   Active
d4495w8xwhtjrdmadm05dip6x *  swarm-mng    Ready   Active      Leader
embruwpk01py52hfceovy37cl  swarm-node2  Ready   Active      Reachable
docker@swarm-mng:~$
```

Swarm Init/Join

- Option:
 - promote <ID>: ทำการ promote node จาก worker กลายเป็น manager
 - demote <ID>: ทำการ depromote node จาก manager กลายเป็น worker ปกติ
 - Inspect <ID>: ตรวจสอบค่าคอนฟิกกูเรชั่นของ 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>: อัปเดตสถานะของ 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

- ทำการสร้าง docker-engine ขึ้นมาใหม่เพื่อรับการสร้าง swarm ดังนี้

```
docker-machine create --driver=virtualbox swarm-mng
```

```
docker-machine create --driver=virtualbox swarm-node1
```

```
docker-machine create --driver=virtualbox swarm-node2
```

- ตรวจสอบและบันทึก IP Address ของ docker-engine แต่ละเครื่องที่สร้างขึ้น

```
docker-machine ls
```

- เริ่ม initial swarm ที่เครื่อง swarm-mng

```
docker swarm init --secret labdocker --listen-addr  
192.168.99.101:2377
```

Workshop 2-4: Swarm Mode

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

```
docker swarm join \
--token SWMTKN-1-
316tn4xze2w2zi7jpuhxjdyoniu18k0onh5ywrxinzmau8t71f-
4750wvm6fqxi8lbd3trlw7pk4 \
192.168.99.102:2377
```

- ตรวจสอบสถานะของ swarm หลังจาก join node เข้าไปเรียบร้อยแล้ว

```
docker node ls
```

ID	HOSTNAME	STATUS	AVAILABILITY	MANAGER STATUS
9d691mgn29newn26phadsavny	swarm-node1	Ready	Active	
d4495w8xwhtjrdmadm05dip6x *	swarm-mng	Ready	Active	Leader
embruwpko1py52hfceovy37cl	swarm-node2	Ready	Active	

Swarm Service

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

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

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

```
Praparn — ssh ▾ docker-machine ssh swarm-mng — bash — 100x
ssh ▾ docker...h swarm-mng      ssh ▾ docker...swarm-node1      ssh ▾ docker...swarm-node2
docker@swarm-mng:~$ docker service create --name nodejs \
|> labdocker/alpineweb:latest node hello.js
5z3df8fx0vvv3es5a51bhslwp
[ docker@swarm-mng:~$ docker service ls
ID          NAME      REPLICAS   IMAGE                      COMMAND
5z3df8fx0vvv  nodejs    0/1        labdocker/alpineweb:latest  node hello.js
docker@swarm-mng:~$ ]
```

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
docker@swarm-mng:~$ █
```

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

Swarm Service

docker service scale nodejs=5

```
praparnlueangphoonlap — Terminal MAC Pro — ssh · docker-machine ssh swarm-mng — 147x30
~ — Terminal MAC Pro — ssh · docker-machine ssh labdocke
docker@swarm-mng:~$ docker service create --name nodejs -dt \
|> labdocke/alpineweb:latest node hello.js
n96f925q4yq9vi94zdpmjff34
docker@swarm-mng:~$ docker service ls
ID          NAME      MODE      REPLICAS      IMAGE
n96f925q4yq9  nodejs    replicated  1/1        labdocke/alpineweb:latest
docker@swarm-mng:~$ docker service ps nodejs
ID          NAME      IMAGE      NODE      DESIRED STATE      CURRENT STATE      ERROR
PORTS
9lrwdm68zgq4  nodejs.1  labdocke/alpineweb:latest  swarm-mng  Running      Running 5 minutes ago

docker@swarm-mng:~$ docker service scale nodejs=5
nodejs scaled to 5
docker@swarm-mng:~$ docker service ls
ID          NAME      MODE      REPLICAS      IMAGE
n96f925q4yq9  nodejs    replicated  5/5        labdocke/alpineweb:latest
docker@swarm-mng:~$ docker service ps nodejs
ID          NAME      IMAGE      NODE      DESIRED STATE      CURRENT STATE      ERROR
PORTS
9lrwdm68zgq4  nodejs.1  labdocke/alpineweb:latest  swarm-mng  Running      Running 6 minutes ago
jmo3g3rfzfxi   nodejs.2  labdocke/alpineweb:latest  swarm-mng  Running      Running 26 seconds ago
zxjo6psq6hnh   nodejs.3  labdocke/alpineweb:latest  swarm-node2  Running      Running 15 seconds ago
tmdlb2878g9j   nodejs.4  labdocke/alpineweb:latest  swarm-node1  Running      Running 26 seconds ago
f6xe6zbprlyt  nodejs.5  labdocke/alpineweb:latest  swarm-node1  Running      Running 26 seconds ago

docker@swarm-mng:~$
```

Swarm Service

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

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

```
docker@swarm-mng:~$ docker service inspect nodejs | more
[
  {
    "ID": "n96f925q4yq9vi94zdpmjff34",
    "Version": {
      "Index": 174
    },
    "CreatedAt": "2017-08-10T15:19:01.295998146Z",
    "UpdatedAt": "2017-08-10T15:28:15.828876222Z",
    "Spec": {
      "Name": "nodejs",
      "Labels": {},
      "TaskTemplate": {
        "ContainerSpec": {
          "Image": "labdocker/alpineweb:latest@sha256:e14b3ab7fd7378177cd790d887656a6623b75bff183df7e2b1b2592691e419d3",
          "Args": [
            "node",
            "hello.js"
          ],
          "TTY": true,
          "StopGracePeriod": 10000000000,
          "DNSConfig": {}
        }
      }
    }
  }
]
```

Swarm Service

```
docker service ps nodejs
```

ID	NAME	IMAGE	NODE	DESIRED STATE	CURRENT STATE	ERROR
bgelggjkcfbt	nodejs.1	labdocker/alpineweb:latest	swarm-mng	Running	Running 2 minutes ago	
9lrdm68zgq4	_ nodejs.1	labdocker/alpineweb:latest	swarm-mng	Shutdown	Shutdown 2 minutes ago	

```
docker rm nodejs
```

ID	NAME	MODE	REPLICAS	IMAGE	PORTS
bgelggjkcfbt	nodejs	replicated	1	labdocker/alpineweb:latest	

Swarm Service

```
[docker@swarm-mng:~$ docker service update --help

Usage: docker service update [OPTIONS] SERVICE

Update a service

Options:
  --args command
  --config-add config      | Service command args
  --config-rm list          | Add or update a config file on a service
  --constraint-add list     | Remove a configuration file
  --constraint-rm list      | Add or update a placement constraint
  --container-label-add list | Remove a constraint
  --container-label-rm list  | Add or update a container label
  --credential-spec credential-spec | Remove a container label by its key
  -d, --detach               | Credential spec for managed service account (Windows only)
  --dns-add list             | Exit immediately instead of waiting for the service to converge (default true)
  --dns-option-add list      | Add or update a custom DNS server
  --dns-option-rm list        | Add or update a DNS option
  --dns-rm list               | Remove a DNS option
  --dns-search-add list       | Remove a custom DNS server
  --dns-search-rm list        | Add or update a custom DNS search domain
  --endpoint-mode string      | Remove a DNS search domain
  --entrypoint command        | Endpoint mode (vip or dnsrr)
  --env-add list              | Overwrite the default ENTRYPOINT of the image
  --env-rm list               | Add or update an environment variable
  --force                     | Remove an environment variable
  --group-add list            | Force update even if no changes require it
  --group-rm list              | Add an additional supplementary user group to the container
  --health-cmd string         | Remove a previously added supplementary user group from the container
                                | 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")
--rollback-parallelism uint                Maximum number of tasks rolled back simultaneously (0 to roll back all at once)
--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
docker@swarm-mng:~$ █
```

- 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
```

```
docker@swarm-mng:~$ docker service create --name nodejs -dt \
[> labdocker/alpineweb:latest node hello.js
n96f925q4yq9vi94zdpmjff34
docker@swarm-mng:~$ docker service ls
ID          NAME      MODE      REPLICAS      IMAGE
n96f925q4yq9    nodejs    replicated    1/1        labdocker/alpineweb:latest
PORTS
```

Orchestrator Assignment

- Swarm สามารถควบคุมการจ่ายงานให้จาก manager ไปยัง worker ได้หลากหลายเทคนิค เพื่อตอบสนองความต้องการของผู้ใช้งาน
- Docker stack deploy (Compose version 3.3)
 - 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

```
docker@swarm-mng:~$ docker service inspect nodejs|grep update
docker@swarm-mng:~$ docker node ls
ID           HOSTNAME   STATUS  AVAILABILITY  MANAGER STATUS
6bei4mbj5pd7yduyhp375b7z4 *  swarm-mng  Ready  Active  Leader
u2eju1jxy5k8qd30ucrhuhsf  swarm-node1  Ready  Active
wyw7pfddcumccm2uojesnjy9v  swarm-node2  Ready  Active
docker@swarm-mng:~$ docker service create -dt --constraint 'node.id==6bei4mbj5pd7yduyhp375b7z4' --name nodejs --replicas=5 labdocker/alpineweb:latest node hello.js
w87s7jaz5i8l944yq1t8v2eo6
docker@swarm-mng:~$ docker service ps nodejs
ID          NAME      IMAGE          NODE          DESIRED STATE    CURRENT STATE          ERROR
OR          PORTS
u8p7dnw2wta8  nodejs.1  labdocker/alpineweb:latest  swarm-mng  Running        Running  6 seconds ago
[           w3bm3p92kmdq  nodejs.2  labdocker/alpineweb:latest  swarm-mng  Running        Running  6 seconds ago
7pkkpmt4ew2l  nodejs.3  labdocker/alpineweb:latest  swarm-mng  Running        Running  6 seconds ago
2m8dmrzxotwr  nodejs.4  labdocker/alpineweb:latest  swarm-mng  Running        Running  6 seconds ago
vwbxkhpftup7  nodejs.5  labdocker/alpineweb:latest  swarm-mng  Running        Running  6 seconds ago
[ docker@swarm-mng:~$ █
```

Orchestrator Assignment

- Remove constrain/Add constrain
 - node.hostname

```
docker@swarm-mng:~$ docker service update -dt --constraint-rm 'node.id==6bei4mbj5pd7yduyhp375b7z4' --constraint-add 'node.hostname!=swarm-mng' nodejs
nodejs
docker@swarm-mng:~$ docker service inspect nodejs|grep update
    "Message": "update in progress"
docker@swarm-mng:~$ docker service inspect nodejs|grep update
    "Message": "update completed"
docker@swarm-mng:~$ docker service ps nodejs
[REDACTED]
ID          NAME      IMAGE           NODE        DESIRED STATE  CURRENT STATE
ERROR       PORTS
pahz73v2ma91  nodejs.1  labdocker/alpineweb:latest  swarm-node1  Running     Running about a minute ago
u8p7dnw2wta8   \_ nodejs.1  labdocker/alpineweb:latest  swarm-mng   Shutdown   Shutdown about a minute ago
oj1koyxqgl55  nodejs.2  labdocker/alpineweb:latest  swarm-node1  Running     Running about a minute ago
w3bm3p92kmdq   \_ nodejs.2  labdocker/alpineweb:latest  swarm-mng   Shutdown   Shutdown about a minute ago
1k57ahejx8pv  nodejs.3  labdocker/alpineweb:latest  swarm-node2  Running     Running about a minute ago
7pkkpmt4ew21   \_ nodejs.3  labdocker/alpineweb:latest  swarm-mng   Shutdown   Shutdown about a minute ago
l11o7jo0xdy6  nodejs.4  labdocker/alpineweb:latest  swarm-node1  Running     Running about a minute ago
2m8dmrzxotwr   \_ nodejs.4  labdocker/alpineweb:latest  swarm-mng   Shutdown   Shutdown about a minute ago
dg4x1dac3jyb  nodejs.5  labdocker/alpineweb:latest  swarm-node2  Running     Running about a minute ago
vwbxkhptup7   \_ nodejs.5  labdocker/alpineweb:latest  swarm-mng   Shutdown   Shutdown about a minute ago
docker@swarm-mng:~$ [REDACTED]
```

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
```

```
docker@swarm-mng:~$ docker service inspect nodejs|grep update  
docker@swarm-mng:~$ docker node ls  
ID          HOSTNAME      STATUS      AVAILABILITY      MANAGER STATUS  
6bei4mbj5pd7yduyhp375b7z4 *  swarm-mng      Ready      Active      Leader  
u2eju1jxy5k8qd30ucrhuhjfs  swarm-node1    Ready      Active  
ywy7pfddcumccm2uojesnjy9v  swarm-node2    Ready      Active  
docker@swarm-mng:~$ docker service create -dt --constraint 'node.id==6bei4mbj5pd7yduyhp375b7z4' --na  
me nodejs --replicas=5 labdocker/alpineweb:latest node hello.js  
w87s7jaz5i8l944yq1t8v2eo6  
docker@swarm-mng:~$ docker service ps nodejs  
ID          NAME      IMAGE      NODE      DESIRED STATE      CURRENT STATE      ER  
OR          PORTS  
u8p7dnw2wta8  nodejs.1  labdocker/alpineweb:latest  swarm-mng  Running      Running 6 seconds ago  
w3bm3p92kmdq  nodejs.2  labdocker/alpineweb:latest  swarm-mng  Running      Running 6 seconds ago  
7pkkpmt4ew2l  nodejs.3  labdocker/alpineweb:latest  swarm-mng  Running      Running 6 seconds ago  
2m8dmrzxotwr  nodejs.4  labdocker/alpineweb:latest  swarm-mng  Running      Running 6 seconds ago  
vwbxkhptup7   nodejs.5  labdocker/alpineweb:latest  swarm-mng  Running      Running 6 seconds ago  
docker@swarm-mng:~$
```

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
```

```
[docker@swarm-mng:~$ docker service rm nodejs  
nodejs  
[docker@swarm-mng:~$ docker node update --label-add 'storage=sas' swarm-mng  
swarm-mng  
[docker@swarm-mng:~$ docker node update --label-add 'storage=nvdimm' swarm-node1  
swarm-node1  
[docker@swarm-mng:~$ docker node update --label-add 'storage=sata' swarm-node2  
swarm-node2  
[docker@swarm-mng:~$ docker node inspect swarm-mng|grep storage  
    "storage": "sas"  
[docker@swarm-mng:~$  
[docker@swarm-mng:~$ docker node inspect swarm-node1|grep storage  
    "storage": "nvdimm"  
[docker@swarm-mng:~$ docker node inspect swarm-node2|grep storage  
    "storage": "sata"  
[docker@swarm-mng:~$ ]
```

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
```

```
docker@swarm-mng:~$ docker service create -dt --constraint 'node.labels.storage==sas' --name nodejs --replicas=5 labdocker/alpineweb:la
|test node hello.js
qhfydixi6h5f0ral5jr4c7trc
docker@swarm-mng:~$ docker service ps nodejs
ID          NAME      IMAGE           NODE        DESIRED STATE   CURRENT STATE
OR          PORTS
w6bbu30v6m2x  nodejs.1  labdocker/alpineweb:latest  swarm-mng  Running
               nodejs.2  labdocker/alpineweb:latest  swarm-mng  Running
               nodejs.3  labdocker/alpineweb:latest  swarm-mng  Running
               nodejs.4  labdocker/alpineweb:latest  swarm-mng  Running
               nodejs.5  labdocker/alpineweb:latest  swarm-mng  Running
docker@swarm-mng:~$
```

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
```

```
docker@swarm-mng:~$ docker service create -dt --constraint 'node.labels.storage==sas' --name nodejs --replicas=5 labdocker/alpineweb:la  
[test node hello.js  
qhfydixi6h5f0ral5jr4c7trc  
docker@swarm-mng:~$ docker service ps nodejs  
ID OR  
w6bbu30v6m2x  
NAME  
PORTS  
nodejs.1  
IMAGE  
labdocker/alpineweb:latest  
NODE  
swarm-mng  
DESIRED STATE  
Running  
CURRENT STATE  
Running 3 seconds ago  
9xpnnj3zqxig  
nodejs.2  
labdocker/alpineweb:latest  
swarm-mng  
Running  
Running 2 seconds ago  
nwuzbcuj912r  
nodejs.3  
labdocker/alpineweb:latest  
swarm-mng  
Running  
Running 2 seconds ago  
m9dwd8bjbjve  
nodejs.4  
labdocker/alpineweb:latest  
swarm-mng  
Running  
Running 2 seconds ago  
rvkv00o4n5yq  
nodejs.5  
labdocker/alpineweb:latest  
swarm-mng  
Running  
Running 2 seconds ago  
docker@swarm-mng:~$
```

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
```

```
[docker@swarm-mng:~$ docker node update --label-add 'physical=DELLPE820' swarm-mng  
swarm-mng  
docker@swarm-mng:~$  
docker@swarm-mng:~$ docker node update --label-add 'physical=DELLPE820' swarm-node1  
swarm-node1  
docker@swarm-mng:~$  
[docker@swarm-mng:~$ docker node update --label-add 'physical=HP' swarm-node2  
swarm-node2  
docker@swarm-mng:~$ docker node inspect swarm-mng|grep physical  
    "physical": "DELLPE820",  
docker@swarm-mng:~$  
docker@swarm-mng:~$ docker node inspect swarm-node1|grep physical  
    "physical": "DELLPE820",  
docker@swarm-mng:~$  
[docker@swarm-mng:~$ docker node inspect swarm-node2|grep physical  
    "physical": "HP",  
docker@swarm-mng:~$ ]
```

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
```

```
docker@swarm-mng:~$ docker service create -dt --placement-pref 'spread=node.labels.physical' \
|> --name nodejs --replicas=10 labdocker/alpineweb:latest node hello.js
s74pj8exuh8skxtuxeeblvel
docker@swarm-mng:~$ docker service ps nodejs
ID          NAME      IMAGE           NODE        DESIRED STATE   CURRENT STATE      ERRO
R          PORTS
p84dv19t4kaq    nodejs.1  labdocker/alpineweb:latest  swarm-node2  Running        Running  1 second ago
pw15nk81ujcf    nodejs.2  labdocker/alpineweb:latest  swarm-mng   Running        Running  1 second ago
tjz75m4as403    nodejs.3  labdocker/alpineweb:latest  swarm-node2  Running        Running  1 second ago
wdrh5phg4xwp    nodejs.4  labdocker/alpineweb:latest  swarm-mng   Running        Running  1 second ago
mfri64z507tm    nodejs.5  labdocker/alpineweb:latest  swarm-node2  Running        Running  1 second ago
totevs2q6dus    nodejs.6  labdocker/alpineweb:latest  swarm-node2  Running        Running  1 second ago
valrbfn79ofi    nodejs.7  labdocker/alpineweb:latest  swarm-mng   Running        Running  1 second ago
m6rxlxh3amq2    nodejs.8  labdocker/alpineweb:latest  swarm-node1  Running        Running  1 second ago
nn1st3cou81j    nodejs.9  labdocker/alpineweb:latest  swarm-node2  Running        Running  1 second ago
mdxigzfaygc0    nodejs.10 labdocker/alpineweb:latest  swarm-node1  Running        Running  1 second ago
docker@swarm-mng:~$
```

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
```

```
docker@swarm-mng:~$ docker service create -dt --placement-pref 'spread=node.labels.physical' \
> --name nodejs --replicas=10 labdocker/alpineweb:latest node hello.js
s74pj8exuh8skxtuxeebllevel
docker@swarm-mng:~$ docker service ps nodejs
ID          NAME      IMAGE           NODE        DESIRED STATE   CURRENT STATE      ERRO
R          PORTS
p84dv19t4kaq  nodejs.1  labdocker/alpineweb:latest  swarm-node2  Running        Running  1 second ago
pw15nk81ujcf  nodejs.2  labdocker/alpineweb:latest  swarm-mng    Running        Running  1 second ago
tjz75m4as403  nodejs.3  labdocker/alpineweb:latest  swarm-node2  Running        Running  1 second ago
wdrh5phg4xwp  nodejs.4  labdocker/alpineweb:latest  swarm-mng    Running        Running  1 second ago
mfri64z507tm  nodejs.5  labdocker/alpineweb:latest  swarm-node2  Running        Running  1 second ago
totevs2q6dus  nodejs.6  labdocker/alpineweb:latest  swarm-node2  Running        Running  1 second ago
valrbfn79ofi  nodejs.7  labdocker/alpineweb:latest  swarm-mng    Running        Running  1 second ago
m6rxlxh3amq2  nodejs.8  labdocker/alpineweb:latest  swarm-node1  Running        Running  1 second ago
nn1st3cou81j  nodejs.9  labdocker/alpineweb:latest  swarm-node2  Running        Running  1 second ago
mdxigzfaygc0  nodejs.10 labdocker/alpineweb:latest  swarm-node1  Running        Running  1 second ago

docker@swarm-mng:~$
```

Config and Secret

- Make secret data and configuration great again !
- การทำงานของ microservice ที่รวมกันเป็น application stack จะมีส่วนของข้อมูลคอนฟิกต่างๆที่จำเป็นต้องใช้งานในระบบ เช่น
 - Root password of database
 - Environment variable
 - Custom variable
 - Path of mount volume data
 - Etc
- docker config จะใช้เพื่อจัดเก็บข้อมูลคอนฟิกต่างๆของ 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 เพื่อรองรับการให้บริการ 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
```

```
docker@swarm-mng:~$ docker config create nginx.conf /Share_DockerToolbox/nginx.conf
xw9bzcm6agbv67x01jkkch1oc
docker@swarm-mng:~$ docker config ls
ID                  NAME          CREATED        UPDATED
xw9bzcm6agbv67x01jkkch1oc  nginx.conf    4 seconds ago  4 seconds ago
docker@swarm-mng:~$ docker secret create labdocker.com.crt /Share_DockerToolbox/labdocker.com.crt
esxxlhdvd6jbtnjkey17xrmwa
docker@swarm-mng:~$ docker secret create labdocker.com.key /Share_DockerToolbox/labdocker.com.key
ony89eke7ifm203kta21jcn7z
docker@swarm-mng:~$ docker secret ls
ID                  NAME          CREATED        UPDATED
esxxlhdvd6jbtnjkey17xrmwa  labdocker.com.crt  6 seconds ago  6 seconds ago
ony89eke7ifm203kta21jcn7z  labdocker.com.key   5 seconds ago  5 seconds ago
```

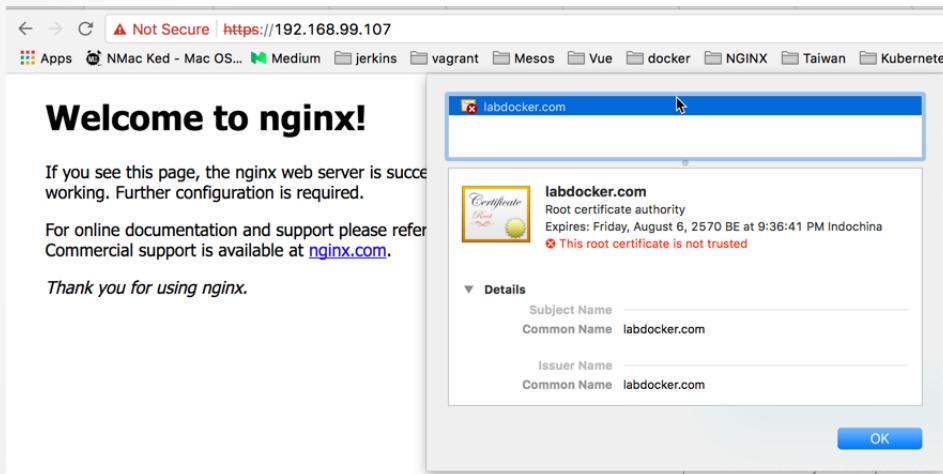
Workshop 2-4: Config and Secret

- 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

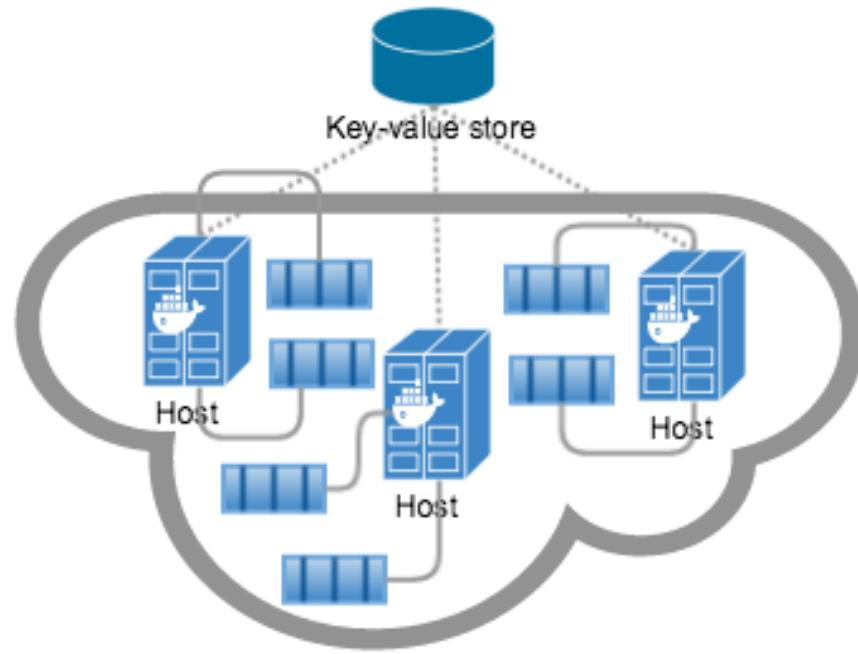
```
docker service -dt create \
--name nginx \
--secret labdocker.com.key \
--secret labdocker.com.crt \
--config source=nginx.conf,target=/etc/nginx/nginx.conf \
-p 443:443 labdocker/nginx:latest
```



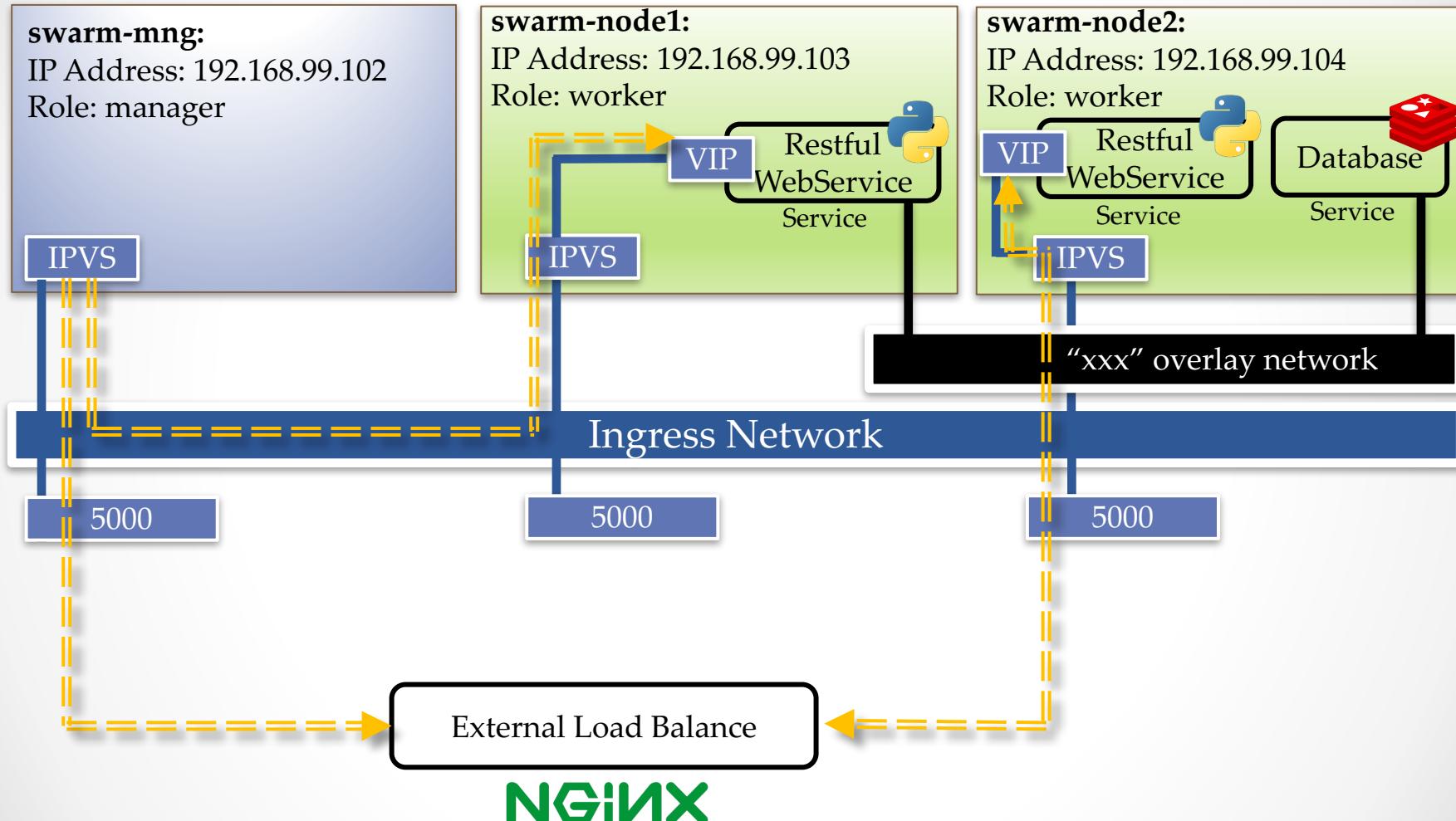
```
praparns-MacBook-Pro% curl https://192.168.99.107 -k
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
body {
    width: 35em;
    margin: 0 auto;
    font-family: Tahoma, Verdana, Arial, sans-serif;
}
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, 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%
```

Overlay and Ingress Network

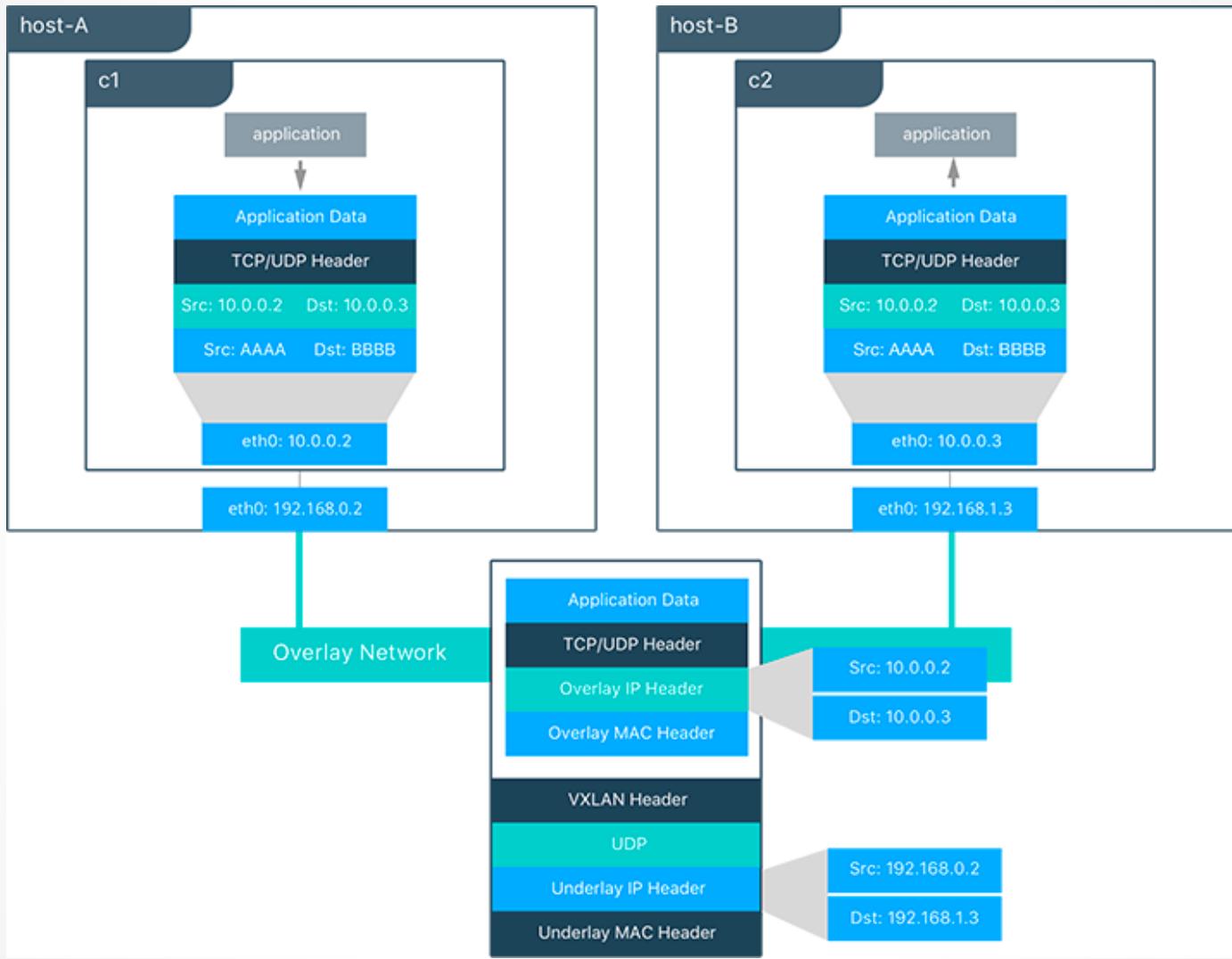
- เพื่อให้ทุกๆ node ภายในได้ swarm cluster สามารถมองเห็นกันได้ docker ได้เตรียมระบบ network แบบ overlay
- การทำงานของ overlay จะอาศัยกลไกของ key value store เป็นหลักในการตรวจสอบ (Discovery / Network status / IP Address etc)
- Default swarm จะสร้าง overlay network ชื่อ “ingress” เพื่อรับรองรับการใช้งาน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

```
docker@swarm-mng:~$ docker network create --driver overlay --subnet=192.168.100.0/24 swarmnet  
43mgygbg4wd15zw8ebxls8qs2  
[docker@swarm-mng:~$ docker network ls  
NETWORK ID      NAME      DRIVER      SCOPE  
1e00be7da218    bridge    bridge      local  
22a6b95d749b    docker_gwbridge    bridge      local  
951ca42f37e7    host      host      local  
gqga8p8t4tf3    ingress    overlay      swarm  
4c38e2fa3110    none      null      local  
43mgygbg4wd1    swarmnet    overlay      swarm  
docker@swarm-mng:~$ █
```

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
```

```
docker@swarm-mng:~$ docker service create -dt --name nodejs \  
> --replicas=2 --network swarmnet -p 3000:3000 \  
> labdocker/alpineweb:latest node hello.js  
m6t4ea8vlnzufy97cg3tr6grt  
docker@swarm-mng:~$ docker service ps nodejs  
ID          NAME      IMAGE      NODE      DESIRED STATE  CURRENT STATE      ERROR      PORTS  
ejng79ett6f  nodejs.1  labdocker/alpineweb:latest  swarm-mng  Running   Running  3 seconds ago  
7xmhszixxqhn nodejs.2  labdocker/alpineweb:latest  swarm-node2  Running   Running  5 seconds ago  
docker@swarm-mng:~$  
  
docker@swarm-mng:~$ docker ps  
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS           NAMES  
47f421456edb      labdocker/alpineweb:latest "node hello.js"  2 minutes ago     Up 2 minutes       3000/tcp        nodejs.1.ejng79ett6fzf3n0o5vp2  
519  
docker@swarm-mng:~$ docker inspect 47f421456edb|grep IPAddress  
  "SecondaryIPAddresses": null,  
  "IPAddress": "",  
  "IPAddress": "10.255.0.6",  
  "IPAddress": "192.168.100.3",  
docker@swarm-mng:~$  
  
docker@swarm-node2:~$ docker ps  
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS           NAMES  
bf10e17d890b      labdocker/alpineweb:latest "node hello.js"  3 minutes ago     Up 3 minutes       3000/tcp        nodejs.2.7xmhszixxqhn7d8xjsfjt4  
111  
docker@swarm-node2:~$ docker inspect bf10e17d890b|grep IPAddress  
  "SecondaryIPAddresses": null,  
  "IPAddress": "",  
  "IPAddress": "10.255.0.7",  
  "IPAddress": "192.168.100.4",  
docker@swarm-node2:~$
```

Workshop 2-4: Overlay and Ingress

- ตรวจสอบ IP Address ของ service และทดสอบ Ping ข้าม node
docker service inspect nodejs|more

```
[root@swarm-mng:~$ docker service inspect nodejs|more
[
  {
    "ID": "m6t4ea8vlnzufy97cg3tr6grt",
    "Version": {
      "Index": 24
    },
    "CreatedAt": "2017-08-13T11:49:49.472002608Z",
    "UpdatedAt": "2017-08-13T11:49:49.472588284Z",
    "Spec": {
      "Name": "nodejs",
      "Labels": {},
      "TaskTemplate": {
        "Ports": [
          {
            "Protocol": "tcp",
            "TargetPort": 3000,
            "PublishedPort": 3000,
            "PublishMode": "ingress"
          }
        ],
        "VirtualIPs": [
          {
            "NetworkID": "gqqa8p8t4tf3gexmb3dm04kx7",
            "Addr": "10.255.0.5/16"
          },
          {
            "NetworkID": "43mgygbg4wd15zw8ebxls8qs2",
            "Addr": "192.168.100.2/24"
          }
        ]
      }
    }
  }
]
```

Workshop 2-4: Overlay and Ingress

- ตรวจสอบ IP Address ของ service และทดสอบ Ping ข้าม node

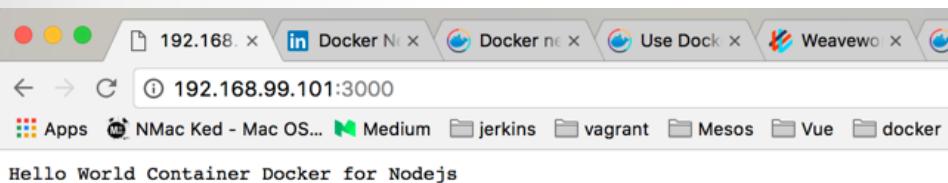
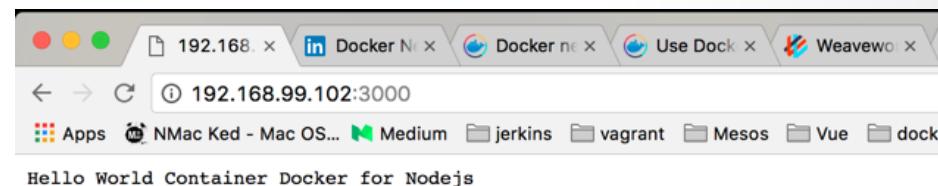
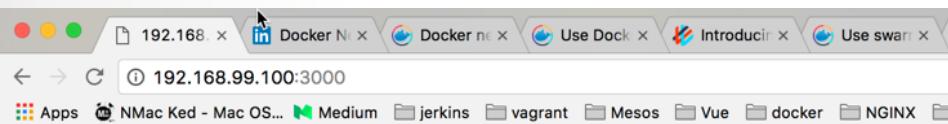
```
docker@swarm-mng:~$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS
47f421456edb      labdockerc/alpineweb:latest   "node hello.js"   11 minutes ago    Up 11 minutes     3000/tcp
519
docker@swarm-mng:~$ docker exec -it nodejs.1.ejng79ettt6fzf3n0o5vp2519 ping 192.168.100.4
PING 192.168.100.4 (192.168.100.4): 56 data bytes
64 bytes from 192.168.100.4: seq=0 ttl=64 time=0.768 ms
^C
--- 192.168.100.4 ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 0.768/0.768/0.768 ms
docker@swarm-mng:~$ docker exec -it nodejs.1.ejng79ettt6fzf3n0o5vp2519 ping 192.168.100.2
PING 192.168.100.2 (192.168.100.2): 56 data bytes
64 bytes from 192.168.100.2: seq=0 ttl=64 time=0.061 ms
^C
--- 192.168.100.2 ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 0.061/0.061/0.061 ms
docker@swarm-mng:~$
```

```
docker@swarm-node2:~$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS
bf10e17d890b      labdockerc/alpineweb:latest   "node hello.js"   12 minutes ago    Up 12 minutes     3000/tcp
4111
docker@swarm-node2:~$ docker exec -it nodejs.2.7xmhszixqhn7d8xjsfjt4111 ping 192.168.100.3
PING 192.168.100.3 (192.168.100.3): 56 data bytes
64 bytes from 192.168.100.3: seq=0 ttl=64 time=0.525 ms
64 bytes from 192.168.100.3: seq=1 ttl=64 time=0.576 ms
^C
--- 192.168.100.3 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 0.525/0.550/0.576 ms
docker@swarm-node2:~$ docker exec -it nodejs.2.7xmhszixqhn7d8xjsfjt4111 ping 192.168.100.2
PING 192.168.100.2 (192.168.100.2): 56 data bytes
64 bytes from 192.168.100.2: seq=0 ttl=64 time=0.055 ms
64 bytes from 192.168.100.2: seq=1 ttl=64 time=0.090 ms
^C
--- 192.168.100.2 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 0.055/0.072/0.090 ms
docker@swarm-node2:~$
```

Workshop 2-4: Overlay and Ingress

- เรียกใช้บริการผ่าน http port ด้วย browser/curl

```
praparns-MacBook-Pro:~ praparn$ docker-machine ls
NAME      ACTIVE   DRIVER      STATE     URL
labdocker - virtualbox Stopped
swarm-mng  - virtualbox Running  tcp://192.168.99.100:2376
swarm-node1 - virtualbox Running  tcp://192.168.99.101:2376
swarm-node2 - virtualbox Running  tcp://192.168.99.102:2376
praparns-MacBook-Pro:~ praparn$ curl http://192.168.99.100:3000
Hello World Container Docker for Nodejs
praparns-MacBook-Pro:~ praparn$ curl http://192.168.99.101:3000
Hello World Container Docker for Nodejs
praparns-MacBook-Pro:~ praparn$ curl http://192.168.99.102:3000
Hello World Container Docker for Nodejs
praparns-MacBook-Pro:~ praparn$
```



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>
```

```
docker version 17.06.2-ce, build 02c2dce  
docker@labdocke:~$ docker swarm join \  
> --token SWMTKN-1-398w160n14ed4od3yiq168yvzb2g7z32nmxvxq47ubqh6sz6-bv4zdksviq68gqh4emh098xn1 \  
> 192.168.99.100:2377  
This node joined a swarm as a manager.  
[docker@labdocke:~$ docker node ls  
ID          HOSTNAME  STATUS  AVAILABILITY  MANAGER STATUS  
izywb24ghey4v6fpp7fv60l9h *  labdocke  Ready  Active  Reachable  
u45isavg9pv02xq2894nqlojs  swarm-mng  Ready  Active  Leader  
ujb06wblj9sw9e463fk4bnju  swarm-node1  Ready  Active    
wdqwo14h1lam1tp6cqauzwxumj  swarm-node2  Ready  Active  
docker@labdocke:~$
```

- ตรวจสอบค่า token เพื่อ join swarm

```
[docker@labdocke:~$ docker swarm join-token -q manager  
SWMTKN-1-398w160n14ed4od3yiq168yvzb2g7z32nmxvxq47ubqh6sz6-bv4zdksviq68gqh4emh098xn1  
[docker@labdocke:~$ docker swarm join-token -q worker  
SWMTKN-1-398w160n14ed4od3yiq168yvzb2g7z32nmxvxq47ubqh6sz6-db21ae2ieqyl5gi25nc6izm6x  
[docker@labdocke:~$
```

HA Manager Role

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

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

```
[root@labdocke:~$ docker node update --role manager swarm-node1  
swarm-node1
```

```
[root@labdocke:~$ docker node ls
```

ID	HOSTNAME	STATUS	AVAILABILITY	MANAGER STATUS
izywb24ghey4v6fpp7fv60l9h *	labdocke	Ready	Active	Reachable
u45isavg9pv02xq2894nqlojs	swarm-mng	Ready	Active	Leader
ujb06wblj9sw9e463fky4bnju	swarm-node1	Ready	Active	Reachable
wdqwo14hlam1tp6cqauzwxumj	swarm-node2	Ready	Active	

```
docker@labdocke:~$
```

HA Manager Role

- Add redundancy swarm manager in swarm cluster
 - Shutting down major manager

```
docker@swarm-mng:~$ docker node ls
ID           HOSTNAME   STATUS    AVAILABILITY  MANAGER STATUS
izywb24ghey4v6fpp7fv60l9h *  labdocker  Ready     Active       Reachable
u45isavg9pv02xq2894nqlojs  swarm-mng  Ready     Active       Leader
ujb06wblj9sw9e463fky4bnju  swarm-node1 Ready     Active       Reachable
wdqwo14h1lam1tp6cqauvwxyzumj  swarm-node2 Ready     Active

[docker@swarm-mng:~$ sudo shutdown -h now
[docker@swarm-mng:~$ Connection to 127.0.0.1 closed by remote host.
exit status 255
praparns-MacBook-Pro:~ praparn$ ]
```

- Other will take role for manager within 5 min.

```
docker@labdocker:~$ docker node ls
ID           HOSTNAME   STATUS    AVAILABILITY  MANAGER STATUS
izywb24ghey4v6fpp7fv60l9h *  labdocker  Ready     Active       Reachable
u45isavg9pv02xq2894nqlojs  swarm-mng  Ready     Active       Leader
ujb06wblj9sw9e463fky4bnju  swarm-node1 Ready     Active       Reachable
wdqwo14h1lam1tp6cqauvwxyzumj  swarm-node2 Ready     Active

[docker@labdocker:~$ docker node ls
ID           HOSTNAME   STATUS    AVAILABILITY  MANAGER STATUS
izywb24ghey4v6fpp7fv60l9h *  labdocker  Ready     Active       Reachable
u45isavg9pv02xq2894nqlojs  swarm-mng  Down      Active       Unreachable
ujb06wblj9sw9e463fky4bnju  swarm-node1 Ready     Active       Leader
wdqwo14h1lam1tp6cqauvwxyzumj  swarm-node2 Ready     Active

[docker@labdocker:~$ ]
```

HA Manager Role

- Add redundancy swarm manager in swarm cluster
 - When old swarm-mng is back to online. It will back to “Reachable state”

```
praparns-MacBook-Pro:~ praparn$ docker-machine start swarm-mng
Starting "swarm-mng"...
(swarm-mng) Check network to re-create if needed...
(swarm-mng) Waiting for an IP...
Machine "swarm-mng" was started.
Waiting for SSH to be available...
Detecting the provisioner...
Started machines may have new IP addresses. You may need to re-run the `docker-machine env` command.
praparns-MacBook-Pro:~ praparn$ docker-machine ssh swarm-mng
##          .
## ## ##   ==
## ## ## ## ## ===
/-----\---/ ===
~~~ {~~ ~~~ ~~~ ~~~ ~~~ / === ~~~
\---- o   /--/
 \----\---/
 [---\---\---[---\---\---]---[---\---\---]
 [---]---\---\---[---]---\---[---]---\---[---]
 [---]---\---\---[---]---\---[---]---\---[---]
 [---]---\---\---[---]---\---[---]---\---[---]
Boot2Docker version 17.06.0-ce, build HEAD : 0672754 - Thu Jun 29 00:06:31 UTC 2017
Docker version 17.06.0-ce, build 02c1d87
docker@swarm-mng:~$ docker node ls
ID           HOSTNAME        STATUS        AVAILABILITY      MANAGER STATUS
izywbghey4v6fpp7fv6019h  labdockeR      Ready       Active       Reachable
u45isavg9pv02xq2894nql0js *  swarm-mng    Ready       Active       Reachable
ujb06wb1j9sw9e463fkY4bnju  swarm-node1   Ready       Active       Leader
wdqwo14h1am1tp6cqauzwxumj  swarm-node2    Ready       Active
docker@swarm-mng:~$
```

HA Manager Role

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

```
docker@swarm-mng:~$ docker node ls
ID          HOSTNAME  STATUS  AVAILABILITY  MANAGER STATUS
izywb24ghey4v6fpp7fv6019h  labdocker  Ready   Active        Reachable
u45isavg9pv02xq2894nqlojs *  swarm-mng  Ready   Active        Reachable
ujb06wblj9sw9e463fky4bnju  swarm-node1  Ready   Active        Leader
wdqwo14h1am1tp6cqauvwxyzumj  swarm-node2  Ready   Active

docker@swarm-mng:~$ docker node update --role worker labdocker
labdocker
docker@swarm-mng:~$ docker node update --availability drain labdocker
labdocker
docker@swarm-mng:~$ docker node ls
ID          HOSTNAME  STATUS  AVAILABILITY  MANAGER STATUS
izywb24ghey4v6fpp7fv6019h  labdocker  Ready   Drain         Reachable
u45isavg9pv02xq2894nqlojs *  swarm-mng  Ready   Active        Leader
ujb06wblj9sw9e463fky4bnju  swarm-node1  Ready   Active
wdqwo14h1am1tp6cqauvwxyzumj  swarm-node2  Ready   Active

docker@swarm-mng:~$
```

- Leave swarm on node “labdocker”

```
docker@labdocker:~$ docker swarm leave
Node left the swarm.
docker@labdocker:~$
```

```
docker@swarm-mng:~$ docker node ls
ID          HOSTNAME  STATUS  AVAILABILITY  MANAGER STATUS
izywb24ghey4v6fpp7fv6019h  labdocker  Down    Drain        Reachable
u45isavg9pv02xq2894nqlojs *  swarm-mng  Ready   Active        Leader
ujb06wblj9sw9e463fky4bnju  swarm-node1  Ready   Active
wdqwo14h1am1tp6cqauvwxyzumj  swarm-node2  Ready   Active

docker@swarm-mng:~$ docker node rm labdocker
labdocker
docker@swarm-mng:~$ docker node ls
ID          HOSTNAME  STATUS  AVAILABILITY  MANAGER STATUS
u45isavg9pv02xq2894nqlojs *  swarm-mng  Ready   Active        Reachable
ujb06wblj9sw9e463fky4bnju  swarm-node1  Ready   Active        Leader
wdqwo14h1am1tp6cqauvwxyzumj  swarm-node2  Ready   Active

docker@swarm-mng:~$
```

Workshop 2-4: HA Manager Role

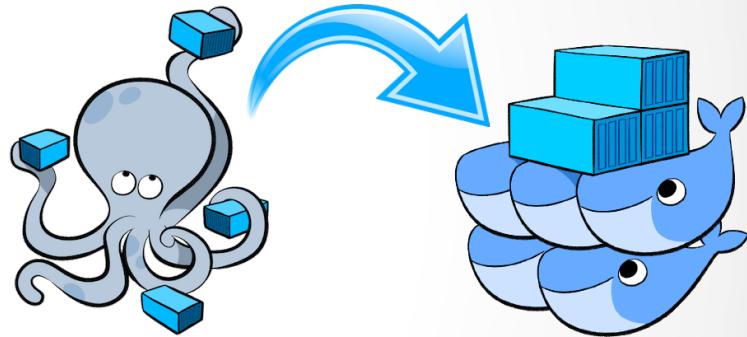
- ทำการ join new manager เข้าสู่ swarm cluster/update/drain

```
[docker@labdocker:~$ docker node ls
ID          HOSTNAME  STATUS  AVAILABILITY  MANAGER STATUS
izywb24ghey4v6fpp7fv6019h *  labdocker  Ready   Active        Reachable
u45isavg9pv02xq2894nqlojs *  swarm-mng  Ready   Active        Leader
ujb06wblj9sw9e463fky4bnju  swarm-node1  Ready   Active
wdqwo14h1am1tp6cqauvwxyzumj  swarm-node2  Ready   Active
docker@labdocker:~$ ]
```

```
[docker@swarm-mng:~$ docker node ls
ID          HOSTNAME  STATUS  AVAILABILITY  MANAGER STATUS
izywb24ghey4v6fpp7fv6019h  labdocker  Ready   Active        Reachable
u45isavg9pv02xq2894nqlojs *  swarm-mng  Ready   Active
ujb06wblj9sw9e463fky4bnju  swarm-node1  Ready   Active
wdqwo14h1am1tp6cqauvwxyzumj  swarm-node2  Ready   Active
docker@swarm-mng:~$ docker node update --role worker labdocker
labdocker
docker@swarm-mng:~$ docker node update --availability drain labdocker
labdocker
docker@swarm-mng:~$ docker node ls
ID          HOSTNAME  STATUS  AVAILABILITY  MANAGER STATUS
izywb24ghey4v6fpp7fv6019h  labdocker  Ready   Drain
u45isavg9pv02xq2894nqlojs *  swarm-mng  Ready   Active        Reachable
ujb06wblj9sw9e463fky4bnju  swarm-node1  Ready   Active        Leader
wdqwo14h1am1tp6cqauvwxyzumj  swarm-node2  Ready   Active
docker@swarm-mng:~$ ]
```

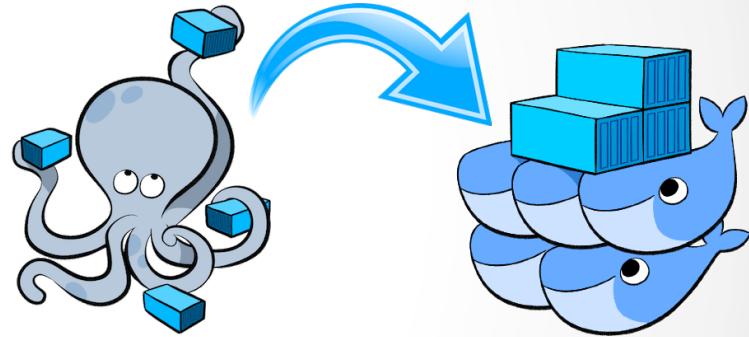
Docker Stack Deploy (Compose)

- 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
```

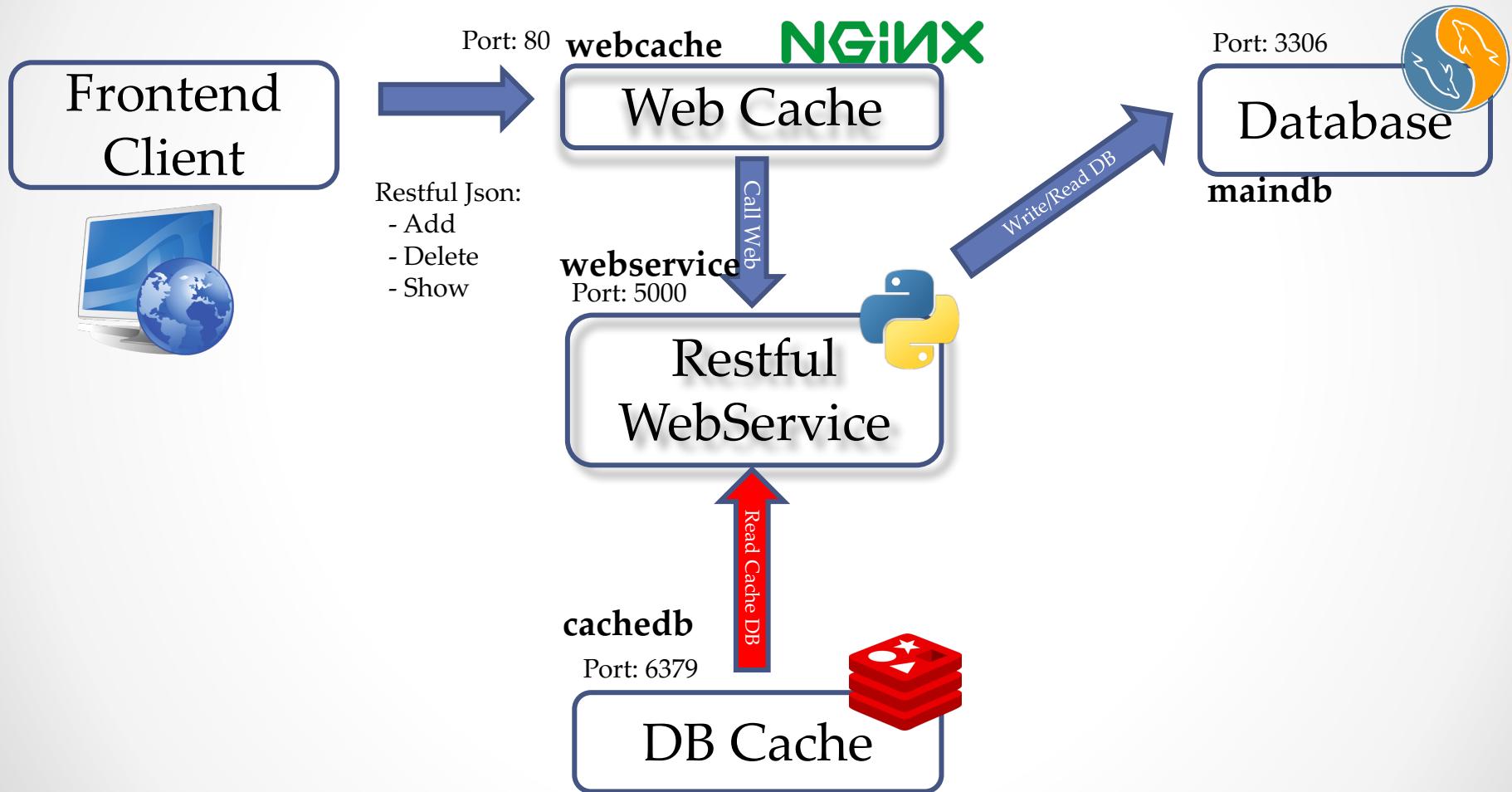
```
docker stack ps <stack name>
```

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

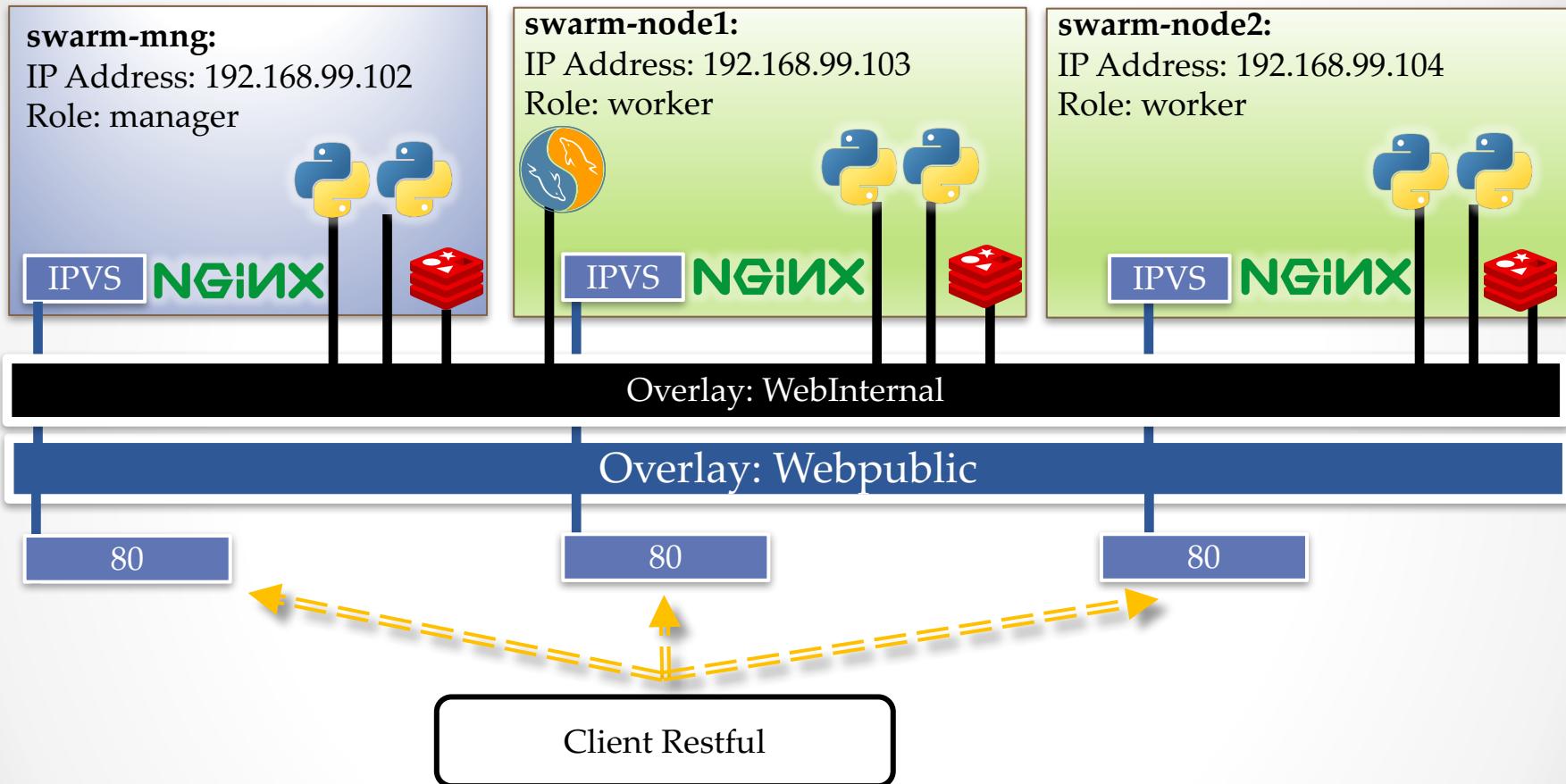
```
docker stack rm <stack name>
```

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

```
56 maindb:
57   image: labdocker/mysql:latest
58   container_name: maindb
59   deploy:
60     mode: replicated
61     replicas: 1
62     endpoint_mode: vip
63   environment:
64     - MYSQL_ROOT_PASSWORD=password
65   networks:
66     webinternal:
67       aliases:
68         - maindb
69
70 cachedb:
71   image: labdocker/redis:latest
72   container_name: cachedb
73   deploy:
74     mode: global
75     endpoint_mode: vip
76   networks:
77     webinternal:
78       aliases:
79         - cachedb
```

```
81 networks:
82   webpublic:
83     driver: overlay
84     ipam:
85       driver: default
86       config:
87         - subnet: 192.168.100.0/24
88   webinternal:
89     driver: overlay
90     ipam:
91       driver: default
92       config:
93         - subnet: 192.168.101.0/24
```

Workshop 2-4: Swarm Compose

```
docker@swarm-mng:/Share_DockerToolbox/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_webservice
Creating service webservice_maindb
Creating service webservice_cachedb
Creating service webservice_webcache
```

NAME	IMAGE	NODE	DESIRED STATE	CURRENT STATE
webservice_webcache.wwzx60rvl5res39v0lxir2gfh	labdocker/cluster:webcache	swarm-node2	Running	Running 46 seconds ago
webservice_webcache.0k95omm28xfef22thummpz3uf	labdocker/cluster:webcache	swarm-mng	Running	Running 49 seconds ago
webservice_webcache.q814dhi3bz14sze4zmjpiy3qa	labdocker/cluster:webcache	swarm-node1	Running	Running 50 seconds ago
webservice_webcache.wwzx60rvl5res39v0lxir2gfh	labdocker/cluster:webcache	swarm-node2	Shutdown	Failed about a minute ago
webservice_webcache.0k95omm28xfef22thummpz3uf	labdocker/cluster:webcache	swarm-mng	Shutdown	Failed about a minute ago
webservice_webcache.q814dhi3bz14sze4zmjpiy3qa	labdocker/cluster:webcache	swarm-node1	Shutdown	Failed about a minute ago
webservice_webcache.wwzx60rvl5res39v0lxir2gfh	labdocker/cluster:webcache	swarm-node2	Shutdown	Failed about a minute ago
webservice_webcache.q814dhi3bz14sze4zmjpiy3qa	labdocker/cluster:webcache	swarm-node1	Shutdown	Failed about a minute ago
webservice_webcache.0k95omm28xfef22thummpz3uf	labdocker/cluster:webcache	swarm-mng	Shutdown	Failed about a minute ago
webservice_cachedb.wwzx60rvl5res39v0lxir2gfh	labdocker/redis:latest	swarm-node2	Running	Running about a minute ago
webservice_cachedb.q814dhi3bz14sze4zmjpiy3qa	labdocker/redis:latest	swarm-node1	Running	Running about a minute ago
webservice_cachedb.0k95omm28xfef22thummpz3uf	labdocker/redis:latest	swarm-mng	Running	Running about a minute ago
webservice_webservice.1	labdocker/cluster:webservice	swarm-node1	Running	Running 39 seconds ago
_ webservice_webservice.1	labdocker/cluster:webservice	swarm-mng	Shutdown	Failed about a minute ago
webservice_maindb.1	labdocker/mysql:latest	swarm-node2	Running	Running about a minute ago

Workshop 2-4: Swarm Compose

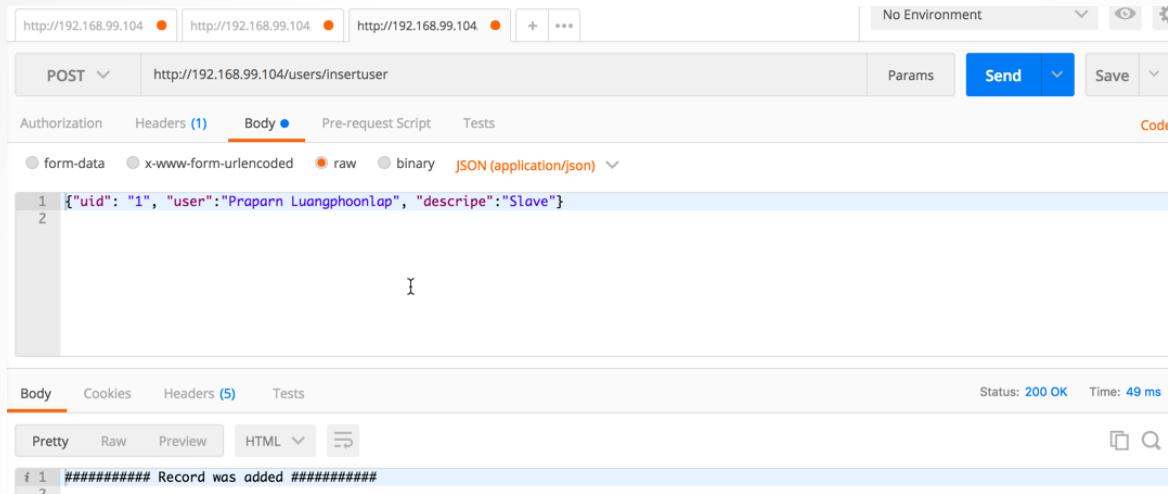
ID	NAME	IMAGE	NODE	DESIRED STATE	CURRENT STATE
	PORTS				
mcj0ux4x25uv o ol223n2c2htp o fj0qu565u8mt o kw8oprgn8b3v ago "task: non-zero exit (1)" a4foun5hzks3 ago "task: non-zero exit (1)" horm2noczrvt ago "task: non-zero exit (1)" vhutisinb7np ago "task: non-zero exit (1)" ktjzx8thiq6k ago "task: non-zero 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	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-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 buttons for 'No Environment', 'Send', 'Save', and 'Code'. Below the header, the URL 'http://192.168.99.104' is entered in the address bar. The main interface has tabs for 'Authorization', 'Headers', 'Body', 'Pre-request Script', and 'Tests'. Under 'Body', there are tabs for 'Pretty', 'Raw', 'Preview', and 'HTML'. The 'Pretty' tab is selected, displaying the response body: '<H1> Welcome Page from Container Python Lab </H1>Checkpoint Date/Time: Sun Aug 13 17:29:12 2017'. Above the body, the status bar shows 'Status: 200 OK' and 'Time: 24 ms'.

Workshop 2-4: Swarm Compose



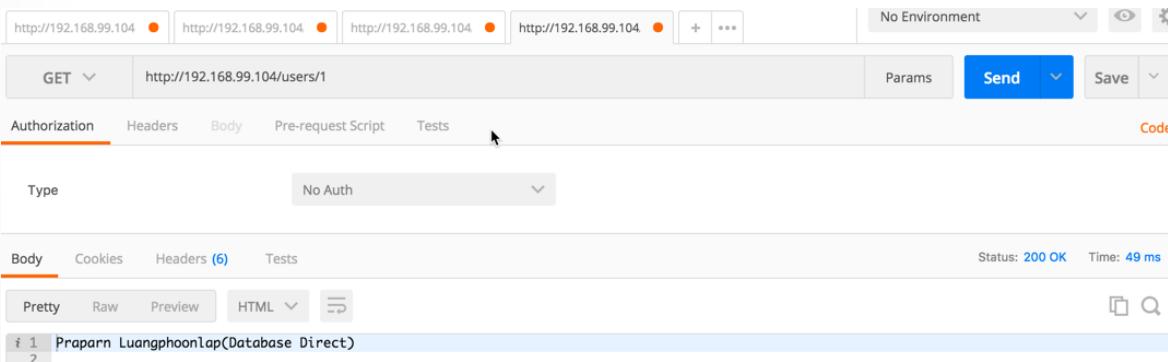
POST http://192.168.99.104/users/insertuser

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

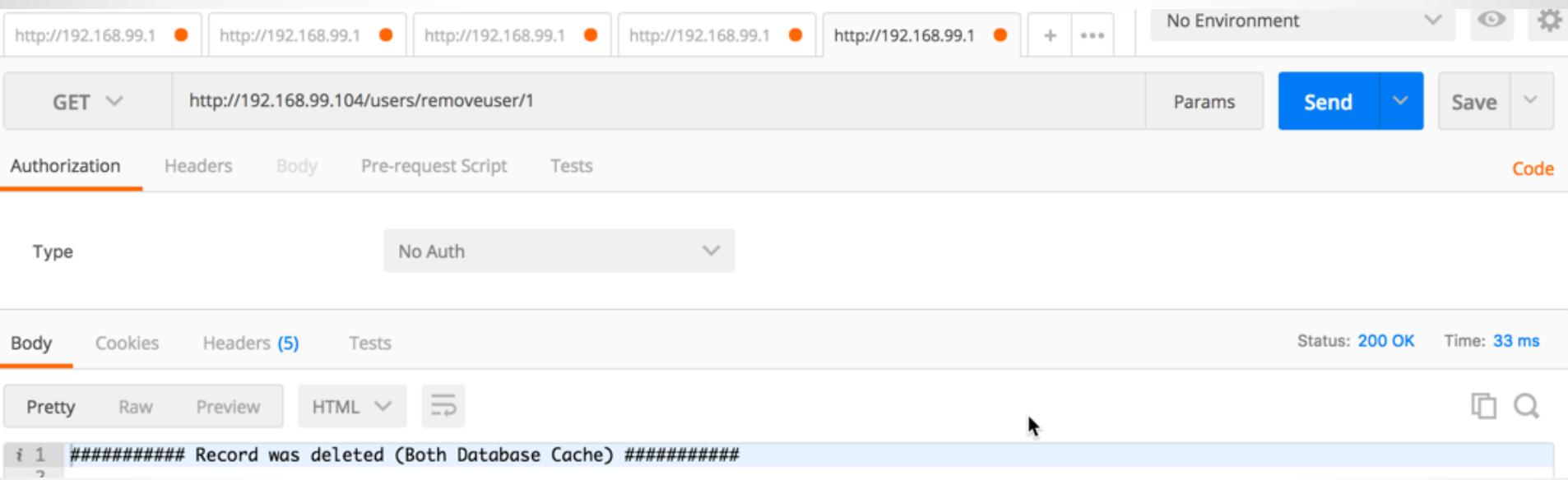
Type: No Auth

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:

- Request 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) #####

Shipyard for Docker

• • •

Shipyard for Docker

- Shipyard เป็น open source management ที่สามารถใช้บริหารจัดการ docker-machine ผ่านหน้า GUI web ได้อย่างมีประสิทธิภาพแบบ On premise
- การใช้งาน shipyard จะใช้การ curl install ผ่านหน้าเว็บ
- curl -sSL https://shipyard-project.com/deploy | sh -s

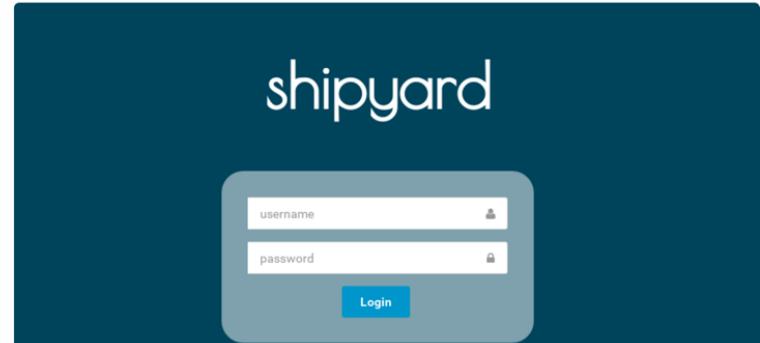


Shipyard

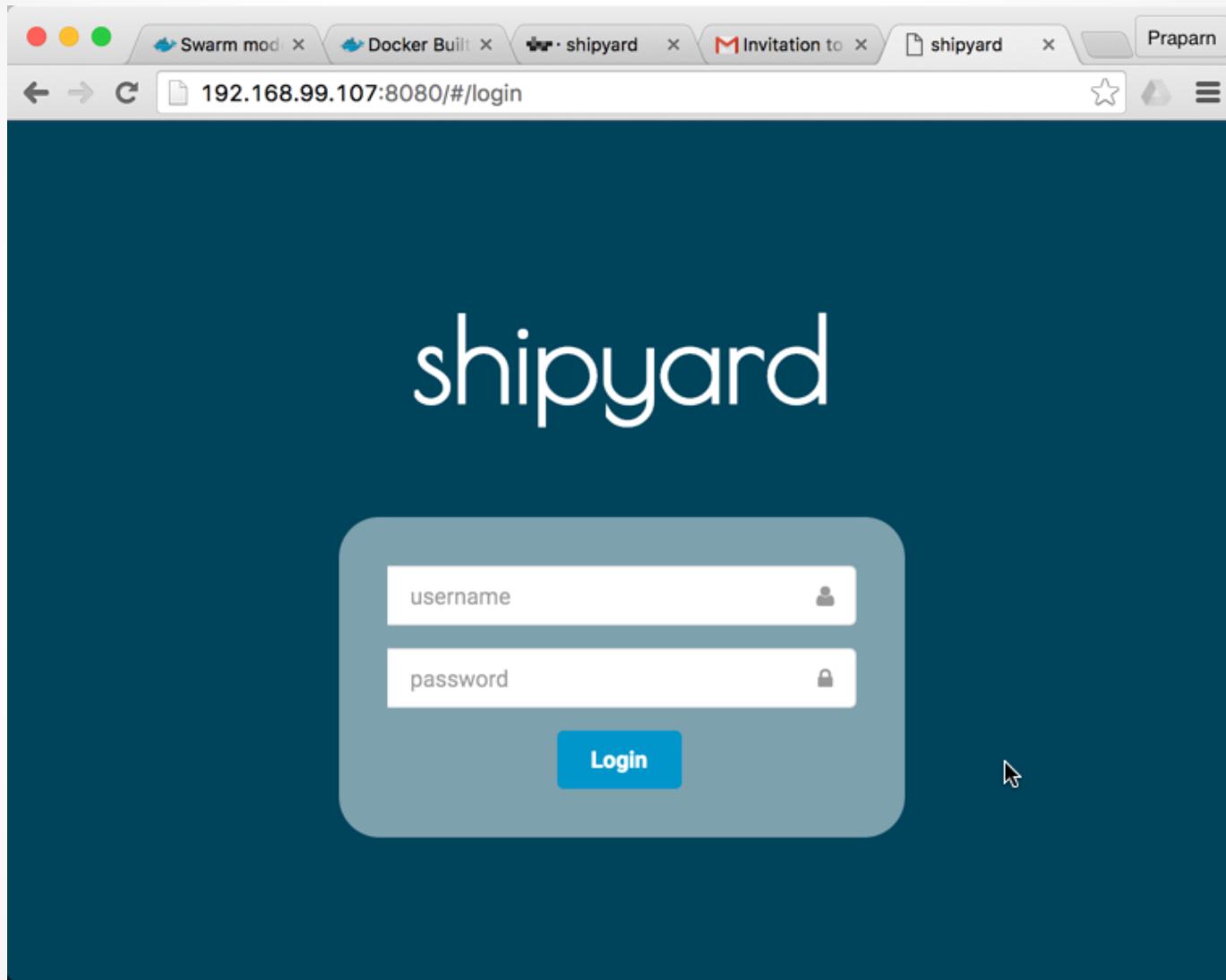
Composable Docker Management

Built on Docker Swarm, Shipyard gives you the ability to manage Docker resources including containers, images, private registries and more.

Shipyard differs from other management applications in that it promotes composability and is 100% compatible with the Docker Remote API. Shipyard manages containers, images, nodes, private registries cluster-wide as well as providing authentication and role based access control.



Shipyard for Docker



Shipyard for Docker

Screenshot of the Shipyard interface for Docker, showing a list of running containers.

The interface includes a top navigation bar with tabs for Swarm mode overview, Docker Built-in Orchestrator, shipyard, Invitation to participate, shipyard, shipyard, and Praparn. Below the navigation bar is a browser address bar showing 192.168.99.107:8080/#/containers.

The main content area features a header with Refresh, Deploy Container, and Search containers... buttons. Below this is a table listing the following container details:

	Heartbeat	Id	Node	Name	Image	Status	Created	Actions
<input type="checkbox"/>	●	34c785052a0f	labdocker	shipyard-controller	shipyard/shipyard:latest	Up 5 minutes	2016-07-29 23:30:51 +0700	🔍 🔧
<input type="checkbox"/>	●	3920186cdbbe	labdocker	shipyard-swarm-agent	swarm:latest	Up 6 minutes	2016-07-29 23:30:28 +0700	🔍 🔧
<input type="checkbox"/>	●	20a671f23af1	labdocker	shipyard-swarm-manager	swarm:latest	Up 6 minutes	2016-07-29 23:30:28 +0700	🔍 🔧
<input type="checkbox"/>	●	241a16f5554c	labdocker	shipyard-proxy	shipyard/docker-proxy:latest	Up 6 minutes	2016-07-29 23:30:15 +0700	🔍 🔧
<input type="checkbox"/>	●	08e650f3db6a	labdocker	shipyard-certs	alpine	Up 6 minutes	2016-07-29 23:30:01 +0700	🔍 🔧
<input type="checkbox"/>	●	35aa2d1e0d51	labdocker	shipyard-discovery	microbox/etcdb:latest	Up 6 minutes	2016-07-29 23:30:01 +0700	🔍 🔧
<input type="checkbox"/>	●	af209c282460	labdocker	shipyard-rethinkdb	rethinkdb	Up 7 minutes	2016-07-29 23:29:48 +0700	🔍 🔧

Shipyard for Docker

The screenshot shows the Shipyard web interface for managing Docker containers. The top navigation bar includes tabs for CONTAINERS, IMAGES, NODES, REGISTRIES, ACCOUNTS, EVENTS, ADMIN, and Help. The main area is titled "Container Deployment" and contains the following fields:

- Image Configuration**:
 - Image Name**: labdocker/nginx:latest
 - Command**: Command
 - Hostname**: nginx
 - Domain**: labdocker.com
- Environment Variables**: A table with columns "Name" and "Value". One row is shown: Environment Variable N and Environment Variable Value.
- Volumes**: A table with columns "Host Path" and "Container Path". One row is shown: Host Path and Container Path.
- Container Links**: A table with columns "Container" and "Alias". One row is shown: Container Name and Link Alias.
- Container Name**: nginx
- Resource Limits**:
 - CPUs**: 1024
 - Memory (MB)**: ∞
- Swarm Constraint**:

Constraint	Rule	Value
storage	$=$	ssd
- Restart Policy**: Do not automatically restart
- Network Mode**: Bridae

Docker: The Next-Gen of Virtualization



Shipyard for Docker

Screenshot of the Shipyard interface for Docker, showing the details of a running container named "nginx".

The container was started today at 11:40 pm.

Container Configuration:

Container ID	Command	Name	Host
15dd63af25c9		labdocker	10.0.2.15:2375

Swarm Node:

Hostname	Domain Name	CPUs	Memory
nginx	labdocker.com	1	996 MB

Environment:

```
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin  
NODE_VERSION=v4.3.0  
NPM_VERSION=2.14.12
```

Port Configuration:

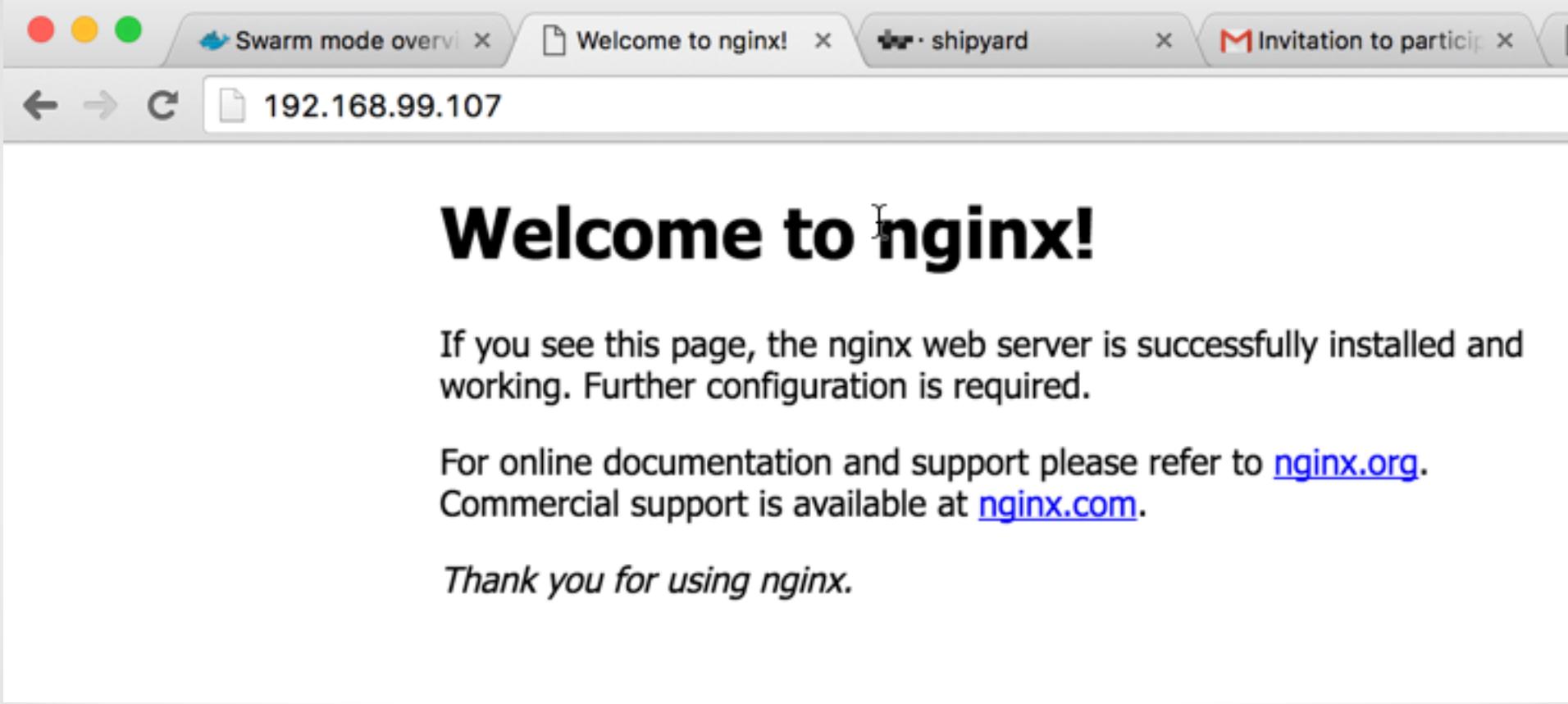
Exposed	Published
10.0.2.15:80	→ 80/tcp

User Defined Container DNS:

Processes:

UID	PID	PPID	C	STIME	TTY	TIME	CMD
root	3139	3128	0	16:40	pts/8	00:00:00	nginx: master process nginx -c /etc/nginx/nginx.conf
dockrem+	3154	3139	0	16:40	pts/8	00:00:00	nginx: worker process

Shipyard for Docker



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.

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)
- Shipyard for Docker
- Q&A



Q&A

By Praparn Luengphoonlap
Email: eva10409@gmail.com

Docker: The Next-Gen of Virtualization

