Docker:Zero to Hero (Part 1)

By Praparn Luengphoonlap
Email: eva10409@gmail.com

docker

Outline Part 1

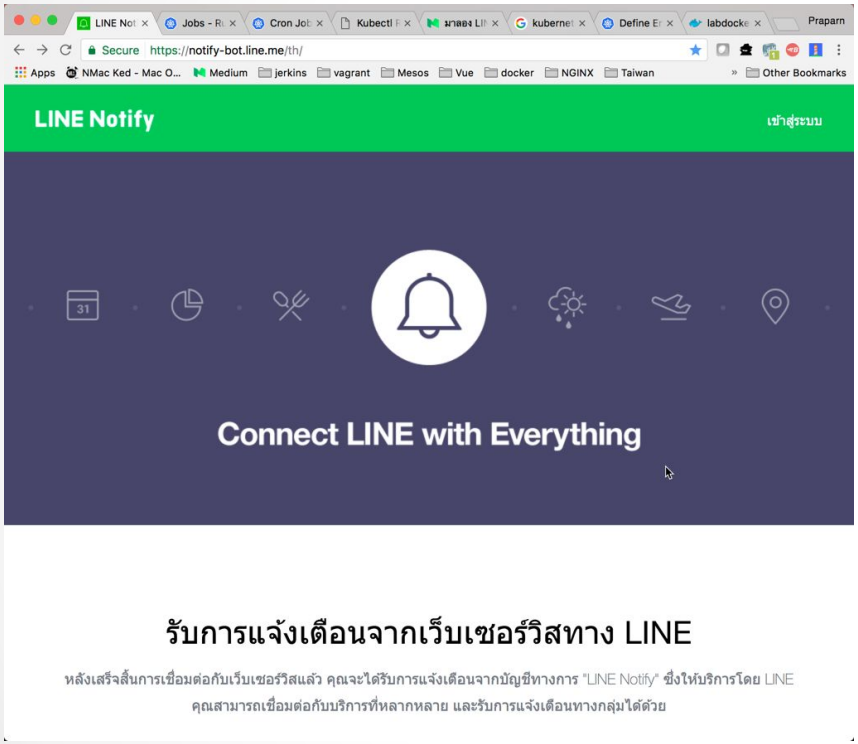
- Docker principle
- Docker machine
- Image, Repository & Tag, container
- CPU, Memory and I/O
- Network
- Volume
- Inspect and Log
- Commit
- Docker Desktop (ShowCase)

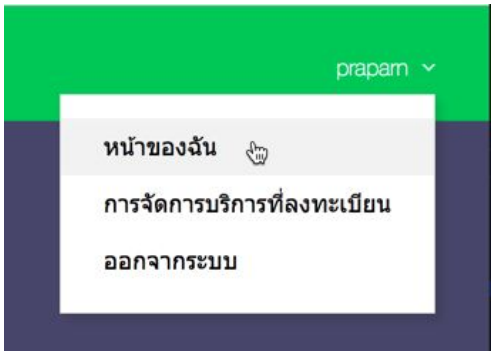
Prerequisite

- 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

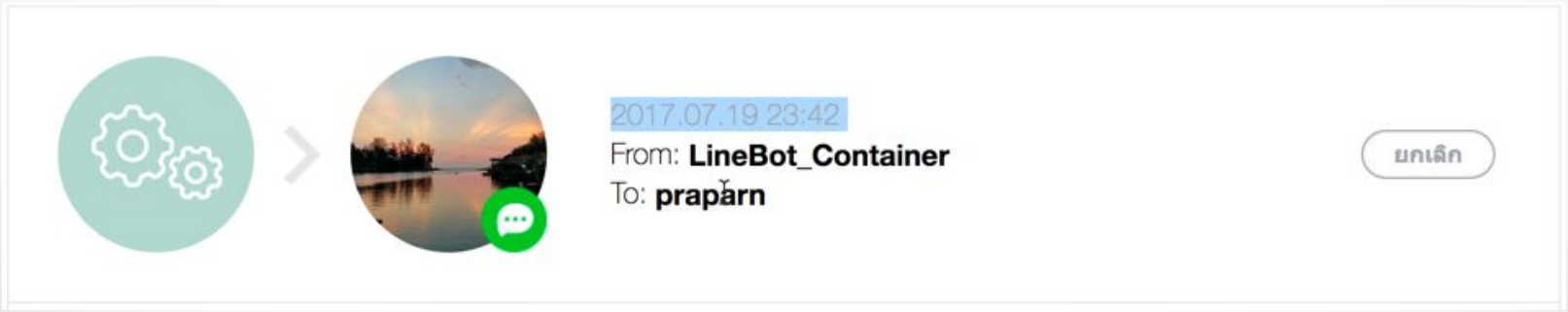
- Generate LINE Token
  - https://notify-bot.line.me





Prerequisite

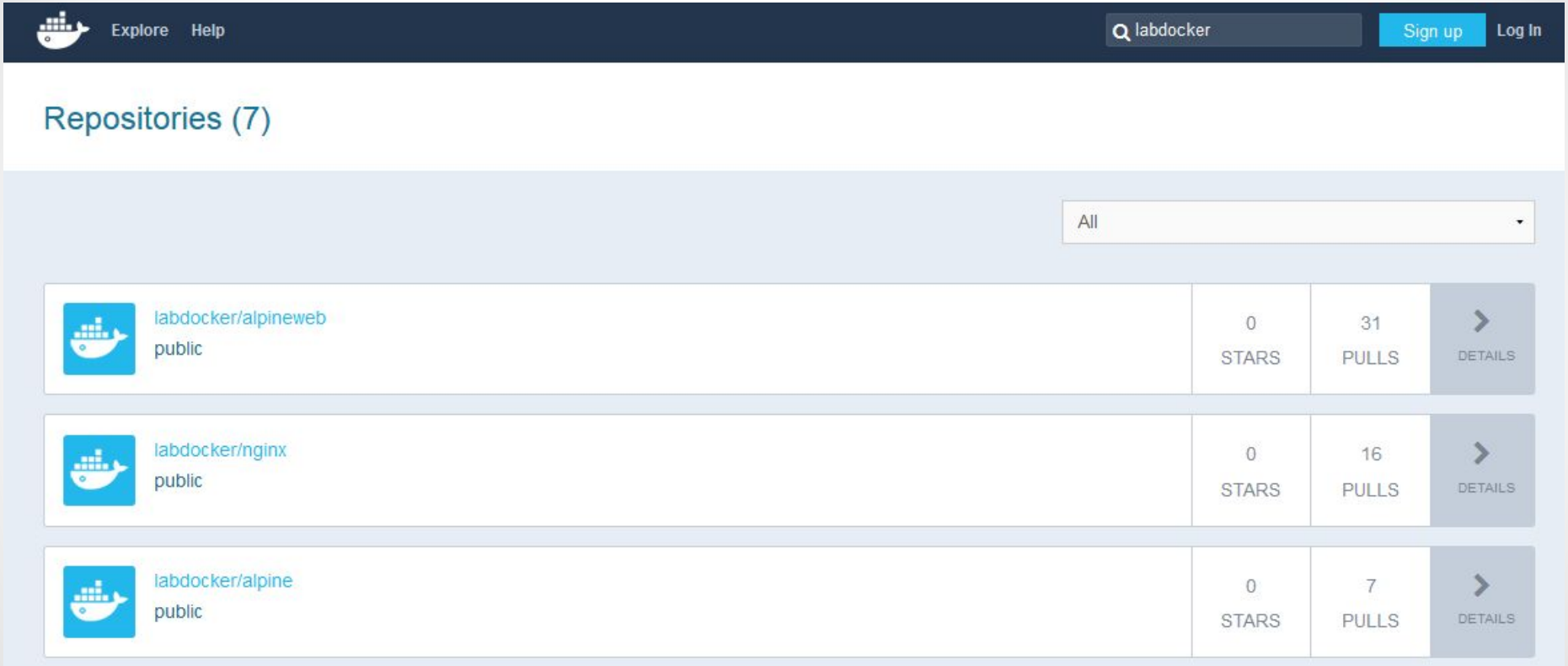
- Generate LINE Token
  - https://notify-bot.line.me



docker

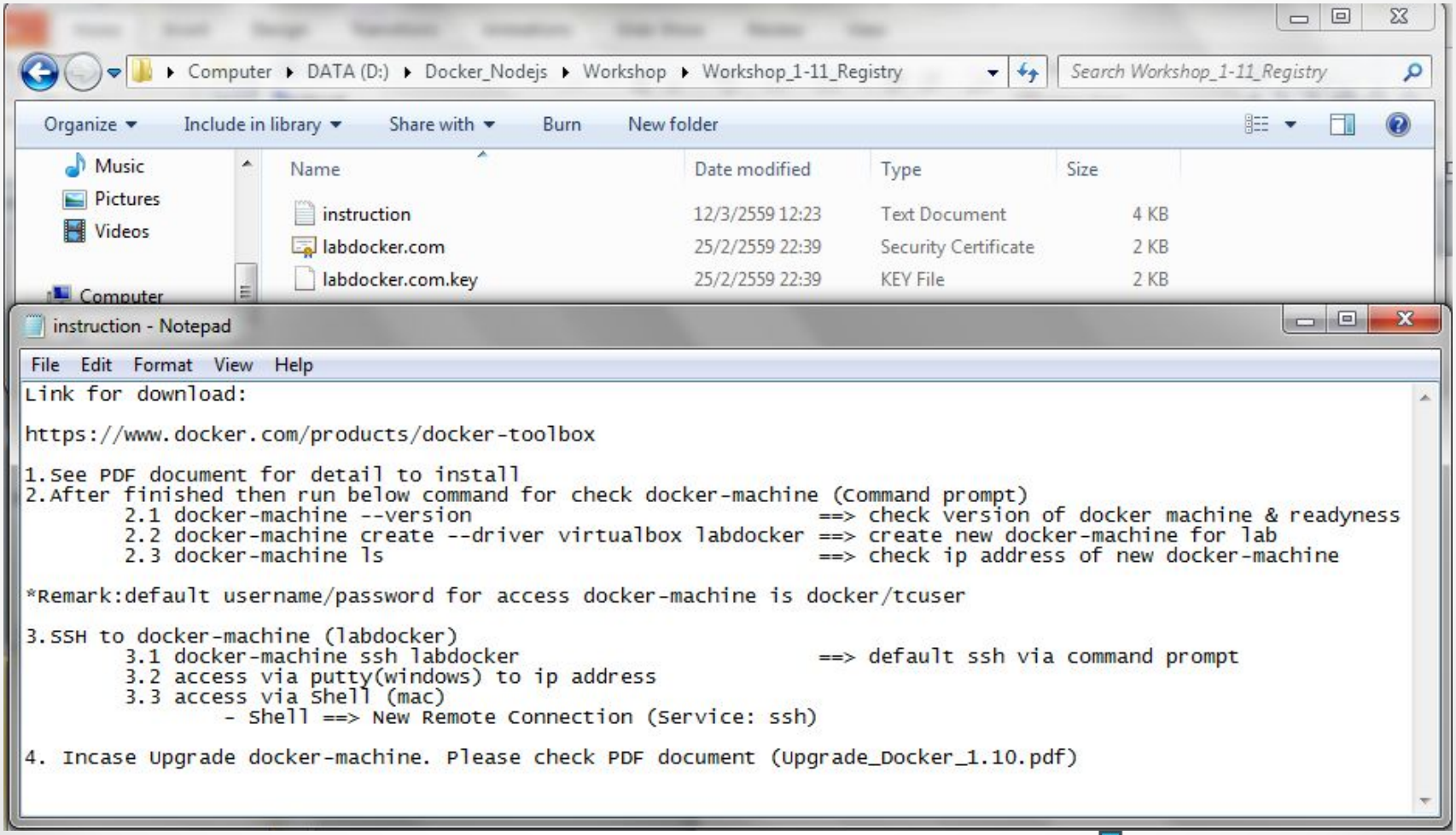
Lab Resource

- Repository for lab



Lab Resource

- Software in lab



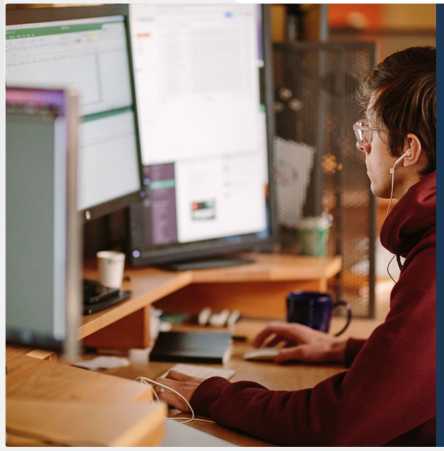
docker

Lab Resource

- Download on GitHub
  - git clone https://github.com/praparn/docker-workshop-online-part1

DockerCon2022

DockerCon 2022



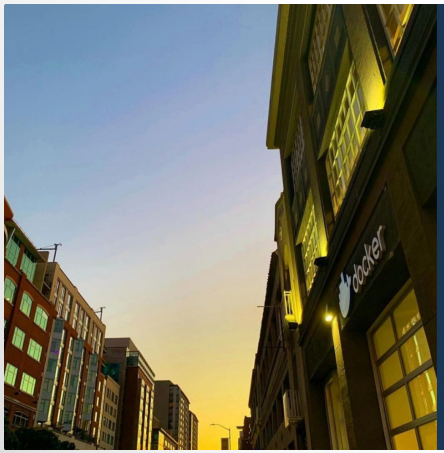
DockerCon: What Makes a Successful CFP Submission

The DockerCon 2022 Call for Papers is now open! DockerCon is one of the largest developer events in the world, with over 80,000 developers registering for each of the last two events. At the core of DockerCon is the chance for members of the community to share their tips, tr...

dockercon, DockerCon 2022, Dockercon CFP

PETER MCKEE
Feb 03 2022

Keep Reading →



DockerCon: What Makes a Successful CFP Submission

The DockerCon 2022 Call for Papers is now open! DockerCon is one of the largest developer events in the world, with over 80,000 developers registering for each of the last two events. At the core of DockerCon is the chance for members of the community to share their tips, tr...

dockercon, DockerCon 2022, Dockercon CFP

PETER MCKEE
Feb 03 2022

Keep Reading →

dockercon

First: it helps to know your audience. At DockerCon, we draw attendees from all over the world with a range of experiences from newbie to guru. In general, however, the majority of the developers who attend DockerCon fall into these three categories:

- **New developers**, who are just getting started with containerization, using Docker, and building cloud native applications. Sessions introducing Docker, ones that put Docker into context of popular programming languages such as JavaScript, Python, Go, .NET and Java, and that help a developer get up and running are quite popular. In these cases, sharing the details you wish you knew when you were getting started make for outstanding sessions and are some of our most viewed content.

- **Experienced developers**, who use Docker but are looking for ways to get their applications built faster and grow their capabilities (both in terms of features as well as individual skills). Proposals for sessions that share best practices, case studies from real-world development experience, as well as ways to incorporate automation, scalability and security within a development inner loop. These all meet the needs of developers with more experience.

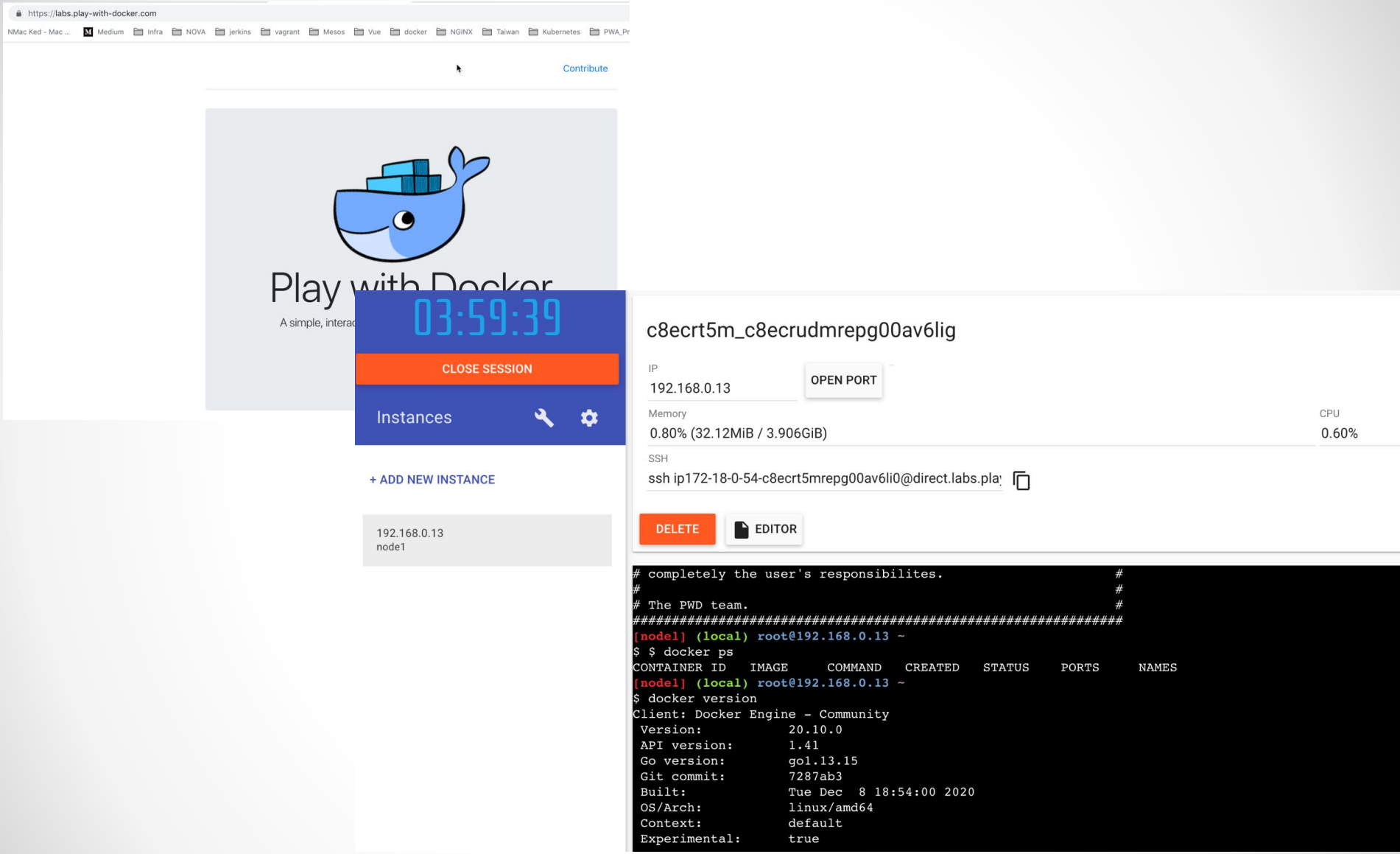
- **Expert developers,** who are digging into the command line, building Docker Images from scratch, and contributing to OSS projects. These developers are attracted to the “black belt” sessions that dive into the internals of the Docker Engine, extensibility and integration into other tools such as CI/CD, repositories, and CSPs. Examples that show how hard problems are solved, how the Docker Engine works, digging into OSS projects, and showing how you can “run with scissors” make for compelling content on cutting-edge topics.

Docker: The Next-Gen of Virtualization

docker

Workshop: Access Docker on AWS

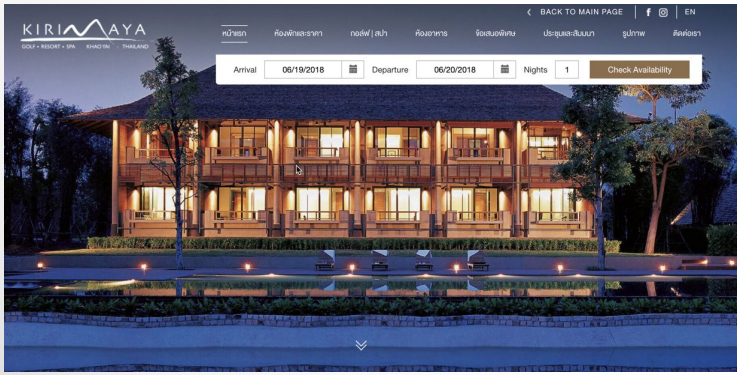
Workshop: Access Docker on AWS



Docker: The Next-Gen of Virtualization

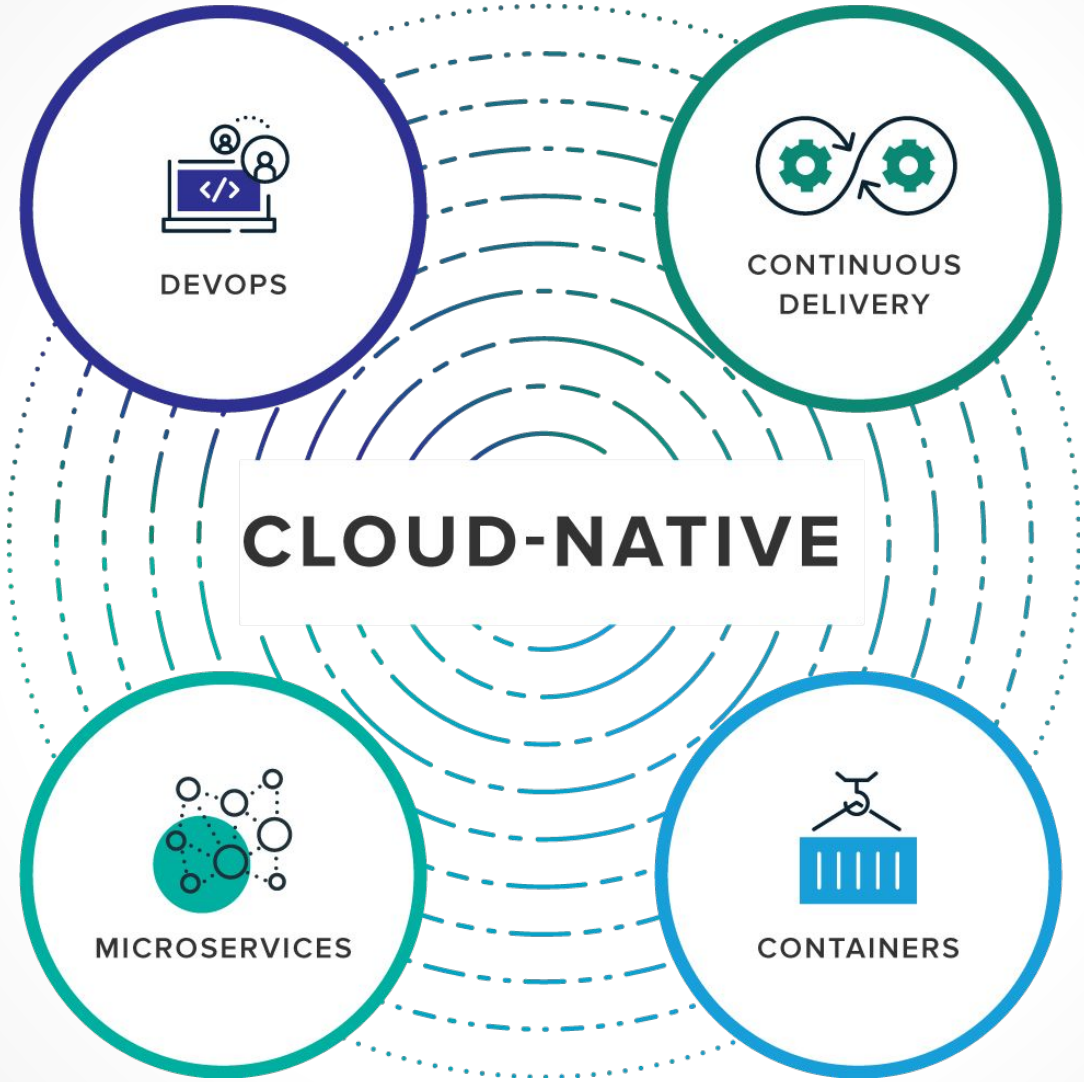
docker

Who are we ? (Opcellent)



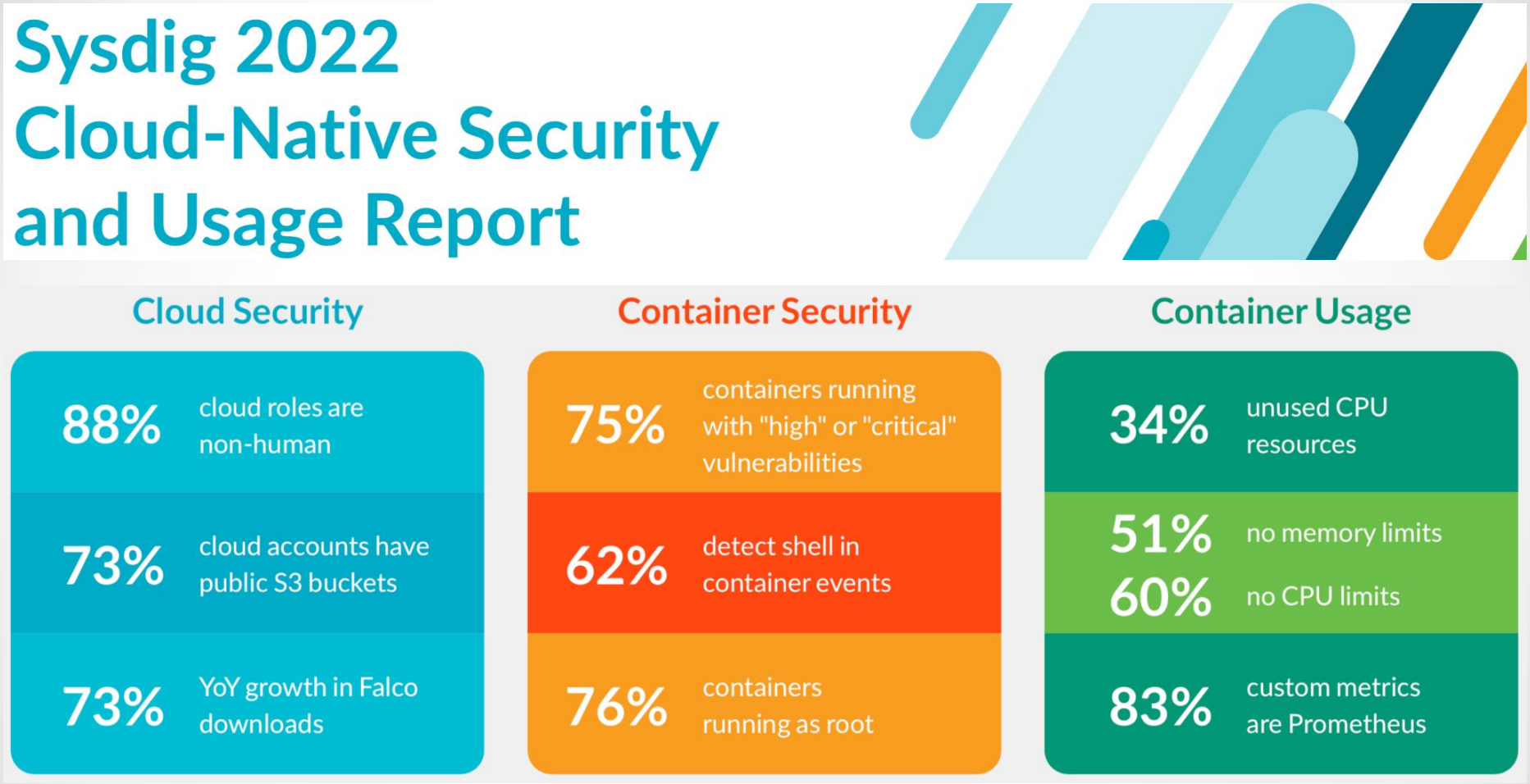
Docker: The Next-Gen of Virtualization

Landscape of the world now



docker

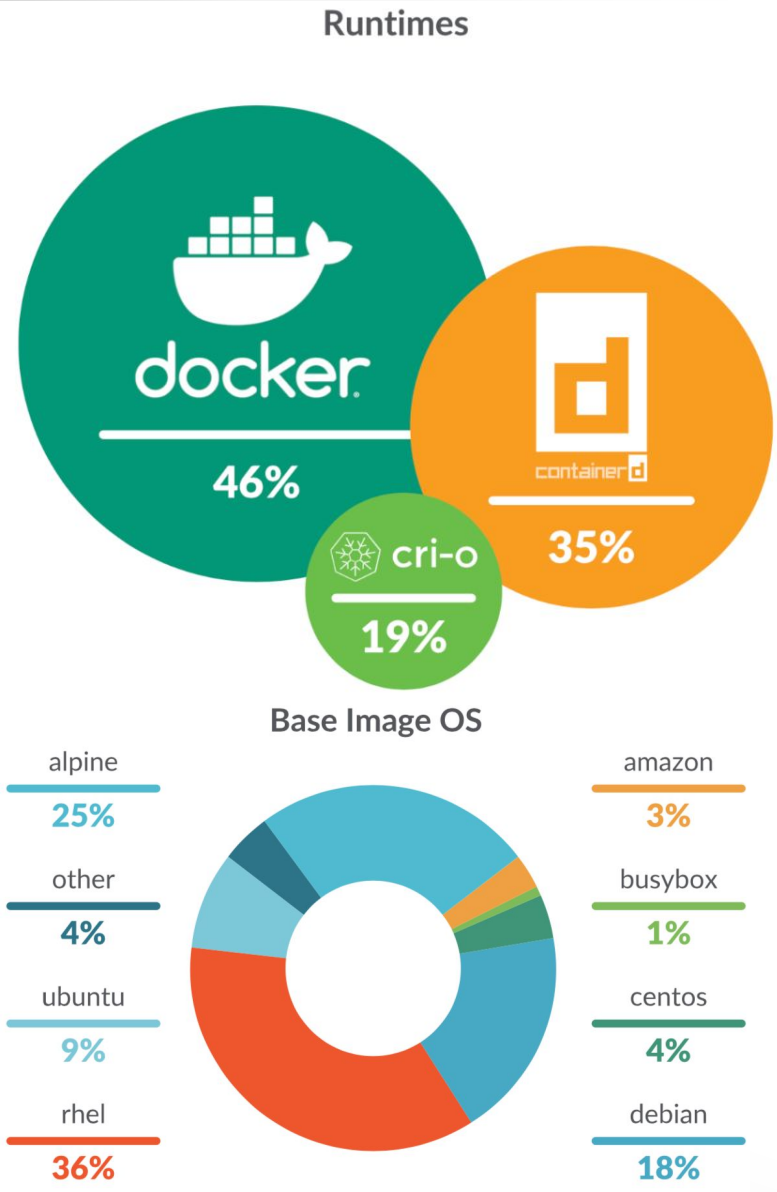
Landscape of the world now

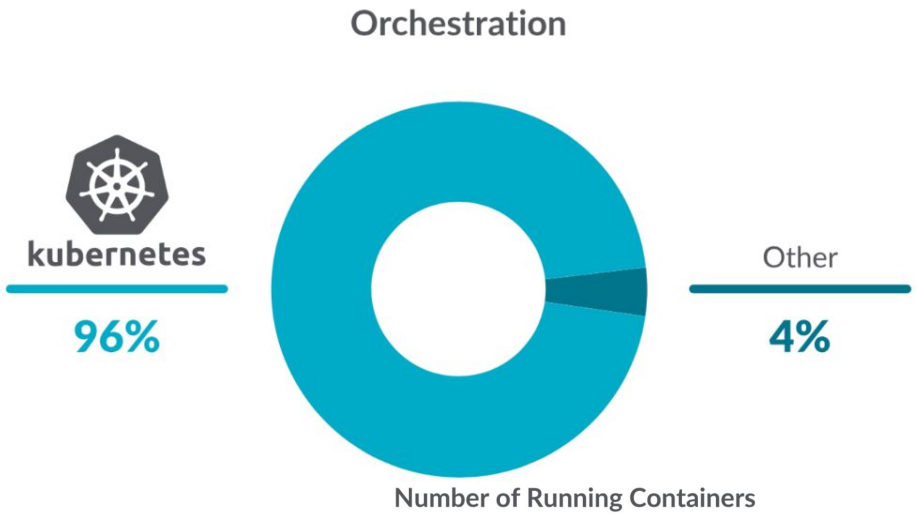


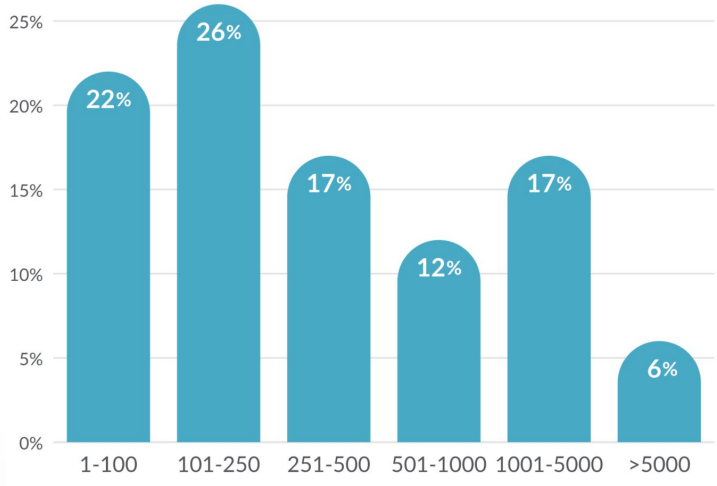
Ref: https://sysdig.com/blog/2022-cloud-native-security-usage-report/

docker

Landscape of the world now

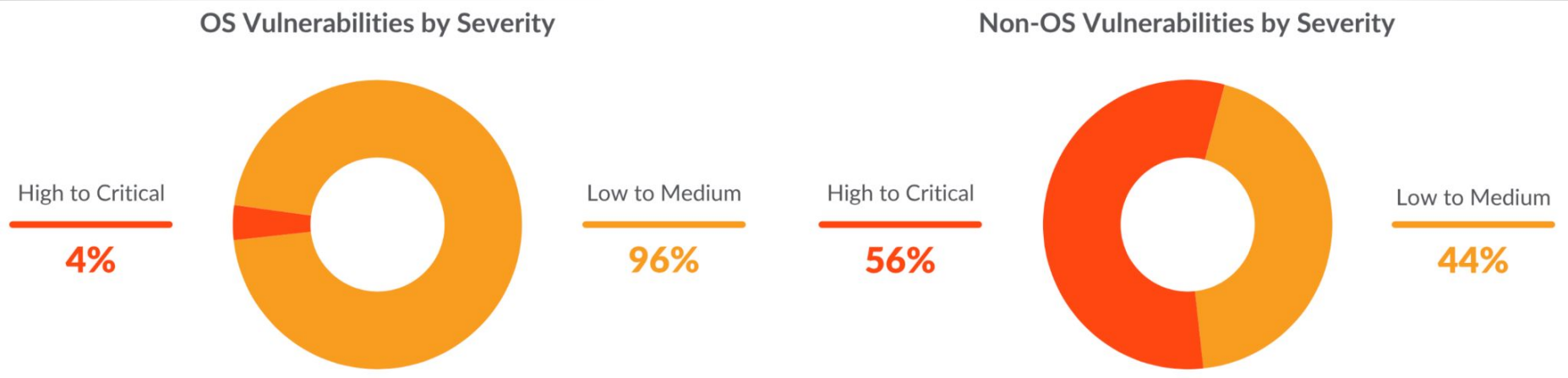


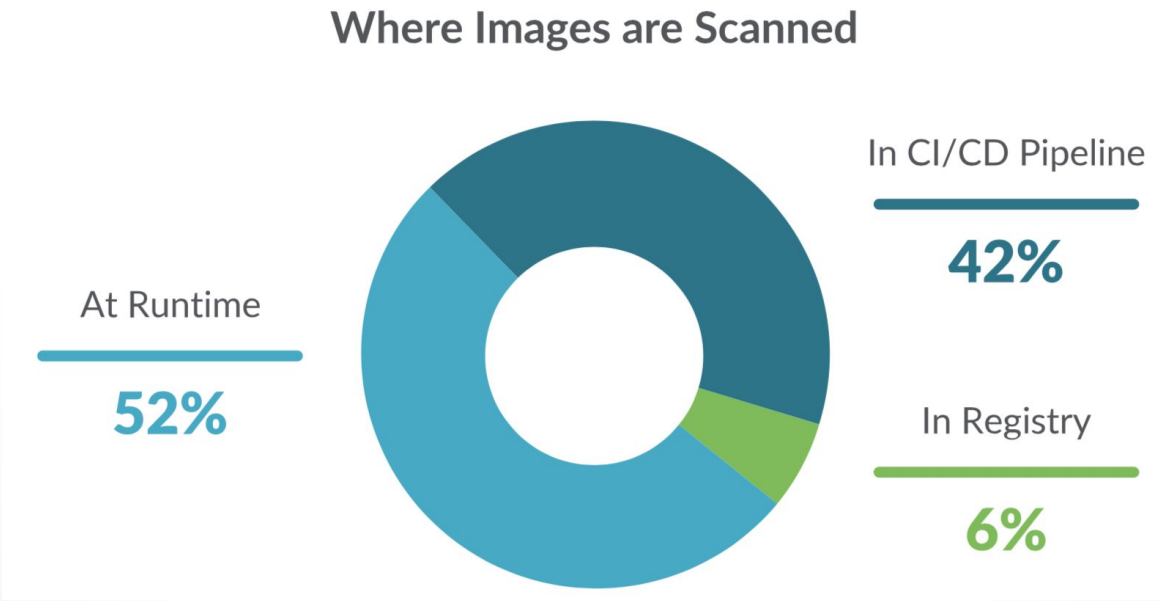




Docker: The Next-Gen of Virtualization

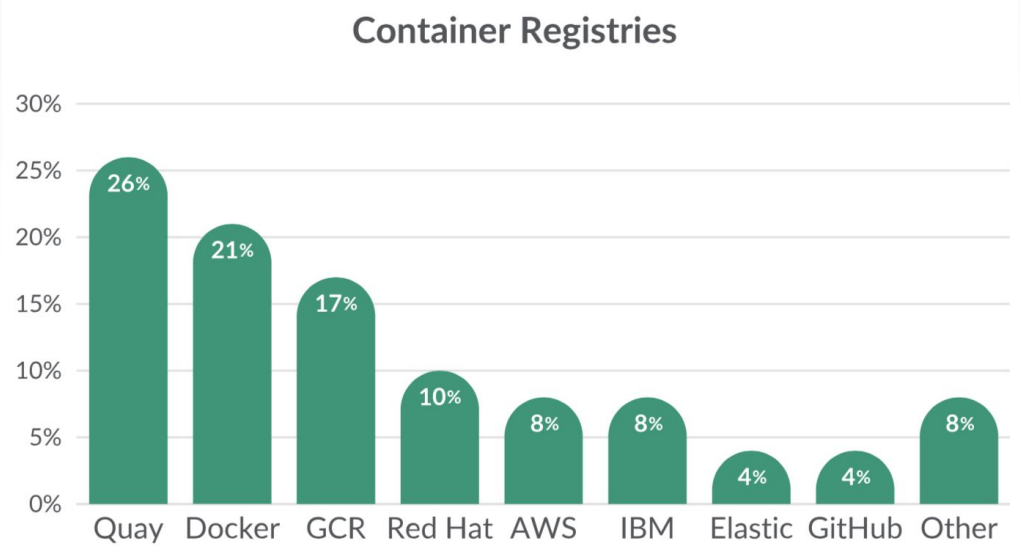
Landscape of the world now

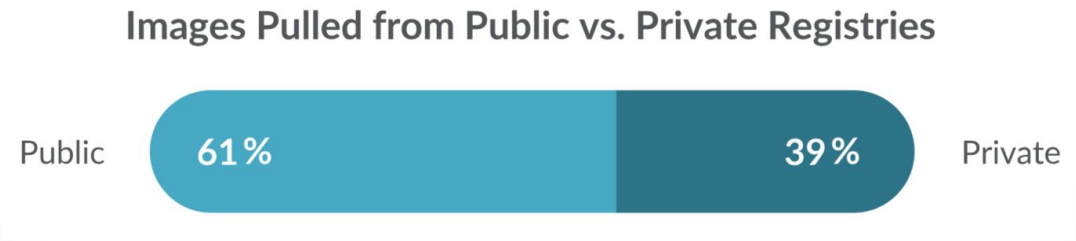




docker

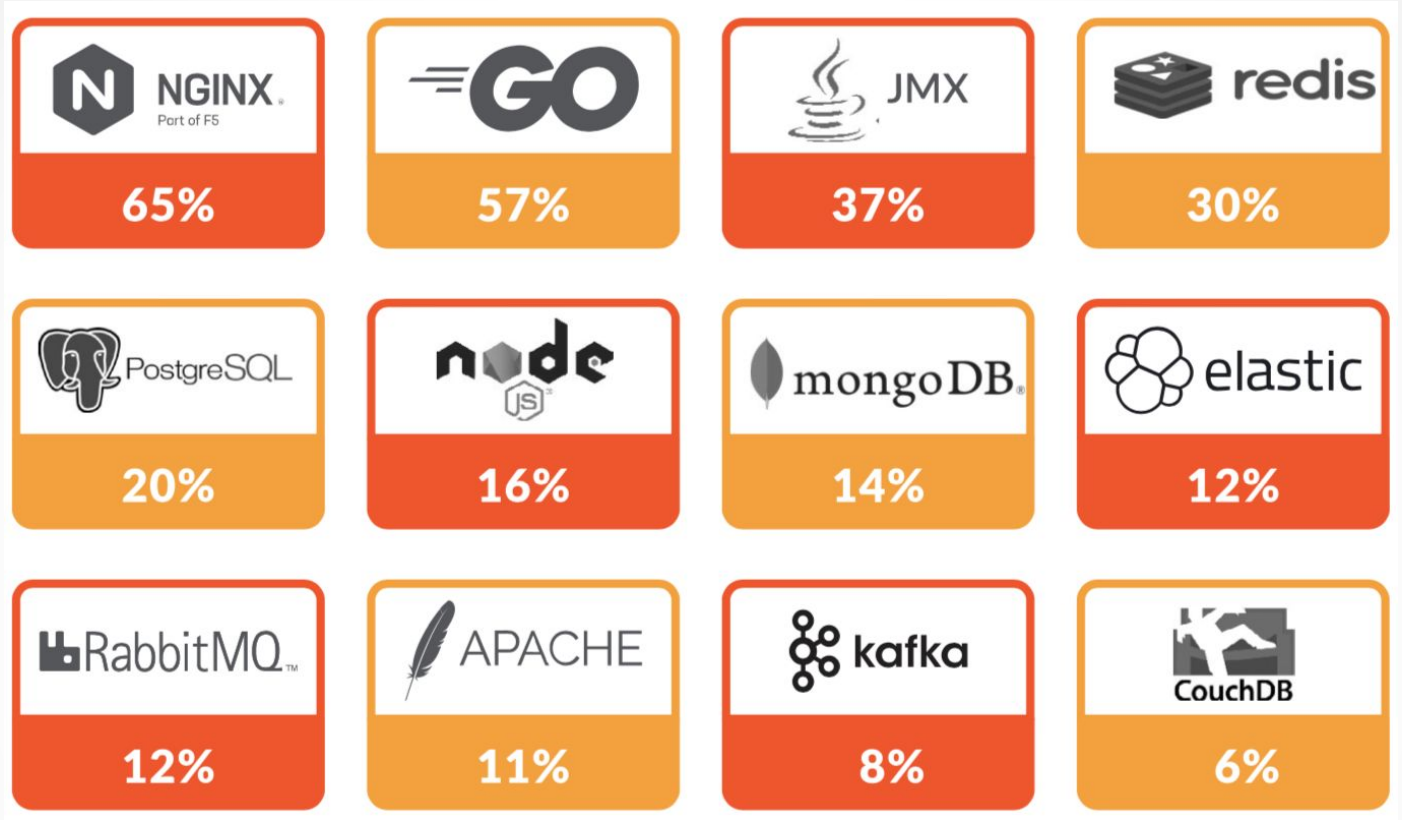
Landscape of the world now





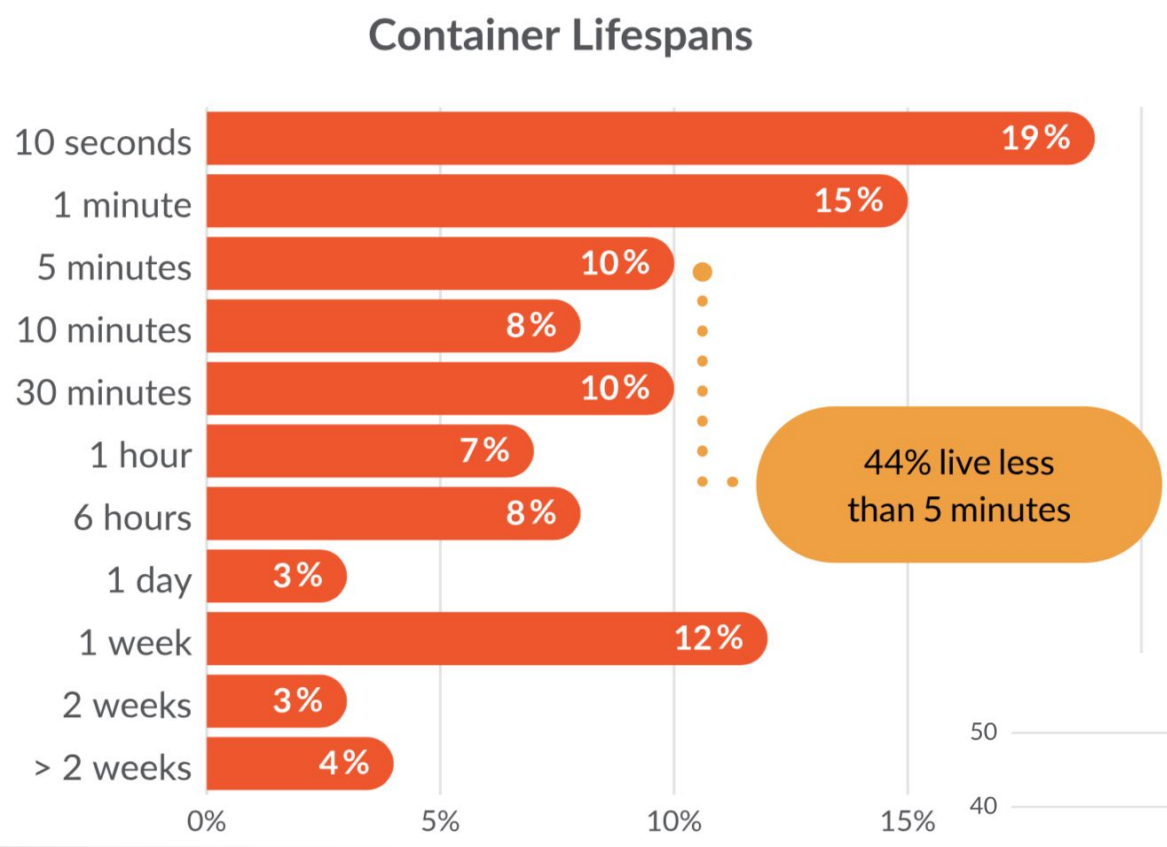
docker

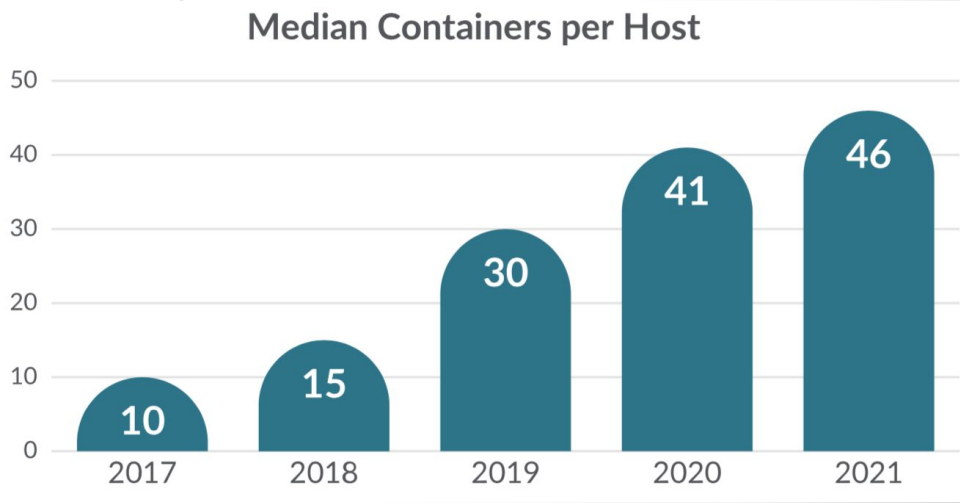
Landscape of the world now



docker

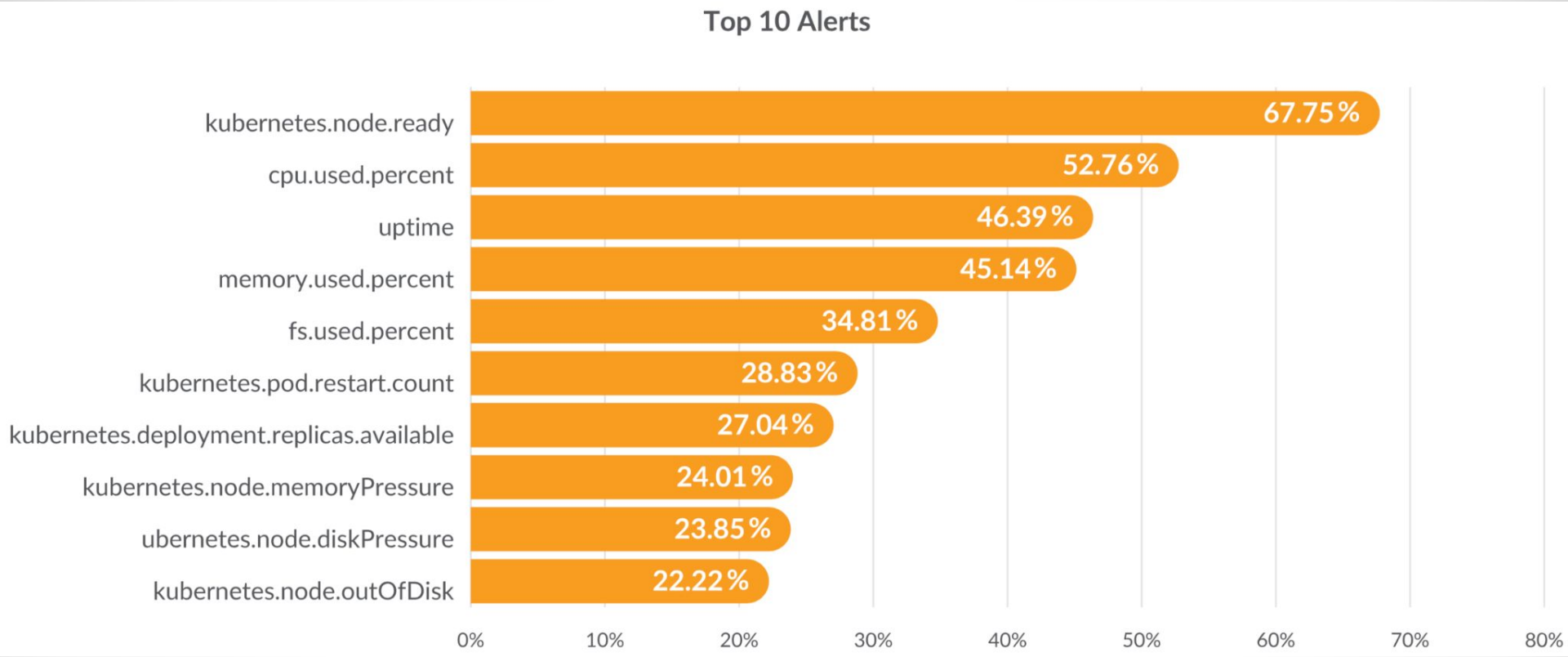
Landscape of the world now





docker

Landscape of the world now

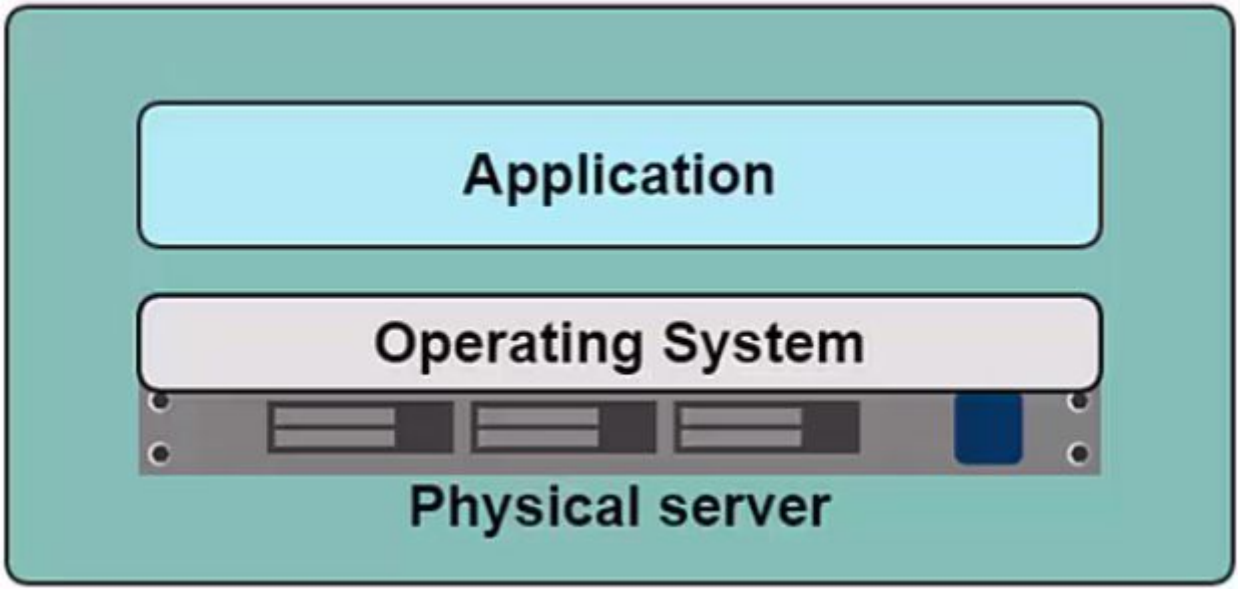


docker

Docker Principle

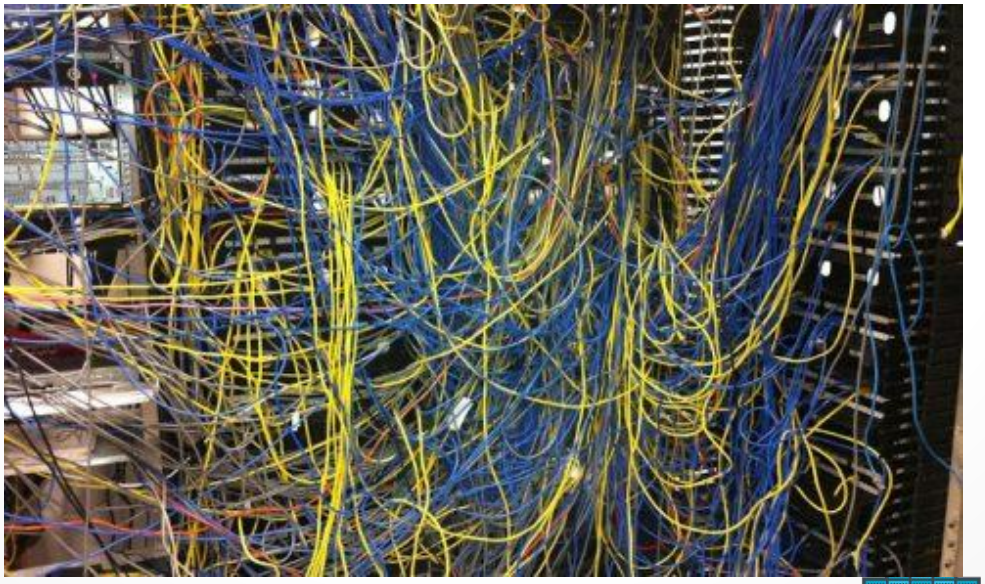
• • •

What is docker ?



docker

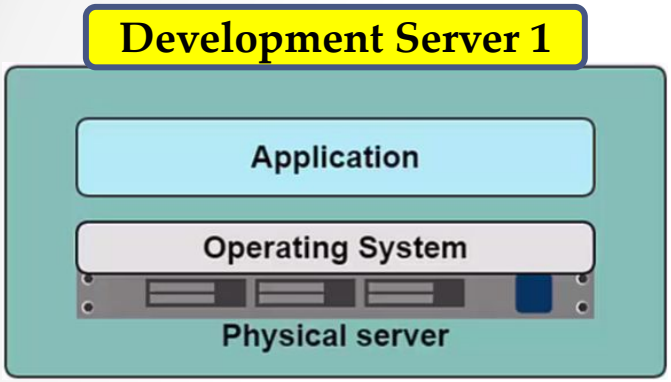
Existing Technology

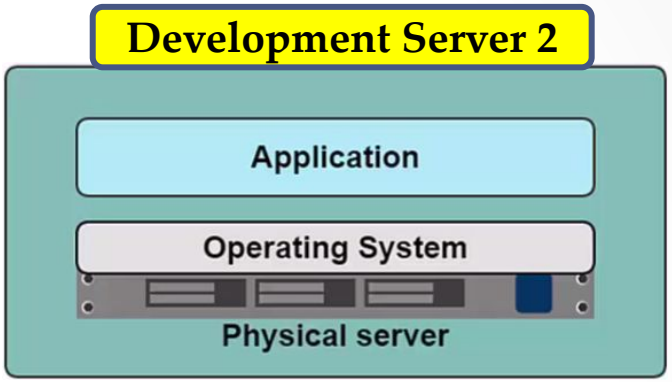


docker

Existing Technology

- Development Environment (Mix everything possible)





- Apache 2.20 Web Server
- PHP 5.5 Engine
- Larevel 4.1 Framework
- MaridDB 5.1
- Etc

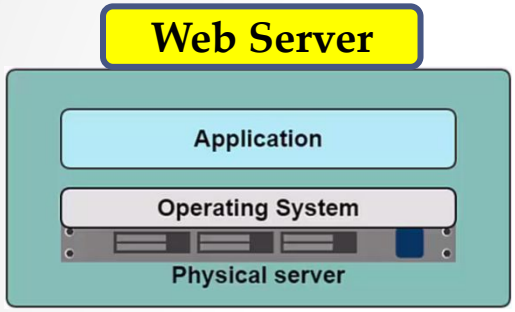
(All-in-One Server)

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

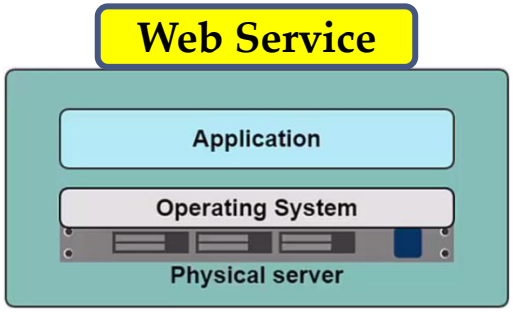
- Need concern about conflict component
- Survive among legacy dependency
- Lack for environment for fulfill develop & test (module test/integrate test/ UAT test / MOT test etc)
- Unexpected software conflict frequently occur
- Incomplete software's integrated test

Existing Technology

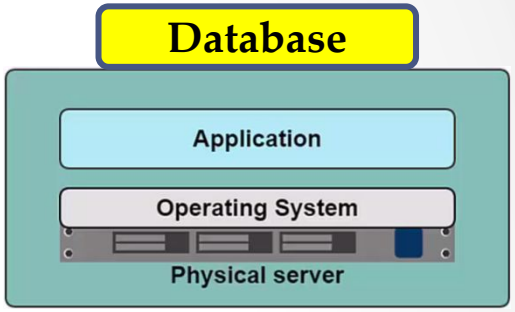
- Production Environment (Best design)
- Day 1: Application 1: Implement



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



- IIS 8
- .Net FrameWork 3.5



- MariaDB 5.1

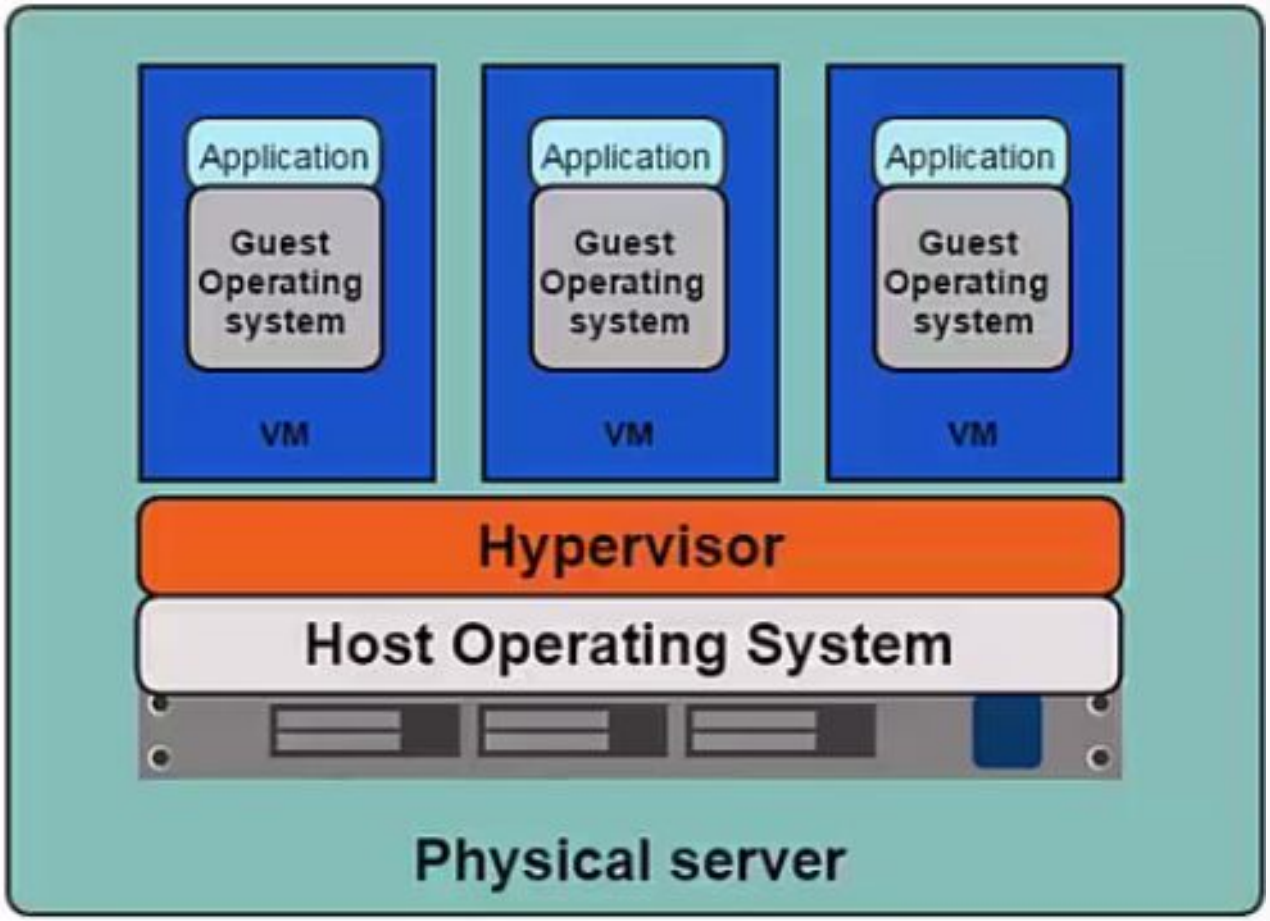
- Day 2: Application 2: Need to implement
  - Need PHP 7.0 ?
  - MariaDB 10.1.14 (Need search feature on 10.1)
- Problem ?
  - Possible to upgrade PHP to 7.0 ? / How to test existing application ?
  - What effect to MariaDB upgrade ?

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)

docker

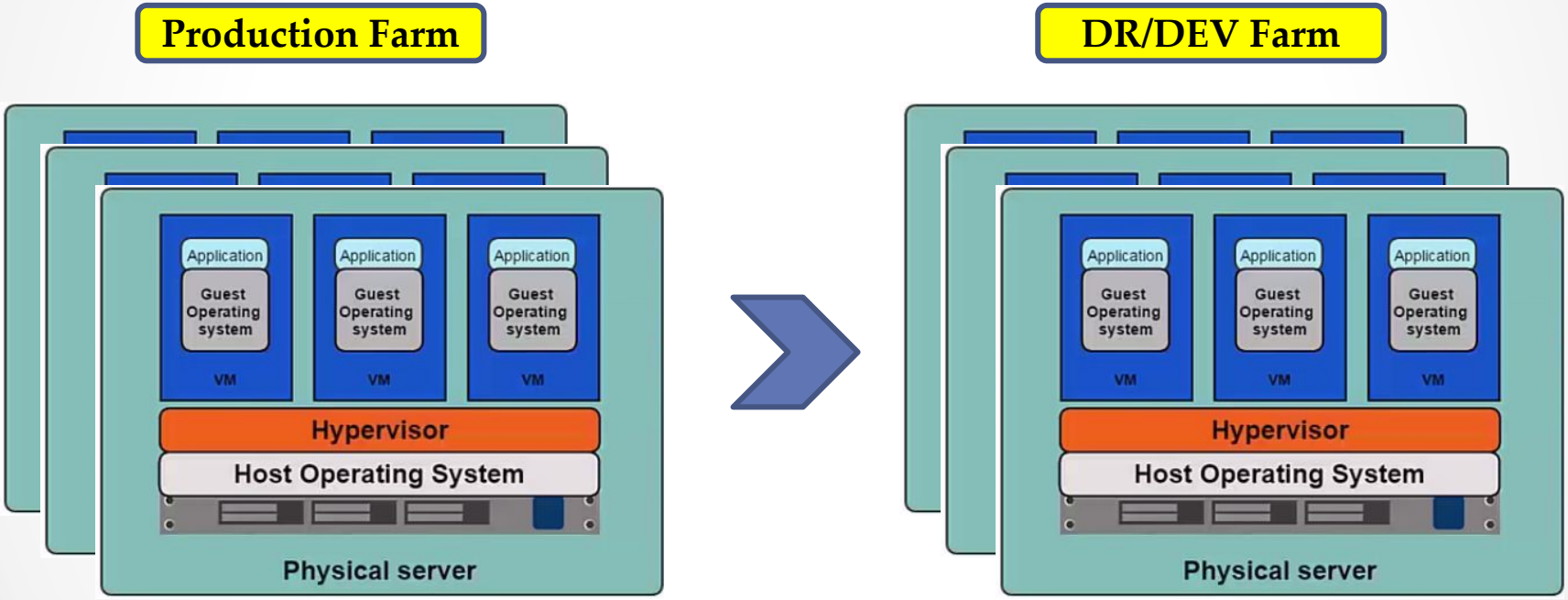
What is docker ?



docker

Existing Technology

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

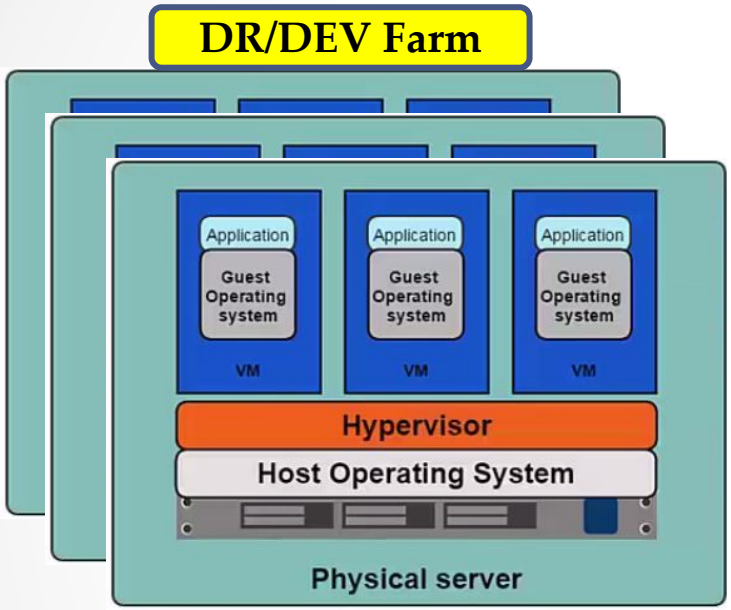


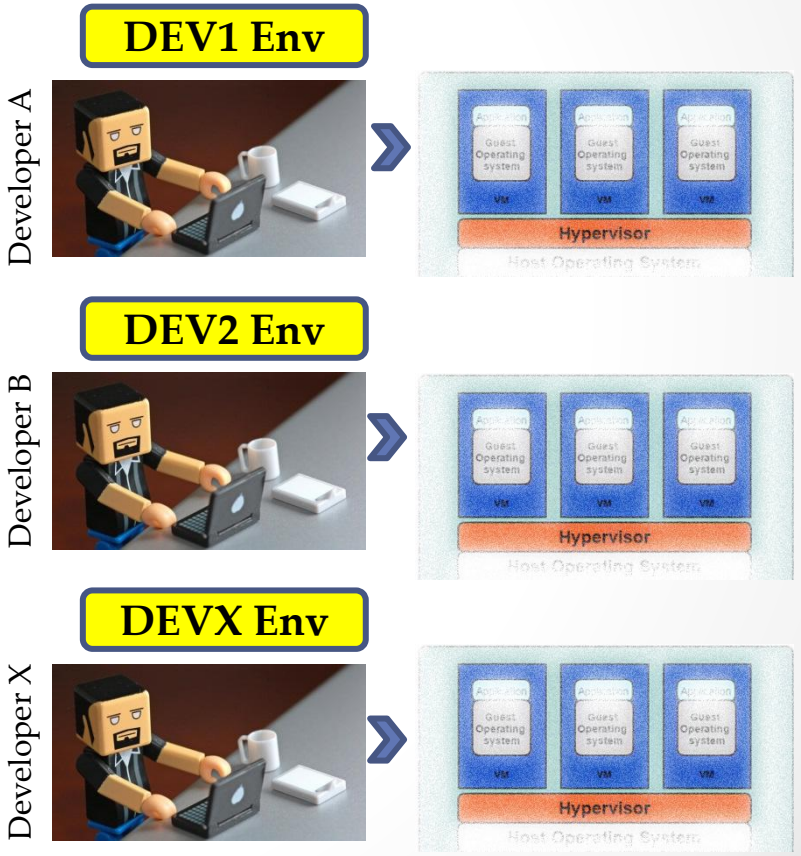
- Kernal-base virtual machine (KVM), Vmware, Virtualbox etc

docker

Existing Technology

- Development Environment (Clone from Production)



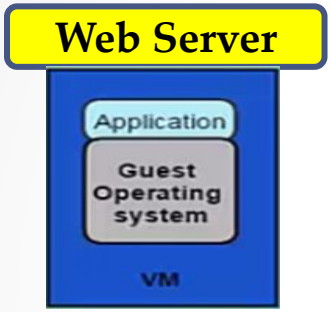


Virtual Machine reach limit:
- Resource insufficient (Almost from disk)
- Conflict version
- Huge of disk duplication
- Conflict & dependency still exist

docker

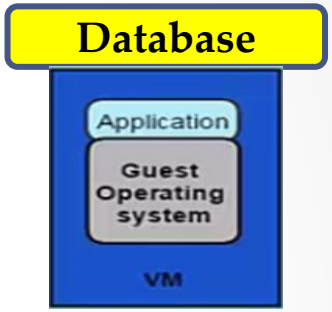
Existing Technology

- Production Environment
- Day 1: Application 1: Implement



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

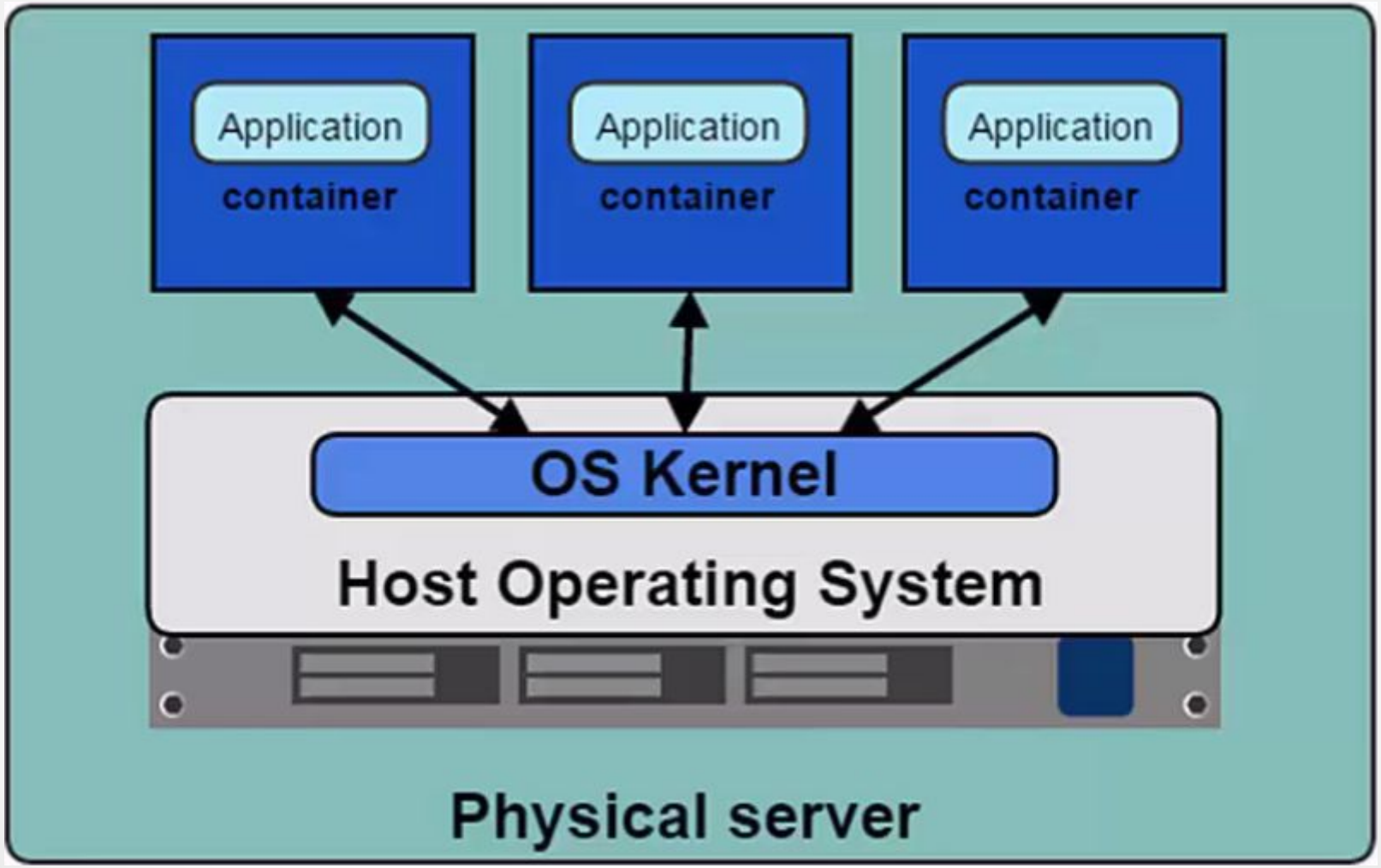
- IIS 8
- .Net FrameWork 3.5



- MariaDB 5.1

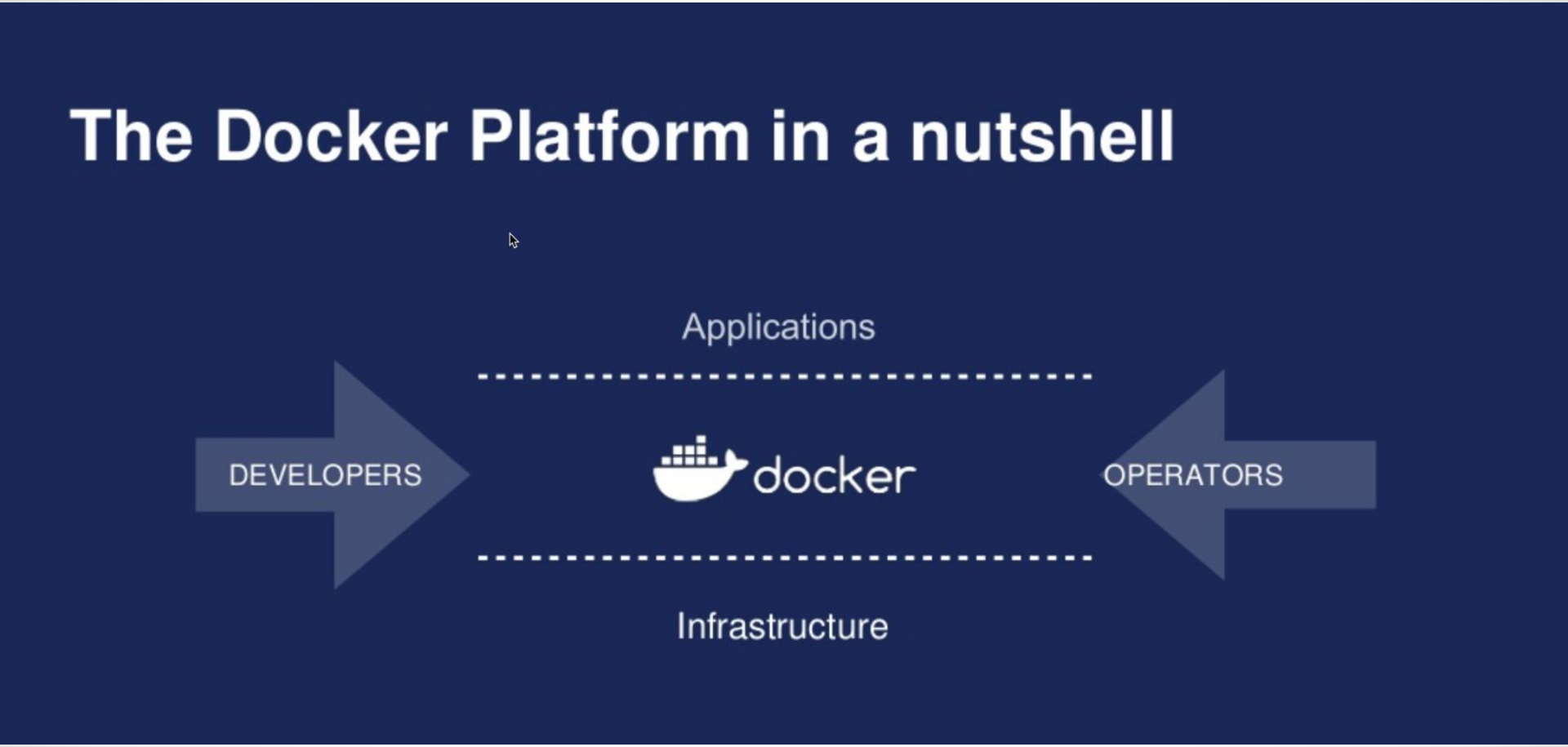
- Day 2: Application 2: Need to implement
  - Need PHP 7.0 ?
  - MariaDB 10.1.14 (Need search feature on 10.1)
- So… The problem still exist.

What is docker ?



docker

What is docker ?



docker

What is docker ?

Core Principles of the Docker Platform

INDEPENDENCE

OPENNESS

SIMPLICITY

dockercon17 EU

docker

What is docker ?

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

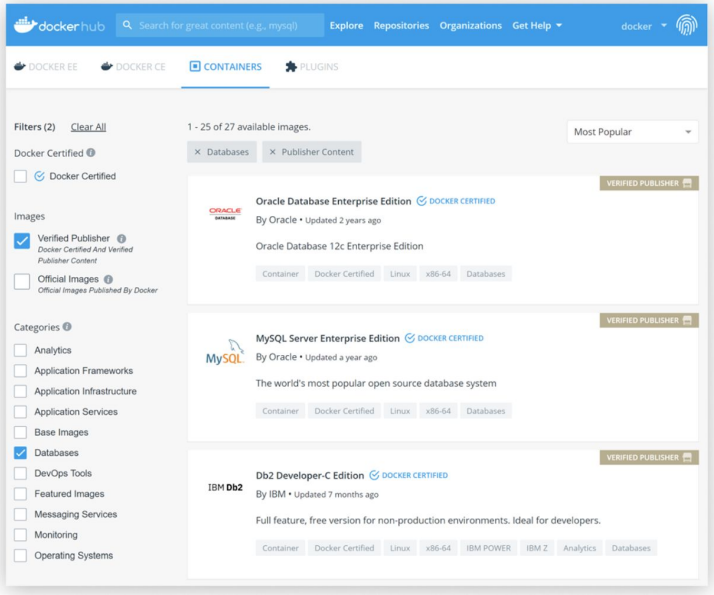
Docker's product

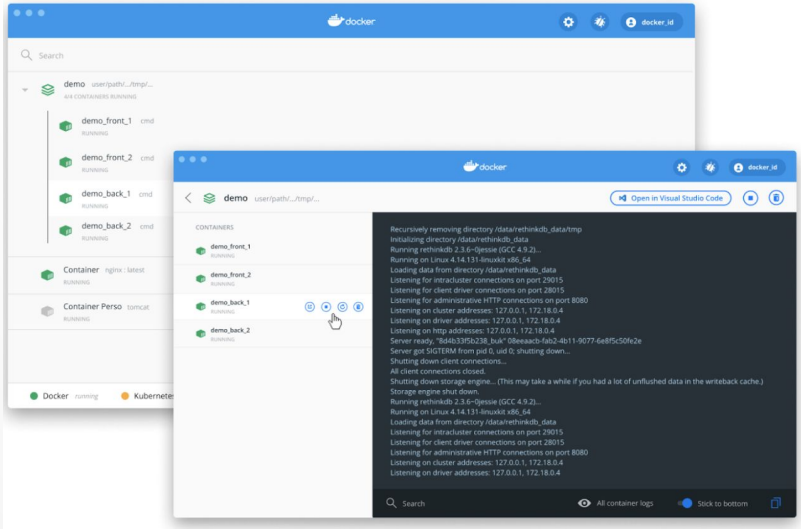
Docker Hub

The world's leading service for finding and sharing container images with your team and the Docker community.

For developers and those experimenting with Docker, Docker Hub is your starting point into Docker containers. Create an account and start exploring the millions of images that are available from the community and verified publishers.

See Docker Hub





Docker Desktop

The preferred choice for millions of developers that are building containerized apps.

Docker Desktop is an application for MacOS and Windows machines for the building and sharing of containerized applications. Access Docker Desktop and follow the guided onboarding to build your first containerized application in minutes.

See Docker Desktop

Docker: The Next-Gen of Virtualization

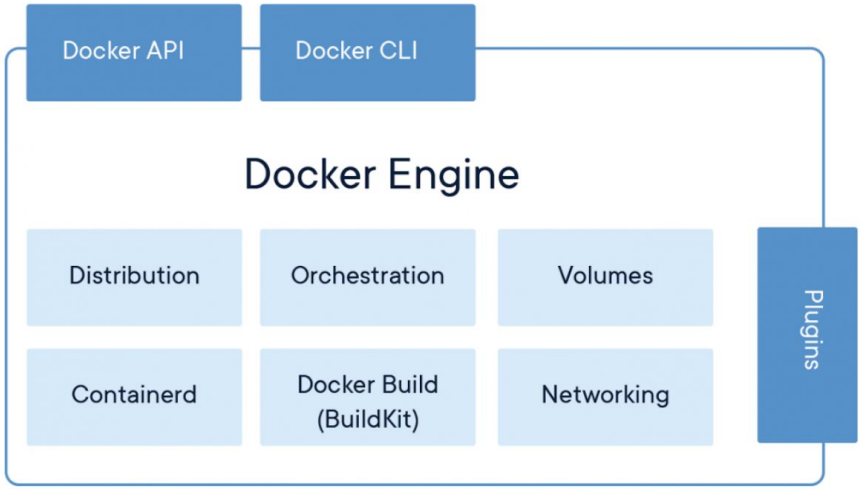
docker

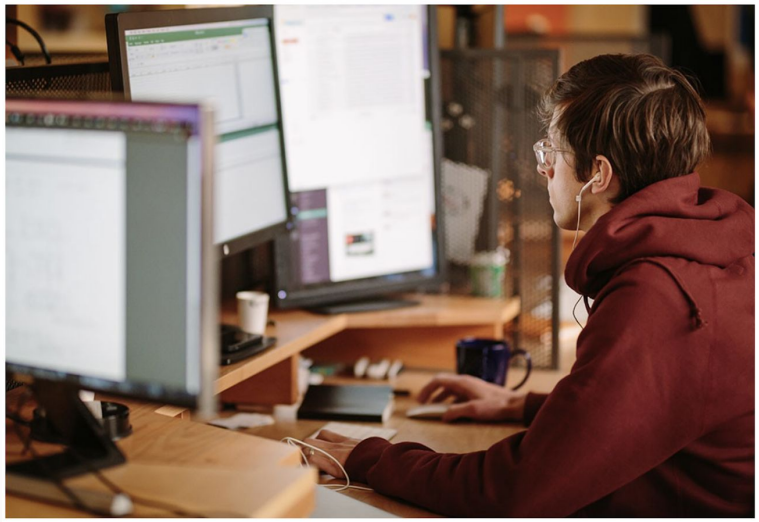
Docker's feature

Container Runtime

Docker Engine powers millions of applications worldwide, providing a standardized packaging format for diverse applications.

See Container Runtime





Developer Tools

The fastest way to securely build, test, and share cloud-ready modern applications from your desktop.

See Developer Tools

docker

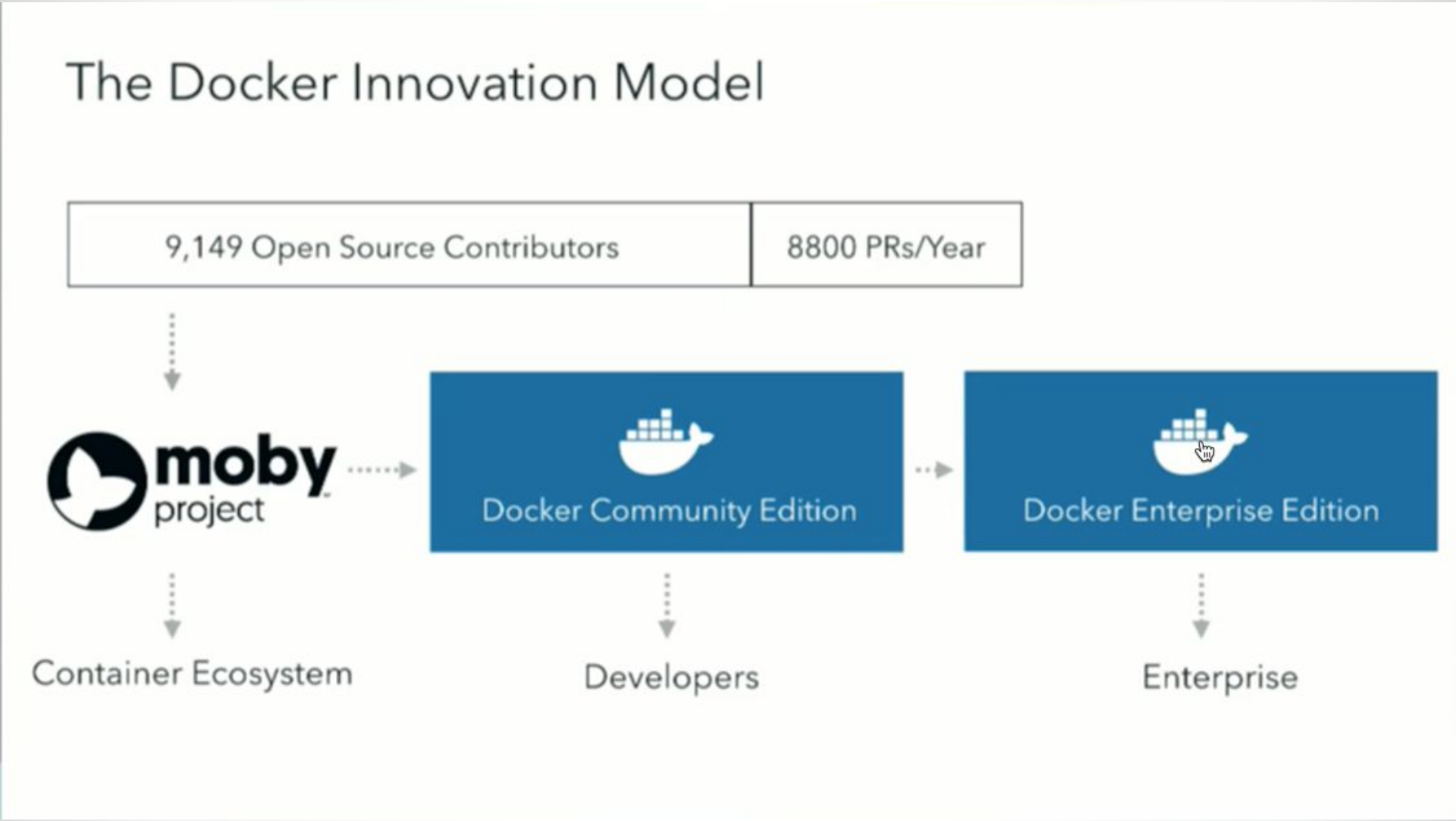
Docker's feature

Kubernetes

Kubernetes has become the standard orchestration platform for containers. All the major cloud providers support it, making it the logical choice for organizations looking to move more applications to the cloud.

See Kubernetes

What is docker ?



docker

What is docker ?

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

Enterprise Edition (EE)	Community Edition (CE)
• 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	• 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

Docker: The Next-Gen of Virtualization

What is docker ?

About Docker CE

Estimated reading time: 7 minutes

Docker Community Edition (CE) is ideal for developers and small teams looking to get started with Docker and experimenting with container-based apps. Docker CE has three types of update channels, **stable**, **test**, and **nightly**:

- **Stable** gives you latest releases for general availability.
- **Test** gives pre-releases that are ready for testing before general availability.
- **Nightly** gives you latest builds of work in progress for the next major release.

For more information about Docker CE, see Docker Community Edition.

Releases

For the Docker CE engine, the open repositories Docker Engine and Docker Client apply.

Releases of Docker Engine and Docker Client for general availability are versioned using dotted triples. The components of this triple are `YY.mm.<patch>` where the `YY.mm` component is referred to as the year-month release. The version numbering format is chosen to illustrate cadence and does not guarantee SemVer, but the desired date for general availability. The version number may have additional information, such as beta and release candidate qualifications. Such releases are considered “pre-releases”.

The cadence of the year-month releases is every 6 months starting with the `18.09` release. The patch releases for a year-month release take place as needed to address bug fixes during its support cycle.

Docker CE binaries for a release are available on download.docker.com as packages for the supported operating systems. Docker EE binaries are available on the Docker Hub for the supported operating systems. The release channels are available for each of the year-month releases and allow users to “pin” on a year-month release of choice. The release channel also receives patch releases when they become available.

Ref: http://dockerdocs.gclearning.cn/install/

What is docker ?

- DOCKER CE

Desktop

Platform	x86_64 / amd64
Docker Desktop for Mac (macOS)	✓
Docker Desktop for Windows	✓

Server

Docker provides `.deb` and `.rpm` packages from the following Linux distributions and architectures:

Platform	x86_64 / amd64	ARM	ARM64 / AARCH64
CentOS	✓		✓
Debian	✓	✓	✓
Fedora	✓		✓
Raspbian		✓	✓
Ubuntu	✓	✓	✓

Docker History

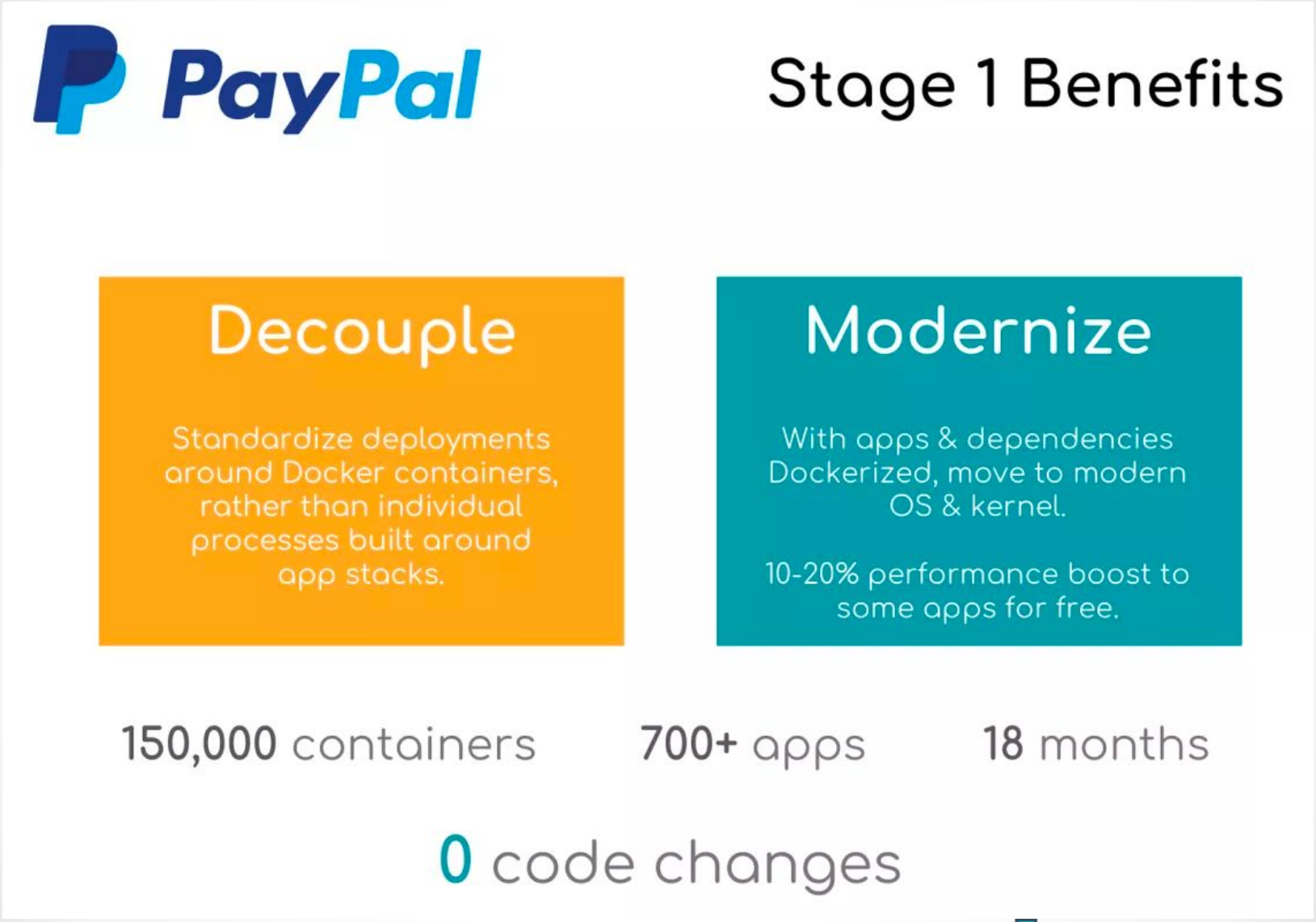
- A dotCloud (PAAS provider) project
- Initial commit January 18, 2013
- Docker 0.1.0 released March 25, 2013
- 18,600+ github stars, 3800+ forks,740 Contributors…. and continues
- dotCloud pivots to docker inc. October 29, 2013

docker

Who use docker now ?

docker

Who use docker now ?

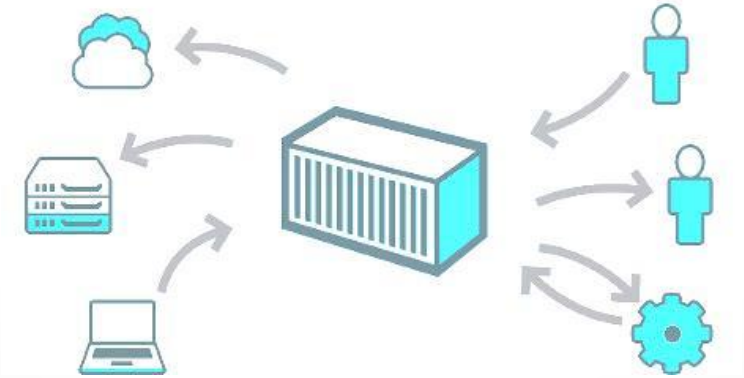


docker

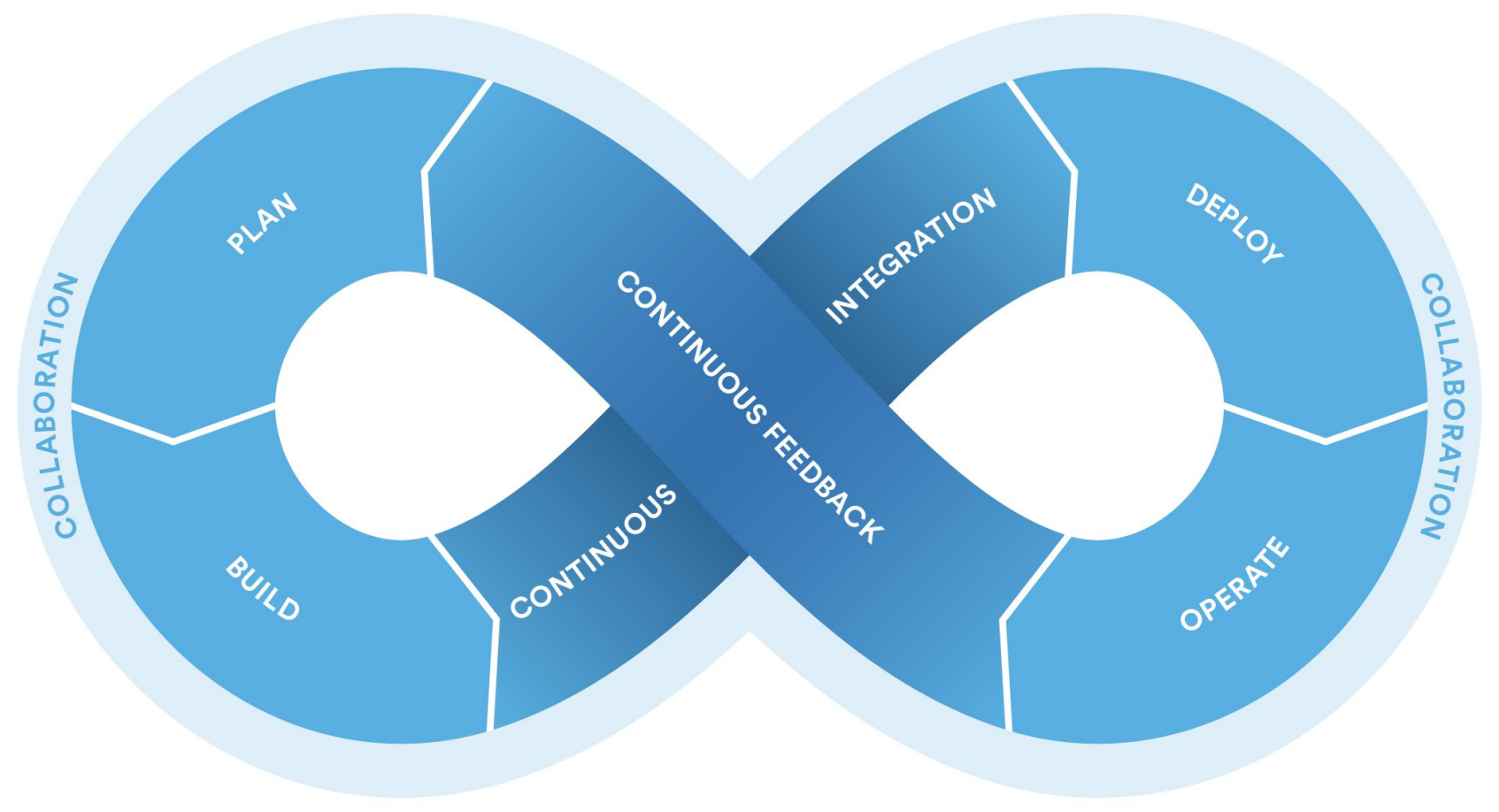
Container Life Cycle

- 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



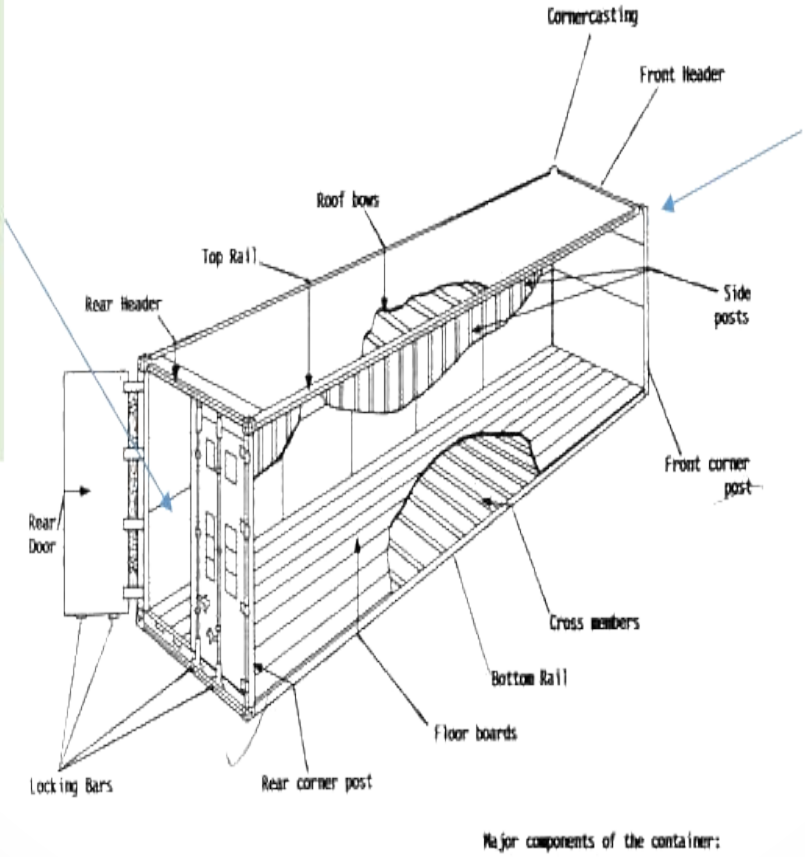
docker

What Cool of Docker ?

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

Separate of Concern

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



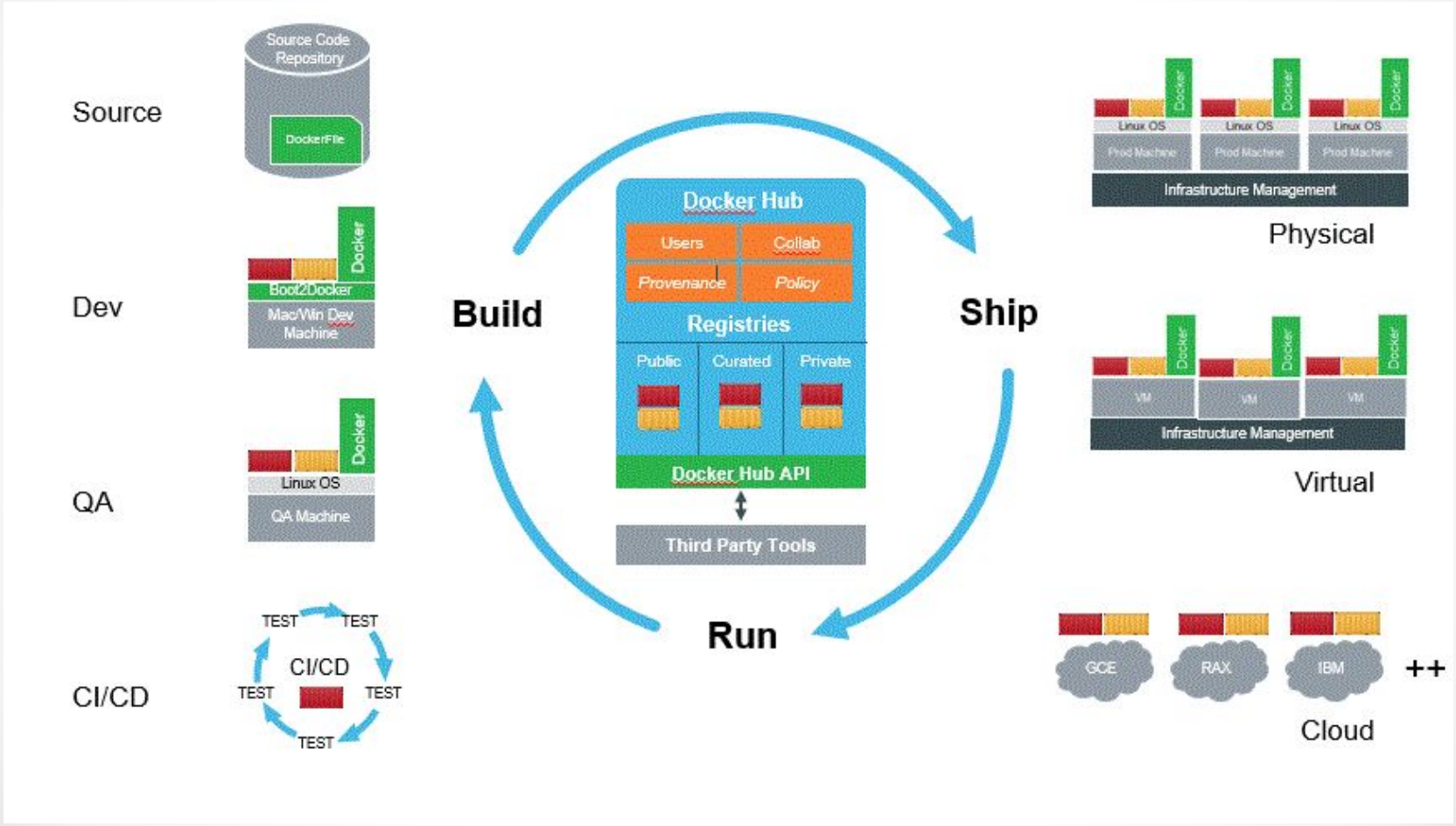
Major components of the container:

- 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

docker

Benefit of docker for DevOps

- Build-Ship-Run



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

docker

Docker Info

```
ubuntu@ip-10-21-1-93:~$ docker version
Client: Docker Engine - Community
 Version:           20.10.12
 API version:       1.41
 Go version:        go1.16.12
 Git commit:        e91ed57
 Built:             Mon Dec 13 11:45:33 2021
 OS/Arch:           linux/amd64
 Context:           default
 Experimental:      true

Server: Docker Engine - Community
 Engine:
  Version:          20.10.12
  API version:      1.41 (minimum version 1.12)
  Go version:       go1.16.12
  Git commit:       459d0df
  Built:            Mon Dec 13 11:43:42 2021
  OS/Arch:          linux/amd64
  Experimental:     false
 containerd:
  Version:          1.4.12
  GitCommit:        7b11cfaabd73bb80907dd23182b9347b4245eb5d
 runc:
  Version:          1.0.2
  GitCommit:        v1.0.2-0-g52b36a2
 docker-init:
  Version:          0.19.0
  GitCommit:        de40ad0
ubuntu@ip-10-21-1-93:~$
```

```
ubuntu@ip-10-21-1-93:~$ docker info
Client:
 Context:    default
 Debug Mode: false
 Plugins:
  app: Docker App (Docker Inc., v0.9.1-beta3)
  buildx: Docker Buildx (Docker Inc., v0.7.1-docker)
  scan: Docker Scan (Docker Inc., v0.12.0)

Server:
 Containers: 0
  Running: 0
  Paused: 0
  Stopped: 0
 Images: 0
 Server Version: 20.10.12
 Storage Driver: overlay2
  Backing Filesystem: extfs
  Supports d_type: true
  Native Overlay Diff: true
  userxattr: false
 Logging Driver: json-file
 Cgroup Driver: systemd
 Cgroup Version: 1
 Plugins:
  Volume: local
  Network: bridge host ipvlan macvlan null overlay
  Log: awslogs fluentd gcplogs gelf journald json-file local logentries splunk syslog
 Swarm: inactive
 Runtimes: io.containerd.runc.v2 io.containerd.runtime.v1.linux runc
 Default Runtime: runc
 Init Binary: docker-init
 containerd version: 7b11cfaabd73bb80907dd23182b9347b4245eb5d
 runc version: v1.0.2-0-g52b36a2
 init version: de40ad0
 Security Options:
  apparmor
  seccomp
   Profile: default
 Kernel Version: 5.11.0-1022-aws
 Operating System: Ubuntu 20.04.3 LTS
 OSType: linux
 Architecture: x86_64
 CPUs: 2
 Total Memory: 1.901GiB
 Name: ip-10-21-1-93
 ID: DD7D:NCAA:PHHS:VXTO:YVE5:BEG4:GLZ3:TVTG:OR4E:PQTU:BOUM:UTE5
 Docker Root Dir: /var/lib/docker
 Debug Mode: false
 Registry: https://index.docker.io/v1/
 Labels:
 Experimental: false
 Insecure Registries:
  127.0.0.0/8
 Live Restore Enabled: false

ubuntu@ip-10-21-1-93:~$
```

Docker Version

Docker Engine release notes

This document describes the latest changes, additions, known issues, and fixes for Docker Engine.

> **Note:** The client and container runtime are now in separate packages from the daemon in Docker Engine 18.09. Users should install and update all three packages at the same time to get the latest patch releases. For example, on Ubuntu: `sudo apt install docker-ce docker-ce-cli containerd.io`. See the install instructions for the corresponding linux distro for details.

Version 20.10

20.10.12

2021-12-13

This release of Docker Engine contains changes in packaging only, and provides updates to the `docker scan` and `docker buildx` commands. Versions of `docker scan` before v0.11.0 are not able to detect the Log4j 2 CVE-2021-44228. We are shipping an updated version of `docker scan` in this release to help you scan your images for this vulnerability.

> **Note**
>
> The `docker scan` command on Linux is currently only supported on x86 platforms. We do not yet provide a package for other hardware architectures on Linux.

The `docker scan` feature is provided as a separate package and, depending on your upgrade or installation method, 'docker scan' may not be updated automatically to the latest version. Use the instructions below to update `docker scan` to the latest version. You can also use these instructions to install, or upgrade the `docker scan` package without upgrading the Docker Engine:

Log4j CVE-2021-44228

docker blog Products Developers Pricing About Us P

Apache Log4j 2 CVE-2021-44228

JUSTIN CORMACK
Dec 11 2021

Update: 13 December 2021

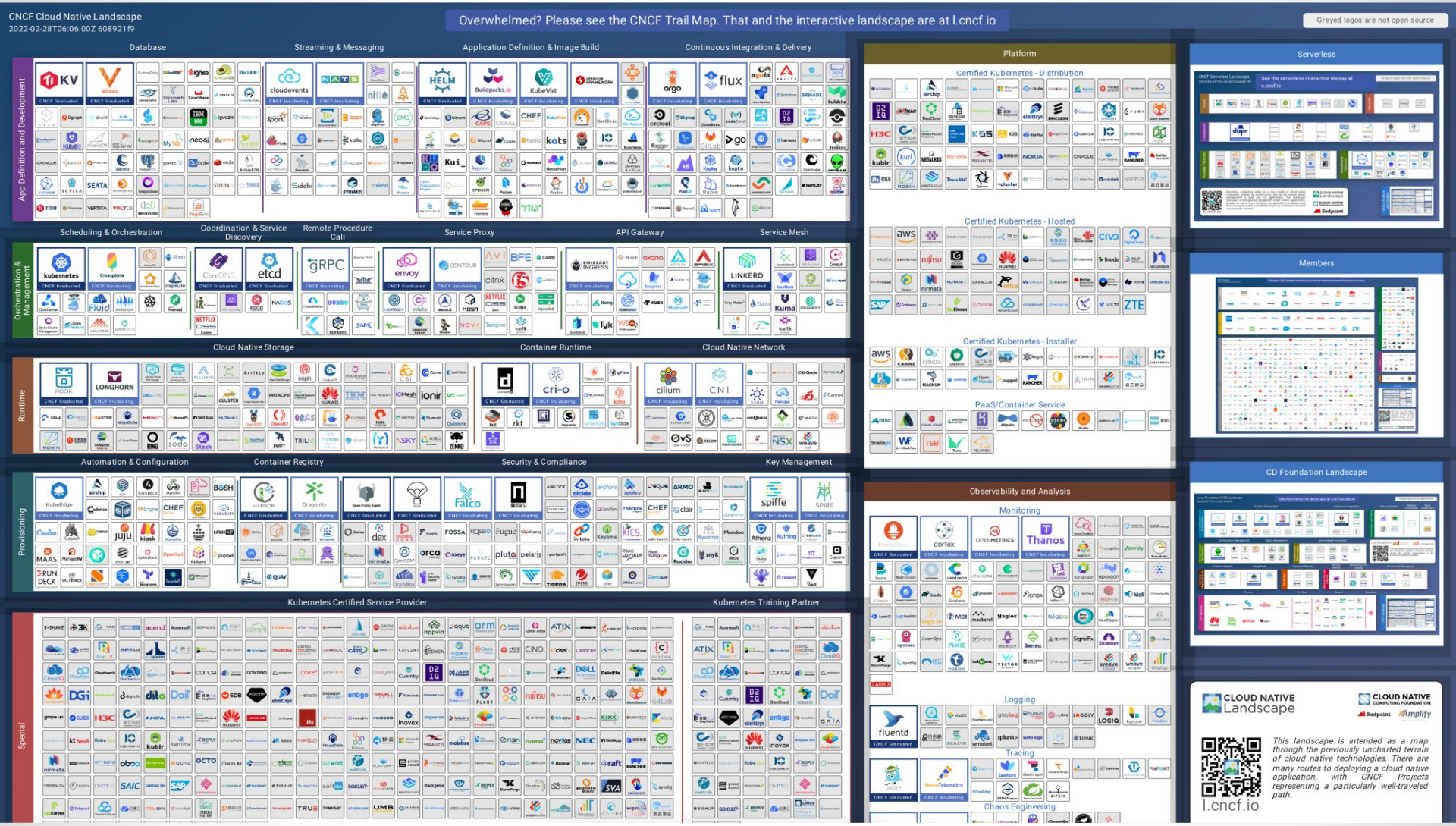
As an update to CVE-2021-44228, the fix made in version 2.15.0 was incomplete in certain non-default configurations. An additional issue was identified and is tracked with **CVE-2021-45046**. For a more complete fix to this vulnerability, it's recommended to update to Log4j2 **2.16.0**.

Original post below has now been updated:

Ref:https://www.docker.com/blog/apache-log4j-2-cve-2021-44228/

docker

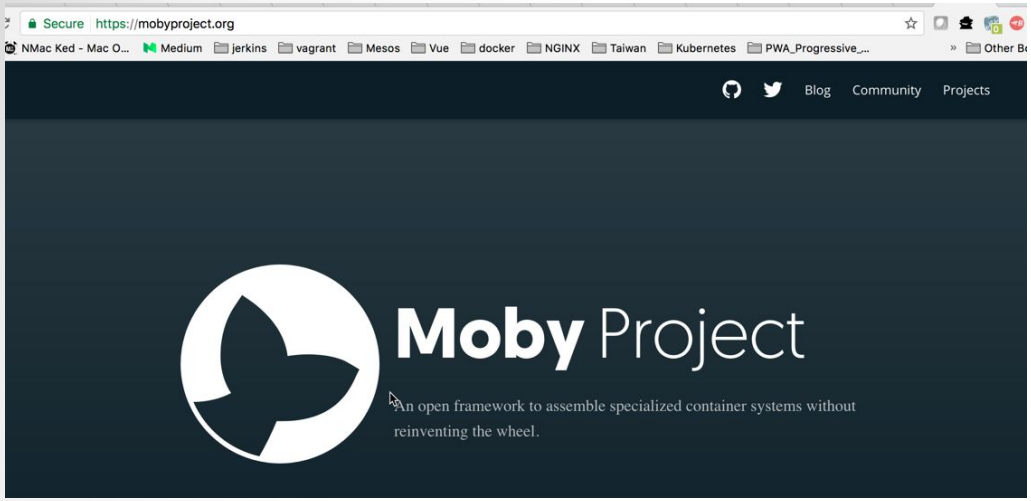
Cloud Native Computing Foundation

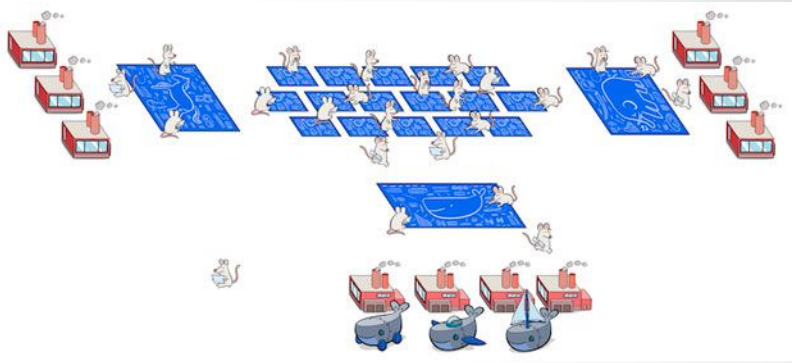


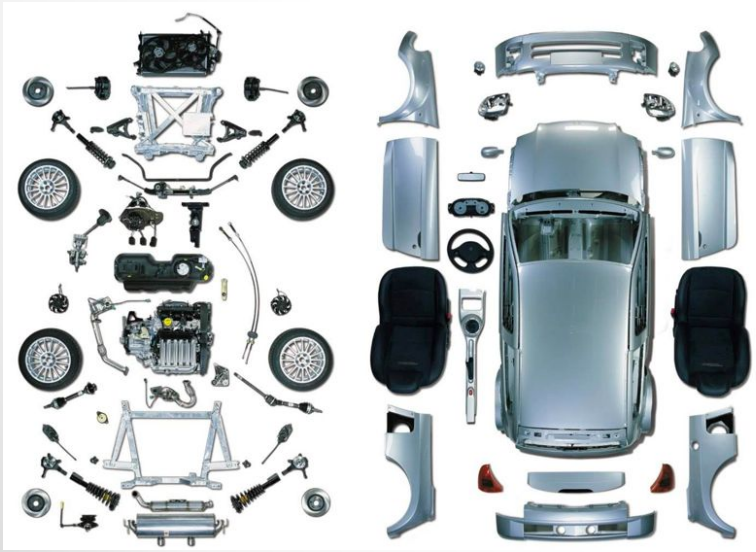
Kubernetes: Production Workload Orchestration

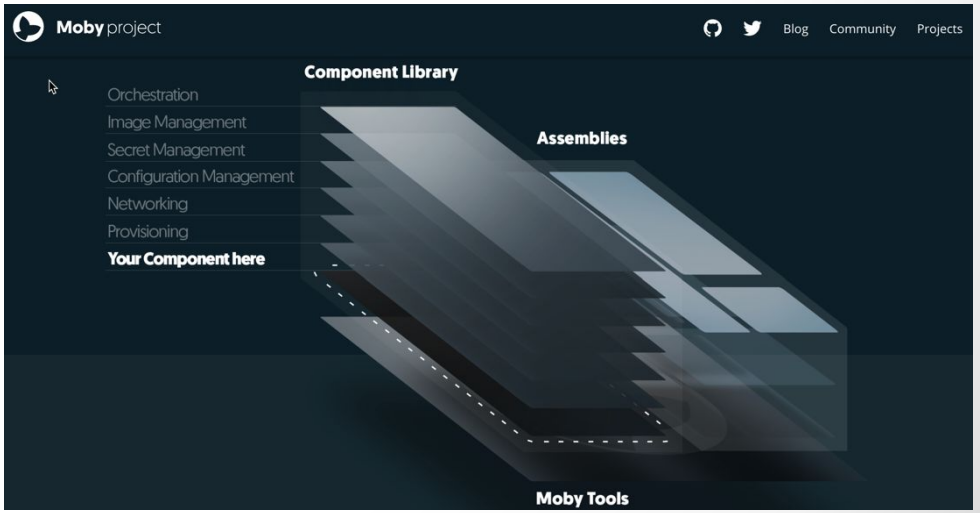
docker

What's New









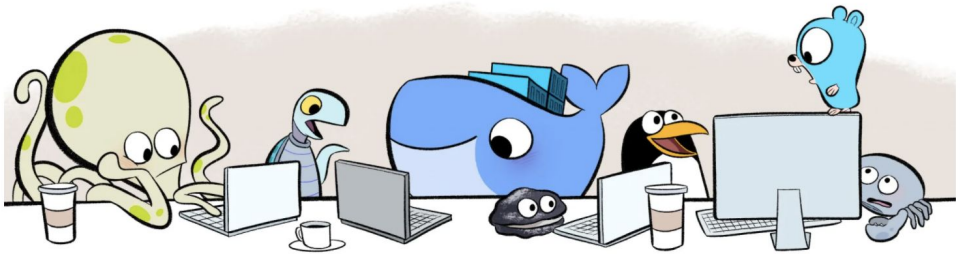
Docker: The Next-Gen of Virtualization

docker

What's New

Docker is Updating and Extending Our Product Subscriptions

SCOTT JOHNSTON
Aug 31 2021

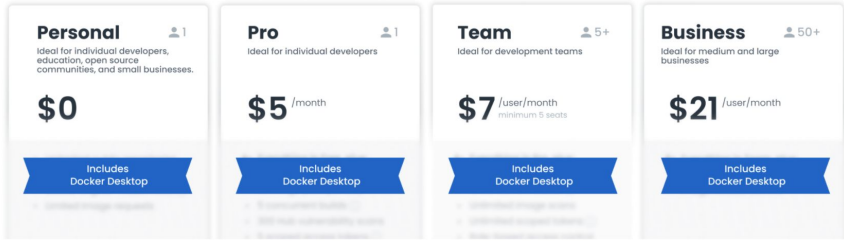


Docker is used by millions of developers to build, share, and run any app, anywhere, and 55% of professional developers use Docker every day at work. In these work environments, the increase in outside attacks on software supply chains is accelerating developer demand for Docker's trusted content, including Docker Official Images and Docker Verified Publisher images. Finally, the rapid global growth in developers – to an estimated 45 million by 2030 – pushes us to scale sustainably so we may continue to provide an innovative, free Docker experience that developers love.

To meet these challenges, today we're announcing updates and extensions to our product subscriptions: **Personal, Pro, Team, and Business.** These updated product subscriptions provide the productivity and collaboration developers rely on with the scale, security, and trusted content businesses require, and do so in a manner sustainable for Docker.

What you need to know:

- We're introducing a **new** product subscription, Docker Business, for organizations using Docker at scale for application development and require features like secure software supply chain management, single sign-on (SSO), container registry access controls, and more.
- Our Docker Subscription Service Agreement includes a change to the terms for **Docker Desktop**:
  - Docker Desktop **remains free** for small businesses (fewer than 250 employees AND less than $10 million in annual revenue), personal use, education, and non-commercial open source projects.
  - It requires a paid subscription (**Pro**, **Team** or **Business**), starting at $5 per user per month, for professional use in larger businesses. You may directly purchase here, or share this post and our solution brief with your manager.
  - While the effective date of these terms is August 31, 2021, there is a grace period until January 31, 2022 for those that require a paid subscription to use Docker Desktop.
- Docker Pro, Docker Team, and Docker Business subscriptions **include** commercial use of Docker Desktop.
- The **existing** Docker Free subscription has been renamed Docker Personal.
- **No changes** to Docker Engine or any upstream open source Docker or Moby project.
- *Check out our FAQ for more information.*



docker

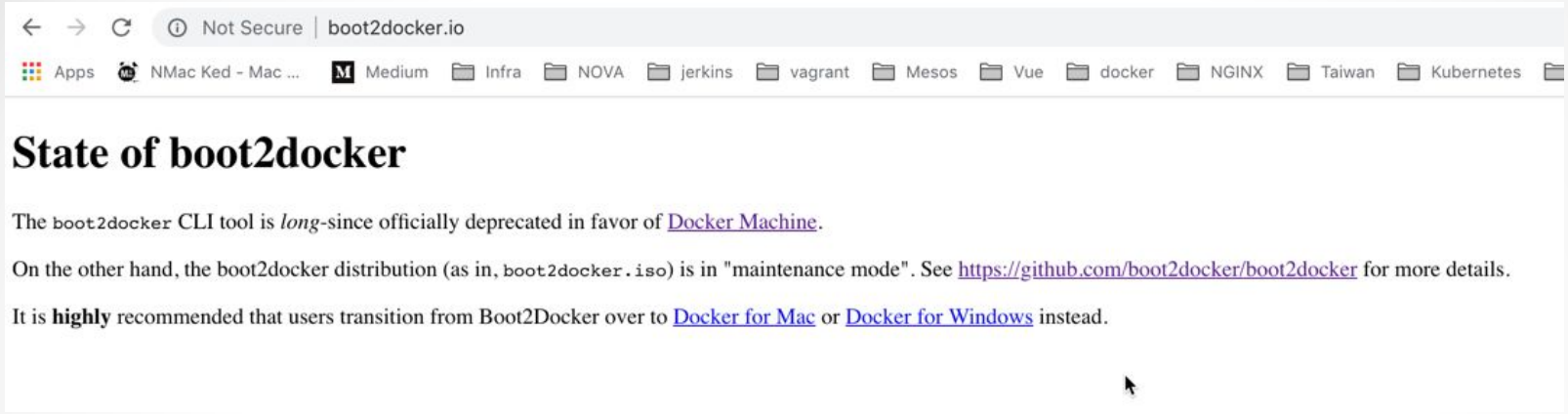
Docker Machine

• • •

Platform of Docker

- Docker Toolbox (Obsolete) for Desktop (Docker CE)
  - Install docker toolbox on machine will Create VM (Oracle VirtualBox)
  - Dockertoolbox for MAC
  - Dockertoolbox for Windows (7,8,10)
- Docker Native for Desktop (Docker Personal, Pro, Team Business)
  - 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 and upper
  - Docker for Ubuntu,Debian
  - Docker for CentOS
  - Docker for Red Hat Enterprise
  - Docker for Fedora

Linux vs Windows vs MAC OS



docker

Docker-machine CLI

- ตรวจสอบสถานะ docker-machine

```
docker-machine ls
```

- สั่งเริ่มการทำงานของ docker-machine

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

- สั่งหยุดการทำงานของ docker-machine

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

- สั่งสร้าง docker-machine เครื่องใหม่

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


Workshop: Pull Image

ทำการ download image ubuntu ลงมาที่เครื่อง boot2docker โดยใช้คำสั่งดังนี้

```
docker image pull labdocker/alpine:3.13.6
```

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

```
docker image pull labdocker/cadvisor:0.40.0
```

Compare with Ubuntu image

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

```
ubuntu@ip-10-0-1-90:~$ docker image ls
REPOSITORY              TAG       IMAGE ID       CREATED          SIZE
labdocker/alpineweb     latest    75f4bf61fa65   51 minutes ago   62.1MB
labdocker/alpine        3.13.6    12adea71a33b   2 days ago       5.61MB
labdocker/ubuntu        latest    fb52e22af1b0   3 days ago       72.8MB
labdocker/cadvisor      0.40.0    1ed215a61956   8 weeks ago      86.9MB
ubuntu@ip-10-0-1-90:~$
```


Workshop: Pull Image

ทำการตรวจสอบประวัติของ image ที่ใช้งานผ่านคำสั่ง docker image history

docker image history labdocker/alpine:3.13.6

docker image history labdocker/alpineweb:latest

```
ubuntu@ip-10-0-1-90:~$ docker image history labdocker/alpine:3.13.6
IMAGE          CREATED      CREATED BY                                      SIZE      COMMENT
12adea71a33b   2 days ago   /bin/sh -c #(nop)  CMD ["/bin/sh"]              0B
<missing>      2 days ago   /bin/sh -c #(nop) ADD file:ecdfb91a737d6c292…   5.61MB
[ubuntu@ip-10-0-1-90:~$     docker image history labdocker/alpineweb:latest
IMAGE          CREATED          CREATED BY                                      SIZE      COMMENT
75f4bf61fa65   53 minutes ago   /bin/sh -c #(nop)  EXPOSE 3000                  0B
<missing>      53 minutes ago   /bin/sh -c #(nop)  HEALTHCHECK &{["CMD-SHELL…   0B
<missing>      53 minutes ago   /bin/sh -c #(nop)  ENTRYPOINT ["node" "hello…   0B
<missing>      53 minutes ago   /bin/sh -c #(nop) WORKDIR /nodejs               0B
<missing>      53 minutes ago   /bin/sh -c #(nop) COPY file:7a1f27eb22f7a835…   263B
<missing>      53 minutes ago   /bin/sh -c mkdir /nodejs                        0B
<missing>      53 minutes ago   /bin/sh -c apk update &&     apk add nodejs …   56.5MB
<missing>      2 hours ago      /bin/sh -c #(nop)  ENV NODE_VERSION=v14.15.4…   0B
<missing>      2 hours ago      /bin/sh -c #(nop)  LABEL Description=NodeJS/…   0B
<missing>      2 hours ago      /bin/sh -c #(nop)  LABEL maintainer=Praparn …   0B
<missing>      2 days ago       /bin/sh -c #(nop)  CMD ["/bin/sh"]              0B
<missing>      2 days ago       /bin/sh -c #(nop) ADD file:ecdfb91a737d6c292…   5.61MB
ubuntu@ip-10-0-1-90:~$
```


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 builder	Manage builds
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 context	Manage contexts
docker cp	Copy files/folders between a container and the local filesystem
docker create	Create a new container
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

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

General Command in Docker

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 manifest	Manage Docker image manifests and manifest lists
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

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

General Command in Docker

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
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
docker unpause	Unpause all processes within one or more containers

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

General Command in Docker

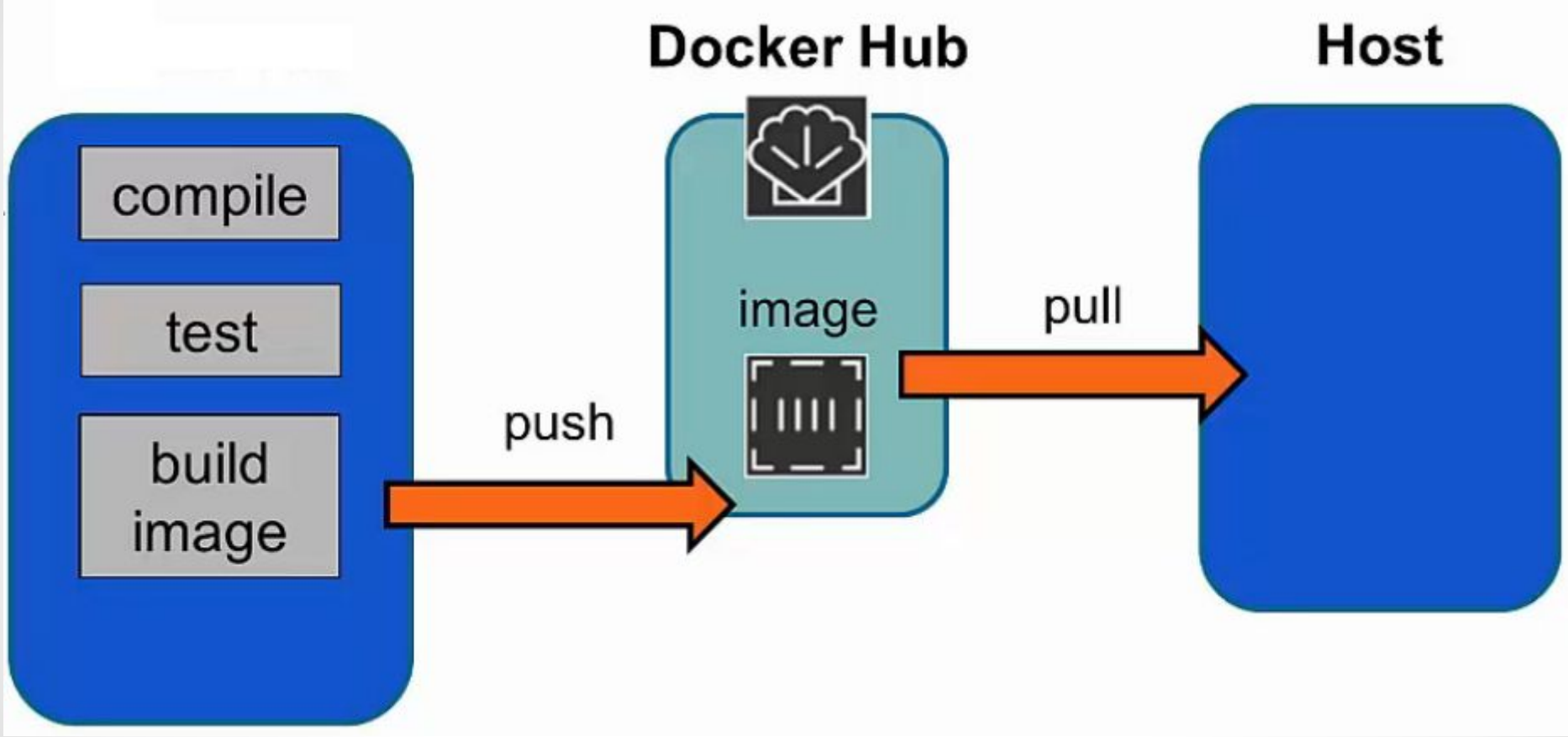
docker update	Update configuration of one or more containers
docker version	Show the Docker version information
docker volume	Manage volumes
docker wait	Block until one or more containers stop, then print their exit codes

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

Image, Repository and Container

• • •

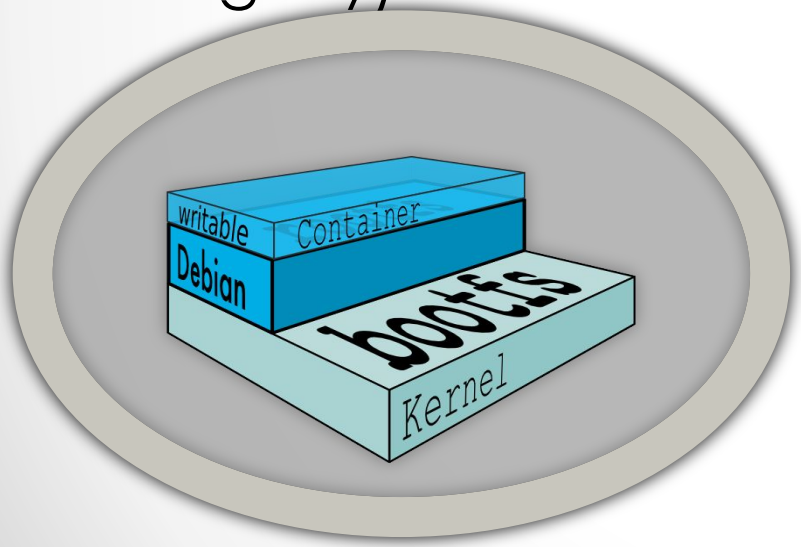
Docker Deployment

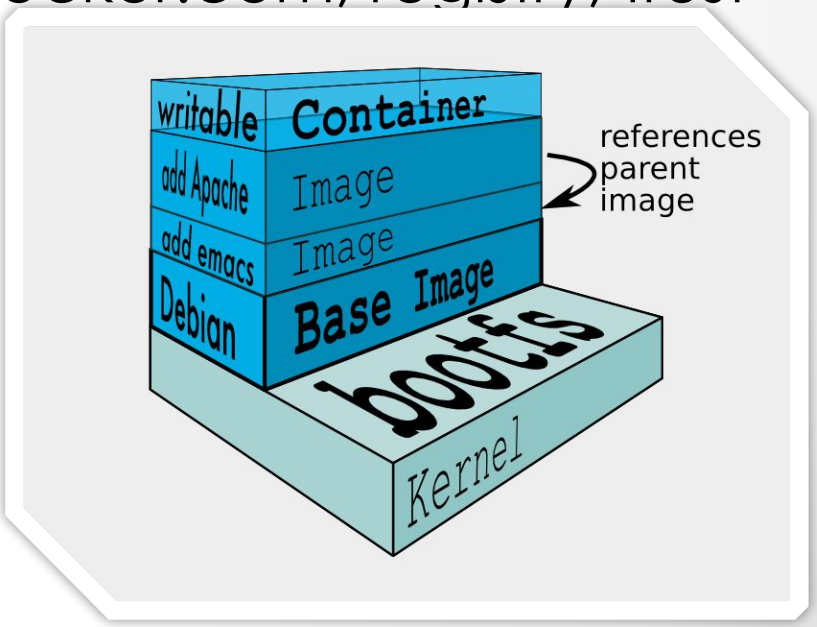


docker

Docker Image

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





docker

Docker Image

- เรียกดู image ที่อยู่ภายในเครื่อง

docker images /docker image ls

```
ubuntu@ip-10-0-1-90:~$ docker image ls
REPOSITORY                TAG                 IMAGE ID       CREATED          SIZE
labdocker/linenotify      bot_v2              7f073d0abe25   8 minutes ago    86.4MB
labdocker/alpineweb       latest              75f4bf61fa65   54 minutes ago   62.1MB
labdocker/alpine          3.13.6              12adea71a33b   2 days ago       5.61MB
labdocker/mariadb         10.5.12             b7b798989225   3 days ago       407MB
labdocker/ubuntu          latest              fb52e22af1b0   3 days ago       72.8MB
labdocker/cadvisor        0.40.0              1ed215a61956   8 weeks ago      86.9MB
labdocker/cluster         webservicelite      837c8f41c918   4 years ago      82.6MB
labdocker/alpinepython    2.7-onbuild         0f4df961b16b   4 years ago      475MB
ubuntu@ip-10-0-1-90:~$
```

- tag image ใหม่เพื่อให้ตรงตามความต้องการในการใช้งาน

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

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

Docker Image

- ตรวจสอบรายละเอียดของ image

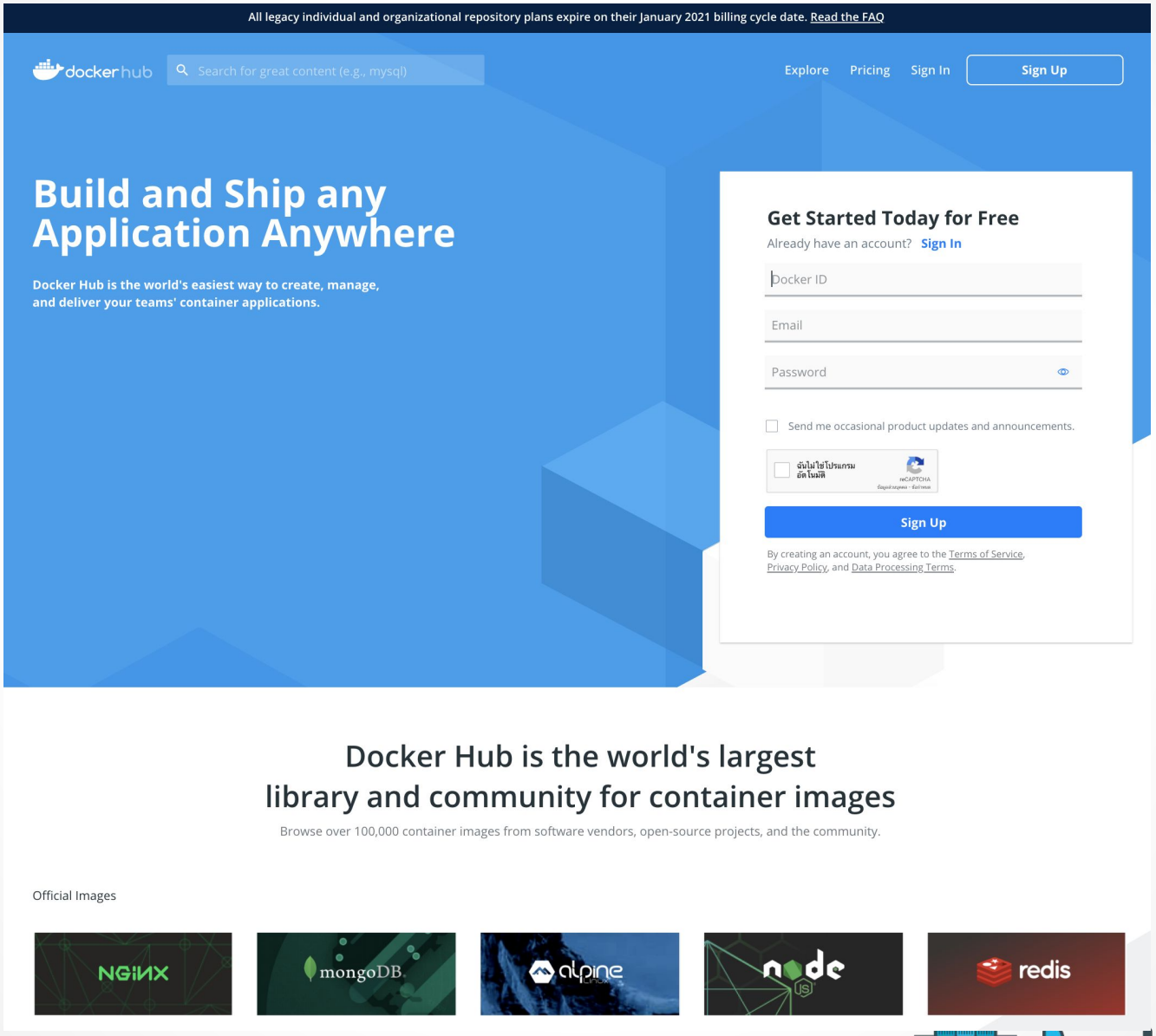
docker images history <image id/image name>

```
ubuntu@ip-10-0-1-90:~$ docker image history labdocker/alpine:3.13.6
IMAGE          CREATED      CREATED BY                                      SIZE      COMMENT
12adea71a33b   2 days ago   /bin/sh -c #(nop)  CMD ["/bin/sh"]              0B
<missing>      2 days ago   /bin/sh -c #(nop) ADD file:ecdfb91a737d6c292…   5.61MB
ubuntu@ip-10-0-1-90:~$     docker image history labdocker/alpineweb:latest
IMAGE          CREATED          CREATED BY                                      SIZE      COMMENT
75f4bf61fa65   53 minutes ago   /bin/sh -c #(nop)  EXPOSE 3000                  0B
<missing>      53 minutes ago   /bin/sh -c #(nop)  HEALTHCHECK &{["CMD-SHELL…   0B
<missing>      53 minutes ago   /bin/sh -c #(nop)  ENTRYPOINT ["node" "hello…   0B
<missing>      53 minutes ago   /bin/sh -c #(nop) WORKDIR /nodejs               0B
<missing>      53 minutes ago   /bin/sh -c #(nop) COPY file:7a1f27eb22f7a835…   263B
<missing>      53 minutes ago   /bin/sh -c mkdir /nodejs                        0B
<missing>      53 minutes ago   /bin/sh -c apk update &&     apk add nodejs …   56.5MB
<missing>      2 hours ago      /bin/sh -c #(nop)  ENV NODE_VERSION=v14.15.4…   0B
<missing>      2 hours ago      /bin/sh -c #(nop)  LABEL Description=NodeJS/…   0B
<missing>      2 hours ago      /bin/sh -c #(nop)  LABEL maintainer=Praparn …   0B
<missing>      2 days ago       /bin/sh -c #(nop)  CMD ["/bin/sh"]              0B
<missing>      2 days ago       /bin/sh -c #(nop) ADD file:ecdfb91a737d6c292…   5.61MB
ubuntu@ip-10-0-1-90:~$
```


Repository (Registry)

- component ในการจัดเก็บ docker (image) ต่างๆรวมไปที่ศูนย์กลางเพื่อให้ docker engine บนเครื่องต่างๆมาเรียก image ไปใช้ทำงาน
- รองรับการเก็บ version ของ image อย่างเป็นระบบ
- สามารถให้ข้อมูลหรือคู่มือแนะนำการใช้งานกับผู้ 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)



docker

Repository (Registry)

docker

Why Docker? Products Developers Pricing Company Sign In Get Started

Understanding Docker Hub Rate Limiting

Learn More

On November 20, 2020, rate limits anonymous and free authenticated use of Docker Hub went into effect. Anonymous and Free Docker Hub users are limited to 100 and 200 container image pull requests per six hours.

If you are affected by these changes you will receive this error message:

```
ERROR: toomanyrequests: Too Many Requests.
```

OR

```
You have reached your pull rate limit. You may increase the limit by authenticating and upgrading: https://www.docker.com/increase-rate-limits.
```

To increase your pull rate limits you can upgrade your account to a Docker Pro or Team subscription.

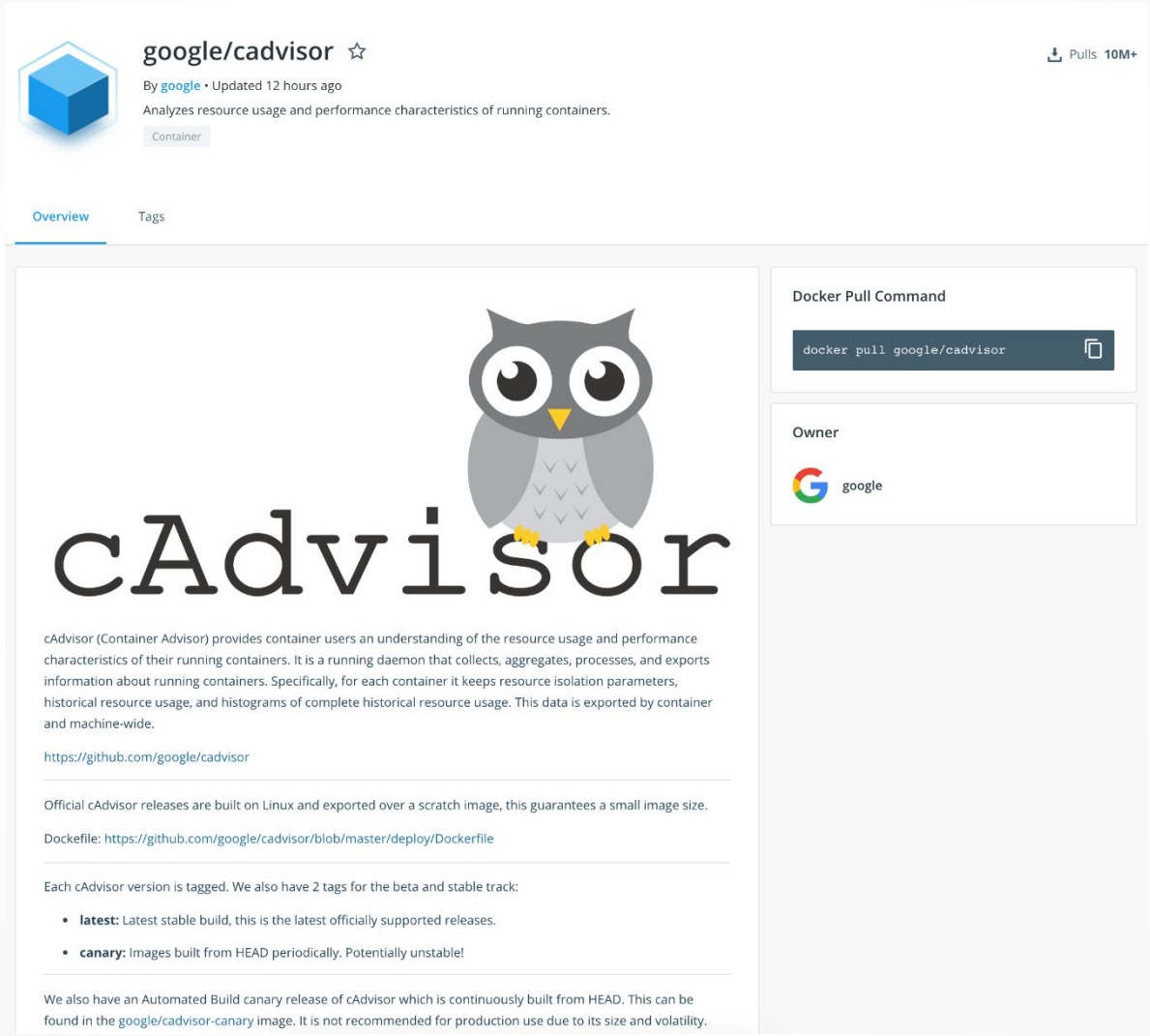
The rate limits of 100 container image requests per six hours for anonymous usage, and 200 container image requests per six hours for free Docker accounts are now in effect. Image requests exceeding these limits will be denied until the six hour window elapses.

NOTE: Docker Pro and Docker Team accounts continue to have unlimited access to pull container images from Docker Hub.

Related Resources

- Rate Limiting Documentation
- Resource Consumption Update FAQ
- Blog post on Resource Consumption and Image Retention Policy Updates
- Docker Subscription Pricing Page
- Pricing FAQ
- Guidance for Setting Up a Mirror
- Understanding Inner Loop Development and Pull Rates
- Docker Partner Information Request for Rate Limiting Exclusions

Repository (Registry)



docker

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 xxxxxxx –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*

Repository (Registry)

- Save image ออกมาเป็น tar file สำหรับ offline machine

```
docker image save -o xxx.tar image name
```

```
Ex: docker image save -o test.tar
labdocker/alpinelab:1.0
```

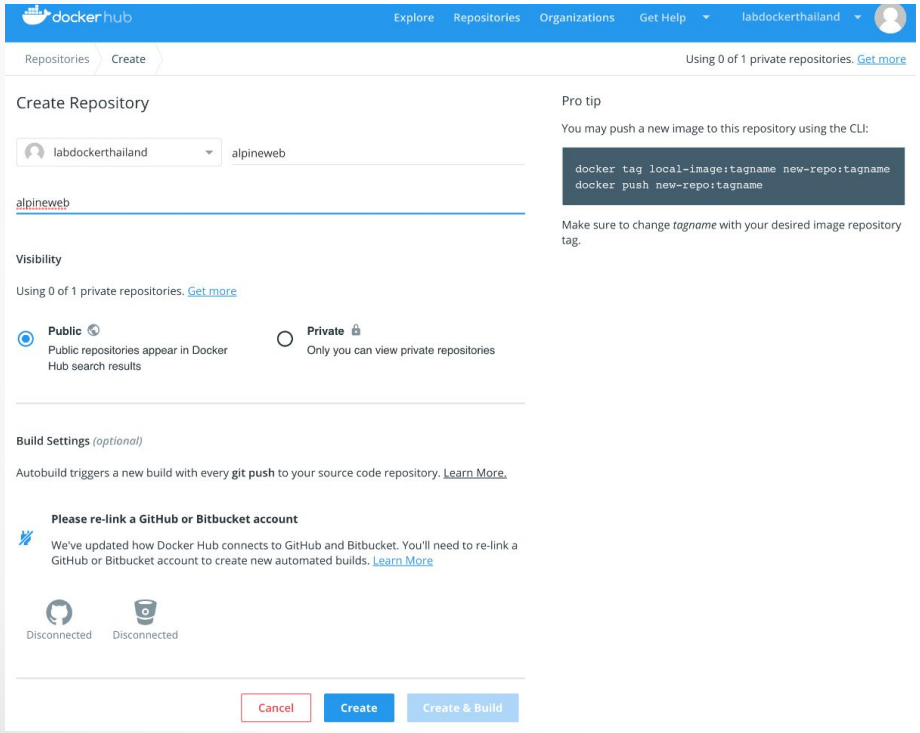
- Load image จาก tar file สำหรับ offline machine

```
docker image load -i xxx.tar
```

```
Ex: docker image load -i test.tar
```


Workshop: 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: Build & Push Image

- ตรวจสอบ image id ด้วยคำสั่ง

```
docker image ls
```

- tag image ใหม่เป็น <accountname>/alpineweb:latest

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

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

```
docker login –u <xxxx>
```

- Push image alpine ที่ tag ไว้ขึ้น repository

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


Workshop: Build & Push Image

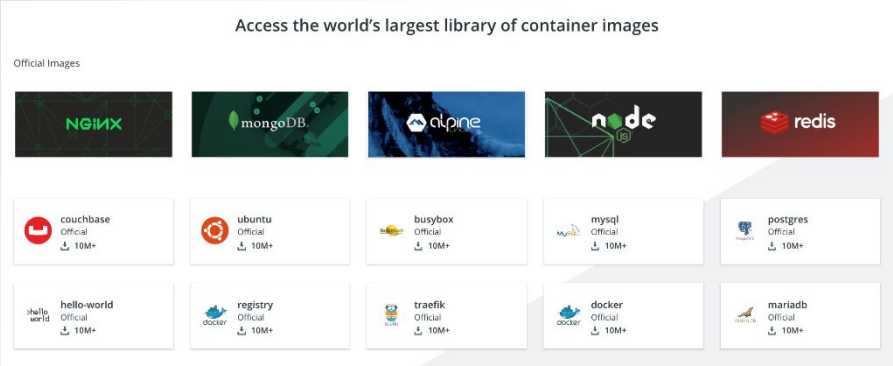
- ทดสอบทำการ save image ออกมาเป็นไฟล์ .tar

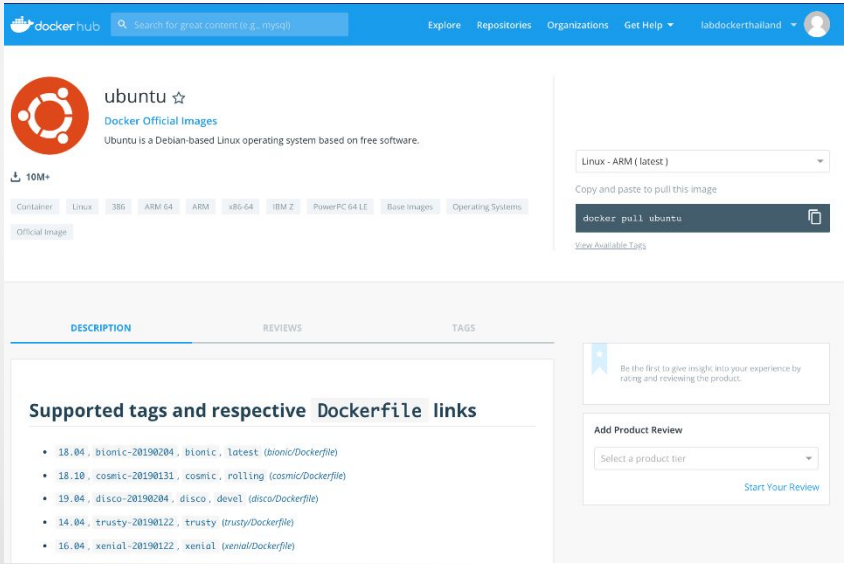
```
docker image save -o ~/alpineweb.tar
<account>/alpineweb:latest
```

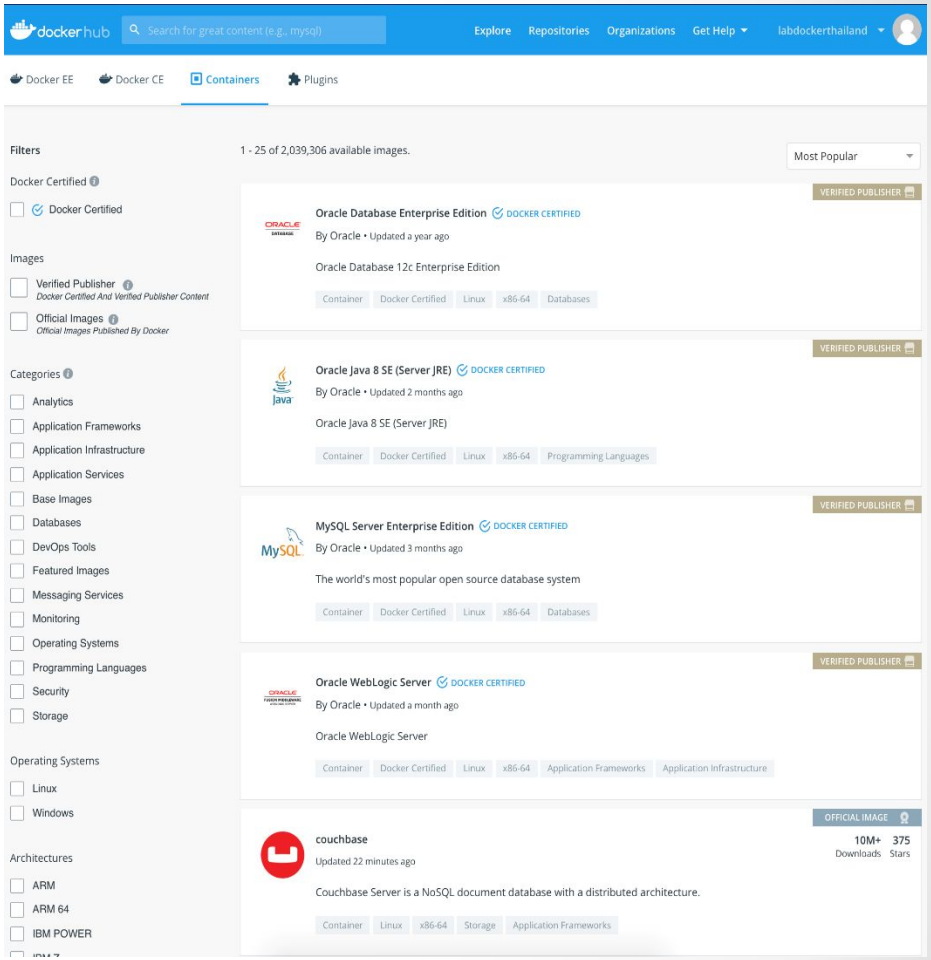
- ดำเนินการลบและ load image จากไฟล์ .tar

```
docker image rm <account>/alpineweb:latest
docker image load -i ~/alpineweb.tar
```


Other service in hub.docker.com

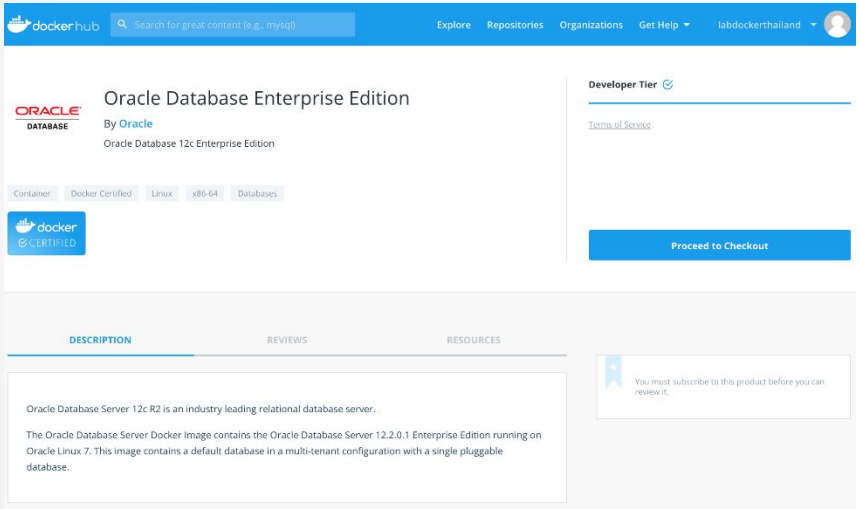


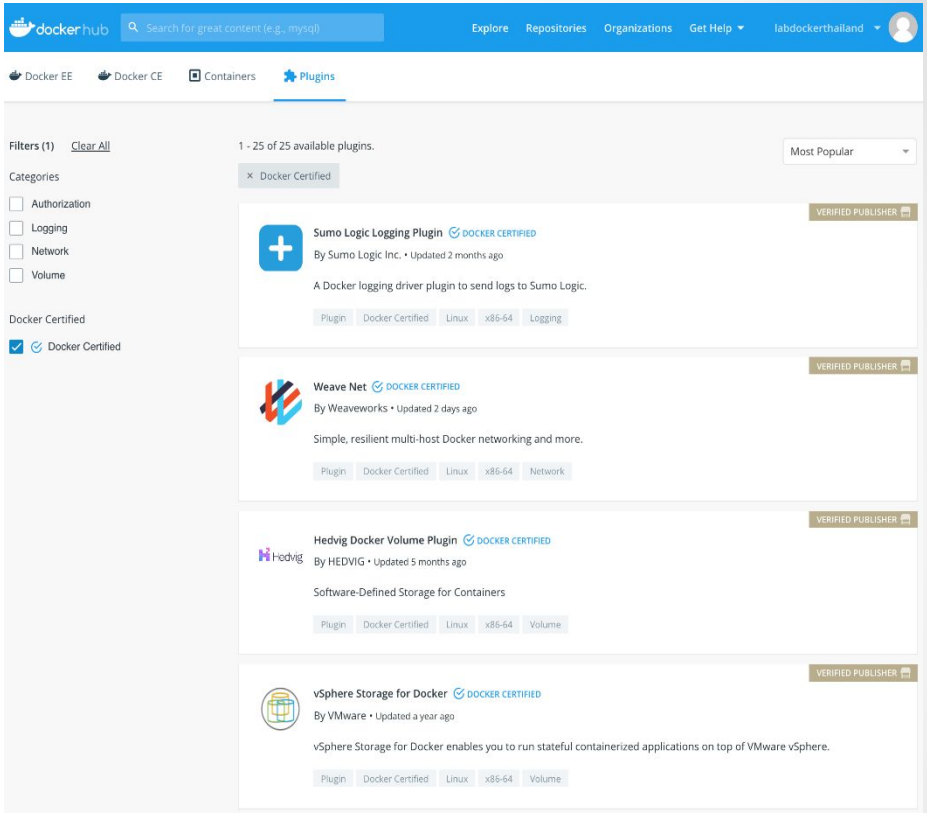




Docker: The Next-Gen of Virtualization

Other service in hub.docker.com





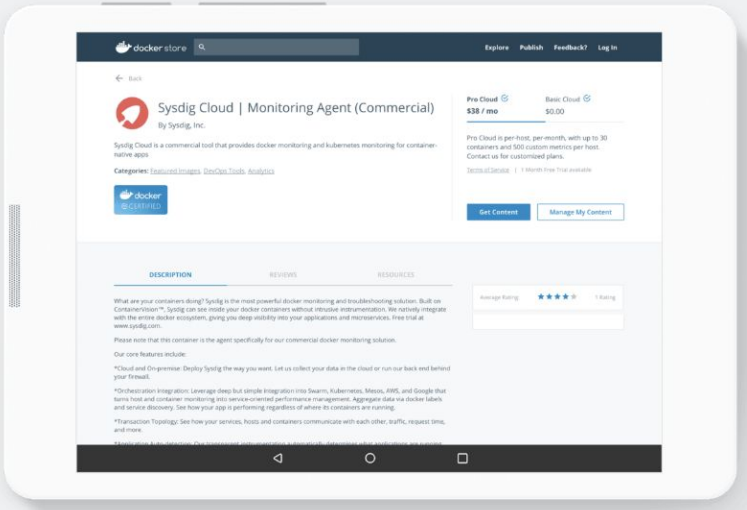
Other service in hub.docker.com

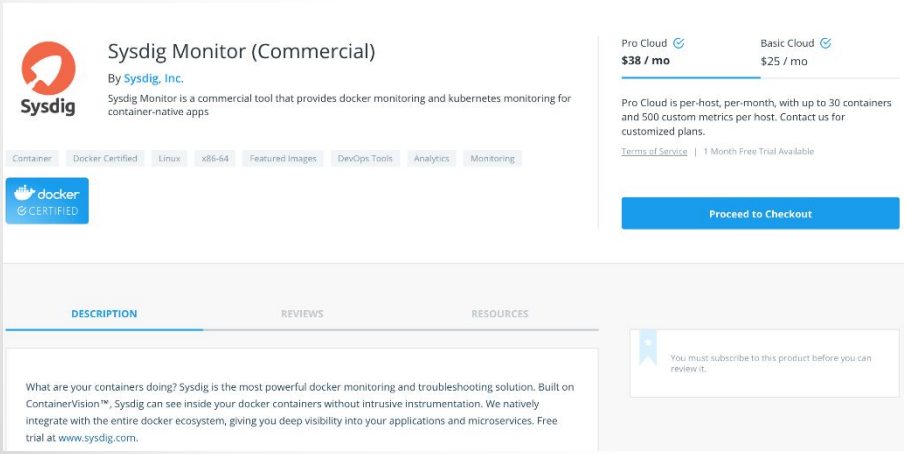
PUBLISHER PROGRAM

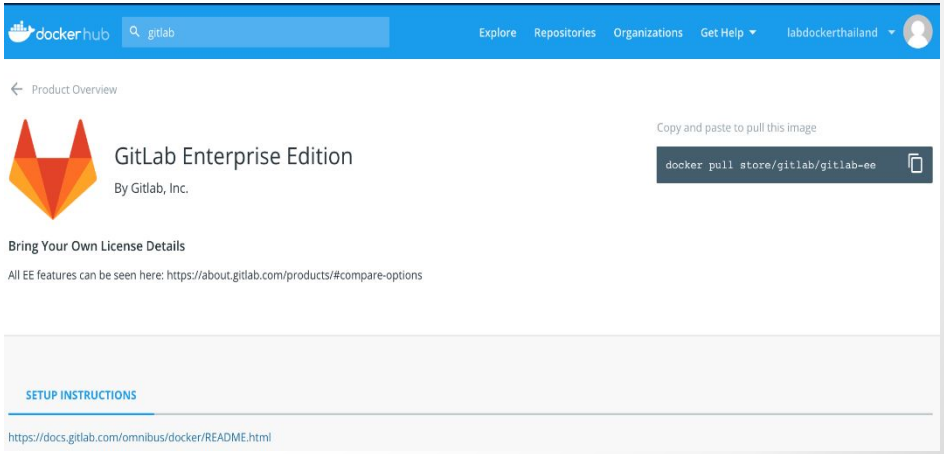
Deliver your business through Docker Hub

Package and publish apps and plugins as containers in Docker Hub for easy download and deployment by millions of Docker users worldwide.

Apply To Publish | Learn More

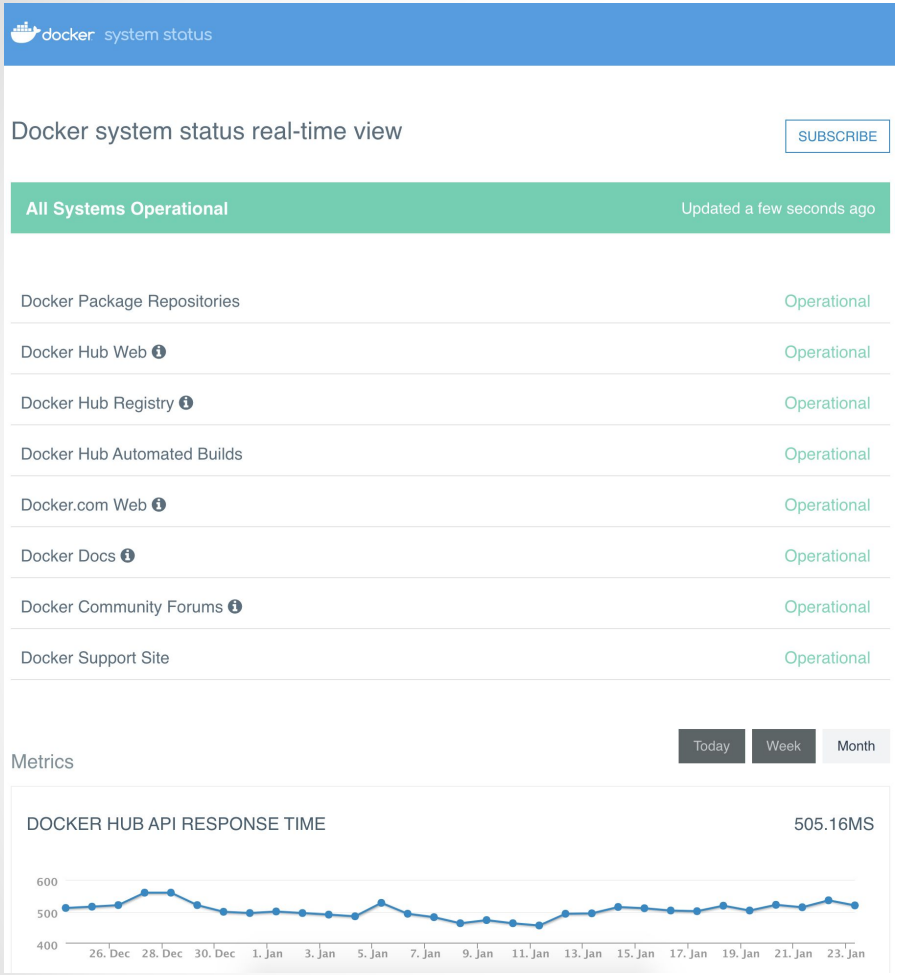


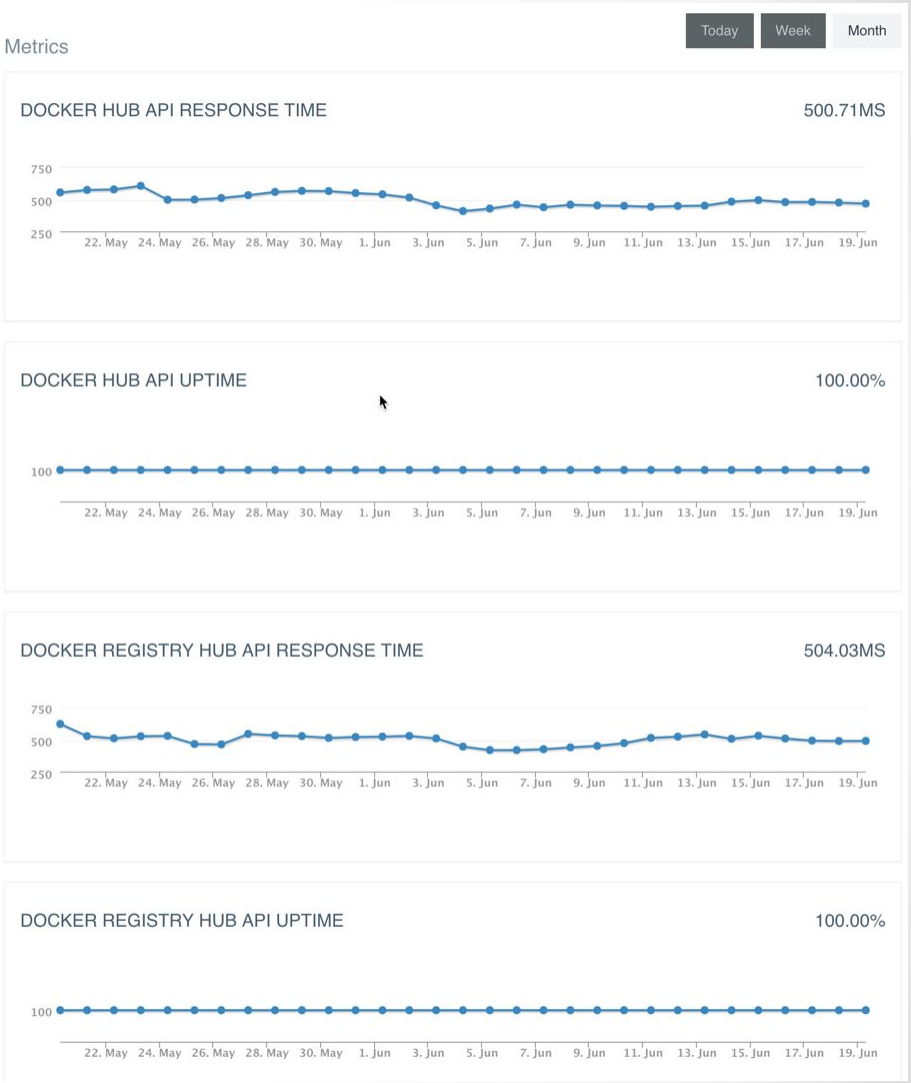




docker

Status.docker.com

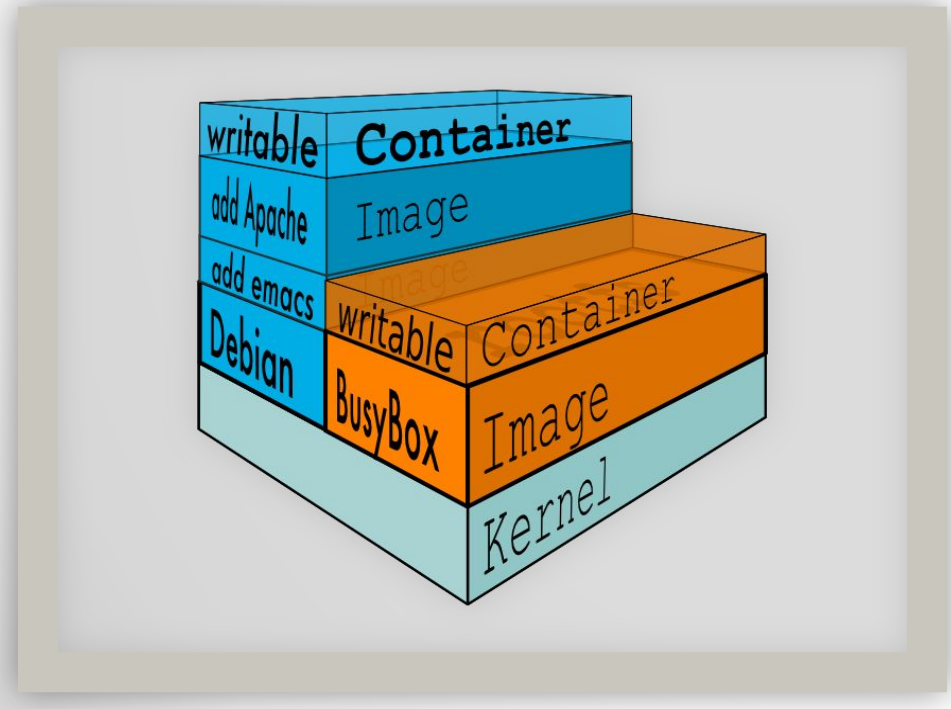


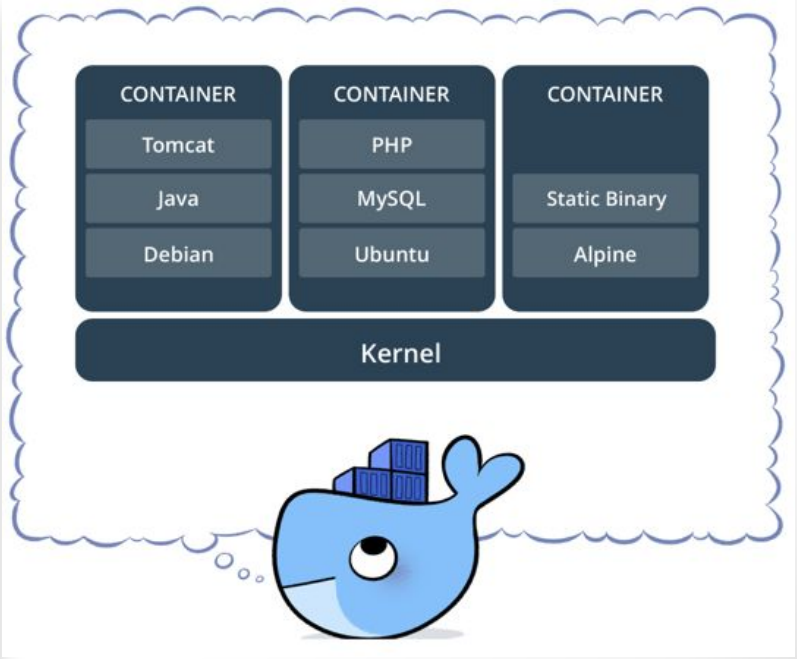


https://status.docker.com/

Docker Container

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



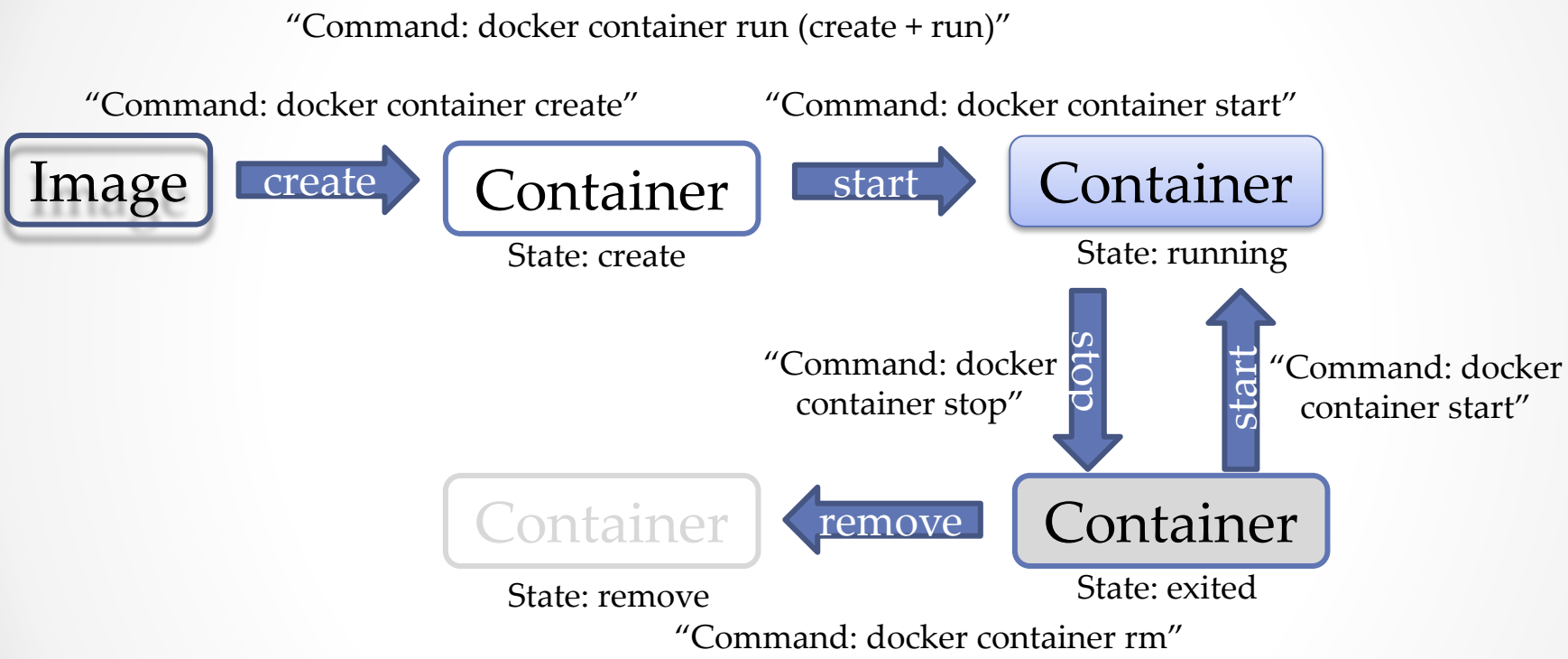


Docker: The Next-Gen of Virtualization

docker

Docker Container

- Container State:



docker

Docker Container

- Container Different (Compare with Image):

docker container diff <container name>

```
ubuntu@ip-10-0-1-90:~$ docker container diff nodejs
C /nodejs
A /nodejs/test.js
C /root
A /root/.ash_history
ubuntu@ip-10-0-1-90:~$ docker container ls
CONTAINER ID   IMAGE                      COMMAND          CREATED          STATUS                    PORTS                                       NAMES
4b04b8ba10e0   labdocker/alpineweb:latest   "node hello.js"   48 seconds ago   Up 47 seconds (healthy)   0.0.0.0:3000->3000/tcp, :::3000->3000/tcp   nodejs
ubuntu@ip-10-0-1-90:~$
```

docker

Docker Container

- Container Performance/Port:

docker container stats <container name>

docker container top <container name>

docker container port <container name>

```
CONTAINER ID        NAME                CPU %               MEM USAGE / LIMIT     MEM %               NET I/O             BLOCK I/O           PIDS
115bb4a5b92b        nodejs              0.00%               8.375MiB / 1.91GiB    0.43%               5.9kB / 2.3kB       36.9kB / 0B         6
^C
ubuntu@ip-10-0-1-58:~$ docker container top nodejs
UID                 PID                 PPID                C                   STIME               TTY                 TIME                CMD
root                3069                3043                0                   14:15               pts/0               00:00:00            node
hello.js
ubuntu@ip-10-0-1-58:~$ docker container port nodejs
3000/tcp -> 0.0.0.0:3000
ubuntu@ip-10-0-1-58:~$
```


Docker Container

- Container change name on-the-fly:

```
docker container rename <old name> <new name>
```

```
ubuntu@ip-10-0-1-90:~$ docker container ls
CONTAINER ID   IMAGE                          COMMAND           CREATED         STATUS                     PORTS                                       NAMES
4b04b8ba10e0   labdocker/alpineweb:latest   "node hello.js"   3 minutes ago   Up 49 seconds (healthy)   0.0.0.0:3000->3000/tcp, :::3000->3000/tcp   nodejs
ubuntu@ip-10-0-1-90:~$ docker container rename nodejs nodejsnew
ubuntu@ip-10-0-1-90:~$ docker container ls
CONTAINER ID   IMAGE                          COMMAND           CREATED         STATUS                     PORTS                                       NAMES
4b04b8ba10e0   labdocker/alpineweb:latest   "node hello.js"   3 minutes ago   Up About a minute (healthy)   0.0.0.0:3000->3000/tcp, :::3000->3000/tcp   nodejsnew
ubuntu@ip-10-0-1-90:~$
```


Docker Container

- สั่ง run docker เพื่อสร้าง container จาก image file

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

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

- สั่ง exec command ไปบน container ที่รันอยู่

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

- เริ่ม/หยุด container

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

- ลบ container

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


Workshop: Run Container

- 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: Run Container

- DAEMON LINE BOT

ธนาคารแห่งประเทศไทย

ข้อมูลทั่วไปของ API

ประเภทของ API :	REST
ประเภทของข้อมูล :	JSON
ความปลอดภัย :	PUBLIC

ตำแหน่งที่อยู่ของ API

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

พารามิเตอร์ใน Header

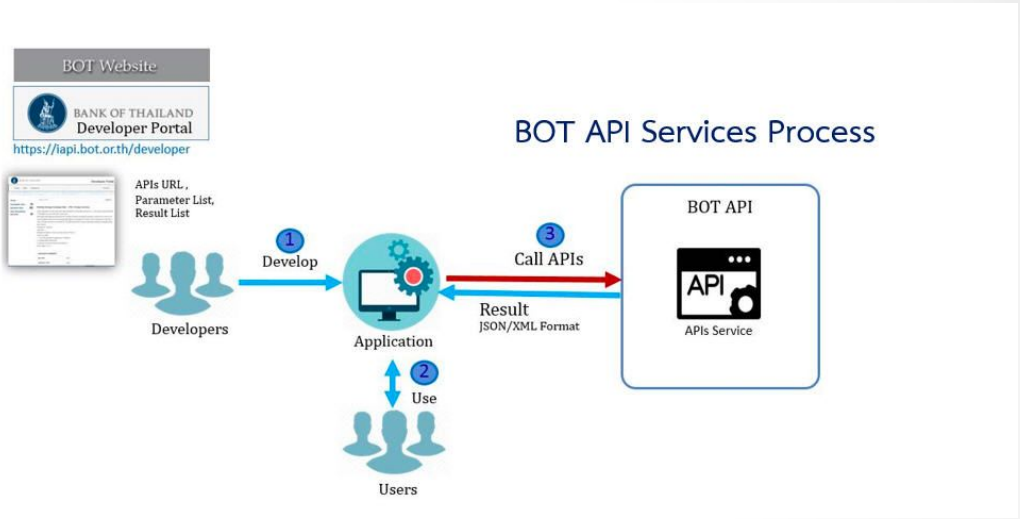
api-key : xk3h6ash-ahw5l3mch6hl3-2hhg6l4mng2346l34l6

ข้อมูลพารามิเตอร์สำหรับการใช้งาน API

ชื่อตัวแปร	ประเภท	คำอธิบาย
start_period	Datetime	วันที่เริ่มต้นงวดข้อมูล (YYYY-MM-DD) เช่น 2017-06-30
end_period	Datetime	วันที่สิ้นสุดงวดข้อมูล (YYYY-MM-DD) เช่น 2017-06-30

ข้อมูลพารามิเตอร์ที่จะได้รับ

ชื่อ	ประเภท	คำอธิบาย
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	แหล่งที่มาของข้อมูล
report_remark_eng	String	หมายเหตุ รายงานภาษาอังกฤษ
report_remark_th	String	หมายเหตุ รายงานภาษาไทย
last_updated	String	วันที่แก้ไขข้อมูลล่าสุด
period	String	วันที่งวดข้อมูล
rate	String	อัตรา

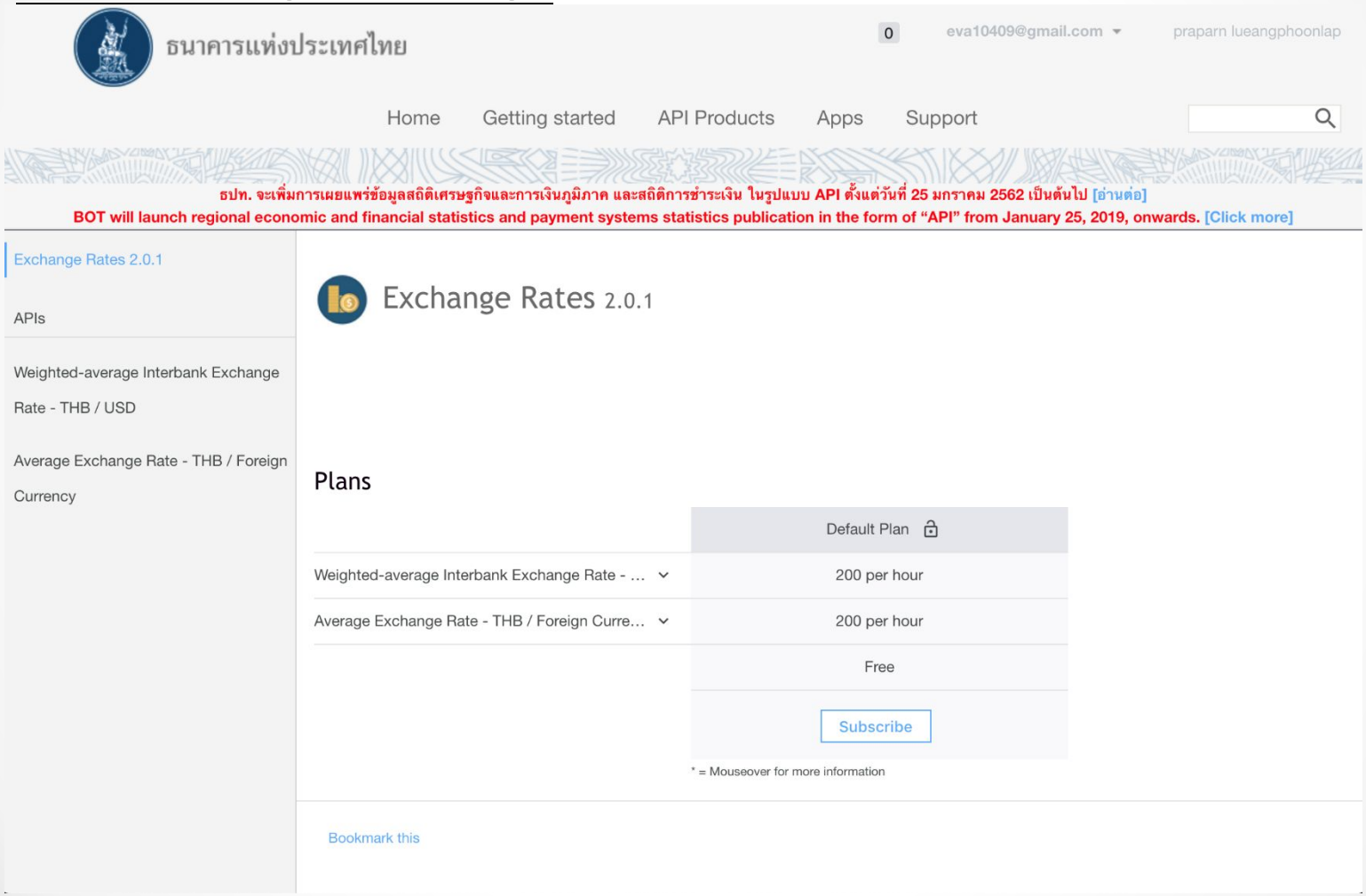


Docker: The Next-Gen of Virtualization

docker

Workshop: Run Container

- Ex 2: DAEMON LINE BOT



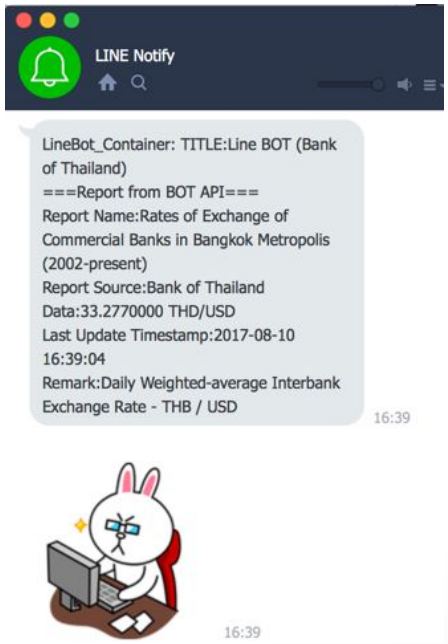
docker

Workshop: Run Container

- DAEMON LINE BOT

```
docker@labdocker:~$ docker container run --rm --name linebot -e "TITLE=Line BOT (Bank of Thailand)" -e "TOKEN=EIeqyillzkzpaCo1ROrebZTcGbIyzDZHpOjcdd0t6CX" 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 EIeqyillzkzpaCo1ROrebZTcGbIyzDZHpOjcdd0t6CX","content-length":466}}}
"{\"status\":200,\"message\":\"ok\"}"
docker@labdocker:~$
```

```
const request = require('request');
const qs = require('querystring');
const TITLE=process.env.TITLE;
const KEY =process.env.KEY; //Input API-KEY to Header
//const KEY="U9G1L457H6DCugT7VmBaEacbHV9RX0PySO05cYaGsm";
const URL =process.env.URL; //Input URL for Request
//const URL="https://iapi.bot.or.th/Stat/Stat-ReferenceRate/DAILY_REF_RATE_V1/?";
//const TOKEN = 'EIeqyillzkzpaCo1ROrebZTcGbIyzDZHpOjcdd0t6CX';
const TOKEN=process.env.TOKEN; //Input TOKEN LINE
//Set Date
var dateFormat = require('dateformat');
var daynow = new Date(); // Today!
daynow.setDate(daynow.getDate() - 1); // Yesterday!
var daynow=dateFormat(daynow, "yyyy-mm-dd");
var DAYSTARTPERIOD=daynow;
var DAYENDPERIOD=daynow;
//Set Date
var stickerPkg=2; //stickerPackageId
var stickerId=161; //stickerId
var Message="";
//Setup QueryString
var querystring = qs.stringify({
    start_period: DAYSTARTPERIOD,
    end_period: DAYENDPERIOD
});
//Setup QueryString
function callback(error, response, body) {
  //console.log(URL+querystring);
  //console.log('Response Code: '+ response.statusCode);
  //console.log(body)
  if (!error && response.statusCode == 200) {
    var info = JSON.parse(body);
```



docker

Workshop: Run Container

- DAEMON LINE BOT
- สั่งรัน container แบบ Daemon โดยตั้งชื่อว่า linebot เพื่อทำการดึงข้อมูลจาก API ของธนาคารแห่งประเทศไทยออกมาแล้วส่ง 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_v2
```


Workshop: Run Container

- 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: Run Container

- Detach Mode NODEJS
- สั่งหยุดเริ่ม container ด้วยคำสั่ง stop

```
docker container stop nodejs
```

```
docker container start nodejs
```

- ทำการตรวจสอบ container offline ด้วยคำสั่ง

```
docker container ls -a
```

- ทดสอบ Shell เข้าสู่ container ด้วยคำสั่ง

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

```
docker container attach nodejs
```


Workshop: Run Container

- Detach Mode NODEJS
- ทำการตรวจสอบ process ภายใน container

```
docker container top nodejs
```

```
docker container stats nodejs
```

- ทำการตรวจสอบความแตกต่างระหว่าง container กับ image ต้นฉบับ

```
docker container diff nodejs
```

- ตรวจสอบการ map port container

```
docker container port nodejs
```

- เปลี่ยนแปลงชื่อ container แบบ online

```
docker container rename nodejs nodejsnew
```


Workshop: Run Container

- 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/');
```

docker

Workshop: Run Container

- Detach Mode Python
- สั่งสร้าง container แบบ detach (daemon) โดยตั้งชื่อว่า pythonและ map network 5000:5000

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

- ทำการรัน

```
docker container ls –a
docker container start python
```

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

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

```
docker container attach python
```


Workshop: Run Container

- Detach Mode Python

```python
from flask import Flask
import os
import time
app = Flask(__name__)

@app.route('/')
def hello():
    return '<H1> Welcome Page from Container Python Lab </H1>Checkpoint Date/Time: ' + time.strftime("%c") +'\n'

if __name__ == "__main__":
    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)(CFS scheduler)

```
--cpu-period=100000 --cpu-quota=500000
```


CPU

- Default container จะมองเห็น CPU resource ทั้งหมดในเครื่องระหว่าง runtime และ share เวลาการทำงานให้ทุก container เท่ากัน

- NUMA (non-uniform memory access) architecture

```
--cpuset-mems="1,3"
```

- กำหนด cpu real-time runtime (ms) (nice system call)

```
--cap-add=sys_nice --cpu-rt-runtime=<value>
```

- กำหนด cpu run-time priority (0-99) (nice system call)

```
--cap-add=sys_nice --ulimit rtprio=xx
```


NVIDIA GPU

- Docker สามารถใช้งาน GPU ของ NVIDIA ในการนำมาประมวลผลได้โดยต้องทำการติดตั้ง nvidia-container-runtime เพิ่มเติมดังนี้ (Need restart)

```
apt-get install nvidia-container-runtime
which nvidia-container-runtime-hook
```

- Check gpu available for use in system by command

```
docker run -it --rm --gpus all ubuntu nvidia-smi
```

- Run docker container with gpu:

```
docker run -it --rm --gpus device=xxxx ubuntu nvidia-smi
```

Ref:https://github.com/NVIDIA/nvidia-docker

Workshop: CPU Configure

- Download cAdvisor

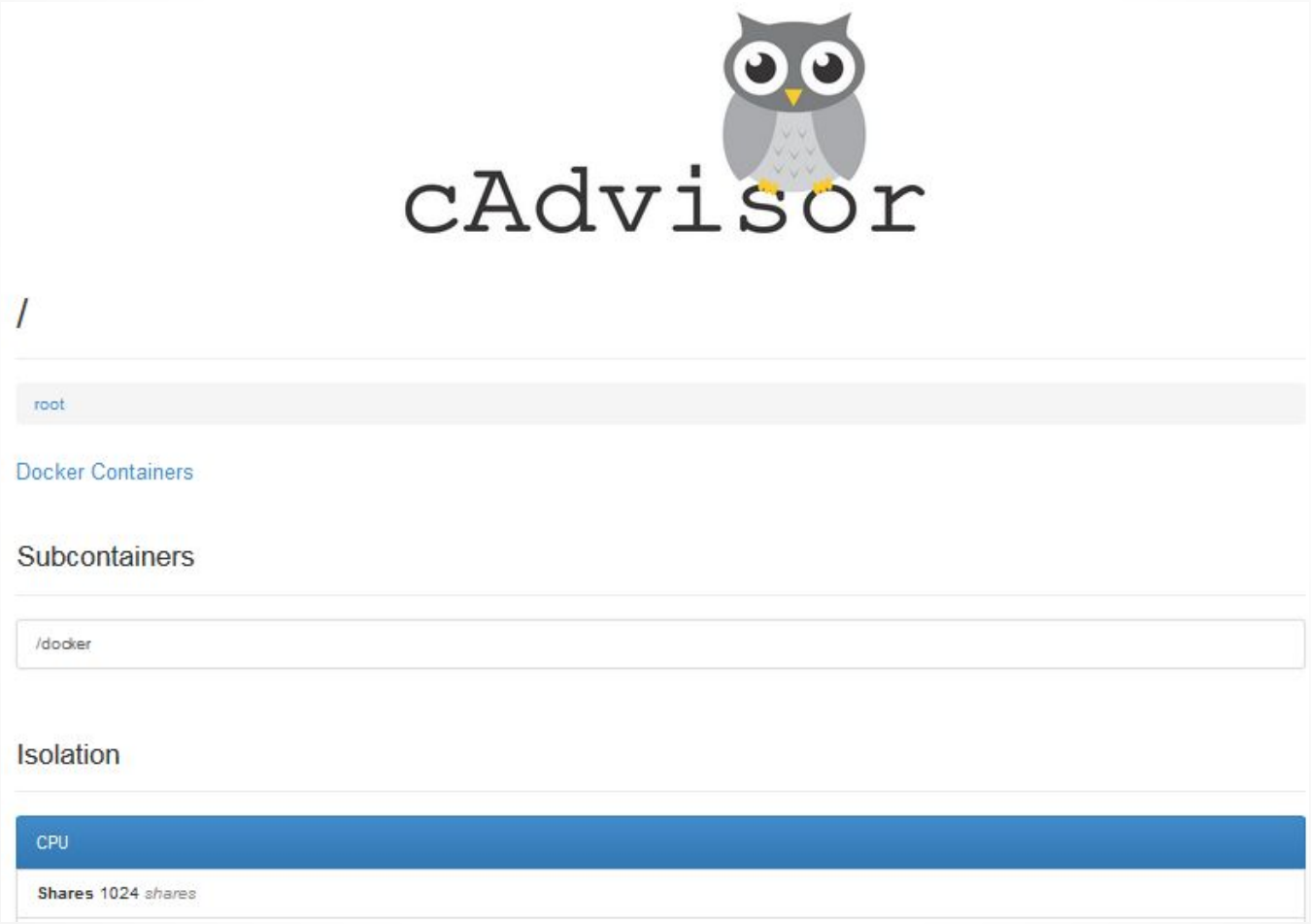
```
docker image pull labdocker/cadvisor:0.40.0
```

- 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:0.40.0
```


Workshop: CPU Configure

- cAdvisor



docker

Workshop: CPU Configure

- Download busybox เพื่อใช้ในการทดสอบ

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

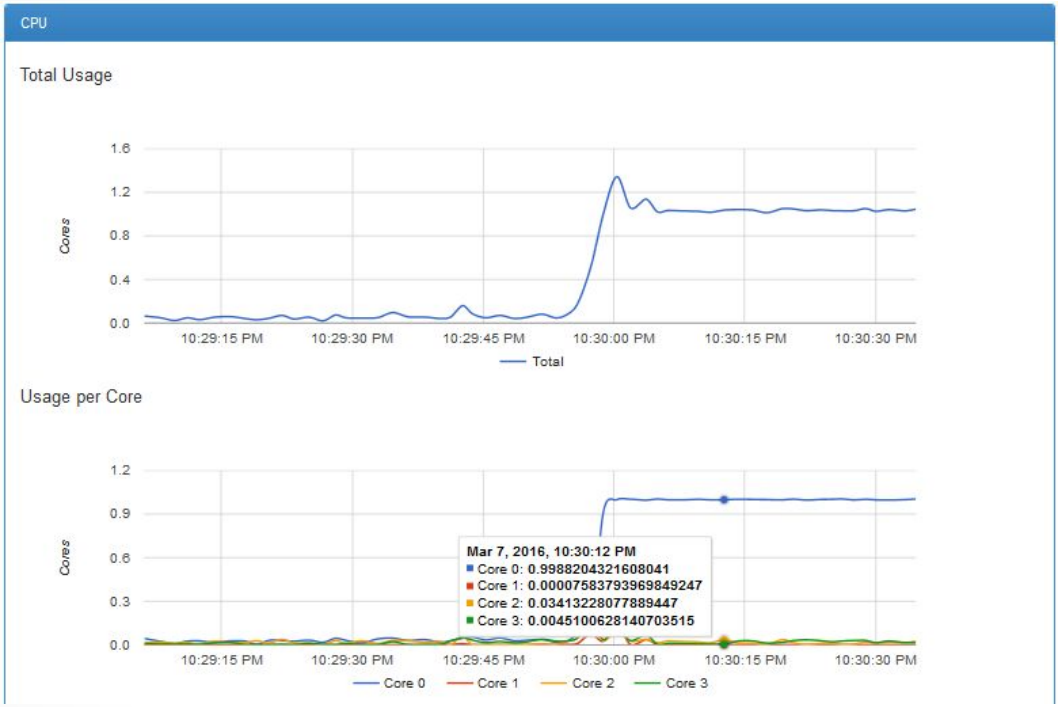
- Scenario 1 (Single CPU: 70/30)

```
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: CPU Configure

- Result




```
CONTAINER           CPU %               MEM USAGE / LIMIT     MEM %               NET I/O             BLOCK I/O           PIDS
01109e1c7b0f        70.65%              76KiB / 995.4MiB      0.01%               648B / 0B           0B / 0B             1
909882f27a59        30.26%              40KiB / 995.4MiB      0.00%               648B / 0B           0B / 0B             1
e247e2e8240a        0.05%               16.86MiB / 995.4MiB   1.69%               718kB / 39.4MB      17.6MB / 0B         11
^C
docker@labdocker:~$ docker ps
CONTAINER ID        IMAGE                         COMMAND                  CREATED             STATUS              PORTS
          NAMES
01109e1c7b0f        labdocker/busybox:latest      "md5sum /dev/urandom"    2 minutes ago       Up 2 minutes
          Share_70
909882f27a59        labdocker/busybox:latest      "md5sum /dev/urandom"    3 minutes ago       Up 3 minutes
          Share_30
e247e2e8240a        labdocker/cadvisor:latest     "/usr/bin/cadvisor..."   4 minutes ago       Up 4 minutes        0.0.0.0:8080->
8080/tcp   cadvisor
docker@labdocker:~$
```


Quiz ?

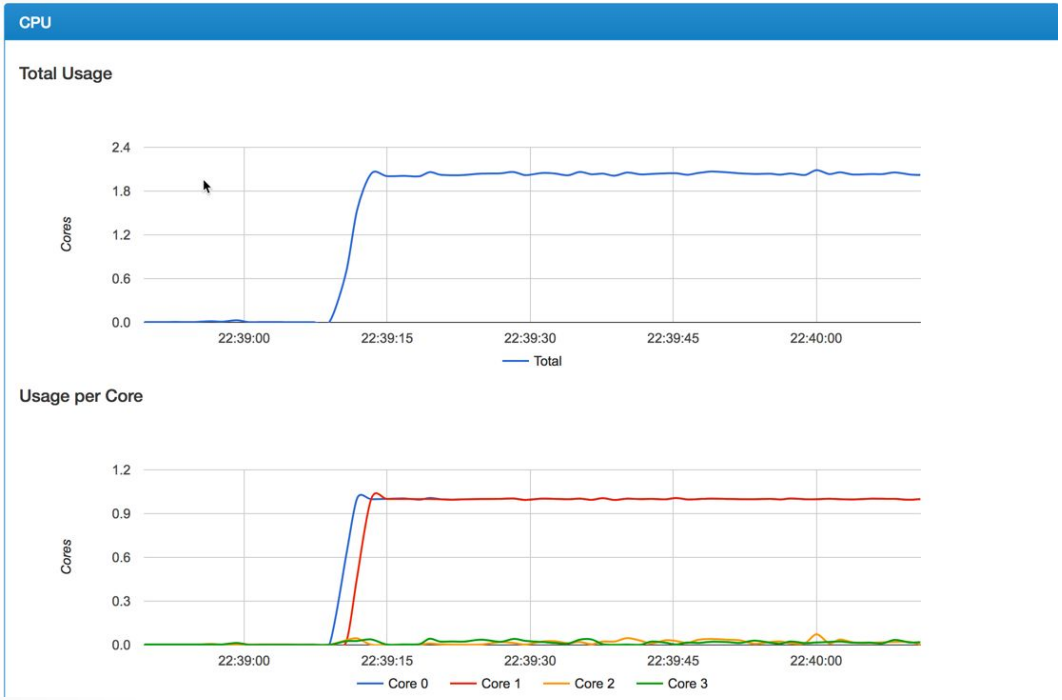
- Scenario (Multiple CPU: 70/30)

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

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


Quiz ?

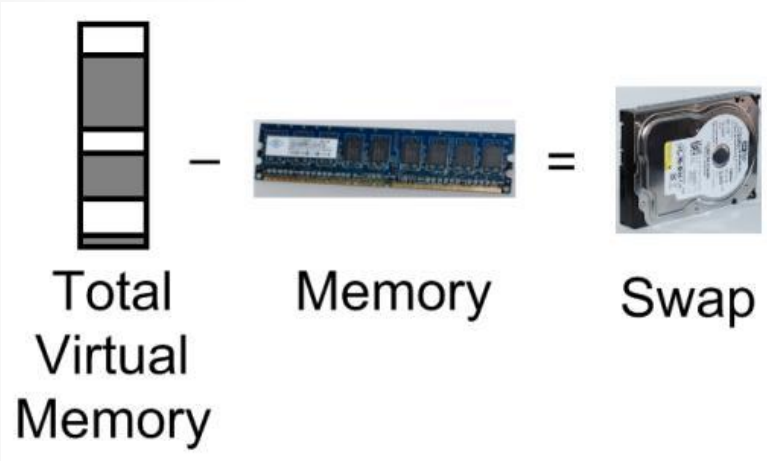
- Result

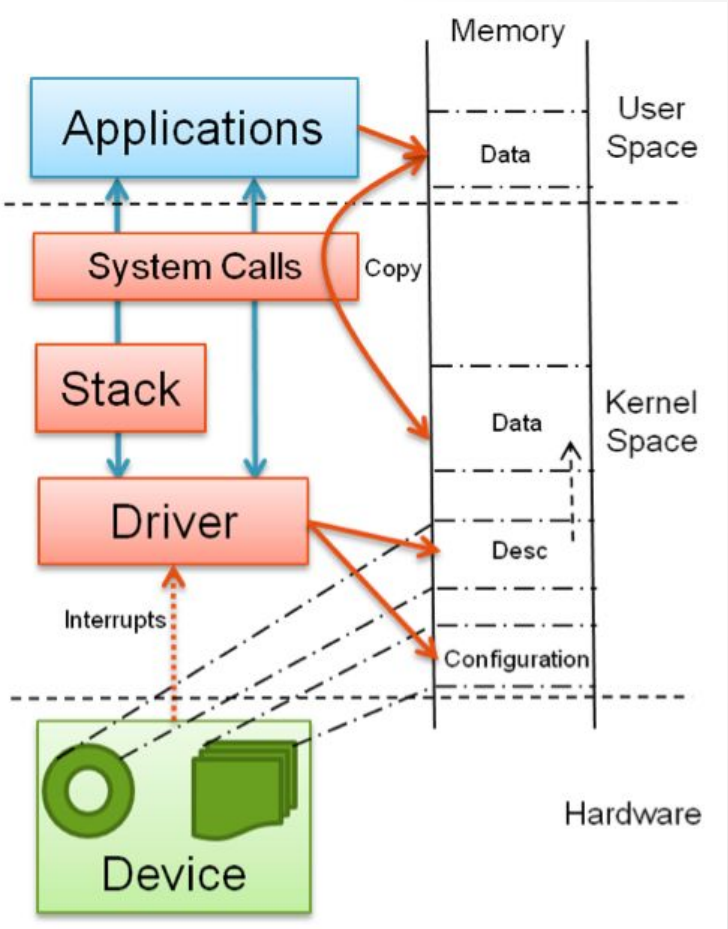



```
CONTAINER           CPU %               MEM USAGE / LIMIT     MEM %               NET I/O             BLOCK I/O           PIDS
02d0b2973cff        99.93%              88KiB / 995.4MiB      0.01%               648B / 0B           0B / 0B             1
b1aebf2a3d71        99.88%              84KiB / 995.4MiB      0.01%               648B / 0B           0B / 0B             1
e247e2e8240a        2.02%               19.34MiB / 995.4MiB   1.94%               1.07MB / 66.7MB     17.6MB / 0B         11
^C
docker@labdocker:~$ docker ps
CONTAINER ID        IMAGE                         COMMAND                  CREATED             STATUS              PORTS
          NAMES
02d0b2973cff        labdocker/busybox:latest      "md5sum /dev/urandom"    41 seconds ago      Up 40 seconds
          Share_70
b1aebf2a3d71        labdocker/busybox:latest      "md5sum /dev/urandom"    42 seconds ago      Up 42 seconds
          Share_30
e247e2e8240a        labdocker/cadvisor:latest     "/usr/bin/cadvisor..."   6 minutes ago       Up 6 minutes        0.0.0.0:8080->
8080/tcp   cadvisor
docker@labdocker:~$
```


Memory

- Default container จะมองเห็น memory resource ทั้งหมดในเครื่อง ระหว่าง runtime และสามารถใช้งาน memory resource ทั้งหมดเท่าที่ต้องการ





docker

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
```

- disable oom kill

```
--oom-kill-disable =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 แล้วกำหนดเป็น “--memory-reservation”
- ทำ capacity testing เพื่อกำหนด maximum memory ที่ต้องการในการใช้งาน (+30%) แล้วกำหนด “--memory“ (Limit)
- สำหรับ critical system ให้กำหนด “--memory-reservation” เท่ากับ “--memory“ (Limit)
- Memory Priority ? (Still waiting)

I/O

- โดย default ทุก container จะมีโอกาสใช้ I/O เท่าๆกัน (500 weight)
- กำหนดค่า weight ในการใช้งาน I/O (สำหรับ Direct IO เท่านั้น)

```
--blkio-weight 300, --blkio-weight-device “/s01/tdb:500”
```

- จำกัดการอ่านข้อมูล เป็น bps (kb: kilobyte, mb: megabyte, gb: gigabyte) / iops

```
--device-read-bps /s01/tdb:1mb
--device-read-iops /s01/tdb:20000
```

- จำกัดการเขียนข้อมูล เป็น bps , iops

```
--device-write-bps /s01/tdb:1mb
--device-write-iops /s01/tdb:20000
```


I/O

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

```
$ iostat -xtc
                 extended device statistics              tty         cpu
device       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
sd6          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
nfs49        0.0    0.0     0.0     0.0  0.0  0.0   15.1   0   0
nfs53        0.0    0.0     0.4     0.0  0.0  0.0   24.5   0   0
nfs54        0.0    0.0     0.0     0.0  0.0  0.0    6.3   0   0
nfs55        0.0    0.0     0.0     0.0  0.0  0.0    4.9   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
15,000	175
10,000	125
7200	75
5400	50

RAID	Write Penalty
0	1
1	2
5	4
6	6
DP	2
10	2

- Raw disk performance: disk iops x unit of disk
  - **Ex:** SAS 128GB (15K) x 6 units => 175 x 6 = 1060 iops

- Raid disk performance:
  - ((raw iops x %write)/penalty)+(raw x %read)
  - Normal situation (write 30%, read 70%)
  - **Ex:** Raid 5 ((1060 x .3)/4)+(1060x.7) □ 821.6 iops
  - **Ex**: Raid 1 ((1060 x .3)/2)+(1060x.7) □ 901 iops

Network

• • •

Network

- Software define network by design (virtual switch) (Support IPv4/6)
- default docker จะจัดเตรียม network มาให้สามรูปแบบ

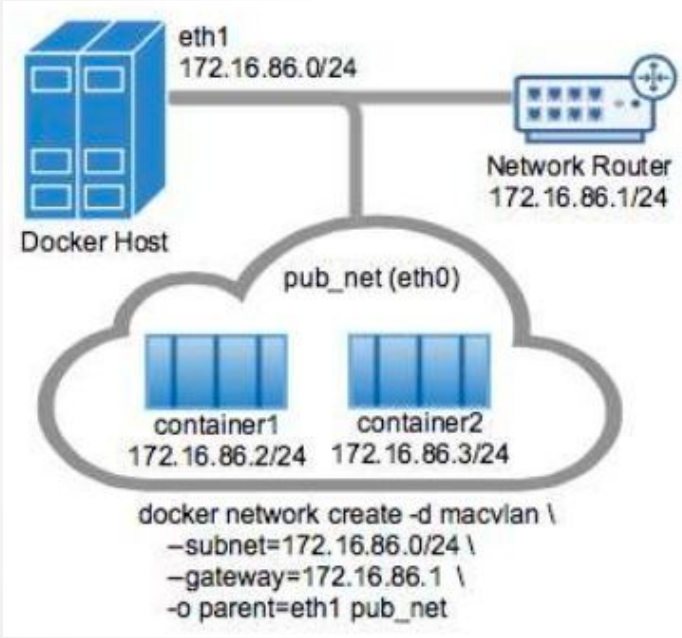
```
docker@labdocker:~$ docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
0dc1bbf00498        bridge              bridge              local
91d6dd4056a9        host                host                local
fa786a00adda        none                null                local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

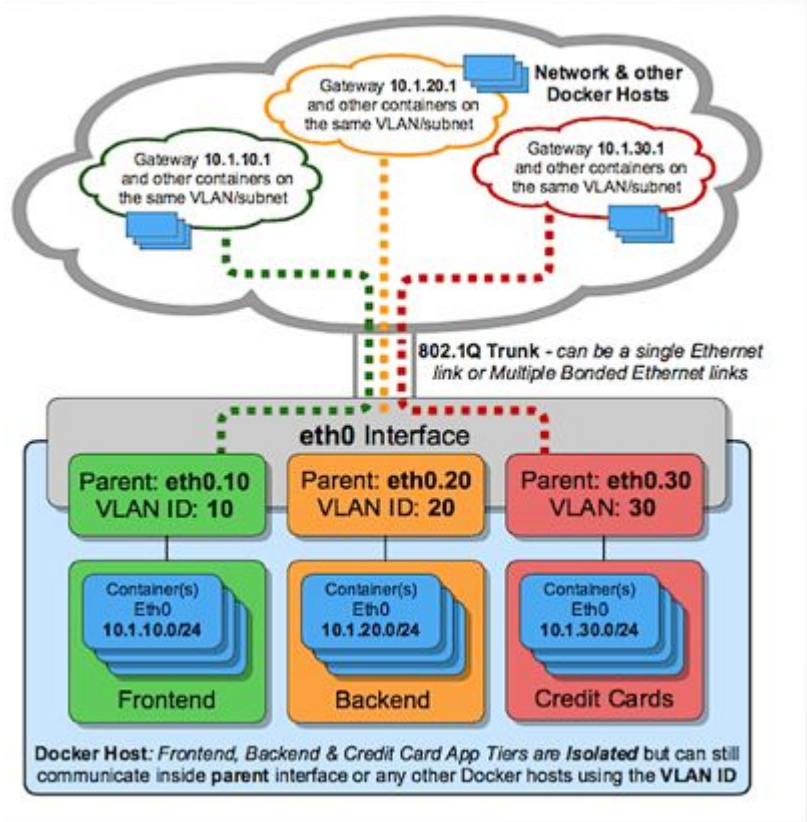
- **Bridge/User Defined** คือ default network สำหรับให้ container เชื่อมต่อออกสู่โลกภายนอกผ่าน virtual switch “docker0” โดยใช้ network stack ของ container เองในการเชื่อมต่อ (route mode)
- **none** คือ network loopback (127.0.0.1) สำหรับ container ที่ไม่มีการเชื่อมต่อออกไปด้านนอก หรือใช้งานกับการเชื่อมต่อแบบอื่นๆ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

  (ใช้ในกรณี ต้องการ network performance สูงสุด) (security concern)

Network

- Additional network type
- **MACVLAN/IPVLAN (L2)**

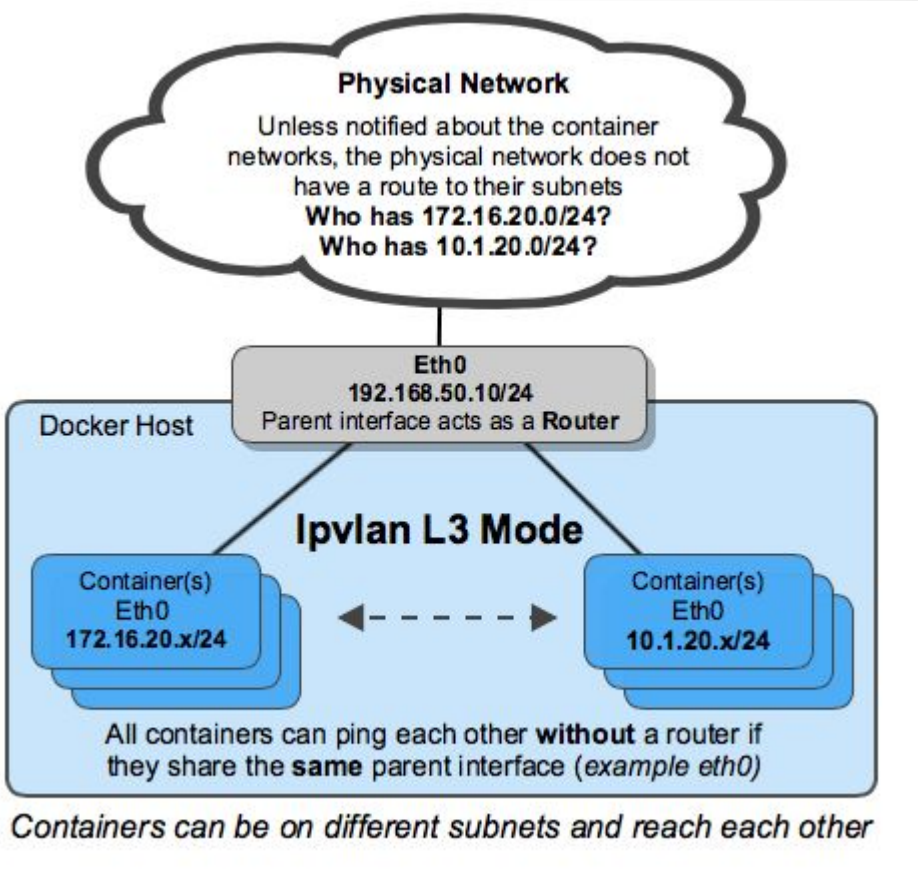




docker

Network

- Additional network type
- **IPVLAN (L3 Mode)**



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

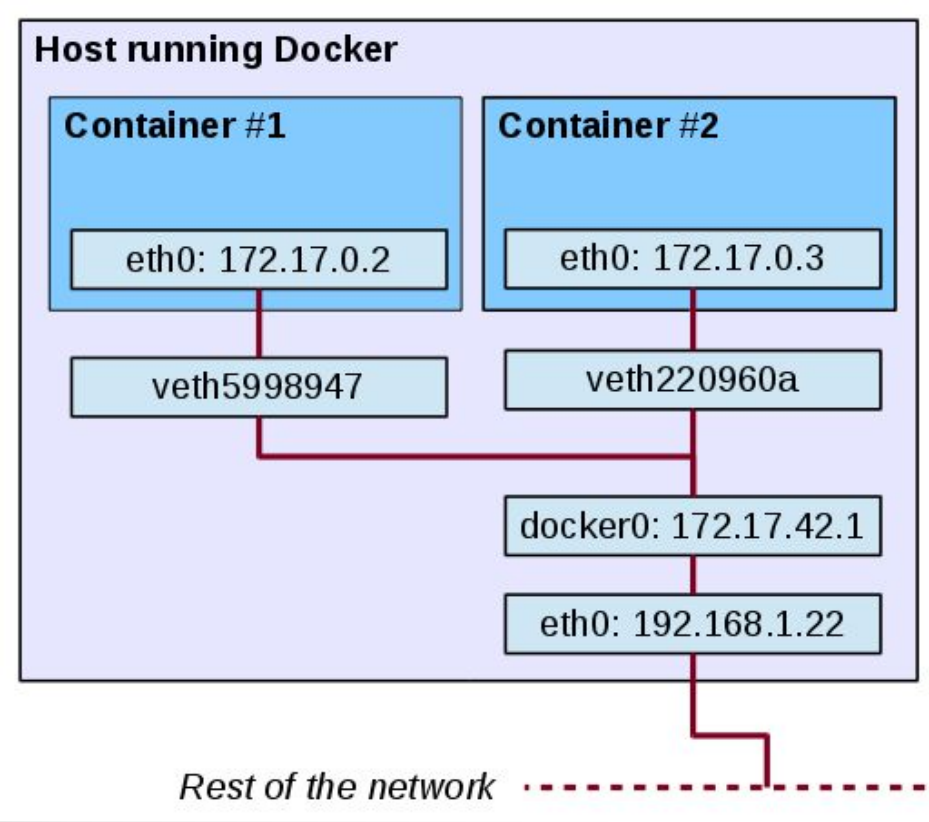
docker

Network

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

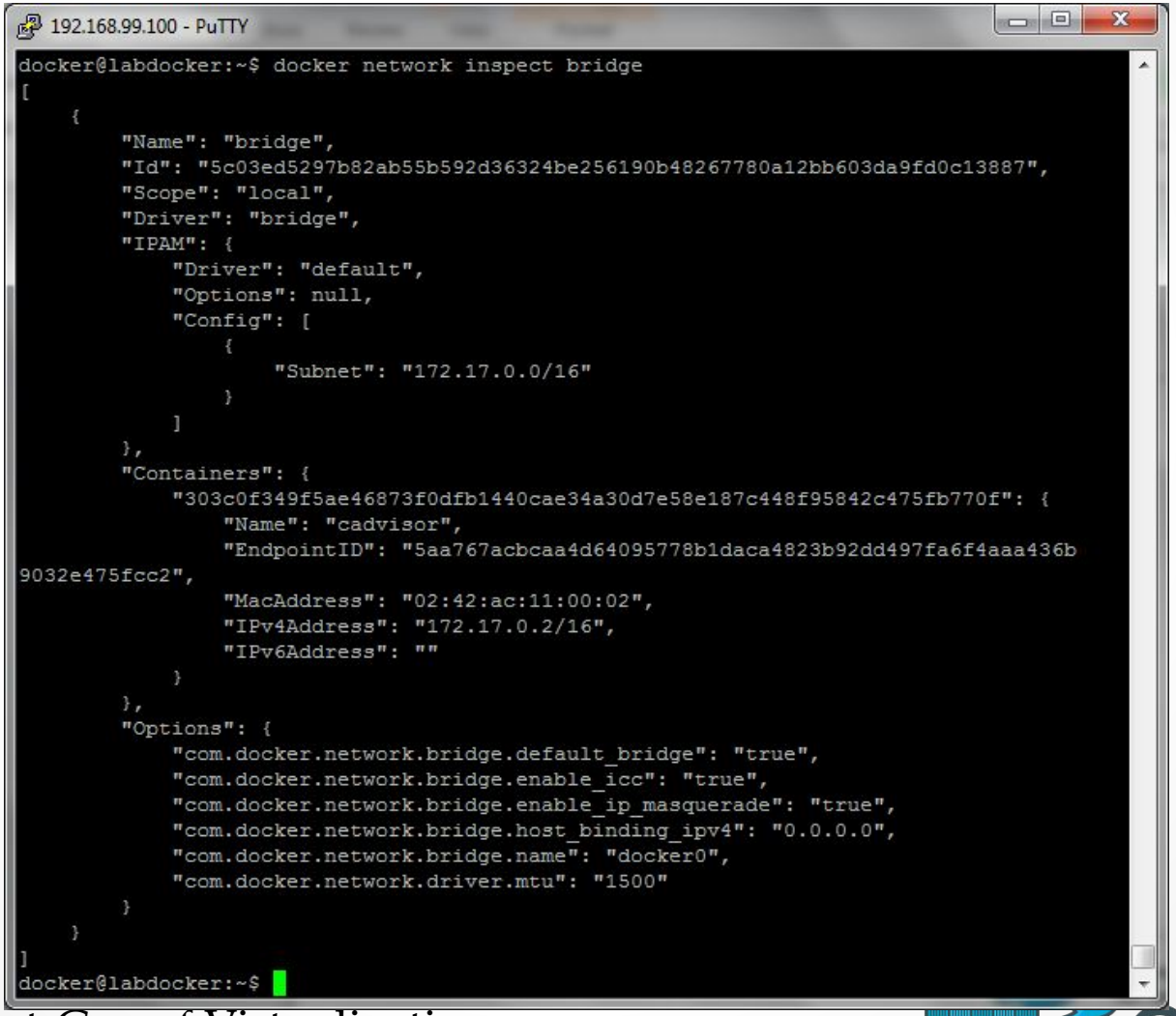
Network

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



Network

- docker network inspect bridge



Network

- Option เกี่ยวกับ network สำหรับสั่ง run container

--dns= x.x.x.x (Default --name,--net-alias,--link also dns internal docker)
--network="<bridge/none/host/custom>"
--network-alias = "xxxx"
--add-host="xxxx"
--mac-address="xx:xx:xx:xx"
--ip="x.x.x.x"
--ipv6="xx:xx:xx:xx"
–link-local-ip="x.x.x.x"
-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
```

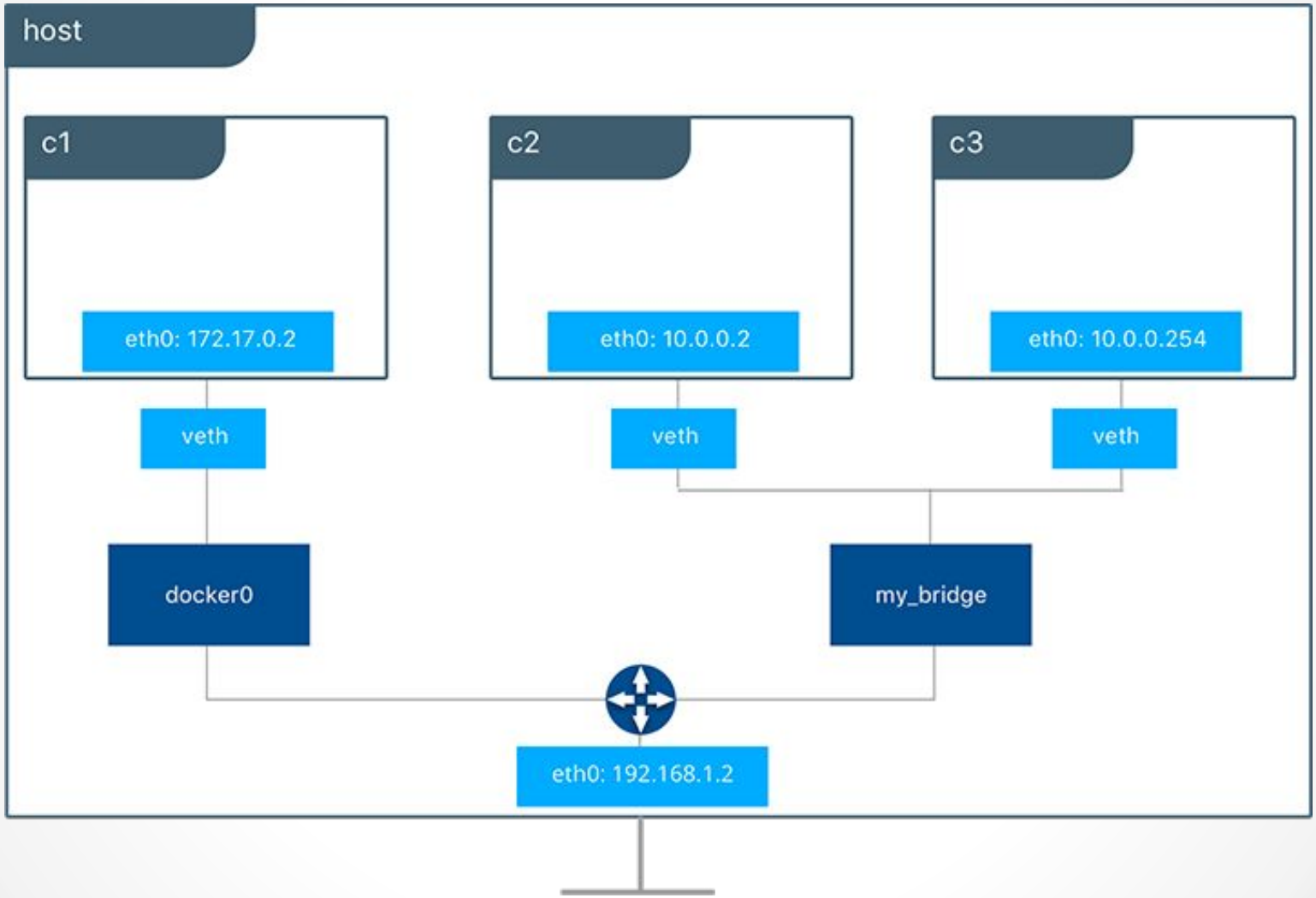
```
Ex: docker network rm netweb
```

- สามารถเพิ่มเติม option ในการสร้าง virtual switch เพิ่มเติม
  - --attachable = กำหนดให้สามารถต่อ container เข้ามาในเน็ตเวิร์คได้แบบ manual
  - --aux-address = กำหนด ip address (v4/v6) สำหรับ network driver
  - --config-from= copy configuration จาก virtual switch อื่นๆ
  - --driver, -d = driver ในการใช้งาน virtual switch (bridge, none, macvlan etc)
  - --ingress = สร้างเพื่อใช้บน switch เพื่อทำ service routeing-mesh
  - --internal = สร้างใช้ภายใน docker เท่านั้น
  - --scope = กำหนด scope การใช้งาน (host,cluster)

Network

- สามารถเพิ่มเติม option ในการสร้าง virtual switch เพิ่มเติม
  - --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



docker

Network

- การเพิ่ม network ใน container

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

```
Ex: docker network connect webnet web1
```

- การลด network ใน container

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

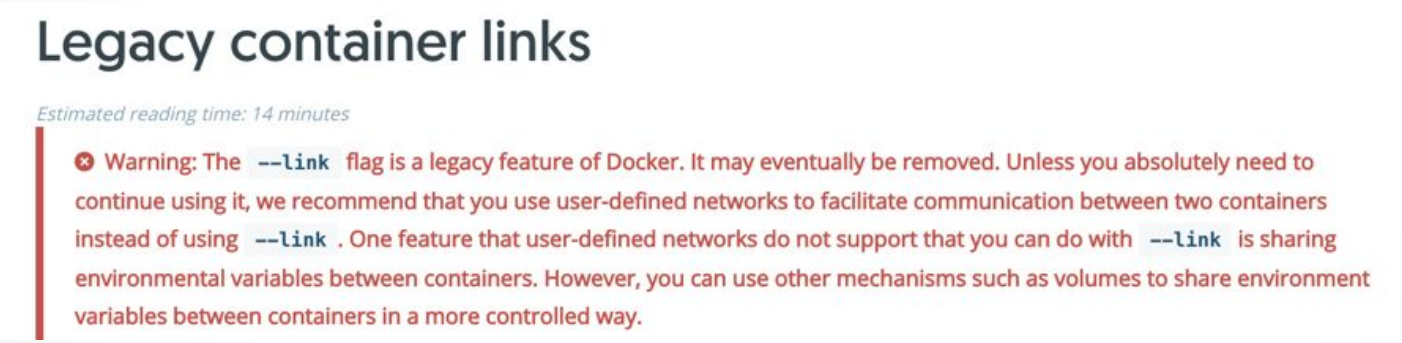
```
Ex: docker network disconnect webnet web1
```


Network

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

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

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



Legacy container links

Estimated reading time: 14 minutes

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

- DNS resolve on docker network (Recommend for Production)

Workshop: Network Configure

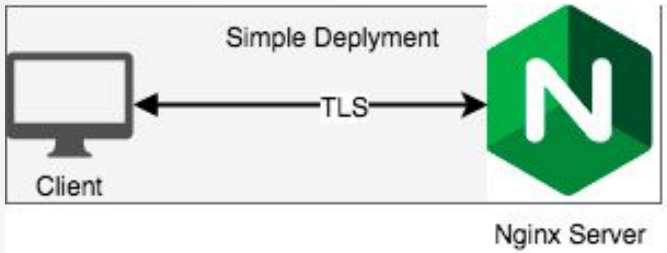
- Part 1: Reverse Proxy Network (DNS)
- Purpose: ทำการ Deploy nodejs webserver ด้วย solution proxy ของ nginx (load balance / reverse proxy)

- **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

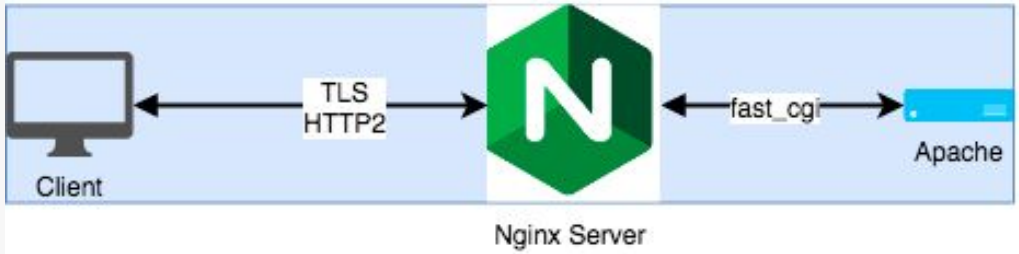
Workshop: Network Configure

docker

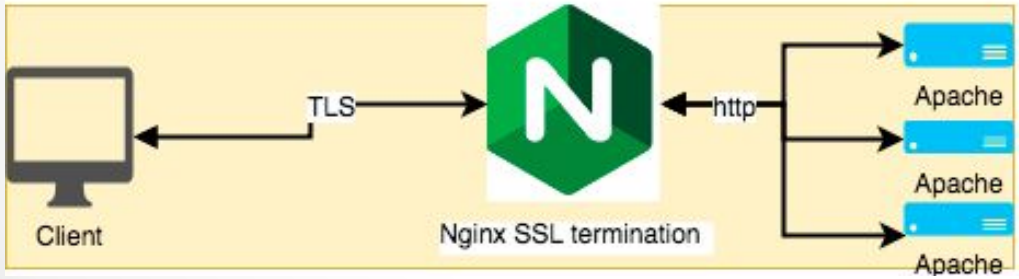
Workshop: Network Configure



Nginx Server A
Plain Deployment



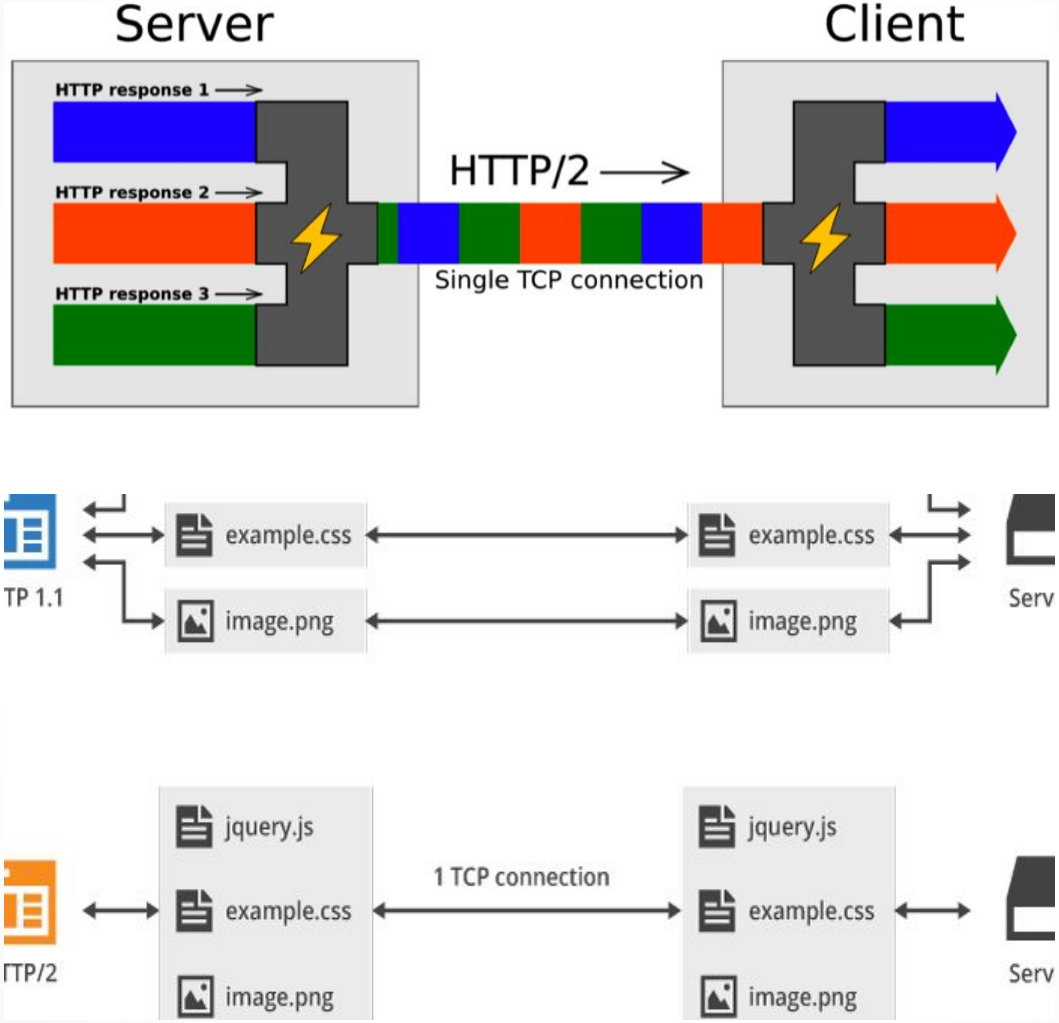
Nginx Reverse Proxy



Nginx SSL Offloading

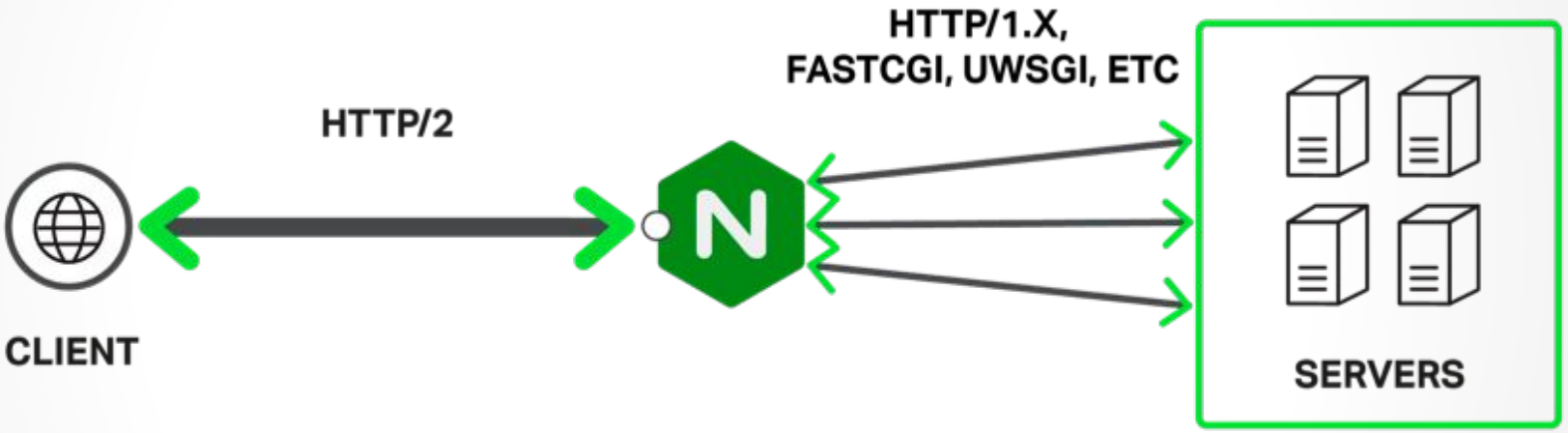
docker

Workshop: Network Configure



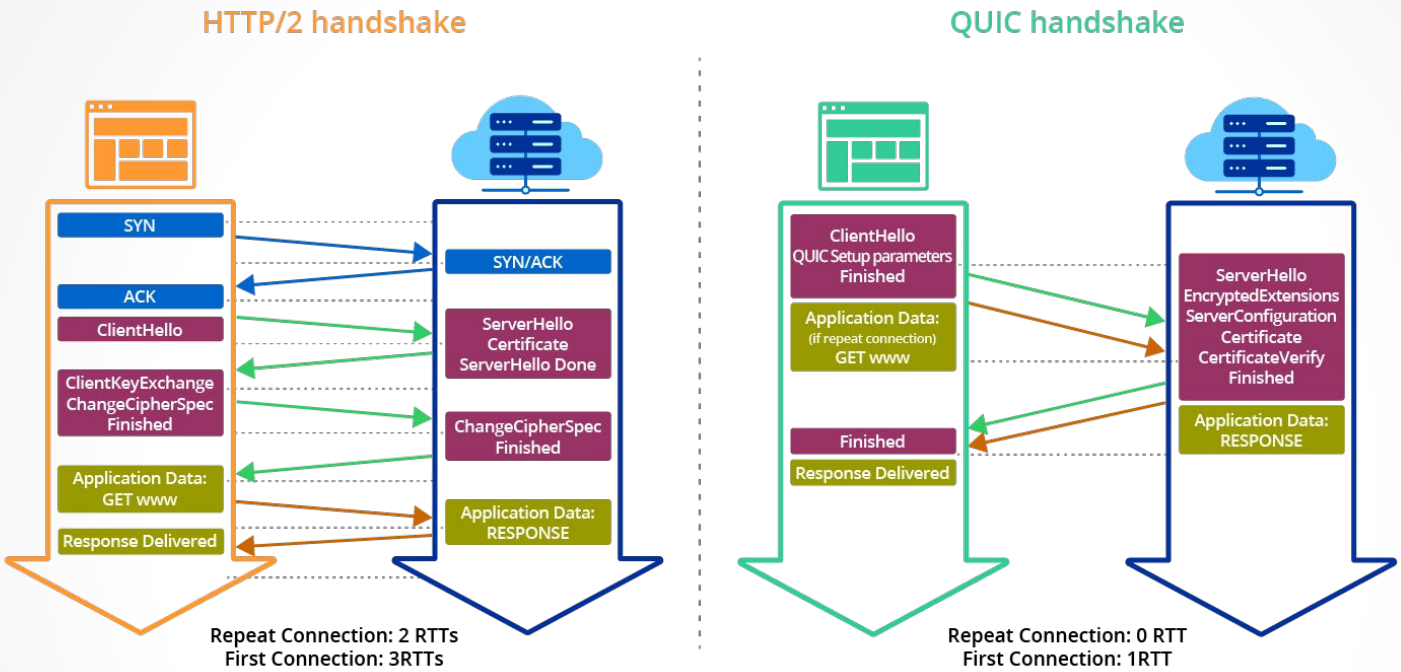
docker

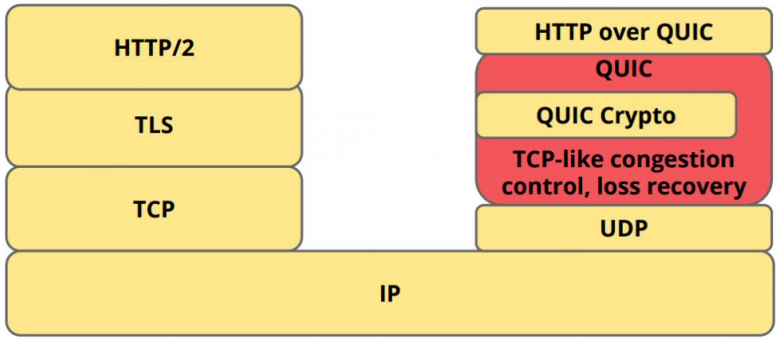
Workshop: Network Configure



docker

Workshop: Network Configure

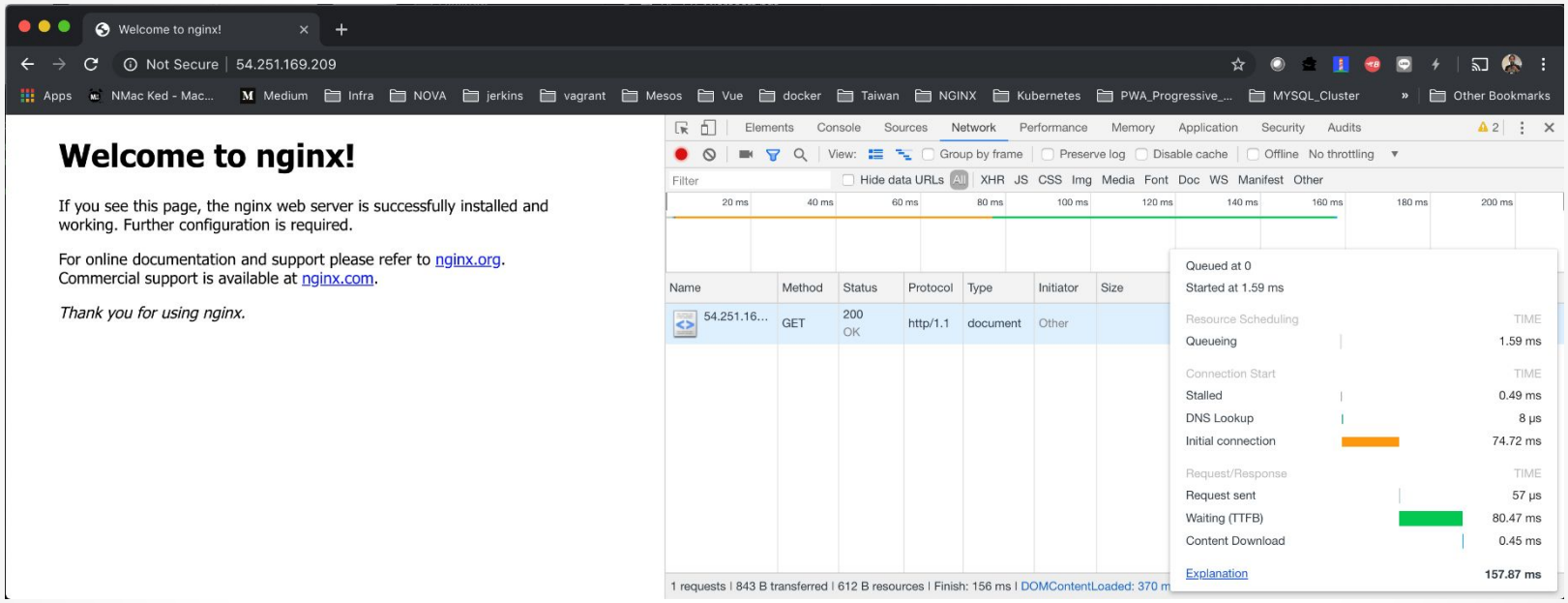




docker

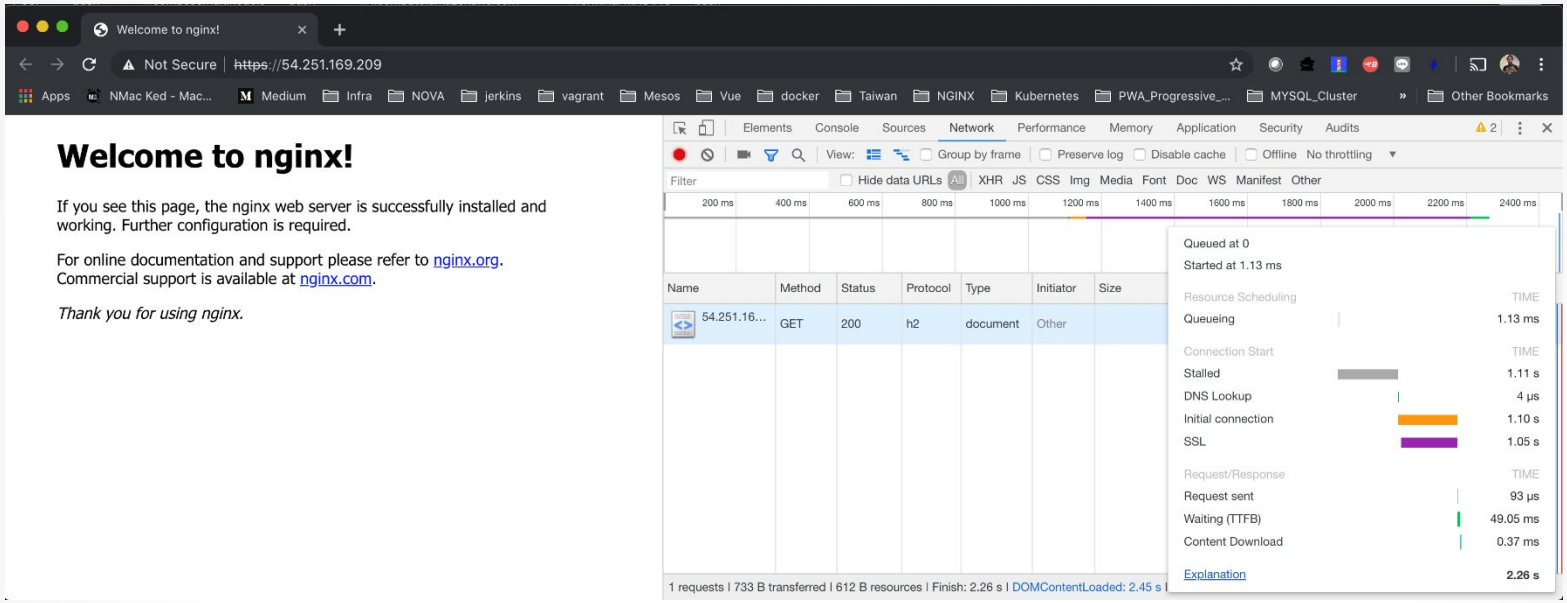
Workshop: Network Configure

- Normal HTTP/1.1

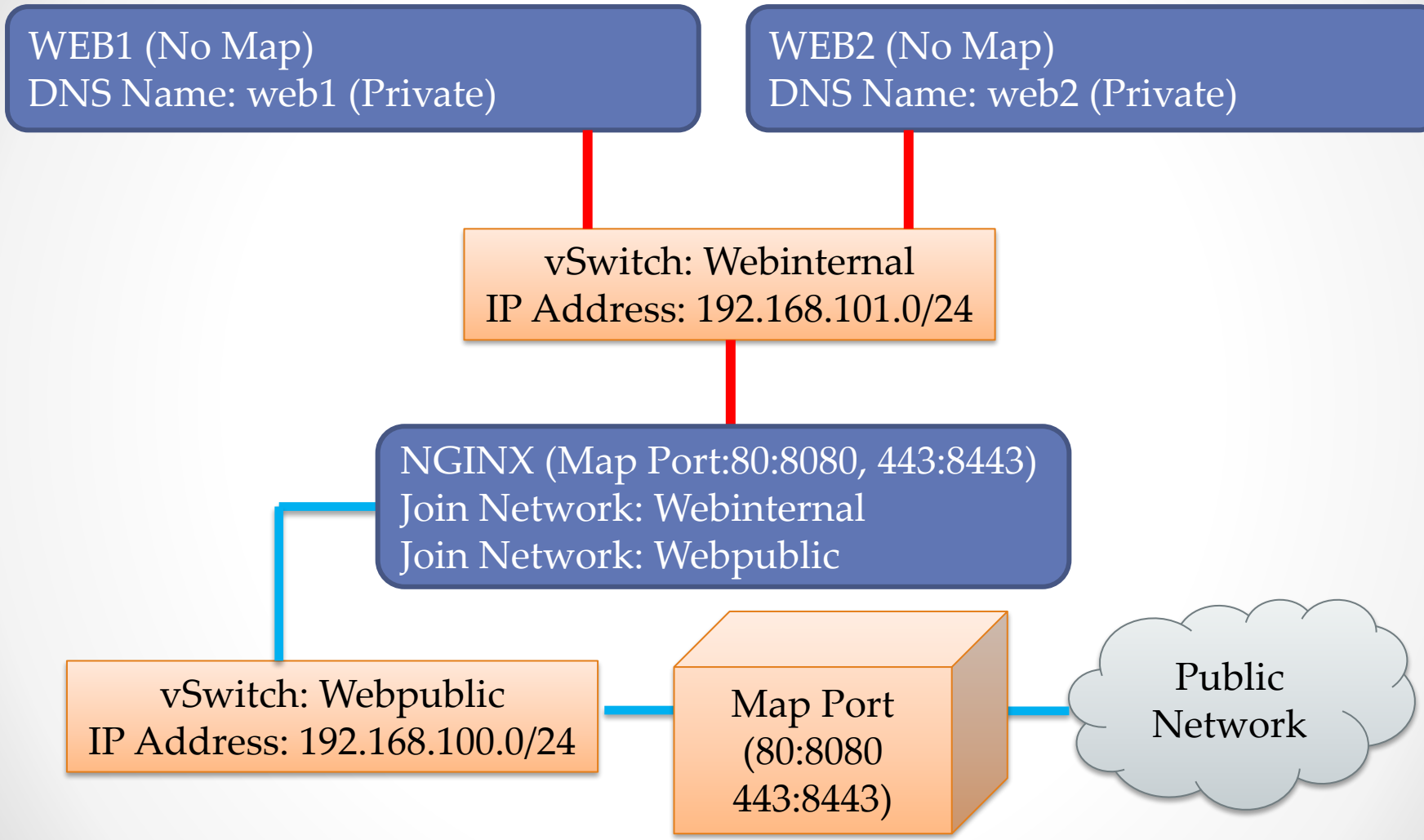


Workshop: Network Configure

- HTTP/2 HTTPS (Old word: SPDY)



Workshop: Network Configure



docker

Workshop: 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: Network Configure

- สร้างเครื่อง Webserver (nodejs) และ Nginx
  - **Web1**
  - docker run -dt --name web1 --net webinternal \
    --net-alias web1 labdocker/alpineweb:web1 node hello.js

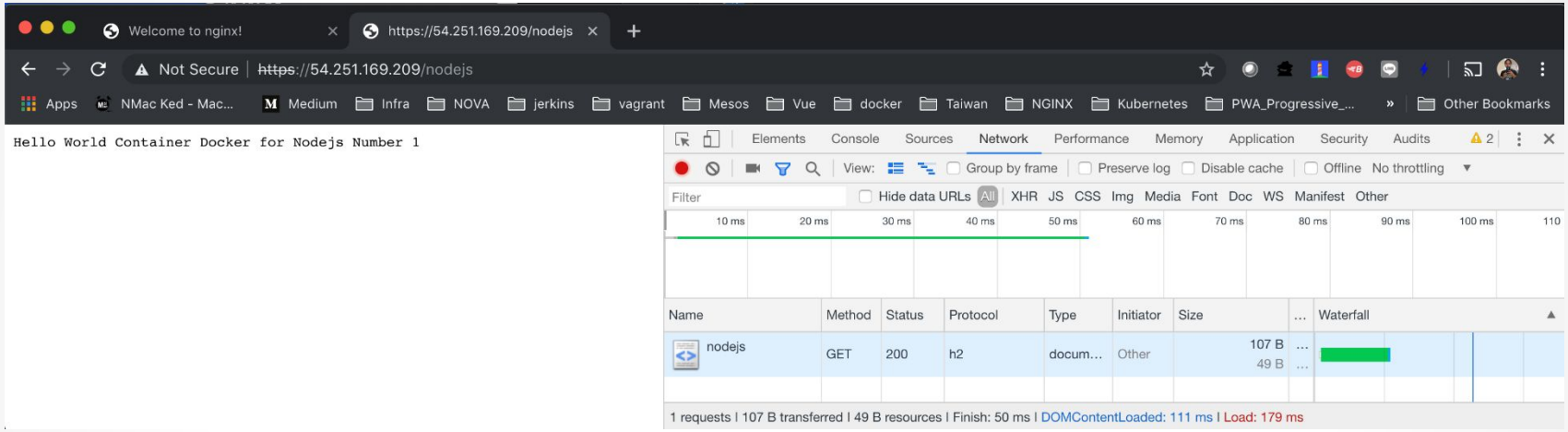
  - **Web2**
  - docker run -dt --name web2 --net webinternal \
    --net-alias web2 labdocker/alpineweb:web2 node hello.js

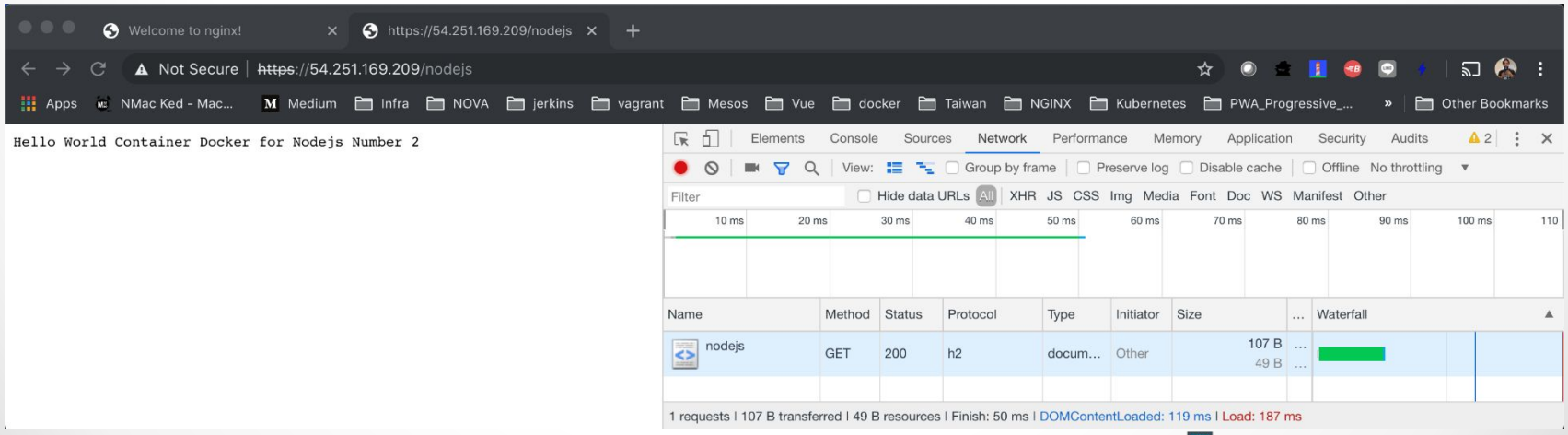
  - **NGINX**
  - docker run -dt --name nginx --net webinternal \
    -p 80:8080 –p 443:8443 labdocker/nginx:labnetworkhttp2

    docker network connect webpublic nginx

Workshop: Network Configure

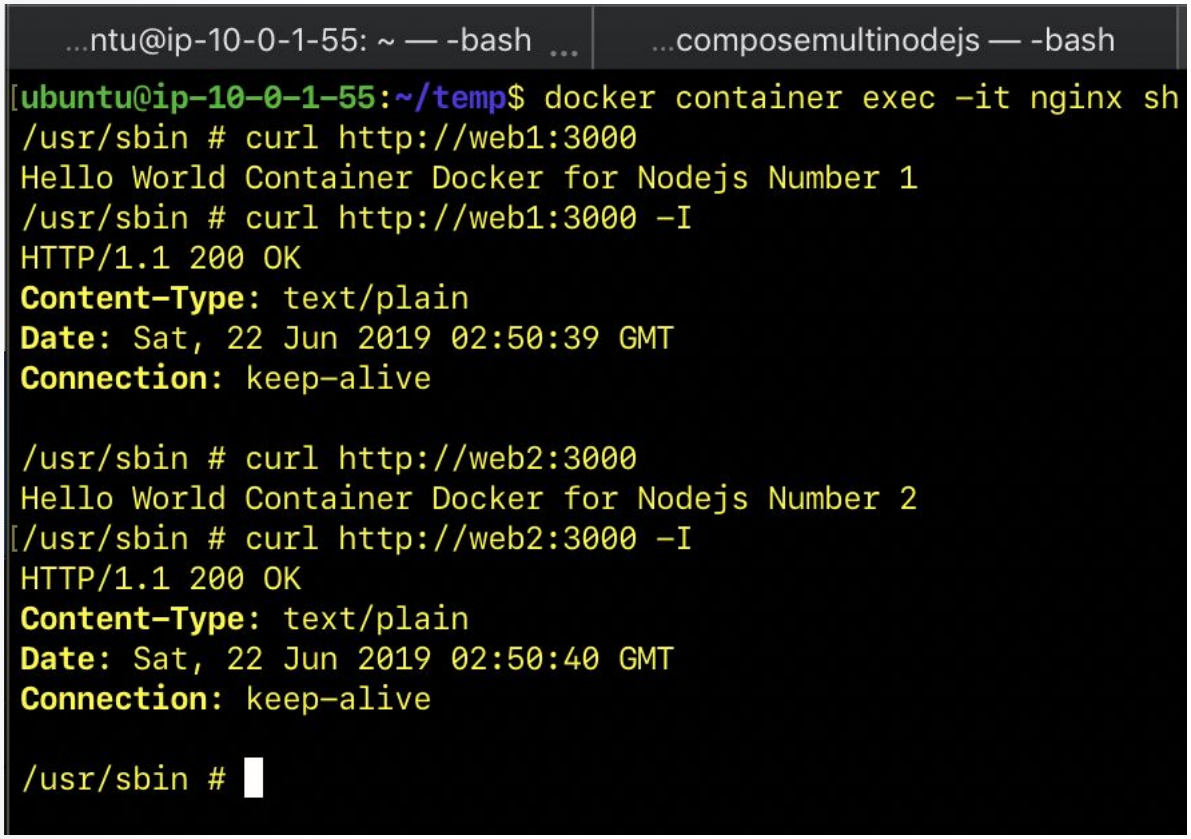
- url: http://<Public IP>/nodejs





Workshop: Network Configure

- curl result:



Workshop: Network Configure

- nginx.conf

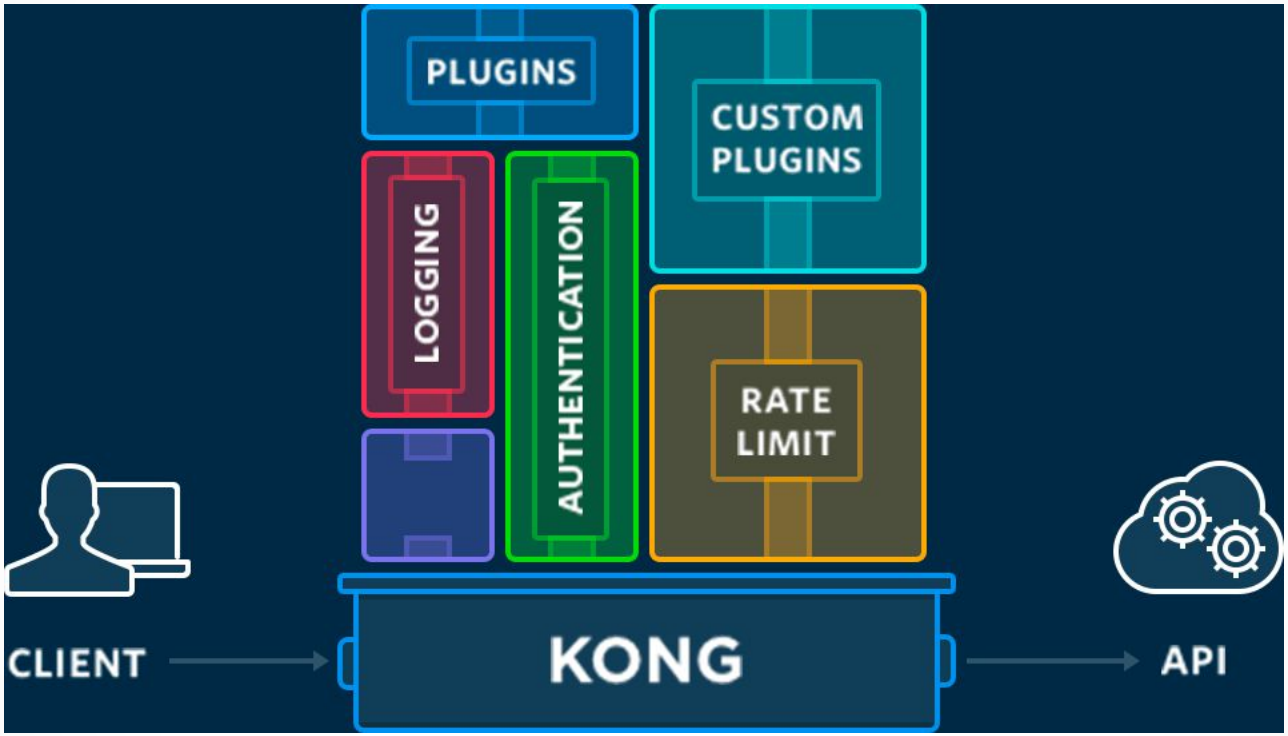
```
upstream nodejs_web {
    server web1:3000;
    server web2:3000;
}

server {
        listen 8080 default_server;
        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;
        }
}

server {
        listen 8443 ssl http2;
        ssl_certificate         labdocker.com.crt;
        ssl_certificate_key     labdocker.com.key;
        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: Network Configure

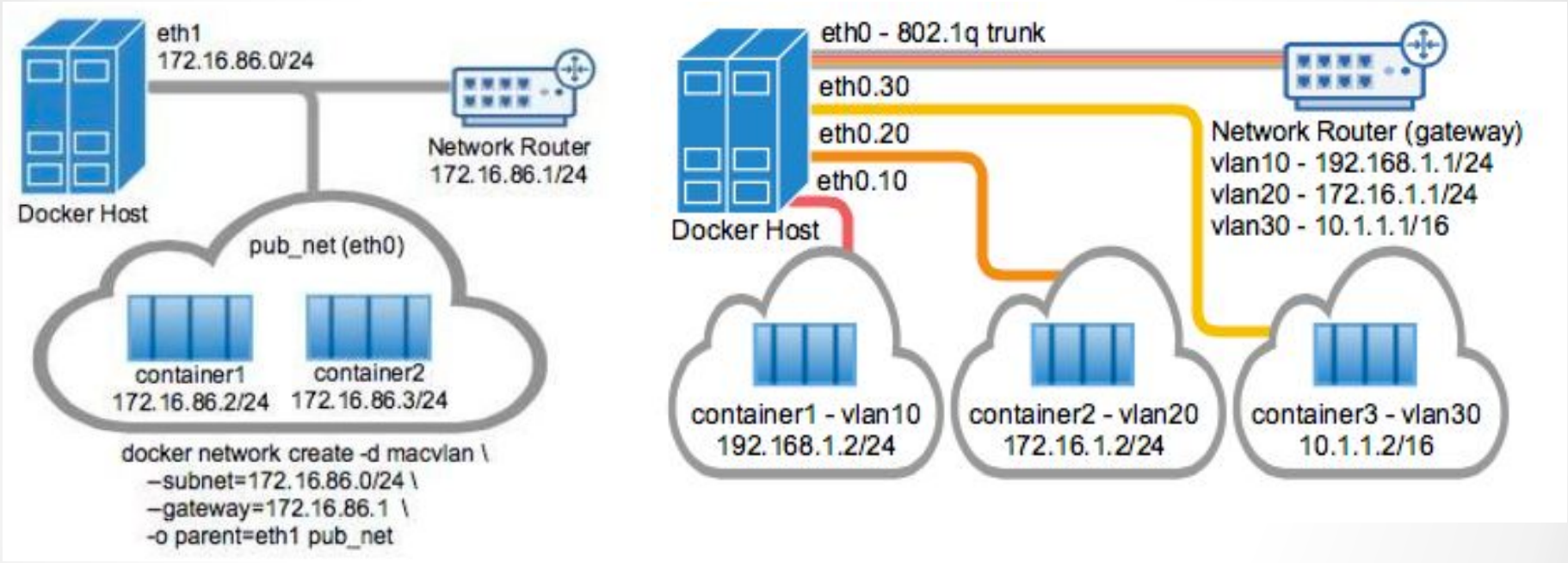
- Bonus: Kong API Gateway the basic



docker

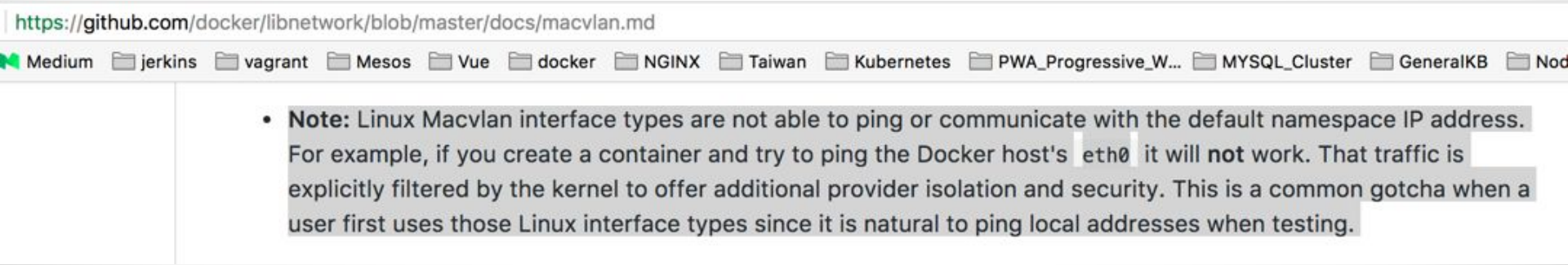
Network

- 802.1q Tagging (MACVLAN)

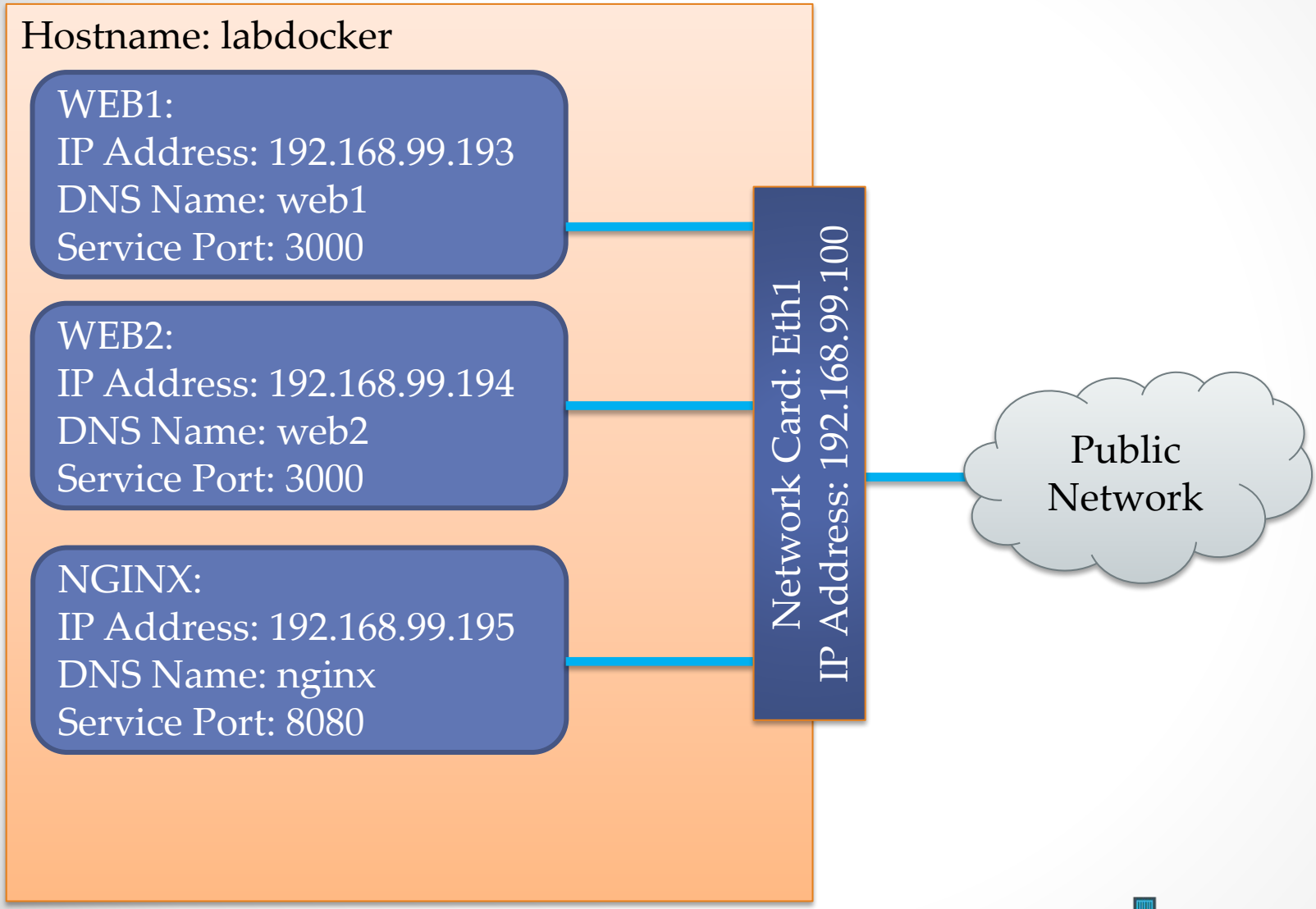


Demo: MACVLAN

- Reverse Proxy Network (MACVLAN) (This need docker-machine)
- Purpose: ทำการ Deploy nodejs webserver ด้วย solution proxy ของ nginx ผ่าน Macvlan
- **Macvlan:**
  - Interface: eth1
  - IP Address: 192.168.99.0/24
  - Gateway: 192.168.99.1 ⇒ Your Machine
  - Labdocker: 192.168.99.100 ⇒ Default
  - Range Address: 192.168.99.192/28
    - 192.168.99.193 – 192.168.99.206



Demo: MACVLAN



docker

Demo: MACVLAN

- ทำการสร้าง Switch สำหรับเชื่อมต่อ Macvlan
  - docker network create -d macvlan \
    --subnet=192.168.99.0/24 \
    --ip-range=192.168.99.192/28 \
    --gateway=192.168.99.1 \
    -o parent=eth1 macvlanlab

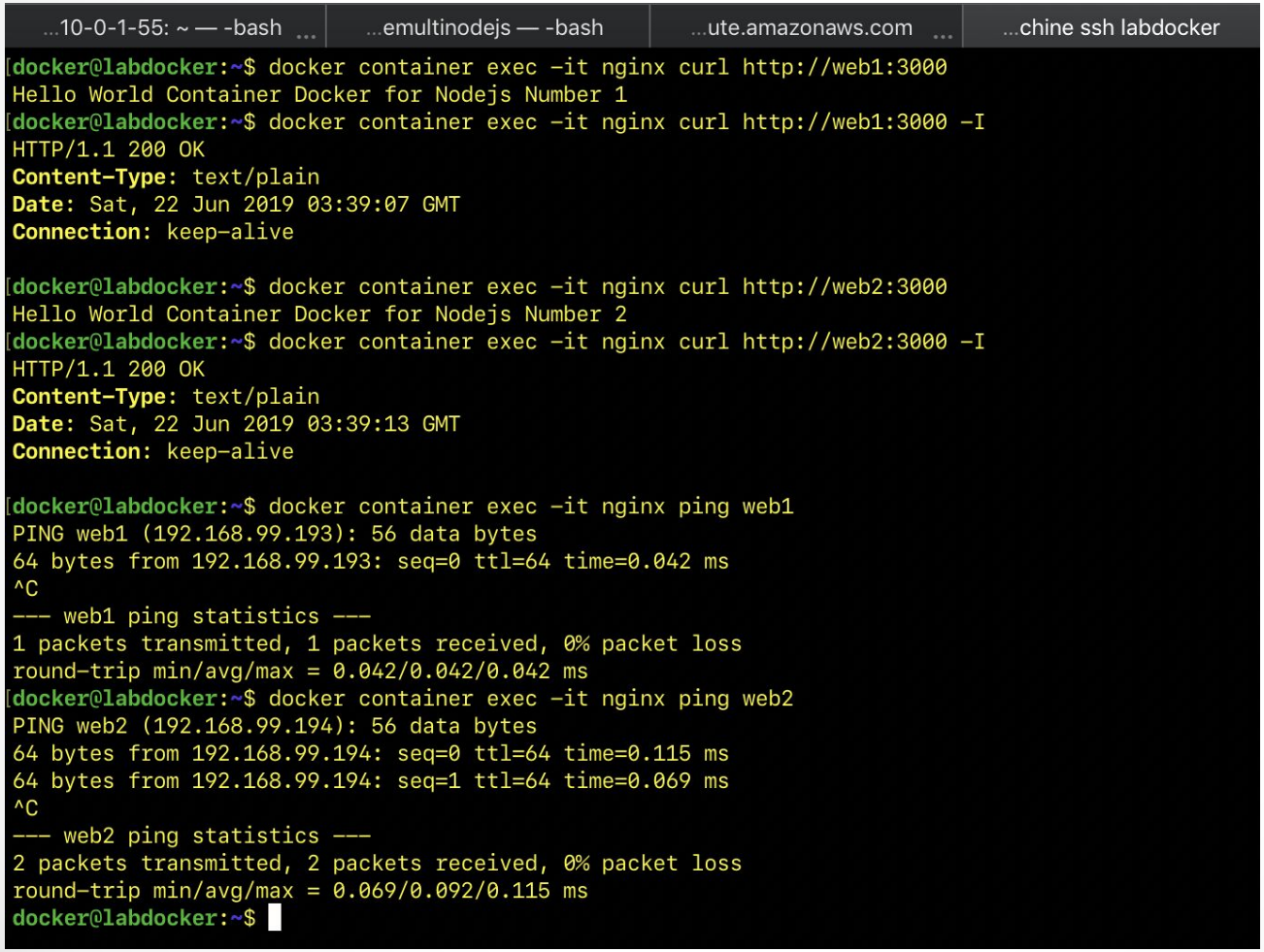
- ในกรณีต้องการทำ Tag VLAN (802.1q) จาก Switch/Router เข้าสู่ Docker Host และทำการสร้าง Sub Interface สามารถทำได้ดังนี้
  - docker network create -d macvlan \
    --subnet=192.168.99.0/24 \
    --ip-range=192.168.99.192/28 \
    --gateway=192.168.99.1 \
    -o parent=eth1.<vlan id> macvlanlab

Demo: MACVLAN

- สร้างเครื่อง Webserver (nodejs) และ Nginx
  - **Web1**
  - docker run -dt --name web1 --net macvlanlab \
    --net-alias web1 labdocker/alpineweb:web1 node hello.js
  - **Web2**
  - docker run -dt --name web2 --net macvlanlab \
    --net-alias web2 labdocker/alpineweb:web2 node hello.js
  - **NGINX**
  - docker run -dt --name nginx --net macvlanlab \
    labdocker/nginx:labnetworkhttp2

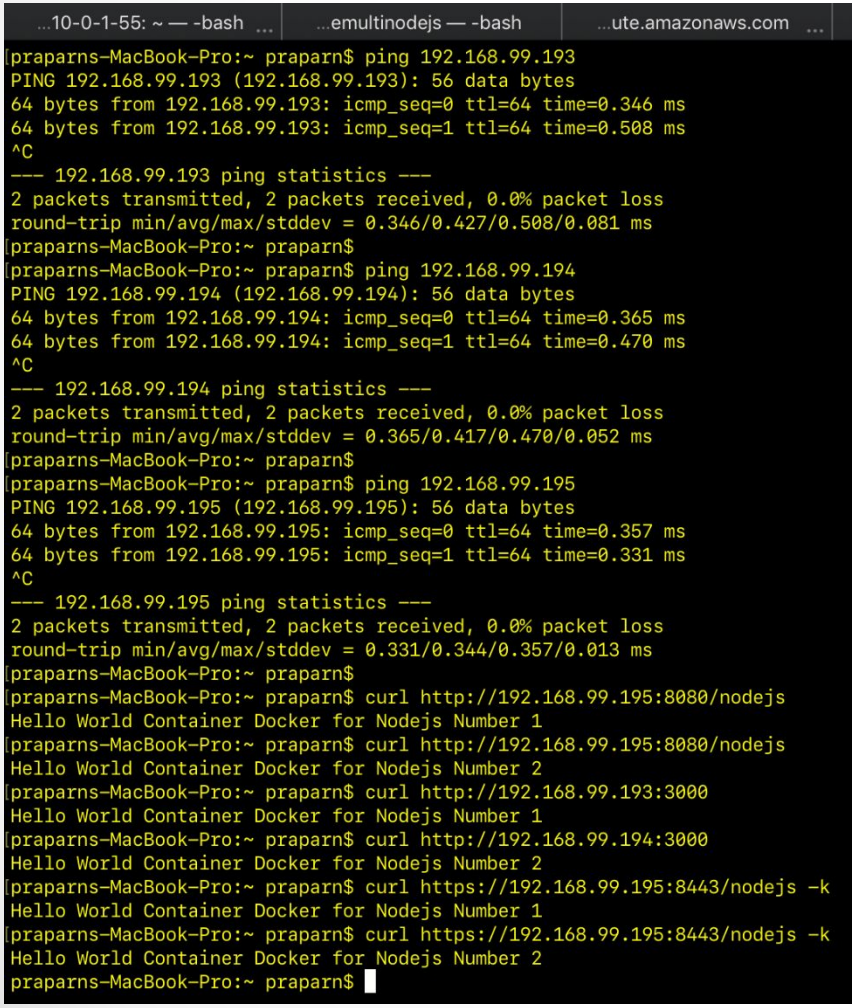
Demo: MACVLAN

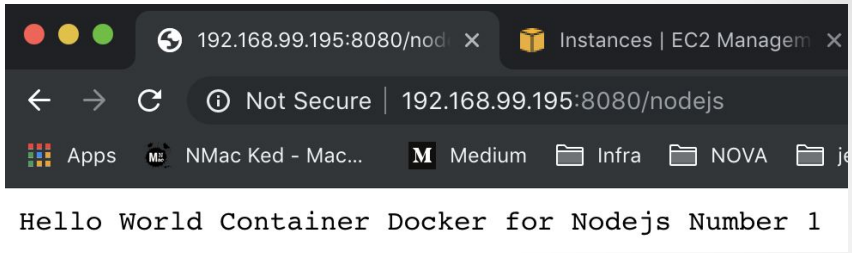
- Internal test:

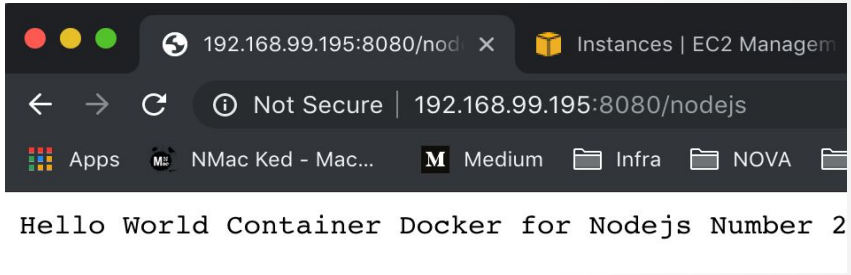


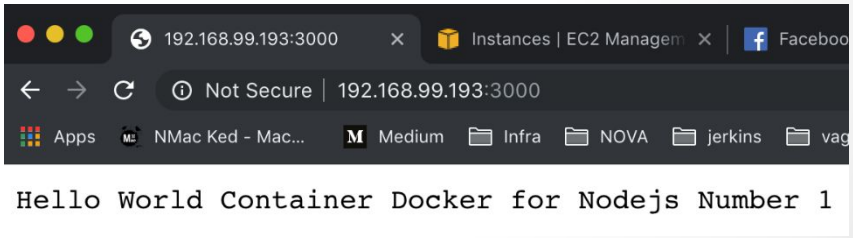
Demo: MACVLAN

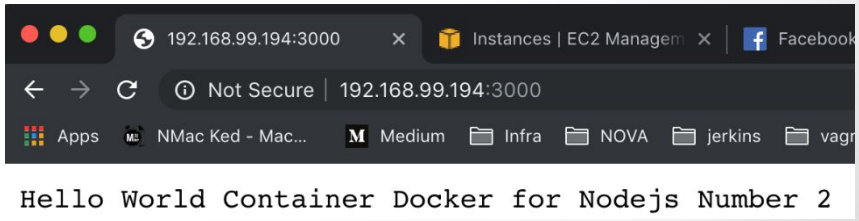
- External Test:







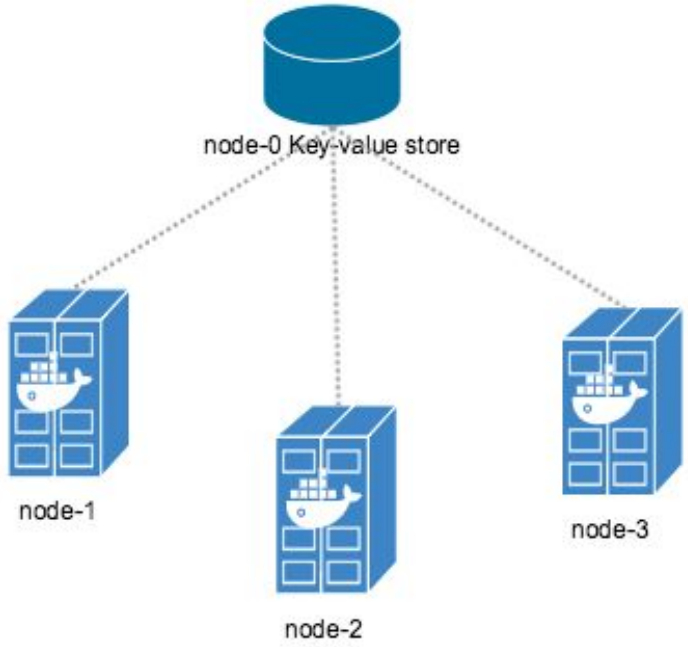




docker

Network

- Network across host (Overlay Network)



Network

- 3rd Party Network Plugin

Plugin	Description
Contiv Networking	An open source network plugin to provide infrastructure and security policies for a multi-tenant micro services deployment, while providing an integration to physical network for non-container workload. Contiv Networking implements the remote driver and IPAM APIs available in Docker 1.9 onwards.
Kuryr Network Plugin	A network plugin is developed as part of the OpenStack Kuryr project and implements the Docker networking (libnetwork) remote driver API by utilizing Neutron, the OpenStack networking service. It includes an IPAM driver as well.
Weave Network Plugin	A network plugin that creates a virtual network that connects your Docker containers - across multiple hosts or clouds and enables automatic discovery of applications. Weave networks are resilient, partition tolerant, secure and work in partially connected networks, and other adverse environments - all configured with delightful simplicity.

Volume

• • •

Volume

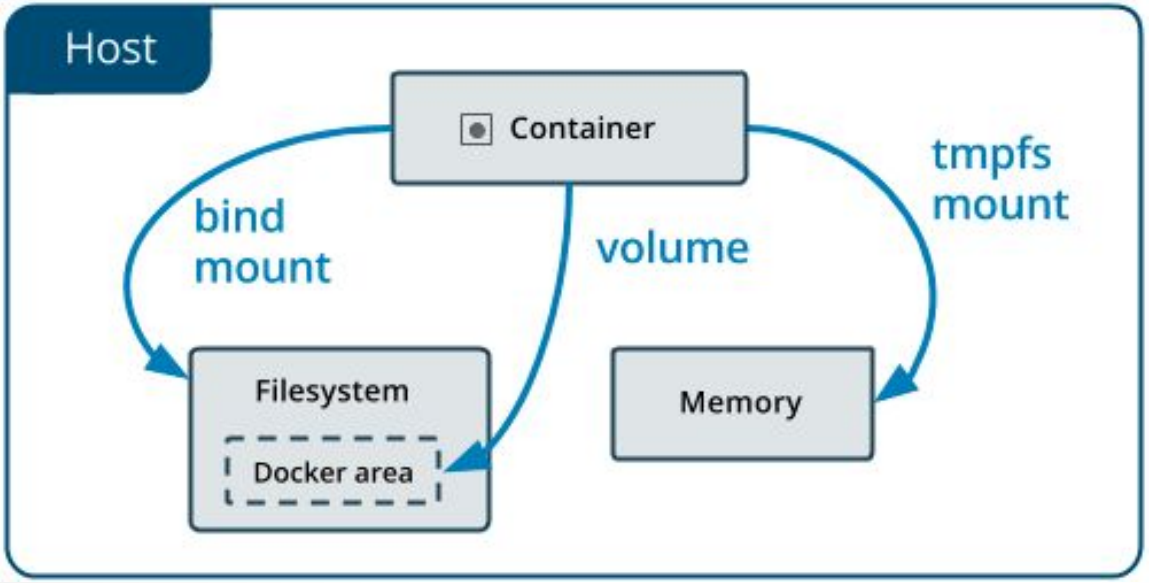
- ตามปกติแล้ว docker จะเก็บข้อมูลทุกอย่างเอาไว้ภายใน container
  - /var/log
  - /sys/data
  - /etc/nagios/
  - /etc/mysql
  - Etc.

- 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
```

- ตรวจสอบการคอนฟิกบน 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

```
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

- NFS3:

```
docker service create -d \
  --name nfs-service \
  --mount
'type=volume,source=nfsvolume,target=/app,volume-driver=local,volume-
opt=type=nfs,volume-opt=device=:/var/docker-nfs,volume-opt=o=addr=10.
0.0.10' nginx:latest
```

- NFS4:

```
docker service create -d \
    --name nfs-service \
    --mount
'type=volume,source=nfsvolume,target=/app,volume-driver=local,volume-
opt=type=nfs,volume-opt=device=:/var/docker-nfs,"volume-opt=o=addr=10
.0.0.10,rw,nfsvers=4,async"' nginx:latest
```


Volume

- CIFS/SAMBA:

```
docker volume create \
--driver local \
--opt type=cifs \
--opt device=//uxxxxx.your-server.de/backup --opt
o=addr=uxxxxx.your-server.de,username=uxxxxxxx,password=*****,file_m
ode=0777,dir_mode=0777 --name cif-volume
```


Volume

lugin	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 *always persist*, 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 environmen

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-persistency space โดย tmpfs ไม่สามารถ Share ระหว่าง container และไม่สามารถใช้งานบน Windows ได้

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

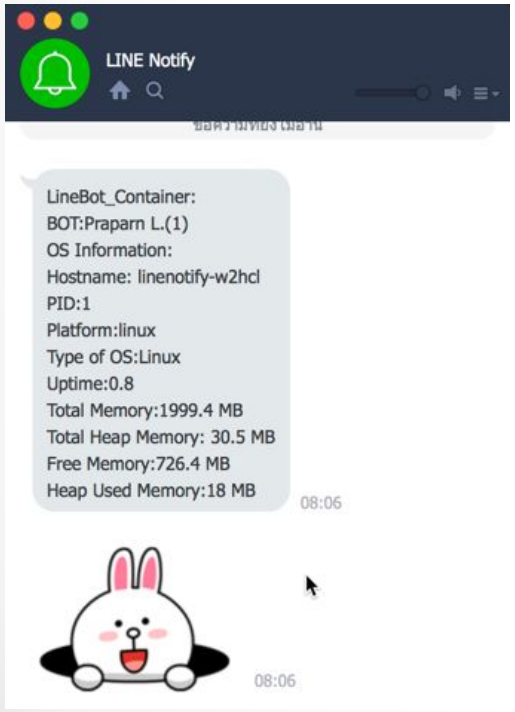
```
Ex: docker volume create --driver local --opt type=tmpfs \
--opt device=tmpfs --opt o=size=100m,uid=1000 tmptest
```

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

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

Workshop: 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”

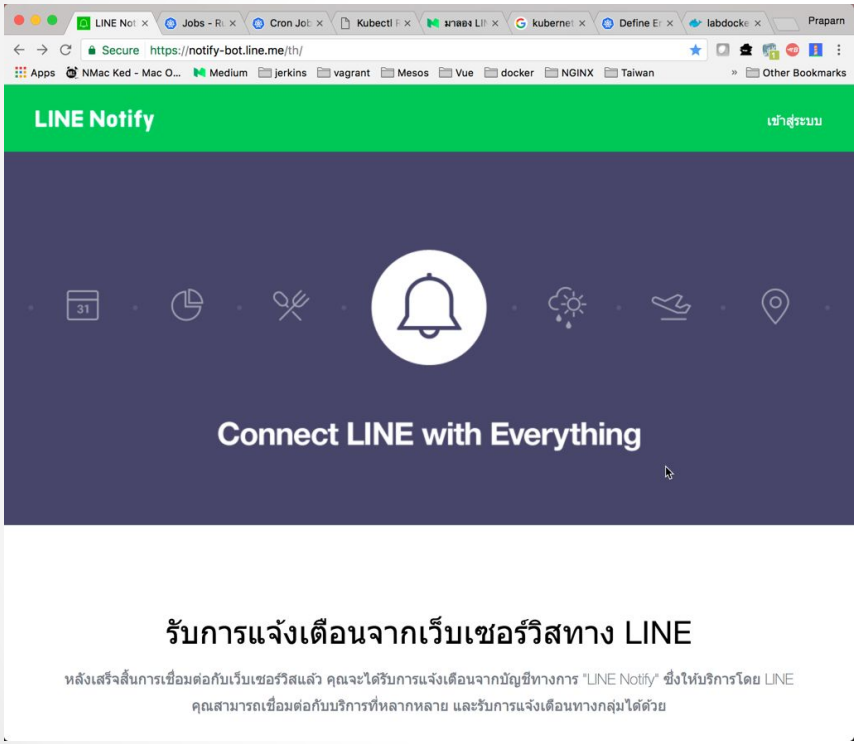


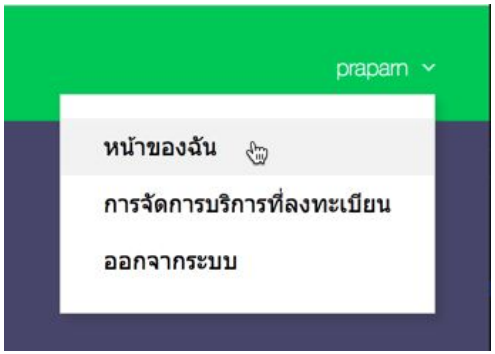
LINE

docker

Workshop: Share Volume

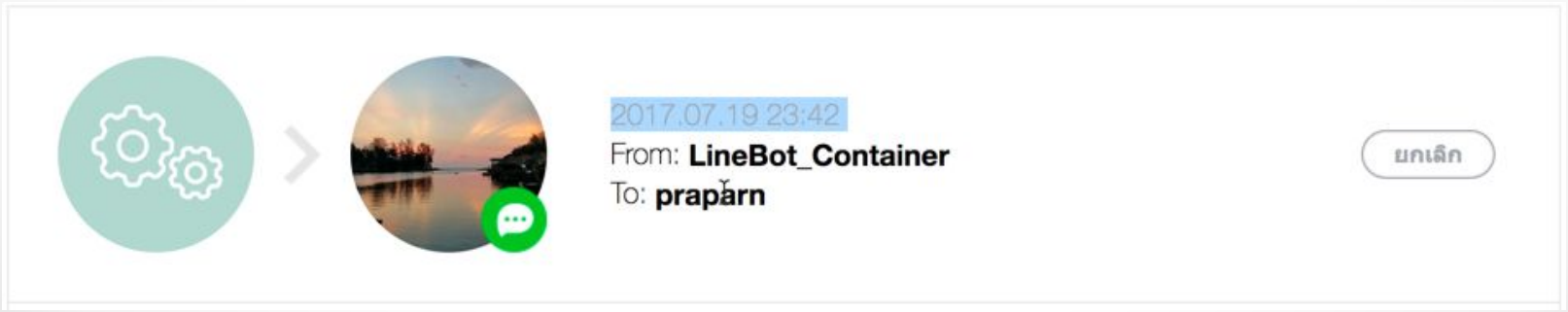
- Generate LINE Token
  - https://notify-bot.line.me





Workshop: Share Volume

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



Workshop: Share Volume

```
const Monitor = require('monitor');
const request = require('request');
const TITLE=process.env.TITLE;
const INTERVAL=process.env.INTERVAL;
const HEAP_HIGH =process.env.HEAP_HIGH;
const MEM_HIGH =process.env.MEM_HIGH;
const SH_OS =process.env.SH_OS;
const TOKEN = process.env.TOKEN;
const critical_StickerPkg = 2;
const critical_StickerId = 39;
const information_StickerPkg = 2;
const information_StickerId = 34;
var options = {
  probeClass: 'Process',
  initParams: {
    pollInterval: INTERVAL
  }
}
var processMonitor = new Monitor(options);
var stickerPkg=0; //stickerPackageId
var stickerId=0; //stickerId
processMonitor.on('change', () => {
  var sendNotify="N" //final decision to send notify
  var heapWarning="N" //flag for heap memory warning
  var memoryWarning="N" //flag for memory warning
```

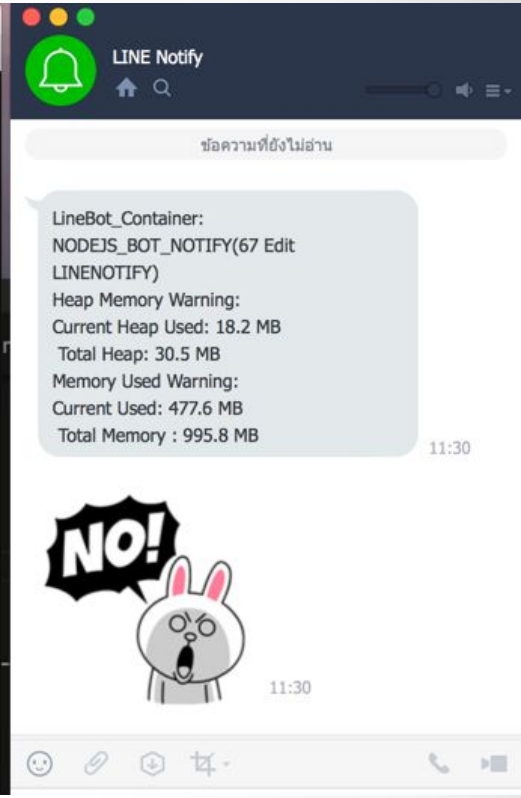
```
var processMonitor = new Monitor(options);
var stickerPkg=0; //stickerPackageId
var stickerId=0; //stickerId
processMonitor.on('change', () => {
  var sendNotify="N" //final decision to send notify
  var heapWarning="N" //flag for heap memory warning
  var memoryWarning="N" //flag for memory warning
  //collect OS information
  var hostName = processMonitor.get('hostname');
  var pID = processMonitor.get('pid');
  var platForm = processMonitor.get('platform');
  var upTime = processMonitor.get('uptime');
  var type = processMonitor.get('type');
  var totalMem = processMonitor.get('totalmem');
  totalMem=Number(((totalMem/1024)/1024).toFixed(1)); //Convert to MB
  var heapTotal = processMonitor.get('heapTotal');
  heapTotal=Number(((heapTotal/1024)/1024).toFixed(1)); //Convert to MB
  var titleMSG="\n"+TITLE+"("+pID+")"
  var osMSG="";
  var warnMSG="";
  //collect OS information
  //collect memory information
  var heapUsed = processMonitor.get('heapUsed');
  heapUsed=Number(((heapUsed/1024)/1024).toFixed(1)); //Convert to MB
  if(HEAP_HIGH<((heapUsed/heapTotal)*100)){heapWarning="Y";}
  var freeMem = processMonitor.get('freemem');
  freeMem=Number(((freeMem/1024)/1024).toFixed(1));
  var usedMem = totalMem-freeMem;
  usedMem=Number(usedMem.toFixed(1)); //Convert to MB
  if(MEM_HIGH<((usedMem/totalMem)*100)){memoryWarning="Y";}
  //collect memory information
```


Workshop: Share Volume

praparnlueangphoonlap — Terminal MAC Pro — ssh ‹ docker-machine ssh labdocker — 102×25

Terminal MAC Pro — ssh ‹ docker-machine ss... | ~ — Terminal MAC Pro — zsh | ~ — Terminal MAC Pro — -bash

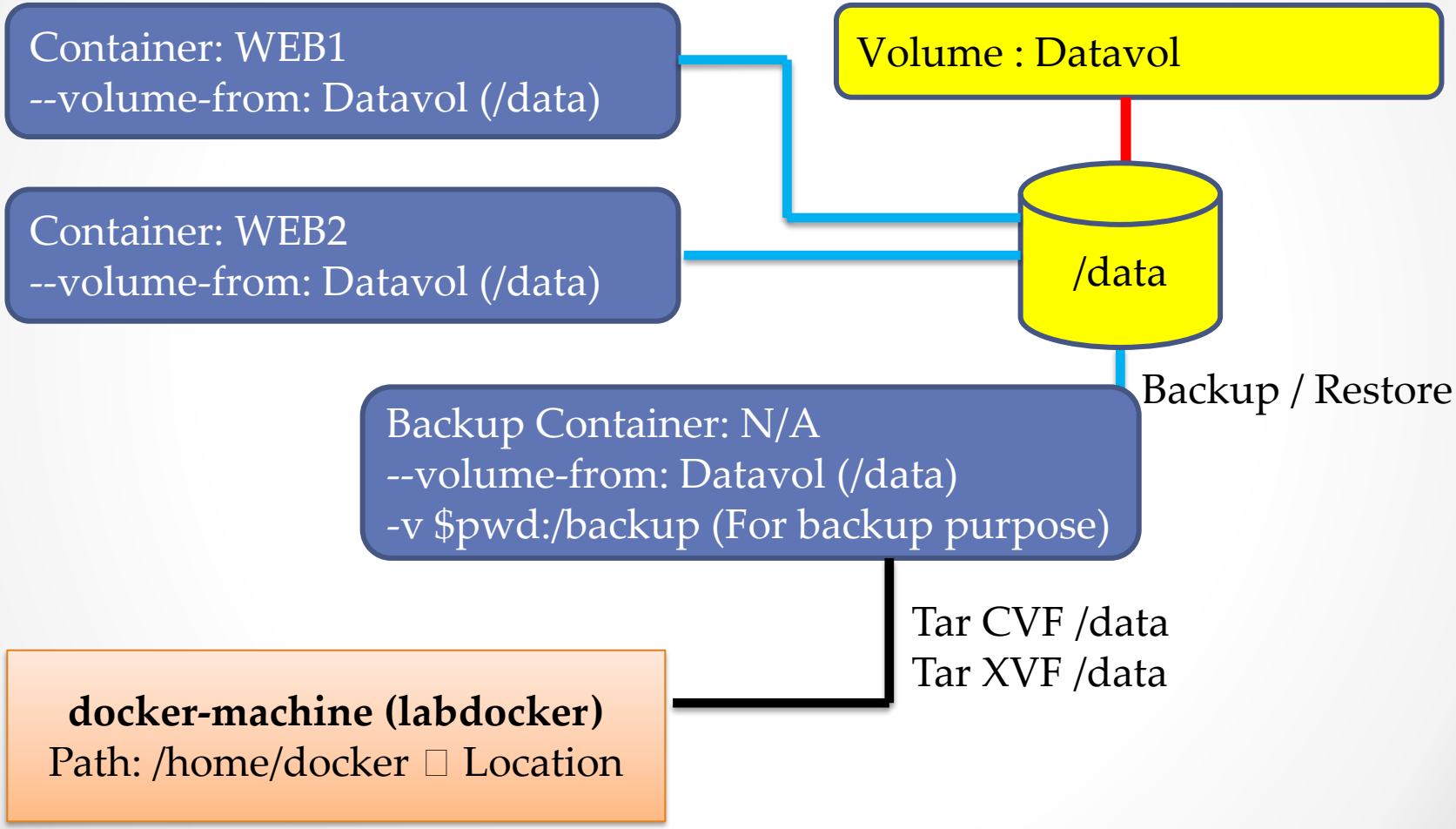

```
mit":"50","x-ratelimit-remaining":"997","x-ratelimit-imageremaining":"50","x-ratelimit-reset":"1501996
936"},"request":{"uri":{"protocol":"https:","slashes":true,"auth":null,"host":"notify-api.line.me","po
rt":443,"hostname":"notify-api.line.me","hash":null,"search":null,"query":null,"pathname":"/api/notify
","path":"/api/notify","href":"https://notify-api.line.me/api/notify"},"method":"POST","headers":{"Con
tent-Type":"application/x-www-form-urlencoded","authorization":"Bearer EIeqyillzkzpaCo1ROrebZTcGbIyzDZ
HpOjcdd0t6CX","content-length":343}}}
"{\"status\":200,\"message\":\"ok\"}"
^C
/nodejs # export HEAP_HIGH=20
/nodejs # export MEM_HIGH=20
/nodejs # export SH_OS=N
/nodejs # node index.js
null
{"statusCode":200,"body":"{\"status\":200,\"message\":\"ok\"}","headers":{"server":"nginx","date":"Sun
, 06 Aug 2017 04:30:41 GMT","content-type":"application/json;charset=UTF-8","transfer-encoding":"chunk
ed","connection":"keep-alive","keep-alive":"timeout=3","x-ratelimit-limit":"1000","x-ratelimit-imageli
mit":"50","x-ratelimit-remaining":"996","x-ratelimit-imageremaining":"50","x-ratelimit-reset":"1501996
936"},"request":{"uri":{"protocol":"https:","slashes":true,"auth":null,"host":"notify-api.line.me","po
rt":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":{"Con
tent-Type":"application/x-www-form-urlencoded","authorization":"Bearer EIeqyillzkzpaCo1ROrebZTcGbIyzDZ
HpOjcdd0t6CX","content-length":290}}}
"{\"status\":200,\"message\":\"ok\"}"
^C
/nodejs #
```

docker

Workshop: 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 หรือกำหนด yyyy-mm-dd
  - --tail = “all” / number of line
  - --until แสดง log ก่อนเวลาที่กำหนด (Ex: 20180619220001) หรือ Relative (Ex: 42m = 42 minute before)
  - --details แสดงรายละเอียดเพิ่มเติมของล็อกเช่น Env variable, labels โดยจะเป็นรายละเอียดเพิ่มเติมจากการใส่ option --log-opt ตอนสร้าง container

Log

- Server side. Big problem will come with logging !!!
- All container log will keep on "/var/lib/docker/containers/<container id>/<container-id>-json.log"and it take mega size on this path until full
- We have 2 option for manage this
- Global Configuration
  - Add daemon.json on path /etc/docker/daemon.json for global apply (need to restart docker daemon for take effect)

```
{
  "exec-opts": ["native.cgroupdriver=systemd"],
  "log-driver": "json-file",
  "log-opts": {
    "max-size": "100m",
    "max-file": "10"
  },
  "storage-driver": "overlay2"
}
```


Log

Driver	Description
`none`	No logs are available for the container and `docker logs` does not return any output.
`local`	Logs are stored in a custom format designed for minimal overhead.
`json-file`	The logs are formatted as JSON. The default logging driver for Docker.
`syslog`	Writes logging messages to the `syslog` facility. The `syslog` daemon must be running on the host machine.
`journald`	Writes log messages to `journald`. The `journald` daemon must be running on the host machine.
`gelf`	Writes log messages to a Graylog Extended Log Format (GELF) endpoint such as Graylog or Logstash.
`fluentd`	Writes log messages to `fluentd` (forward input). The `fluentd` daemon must be running on the host machine.
`awslogs`	Writes log messages to Amazon CloudWatch Logs.
`splunk`	Writes log messages to `splunk` using the HTTP Event Collector.
`etwlogs`	Writes log messages as Event Tracing for Windows (ETW) events. Only available on Windows platforms.
`gcplogs`	Writes log messages to Google Cloud Platform (GCP) Logging.
`logentries`	Writes log messages to Rapid7 Logentries.

Ref https://docs.docker.com/config/containers/logging/configure/

Log

- Container Configuration
  - Specific on run command of docker container
    - --log-driver json-file (default)
    - --log-opt max-size=10m
    - --log-opt max-file=10

```
ubuntu@ip-10-0-1-55:~$ docker container run -dt --name nginx -p 80:80 \
> --log-driver json-file \
> --log-opt max-size=10m \
> --log-opt max-file=10 \
> labdocker/nginx:badversion
07af88d6368bdfd8a74608882963a50052e5c887fc32da812369d78068643e51
ubuntu@ip-10-0-1-55:~$
```


Inspect

- การตรวจสอบค่าคอนฟิกของ container / network / volume etc

```
docker inspect <container >
```

```
docker network inspect
```

```
docker volume inspect
```


Workshop: 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>
```

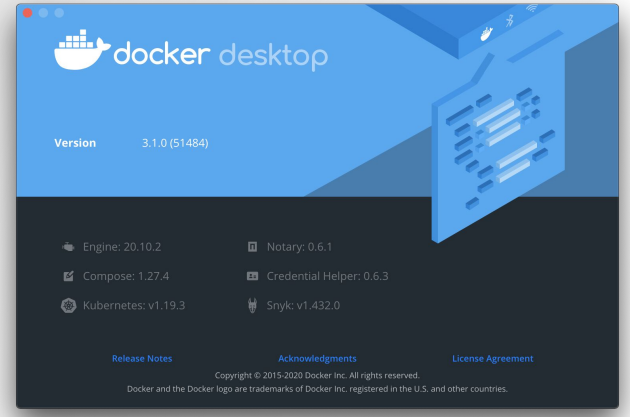
- options
  - -a, --author=“ “ ระบุผู้จัดทำ image
  - -c, --change = [ ] เพิ่มเติม command พิเศษที่จะจัดเก็บภายใน image
  - -m, --message = “ “ ระบุ comment ใน image ที่จัดเก็บ
  - -p, --pause=true กำหนดให้หยุดการใช้งาน container ชั่วคราวในระหว่างทำการ commit

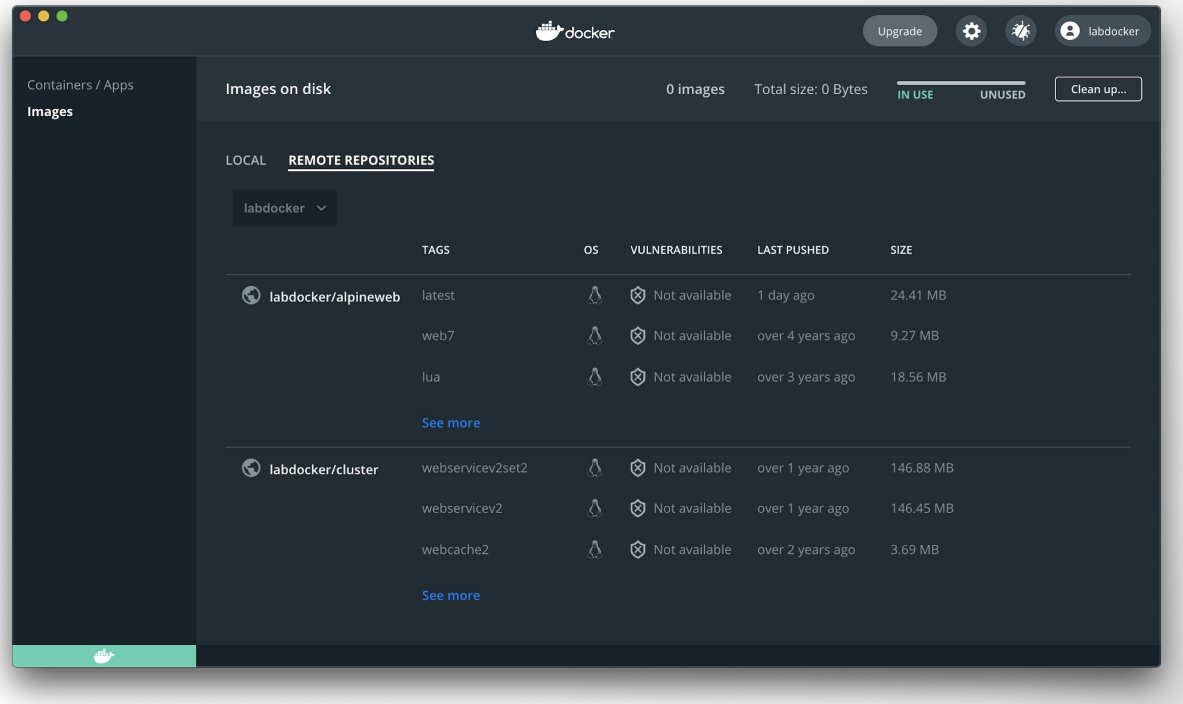
Docker Desktop

• • •

Docker Desktop

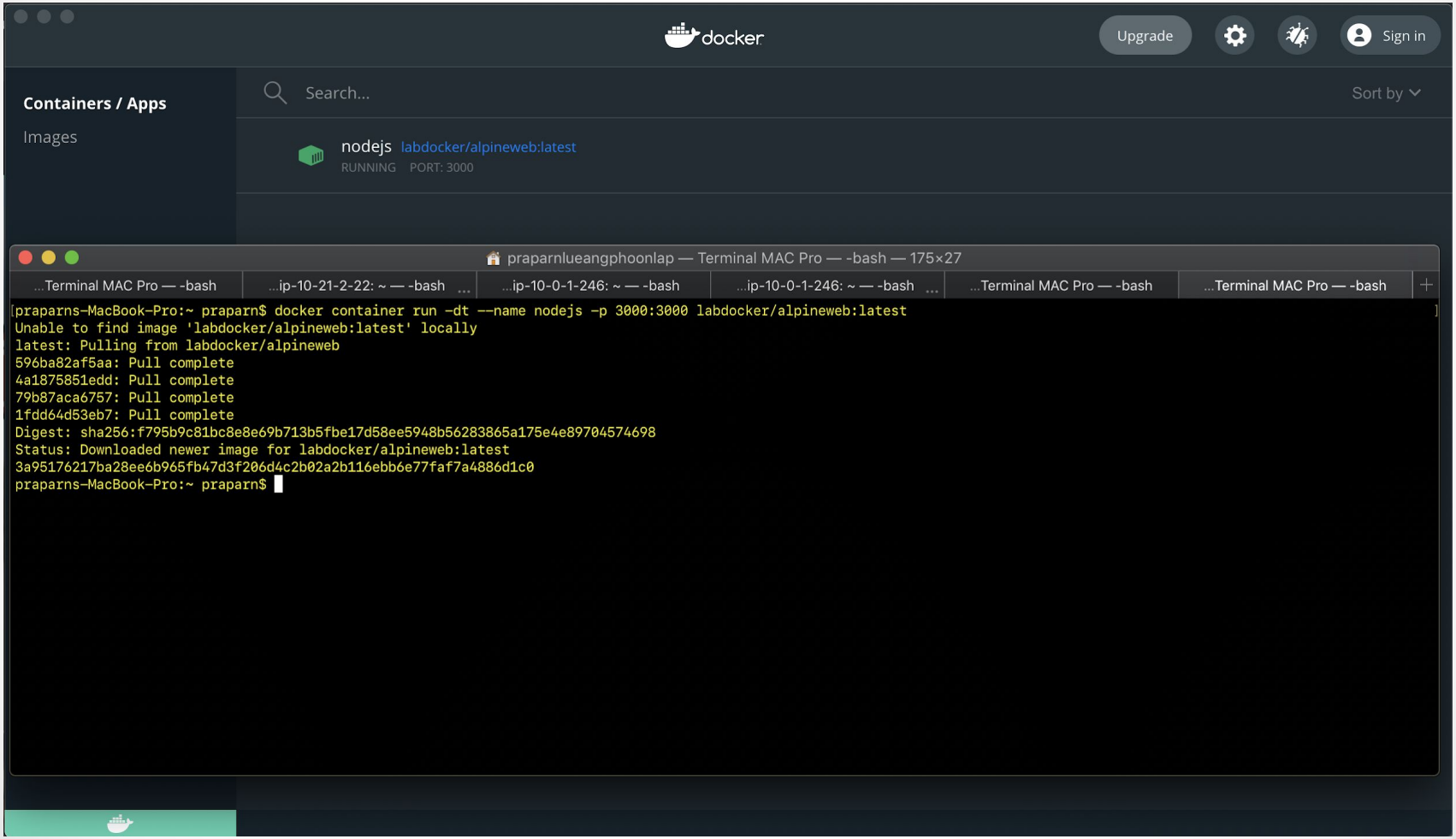
- GUI tool for developer/devops on your machine





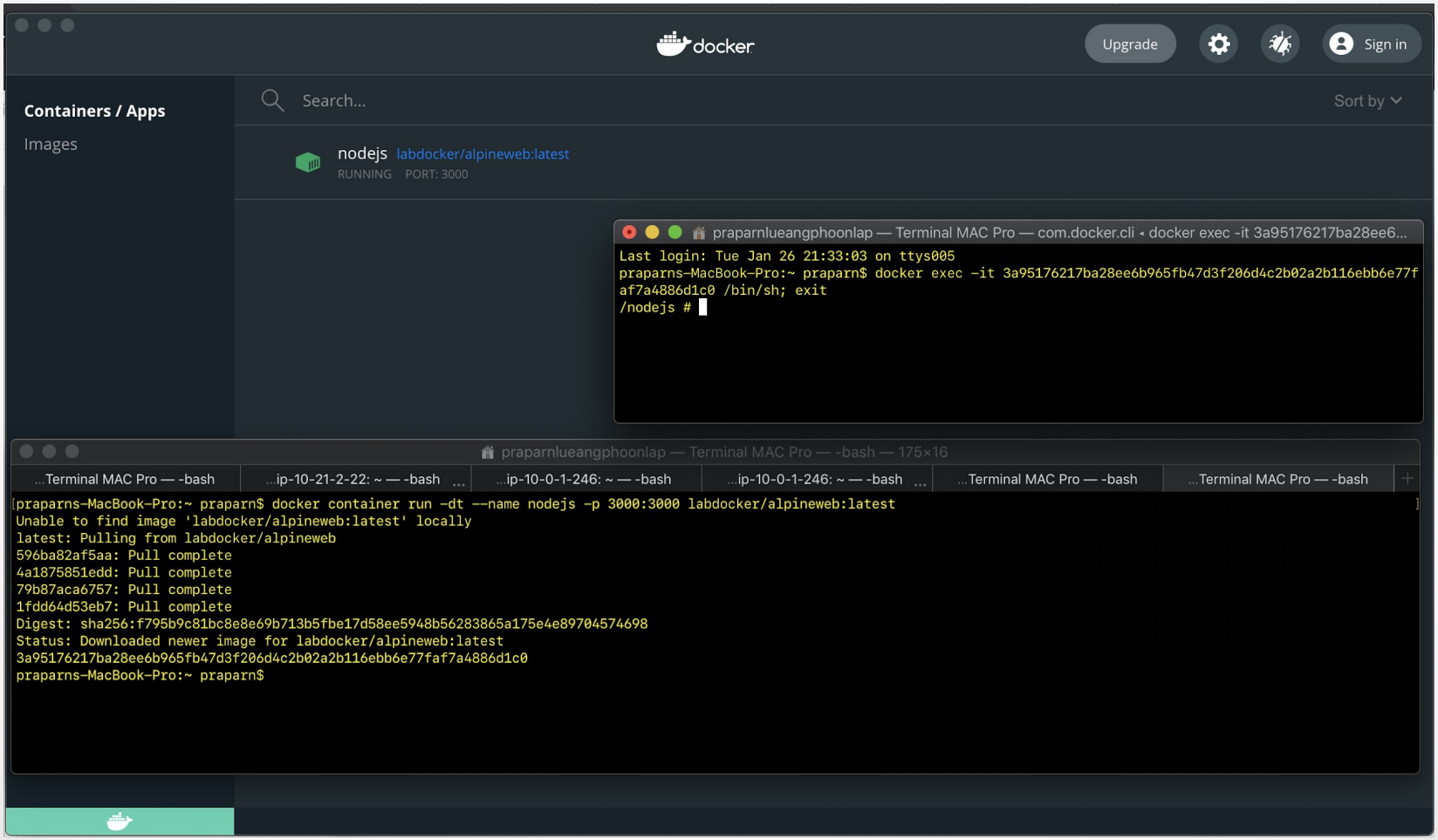
Docker Desktop

- GUI tool for developer/devops on your machine



Docker Desktop

- GUI tool for developer/devops on your machine

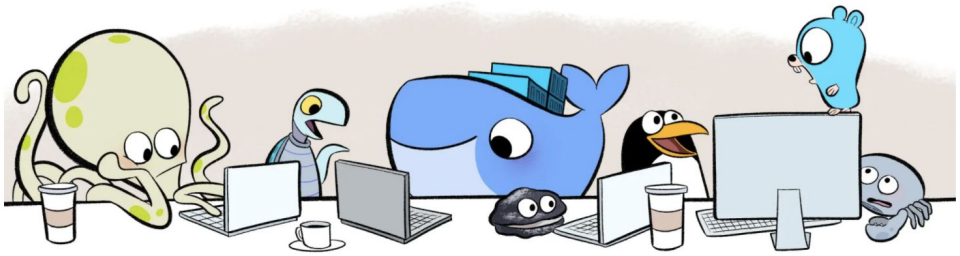


docker

Docker Desktop

Docker is Updating and Extending Our Product Subscriptions

SCOTT JOHNSTON
Aug 31 2021

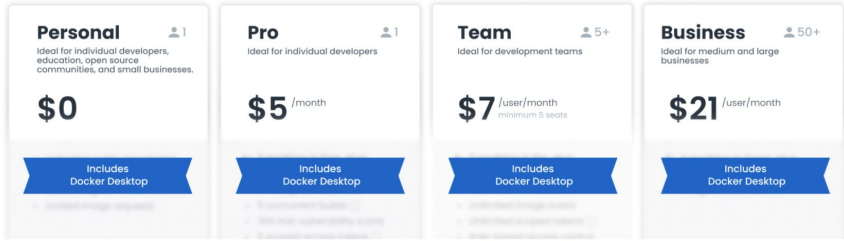


Docker is used by millions of developers to build, share, and run any app, anywhere, and 55% of professional developers use Docker every day at work. In these work environments, the increase in outside attacks on software supply chains is accelerating developer demand for Docker's trusted content, including Docker Official Images and Docker Verified Publisher images. Finally, the rapid global growth in developers – to an estimated 45 million by 2030 – pushes us to scale sustainably so we may continue to provide an innovative, free Docker experience that developers love.

To meet these challenges, today we're announcing updates and extensions to our product subscriptions: **Personal, Pro, Team, and Business.** These updated product subscriptions provide the productivity and collaboration developers rely on with the scale, security, and trusted content businesses require, and do so in a manner sustainable for Docker.

What you need to know:

- We're introducing a **new** product subscription, Docker Business, for organizations using Docker at scale for application development and require features like secure software supply chain management, single sign-on (SSO), container registry access controls, and more.
- Our Docker Subscription Service Agreement includes a change to the terms for **Docker Desktop**:
  - Docker Desktop **remains free** for small businesses (fewer than 250 employees AND less than $10 million in annual revenue), personal use, education, and non-commercial open source projects.
  - It requires a paid subscription (**Pro**, **Team** or **Business**), starting at $5 per user per month, for professional use in larger businesses. You may directly purchase here, or share this post and our solution brief with your manager.
  - While the effective date of these terms is August 31, 2021, there is a grace period until January 31, 2022 for those that require a paid subscription to use Docker Desktop.
- Docker Pro, Docker Team, and Docker Business subscriptions **include** commercial use of Docker Desktop.
- The **existing** Docker Free subscription has been renamed Docker Personal.
- **No changes** to Docker Engine or any upstream open source Docker or Moby project.
- *Check out our FAQ for more information.*



docker

Recapture for Part 1

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

Q&A for Day 1

• • •

Outline Part 2

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