Docker:Zero to Hero (Part 2)

By Praparn Luengphoonlap
Email: eva10409@gmail.com

docker

Outline Day 2

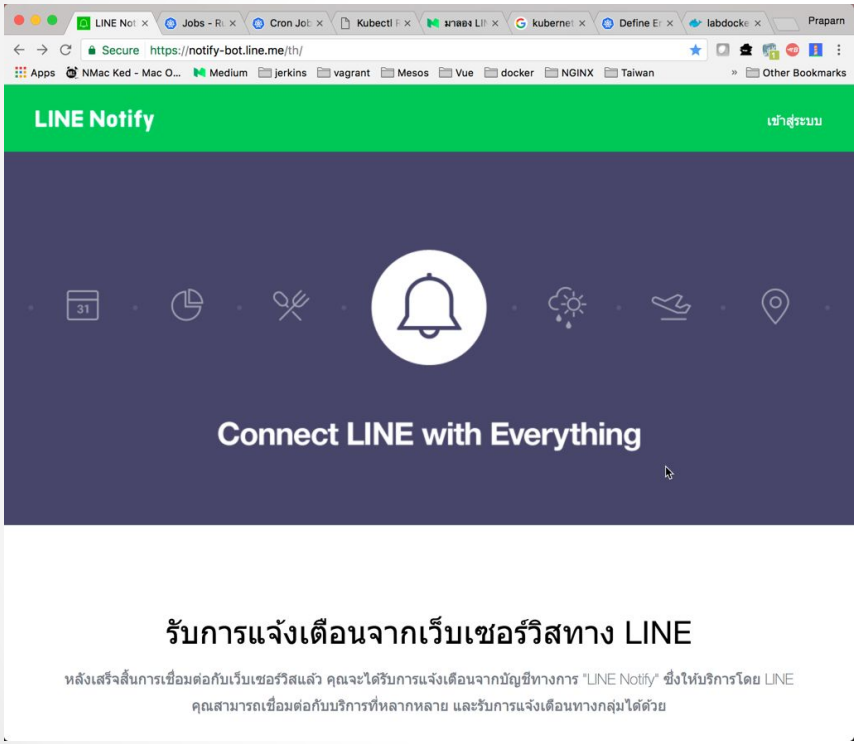
- Dockerfile and Build
- Compose
- Docker Security
- Registry
- Swarm Mode
  - Conceptual of Swarm
  - Swarm Mode Architecture
  - Swarm init/join cluster system
  - Swarm service
  - Orchestrator Assignment
  - Config and Secret
  - Network Overlay and Ingress
  - HA Manager Role
  - Docker Stack Deploy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

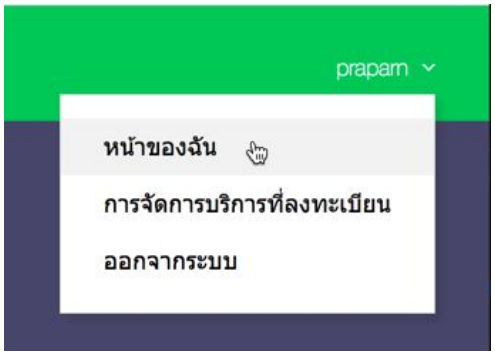
Prerequisite

- 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

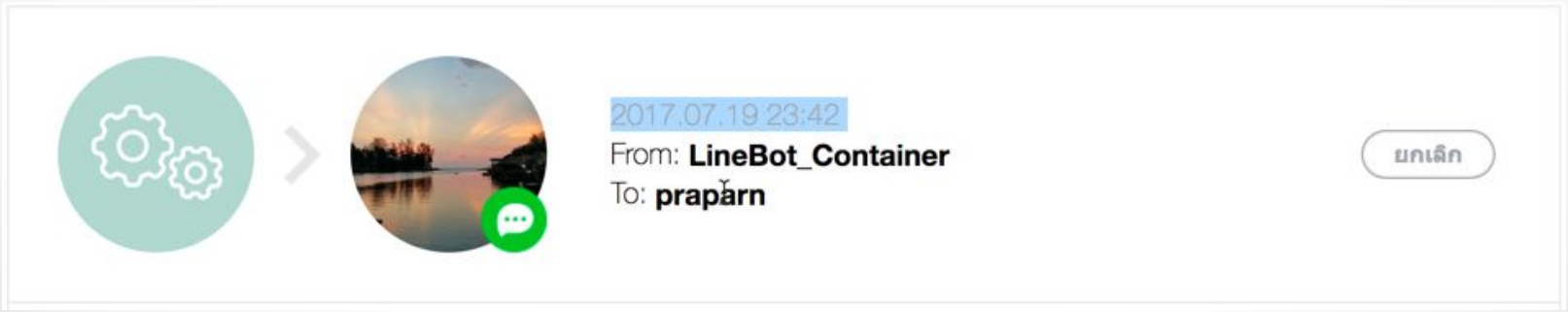
- Generate LINE Token
  - https://notify-bot.line.me





Prerequisite

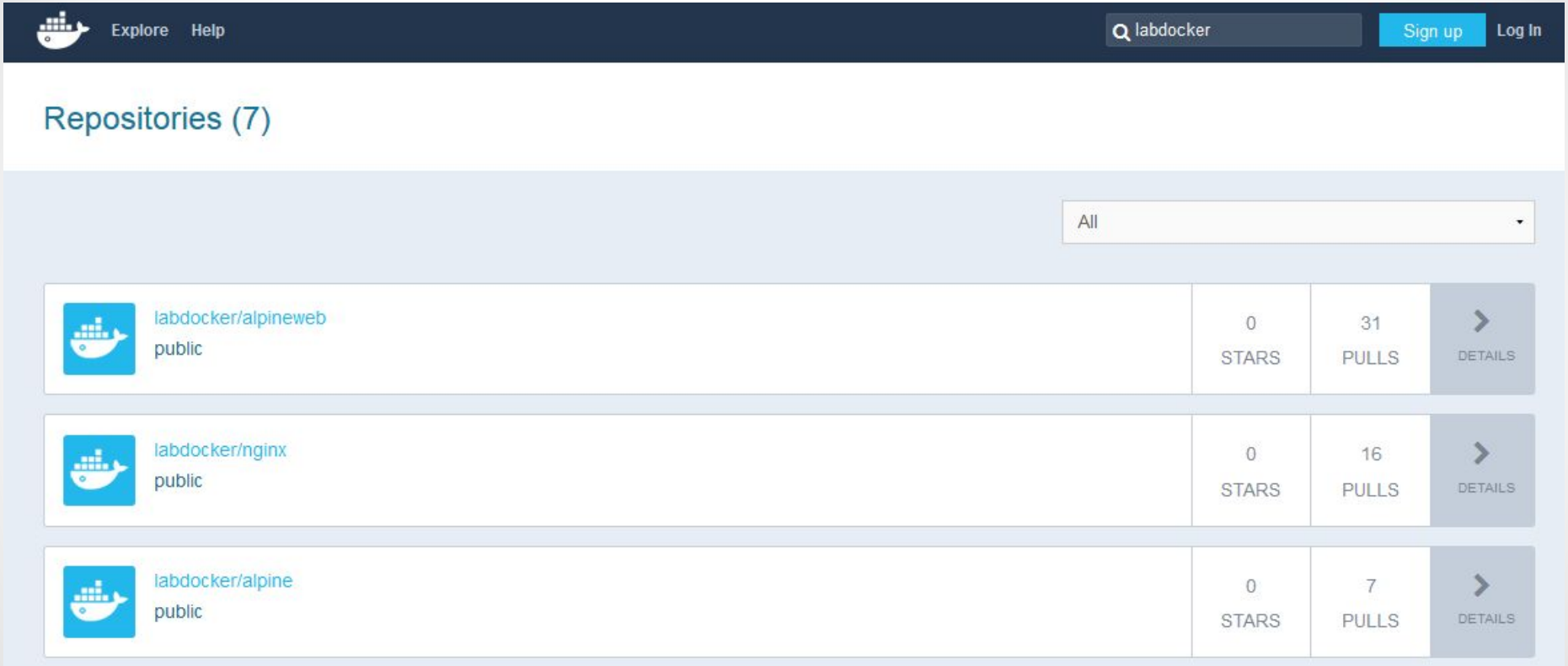
- Generate LINE Token
  - https://notify-bot.line.me



docker

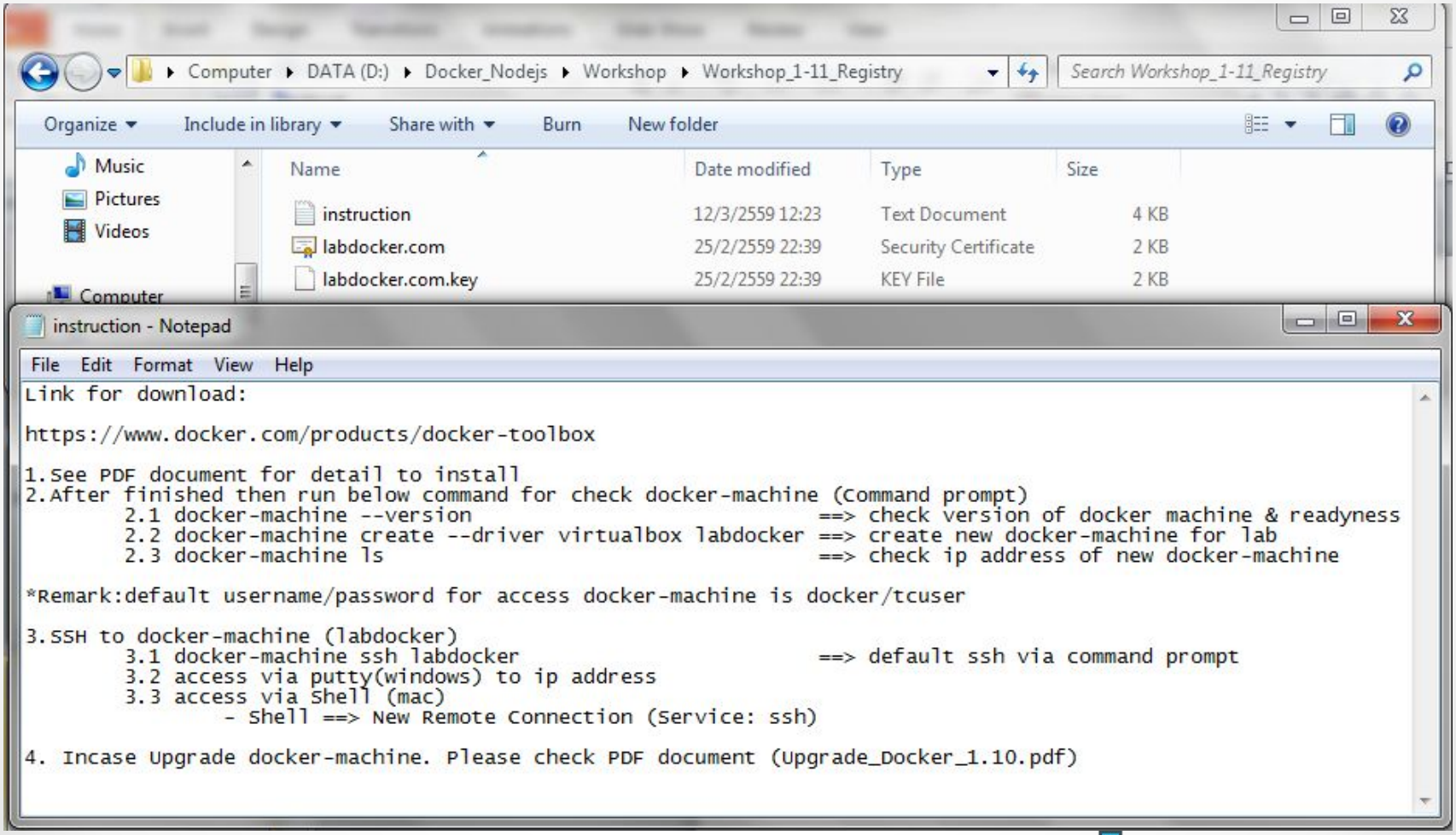
Lab Resource

- Repository for lab



Lab Resource

- Software in lab



docker

Lab Resource

- Download on GitHub
  - git clone https://github.com/praparn/docker-workshop-online-part2

Dockerfile and Build

• • •

Dockerfile

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

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


Dockerfile

- .dockerignore file exclude some file/folder that not sent to build in image

```
# comment
*/temp*
*/*/temp*
temp?
*.md
!README*.md
```


Dockerfile

- Option สำหรับคำสั่ง docker image build (Partial)
  - --add-host Add customer /etc/hosts
  - --build-arg Set build-time variables
  - --cache-from Image to consider
  - --cgroup-parent Specific parent control group
  - --compress Compress context with gzip
  - --cpu-shares CPU shares
  - --cpuset-cpus CPU no. for use
  - --memory, -m Memory limit
  - --disable-content -trust image verification
  - --file, -f Specify docker file
  - --rm Remove intermediate image after build finished
  - --force-rm Force remove intermediate image
  - --network Set network in build operate (RUN)
  - --no-cache Never use cache
  - --pull Always attempt pull new version image
  - --tag, -t Tag image name

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

Dockerfile

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

Ref: https://docs.docker.com/develop/develop-images/dockerfile_best-practices/

Dockerfile

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


Dockerfile

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

RUN a2enmod rewrite

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

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

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


Dockerfile

- ควรระบุ WORKDIR เพื่อกำหนด path เริ่มต้นในการทำงานทุกครั้ง (before ENTRYPOINT, RUN, CMD, COPY etc)

- ADD/COPY operation
  - COPY เป็น basic transfer simple file
  - ADD ใช้ในกรณี remote file URL, self extract (.tar) (Not recommend)

- ENV PATH ควรระบุ path ของ executable ที่จำเป็นต้องใช้งาน

- พิจารณาการใช้ cache ของ image layer โดยเราสามารถ ignore cache ด้วย option "--no-cache=true"
  - ตามปกติ docker จะเปรียบเทียบ instruction ที่กำหนดไว้กับ image layer ที่
  - Consider: instruction, base image
- ควรระบุ HealthCheck เพื่อตรวจสอบการทำงานของ container

Docker Image Version

- Docker start to introduce standard of image manifest and schema on version 1.13.0 with version1 (deprecate)
- So standard image version is base on docker version that install on machine to build image
- In case that we run the deprecate version of docker image (schema v1). Docker will show warning during start container
- Converting image manifest is possible just "docker image pull" and "docker image push" from docker machine with up-to-date version. Docker will convert image format automatically
- Current docker image have 3 release of manifest
  - Manifest v1, Schema v1 (Deprecate)
  - Manifest v2, Schema v1 (Many vulnerability)
  - Manifest v2, Schema v2 (Recommend)

Docker Image Version

- Main feature of "Manifest v2, Scheme v2
  - Support multi-architecture (fat manifest) image for contain single image
  - Keep image configuration to hash as id and embed on image
- Verify manifest and schema version by command:

```
docker manifest inspect <image version>
```

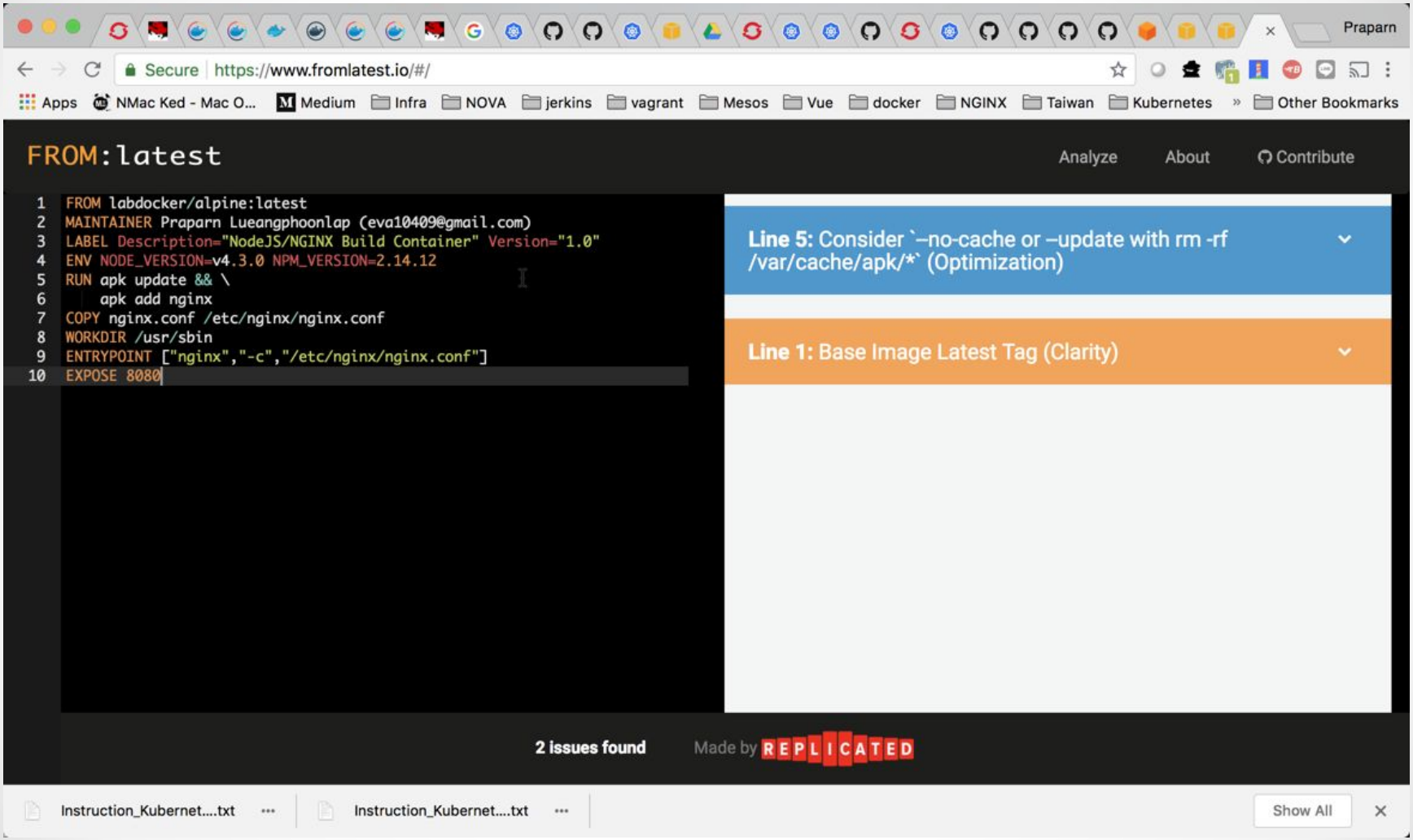
  - application/vnd.docker.distribution.manifest.v1+json
  - application/vnd.docker.distribution.manifest.v2+json
  - application/vnd.docker.distribution.manifest.list.v2+json
  - application/vnd.docker.container.image.v1+json
  - application/vnd.docker.image.rootfs.diff.tar.gzip
  - application/vnd.docker.image.rootfs.foreign.diff.tar.gzip
  - application/vnd.docker.plugin.v1+json

Docker Image Version

- Verify manifest and schema version by command:

```
ubuntu@ip-10-0-1-65:~$ docker manifest inspect labdocker/alpine:latest
{
        "schemaVersion": 2,
        "mediaType": "application/vnd.docker.distribution.manifest.v2+json",
        "config": {
                "mediaType": "application/vnd.docker.container.image.v1+json",
                "size": 1472,
                "digest": "sha256:7731472c3f2a25edbb9c085c78f42ec71259f2b83485aa60648276d408865839"
        },
        "layers": [
                {
                        "mediaType": "application/vnd.docker.image.rootfs.diff.tar.gzip",
                        "size": 2810825,
                        "digest": "sha256:596ba82af5aaa3e2fd9d6f955b8b94f0744a2b60710e3c243ba3e4a467f051d1"
                }
        ]
}
ubuntu@ip-10-0-1-65:~$
```

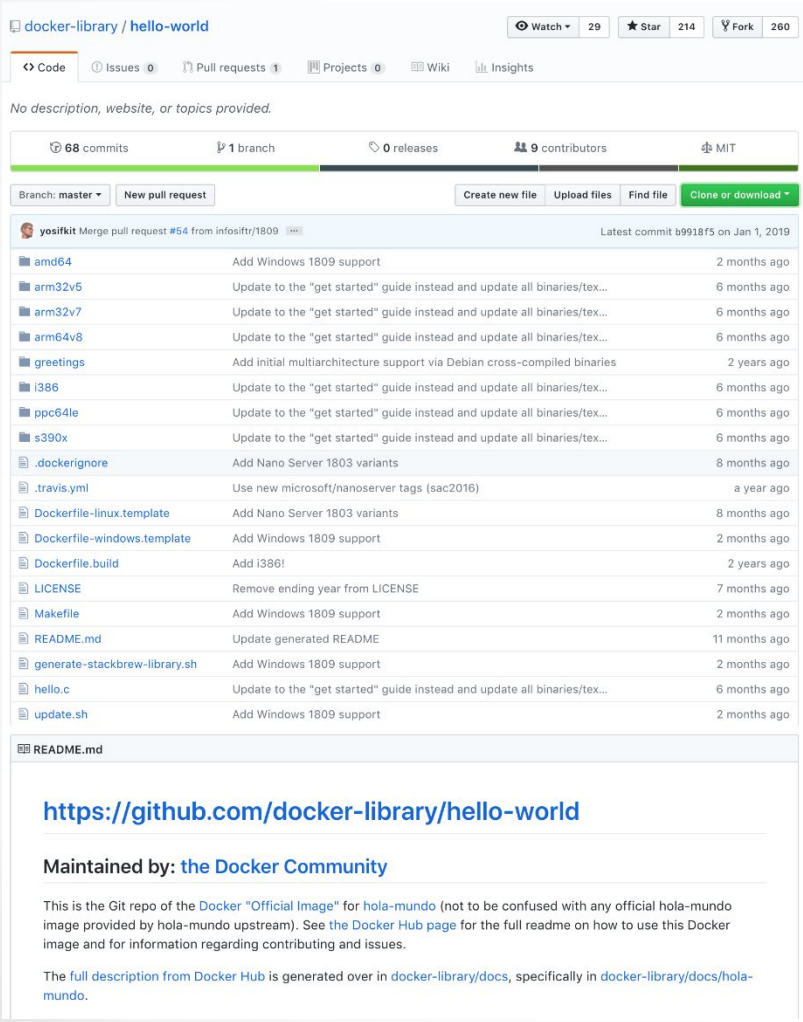

How can check dockerfile ?



docker

Workshop: DockerFile

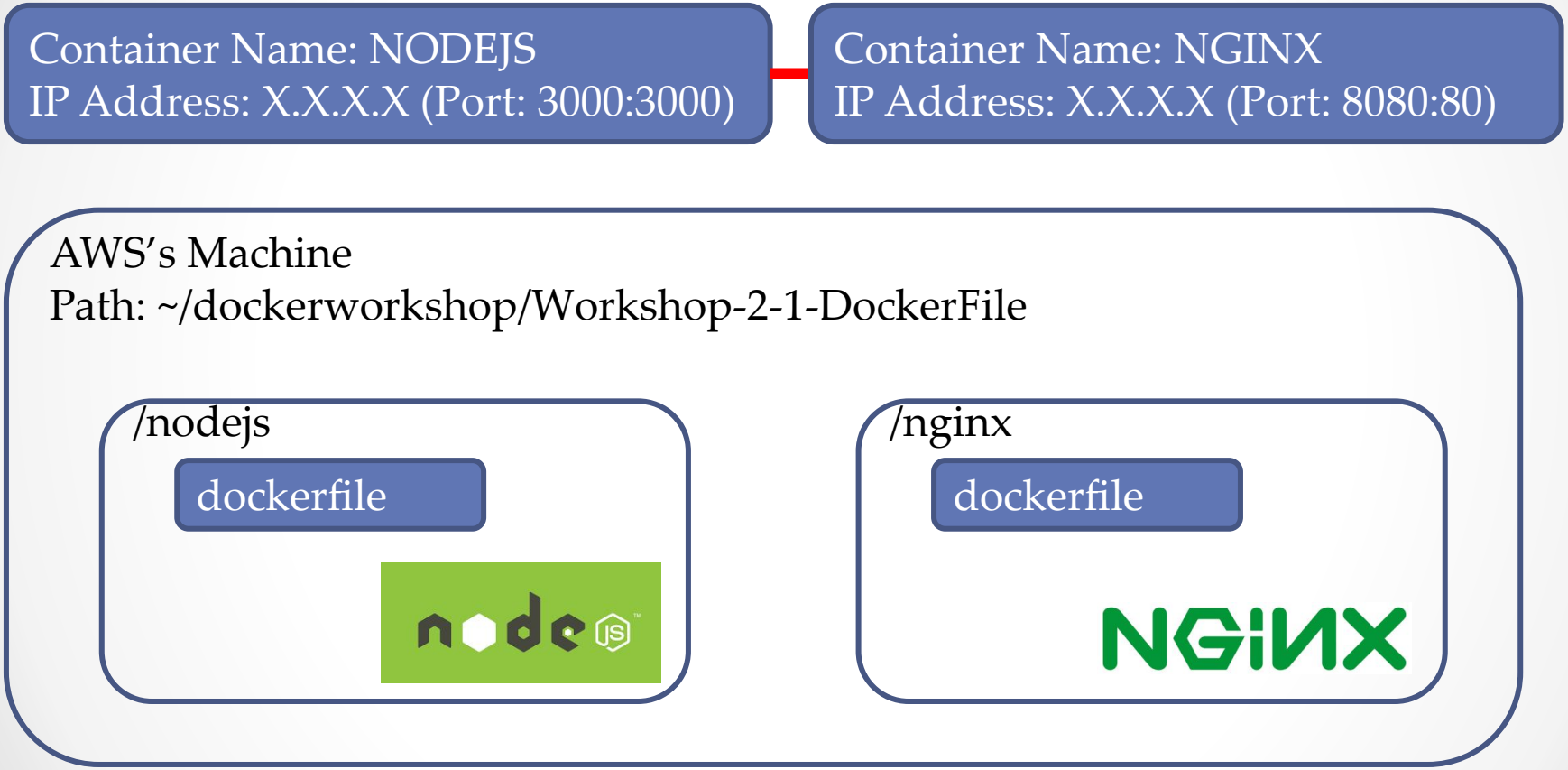
- Part1: SCRATCH




```
instruction.txt  nginx  nodejs  python_restfulset  scratch
ubuntu@ip-10-0-1-104:~$ docker image build -t labdocker/hello-world:1.0 ~/docker-workshop-042019/Workshop-2-1-DockerFile/scratch/
Sending build context to Docker daemon   16.9kB
Step 1/3 : FROM scratch
 --->
Step 2/3 : COPY hello /
 ---> Using cache
 ---> 937200b29047
Step 3/3 : CMD ["/hello"]
 ---> Using cache
 ---> e759df7ef1b1
Successfully built e759df7ef1b1
Successfully tagged labdocker/hello-world:1.0
ubuntu@ip-10-0-1-104:~$ docker container run labdocker/hello-world:1.0

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/

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


Workshop: DockerFile

- Part2: NGINX/NODEJS



Workshop: DockerFile

- ตรวจสอบ dockerfile ตามตัวอย่างด้านล่าง

```
FROM labdocker/alpine:3.13.6
LABEL maintainer="Praparn Lueangphoonlap (eva10409@gmail.com)"
LABEL Description="NodeJS/NGINX Build Container" Version="1.0"
ENV NODE_VERSION=v14.15.4 NPM_VERSION=v14.15.4
RUN apk update && \
    apk add nodejs nodejs-npm curl --no-cache && \
    rm -rf /var/cache/apk/*
RUN mkdir /nodejs
COPY hello.js /nodejs/
WORKDIR /nodejs
ENTRYPOINT ["node","hello.js"]
EXPOSE 3000
HEALTHCHECK --interval=15s --timeout=5s --start-period=10s --retries=3 \
    CMD curl -f http://localhost:3000/ || exit 1
```

docker image build -t labdocker/node:1.0
~/docker-workshop-072019/Workshop-2-1-DockerFile/nodejs

Workshop: DockerFile

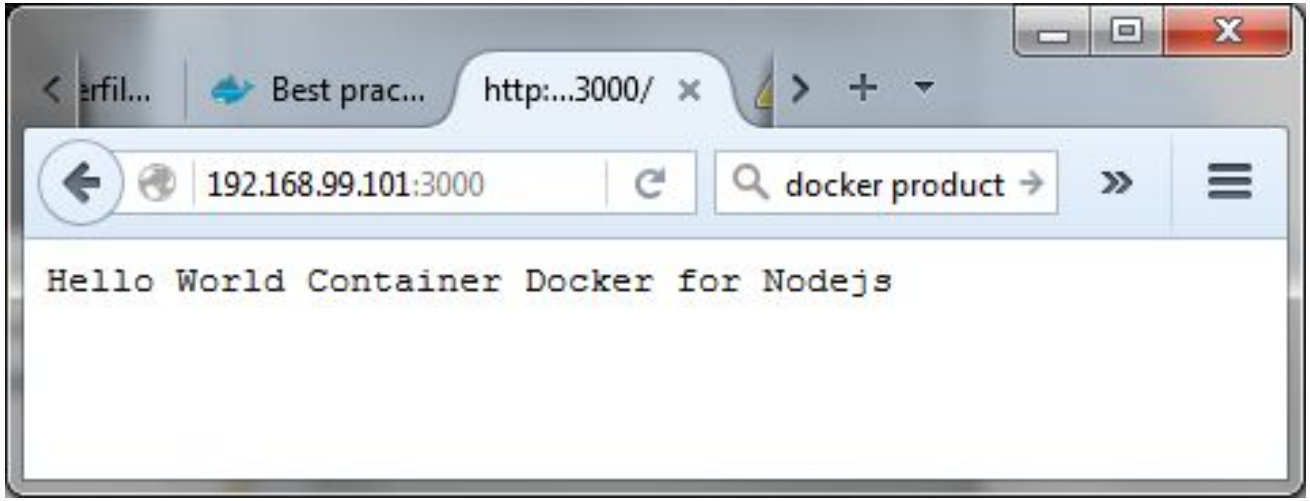
- Output

```
Sending build context to Docker daemon 3.072 kB
Step 1 : FROM labdocker/alpine:latest
 ---> 14f89d0e6257
Step 2 : MAINTAINER Praparn Lueangphoonlap (eva10409@gmail.com)
 ---> Using cache
 ---> 10f80fb4997d
Step 3 : LABEL Description "NodeJS/NGINX Build Container" Version "1.0"
 ---> Using cache
 ---> 59bc5a927e19
Step 4 : ENV NODE_VERSION v4.3.0 NPM_VERSION 2.14.12
 ---> Using cache
 ---> ce23d23959b7
Step 5 : RUN apk update &&     apk add nodejs
 ---> Running in b12c752ef6a0
fetch http://dl-4.alpinelinux.org/alpine/v3.3/main/x86_64/APKINDEX.tar.gz
fetch http://dl-4.alpinelinux.org/alpine/v3.3/community/x86_64/APKINDEX.tar.gz
v3.3.1-99-gc7d905f [http://dl-4.alpinelinux.org/alpine/v3.3/main]
v3.3.1-59-g48b0368 [http://dl-4.alpinelinux.org/alpine/v3.3/community]
OK: 5857 distinct packages available
(1/4) Installing libgcc (5.3.0-r0)
(2/4) Installing libstdc++ (5.3.0-r0)
(3/4) Installing libuv (1.7.5-r0)
(4/4) Installing nodejs (4.3.0-r0)
Executing busybox-1.24.1-r7.trigger
OK: 29 MiB in 15 packages
 ---> 8f2bf208ab86
Removing intermediate container b12c752ef6a0
Step 6 : RUN mkdir /nodejs
 ---> Running in 2ee51d1af642
 ---> d3edf2c2f511
Removing intermediate container 2ee51d1af642
Step 7 : COPY hello.js /nodejs/
 ---> f703ccbc7dd4
Removing intermediate container 952935d69cce
Step 8 : CMD nginx -c /etc/nginx/nginx.conf
 ---> Running in 67940385f5ac
 ---> edfba609e20a
```


Workshop: DockerFile

- Test run

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



Workshop: DockerFile

- ตรวจสอบ dockerfile ตามตัวอย่างด้านล่าง

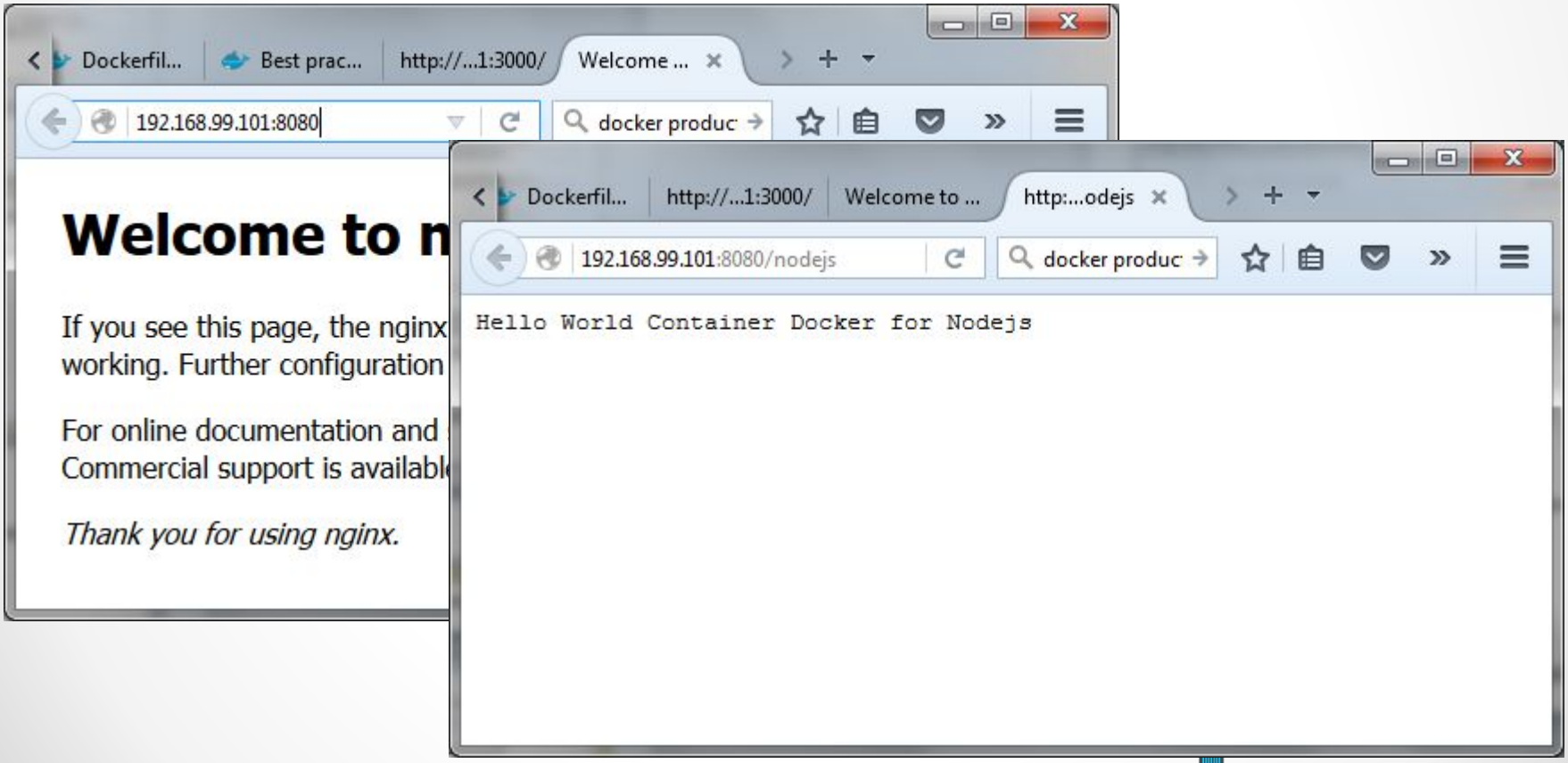
```
FROM labdocker/alpine:3.13.6
LABEL maintainer="Praparn Lueangphoonlap (eva10409@gmail.com)"
LABEL Description="NodeJS/NGINX Build Container" Version="1.0"
ENV NODE_VERSION=v14.15.4 NPM_VERSION=v14.15.4
RUN mkdir -p /run/nginx
RUN apk update && \
    apk add nginx curl --no-cache && \
    rm /etc/nginx/conf.d/default.conf && \
    rm -rf /var/cache/apk/*
COPY nginx.conf /etc/nginx/nginx.conf
WORKDIR /usr/sbin
ENTRYPOINT ["nginx","-c","/etc/nginx/nginx.conf"]
EXPOSE 8080
HEALTHCHECK --interval=15s --timeout=5s --start-period=10s --retries=3 \
    CMD curl -f http://localhost:8080/ || exit 1
```

docker build -t labdocker/web:1.0
~/docker-workshop-072019/Workshop-2-1-DockerFile/nginx

Workshop: DockerFile

- Test run

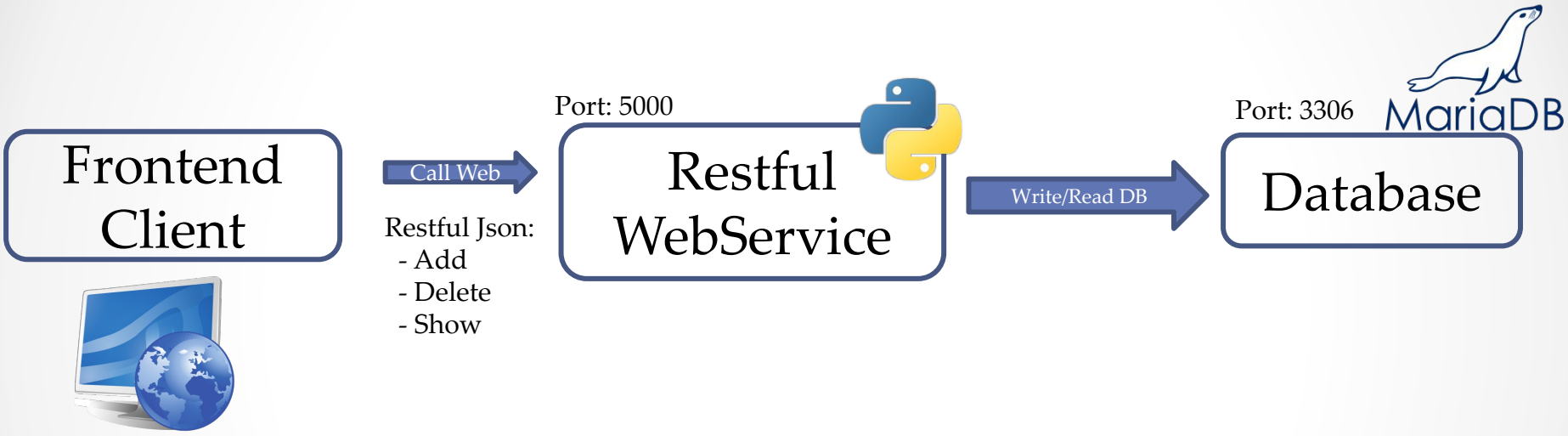
```
docker container run -dt --name NGINX -p 8080:8080 -p
8443:8443labdocker/web:1.0
```



docker

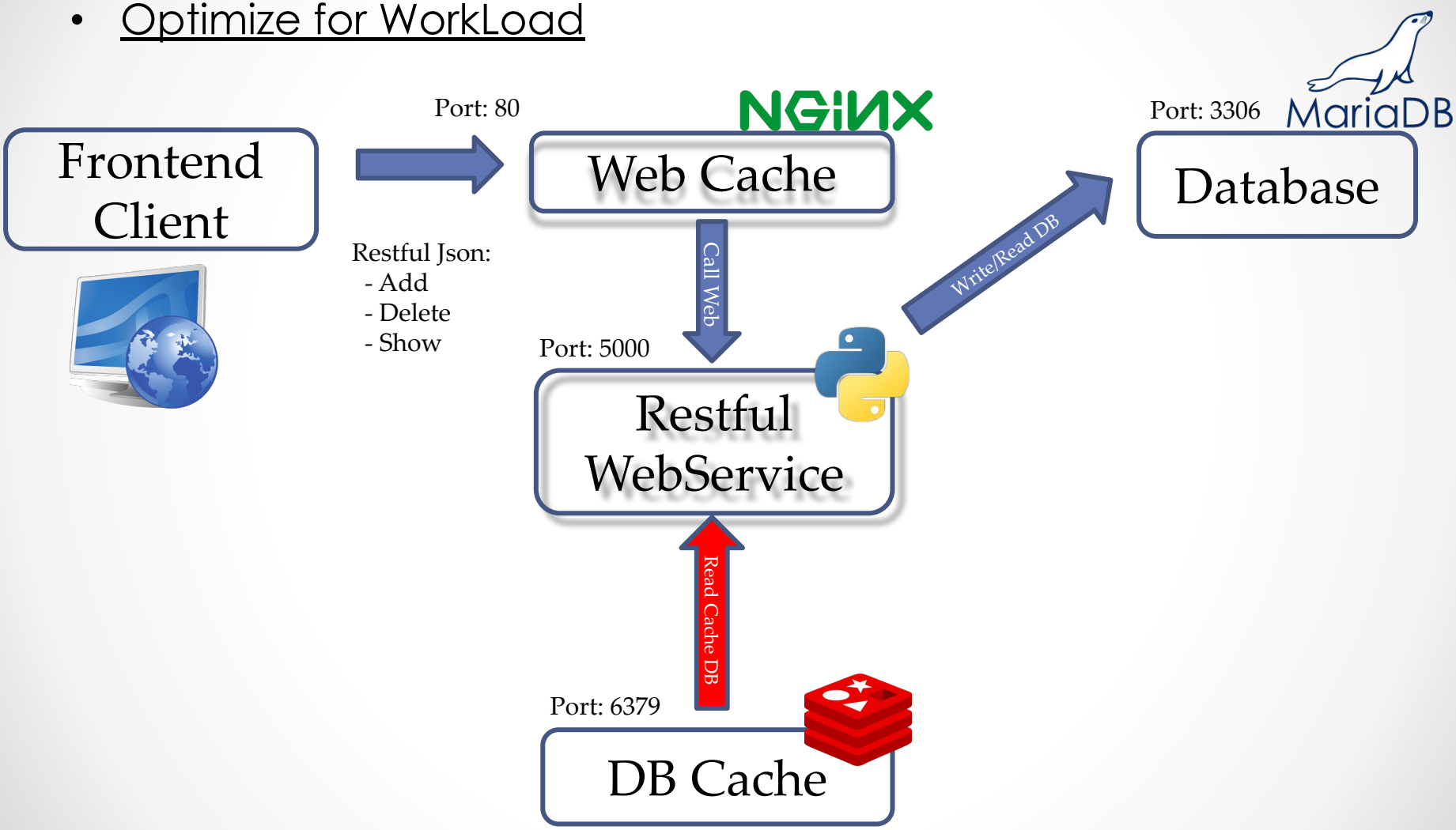
Workshop: DockerFile

- Part3: Python RESTFUL
- Basic Component



Workshop: DockerFile

- Optimize for WorkLoad



docker

Workshop: DockerFile

- Restful WebService
  - /init

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

  - /insertuser

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


Workshop: DockerFile

- Restful WebService
  - /removeuser/<uid>

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


Workshop: DockerFile

- Restful WebService
  - /users/<uid>

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


Workshop: DockerFile

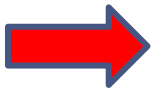
- Web Cache (NGINX)

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

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

    keepalive_timeout 65;

    gzip on;

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

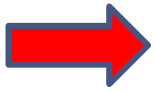

Workshop: DockerFile

- Database / Database Cache

```
maindb:
  image: labdocker/mysql:latest
  container_name: maindb
  environment:
      MYSQL_ROOT_PASSWORD: password

cachedb:
  image: labdocker/redis:latest
  container_name: cachedb

webservice:
  build: .
  dockerfile: dockerfile_python
  container_name: webservice
  ports:
   - "5000:5000"
  links:
   - cachedb:cachedb
   - maindb:maindb

webcache:
  build: .
  dockerfile: dockerfile_nginx
  container_name: webcache
  ports:
   - "80:80"
  links:
   - webservice:webservice
```



Maria Database



Redis Key-Value Database

Workshop: DockerFile

- Dockerfile: WebService

```
FROM labdocker/alpinepython:2.7-onbuild
LABEL maintainer="Praparn Lueangphoonlap (eva10409@gmail.com)"
LABEL Description="NodeJS/NGINX Build Container" Version="1.0"
RUN apk update && \
    apk add curl --no-cache && rm -rf /var/cache/apk/*
EXPOSE 5000
CMD [ "python", "main.py" ]
HEALTHCHECK --interval=30s --timeout=5s --start-period=30s --retries=3 \
    CMD curl -f http://localhost:5000/ || exit 1
```

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

Workshop: DockerFile

- Dockerfile: Webcache

```
FROM labdocker/alpine:3.13.6
LABEL maintainer="Praparn Lueangphoonlap (eva10409@gmail.com)"
LABEL Description="NodeJS/NGINX Build Container" Version="1.0"
ENV NODE_VERSION=v4.3.0 NPM_VERSION=2.14.12
RUN apk update && \
    apk add nginx nginx curl --no-cache && rm -rf /var/cache/apk/*
COPY nginx.conf /etc/nginx/nginx.conf
WORKDIR /usr/sbin
ENTRYPOINT ["nginx","-c","/etc/nginx/nginx.conf"]
EXPOSE 8080
HEALTHCHECK --interval=30s --timeout=5s --start-period=30s --retries=3 \
    CMD curl -f http://localhost:8080/ || exit 1
```

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


Workshop: DockerFile

- Test run container

```
ubuntu@ip-10-0-1-90:~/docker-workshop-092021/Workshop-2-1-DockerFile/python_restfulset$ docker container ls
CONTAINER ID   IMAGE                      COMMAND                  CREATED          STATUS                    PORTS                                       NAMES
ab037be198e2   labdocker/webcache:1.0     "nginx -c /etc/nginx…"   43 seconds ago   Up 42 seconds (healthy)   0.0.0.0:80->8080/tcp, :::80->8080/tcp       webcache
e7b40d9b3683   labdocker/webservice:1.0   "python main.py"         13 minutes ago   Up 13 minutes (healthy)   0.0.0.0:5000->5000/tcp, :::5000->5000/tcp   webservice
00b582c4da1d   labdocker/redis:latest     "docker-entrypoint.s…"   14 minutes ago   Up 14 minutes             6379/tcp                                    cachedb
82b1e278d02f   labdocker/mariadb:10.5.12  "docker-entrypoint.s…"   14 minutes ago   Up 14 minutes             3306/tcp                                    maindb
ubuntu@ip-10-0-1-90:~/docker-workshop-092021/Workshop-2-1-DockerFile/python_restfulset$
```

```
.ssh — ubuntu@ip-10-0-1-90: ~/docker-workshop-092021 — ssh -i docker_lab.pem ubuntu@ec2-54-254-231-32.ap-southeast-1.compute.amazonaws.com — 187×31
...092021 — Terminal MAC Pro — -bash   ...cesetup — Terminal MAC Pro — -bash   ...eacher — Terminal MAC Pro — -bash   ...utheast-1.compute.amazonaws.com   ...utheast-1.compute.amazonaws.com
ubuntu@ip-10-0-1-90:~/docker-workshop-092021$ curl ifconfig.co
54.254.231.32
ubuntu@ip-10-0-1-90:~/docker-workshop-092021$ export Server_IP=54.254.231.32
ubuntu@ip-10-0-1-90:~/docker-workshop-092021$ export Server_Port=80
ubuntu@ip-10-0-1-90:~/docker-workshop-092021$ curl http://$Server_IP:$Server_Port/
<H1> Welcome Page from Container Python Lab </H1>Checkpoint Date/Time: Fri Sep  3 18:33:20 2021
ubuntu@ip-10-0-1-90:~/docker-workshop-092021$  curl http://$Server_IP:$Server_Port/init
###########   Database Create New Account Table Done  ###########
ubuntu@ip-10-0-1-90:~/docker-workshop-092021$
```

Not Secure | 54.254.231.32

Apps | NMac Ked - Mac... | Medium | Infra | NOVA | jerkins | vagrant | Mesos | Vue | docker | Taiwan | NGINX | Other Bookmarks | Reading List

Welcome Page from Container Python Lab

Checkpoint Date/Time: Fri Sep 3 18:33:56 2021

Workshop: DockerFile

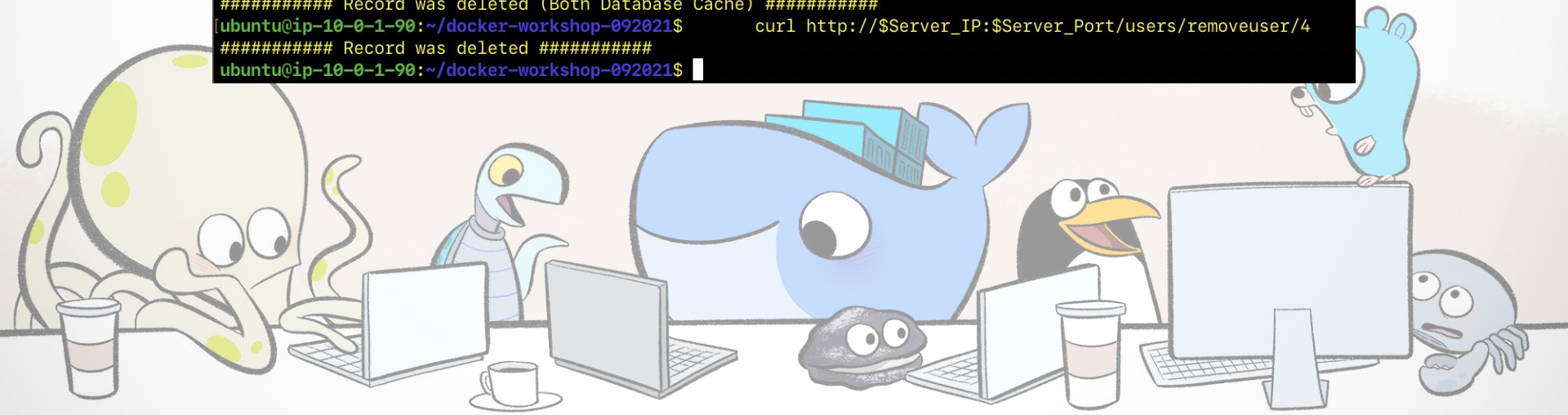
- Init / Insert / Delete Data

```
########### Record was added ###########
ubuntu@ip-10-0-1-90:~/docker-workshop-092021$       curl -i -H "Content-Type: application/json" -X POST -d '{"uid": "8", "user":"Pansa Bunsong", "descripe":"Secuirty Guard"}' http://$Server_IP:$Server_Port/users/insertuser
HTTP/1.1 200 OK
Server: nginx/1.18.0
Date: Fri, 03 Sep 2021 18:34:55 GMT
Content-Type: application/json
Content-Length: 41
Connection: keep-alive

########### Record was added ###########
ubuntu@ip-10-0-1-90:~/docker-workshop-092021$       curl -i -H "Content-Type: application/json" -X POST -d '{"uid": "9", "user":"Wiphanee Wongsaisawan", "descripe":"Administrator"}' http://$Server_IP:$Server_Port/users/insertuser
HTTP/1.1 200 OK
Server: nginx/1.18.0
Date: Fri, 03 Sep 2021 18:34:57 GMT
Content-Type: application/json
Content-Length: 41
Connection: keep-alive

########### Record was added ###########
ubuntu@ip-10-0-1-90:~/docker-workshop-092021$ curl http://$Server_IP:$Server_Port/users/1
Praparn Luangphoonlap(Database Direct)
ubuntu@ip-10-0-1-90:~/docker-workshop-092021$ curl http://$Server_IP:$Server_Port/users/1
Praparn Luangphoonlap(Database Cache)
ubuntu@ip-10-0-1-90:~/docker-workshop-092021$ curl http://$Server_IP:$Server_Port/users/2
Somchai Sunsukwan(Database Direct)
ubuntu@ip-10-0-1-90:~/docker-workshop-092021$ curl http://$Server_IP:$Server_Port/users/2
Somchai Sunsukwan(Database Cache)
ubuntu@ip-10-0-1-90:~/docker-workshop-092021$ curl http://$Server_IP:$Server_Port/users/3
Sanyachan Panrudee(Database Direct)
ubuntu@ip-10-0-1-90:~/docker-workshop-092021$ curl http://$Server_IP:$Server_Port/users/3
Sanyachan Panrudee(Database Cache)
ubuntu@ip-10-0-1-90:~/docker-workshop-092021$ curl http://$Server_IP:$Server_Port/users/removeuser/1
########### Record was deleted (Both Database Cache) ###########
ubuntu@ip-10-0-1-90:~/docker-workshop-092021$       curl http://$Server_IP:$Server_Port/users/removeuser/2
########### Record was deleted (Both Database Cache) ###########
ubuntu@ip-10-0-1-90:~/docker-workshop-092021$       curl http://$Server_IP:$Server_Port/users/removeuser/3
########### Record was deleted (Both Database Cache) ###########
ubuntu@ip-10-0-1-90:~/docker-workshop-092021$       curl http://$Server_IP:$Server_Port/users/removeuser/4
########### Record was deleted ###########
ubuntu@ip-10-0-1-90:~/docker-workshop-092021$
```


Workshop: DockerFile

- Get Data (Direct/Cache) and Delete Data

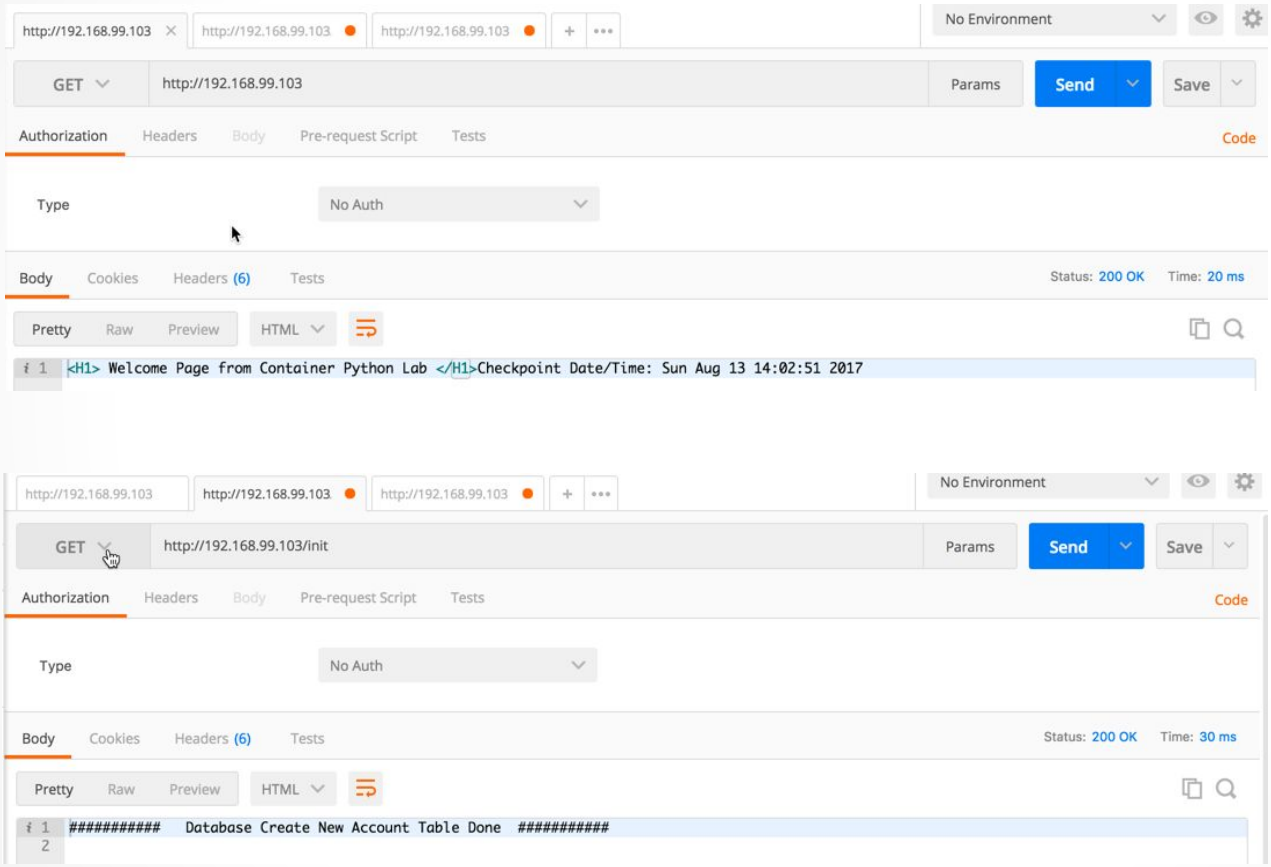
```
ubuntu@ip-10-0-1-90:~/docker-workshop-092021$ curl http://$Server_IP:$Server_Port/users/1
Praparn Luangphoonlap(Database Direct)
ubuntu@ip-10-0-1-90:~/docker-workshop-092021$ curl http://$Server_IP:$Server_Port/users/1
Praparn Luangphoonlap(Database Cache)
ubuntu@ip-10-0-1-90:~/docker-workshop-092021$ curl http://$Server_IP:$Server_Port/users/2
Somchai Sunsukwan(Database Direct)
ubuntu@ip-10-0-1-90:~/docker-workshop-092021$ curl http://$Server_IP:$Server_Port/users/2
Somchai Sunsukwan(Database Cache)
ubuntu@ip-10-0-1-90:~/docker-workshop-092021$ curl http://$Server_IP:$Server_Port/users/3
Sanyachan Panrudee(Database Direct)
ubuntu@ip-10-0-1-90:~/docker-workshop-092021$ curl http://$Server_IP:$Server_Port/users/3
Sanyachan Panrudee(Database Cache)
ubuntu@ip-10-0-1-90:~/docker-workshop-092021$ curl http://$Server_IP:$Server_Port/users/removeuser/1
########### Record was deleted (Both Database Cache) ###########
ubuntu@ip-10-0-1-90:~/docker-workshop-092021$       curl http://$Server_IP:$Server_Port/users/removeuser/2
########### Record was deleted (Both Database Cache) ###########
ubuntu@ip-10-0-1-90:~/docker-workshop-092021$       curl http://$Server_IP:$Server_Port/users/removeuser/3
########### Record was deleted (Both Database Cache) ###########
ubuntu@ip-10-0-1-90:~/docker-workshop-092021$       curl http://$Server_IP:$Server_Port/users/removeuser/4
########### Record was deleted ###########
ubuntu@ip-10-0-1-90:~/docker-workshop-092021$
```



docker

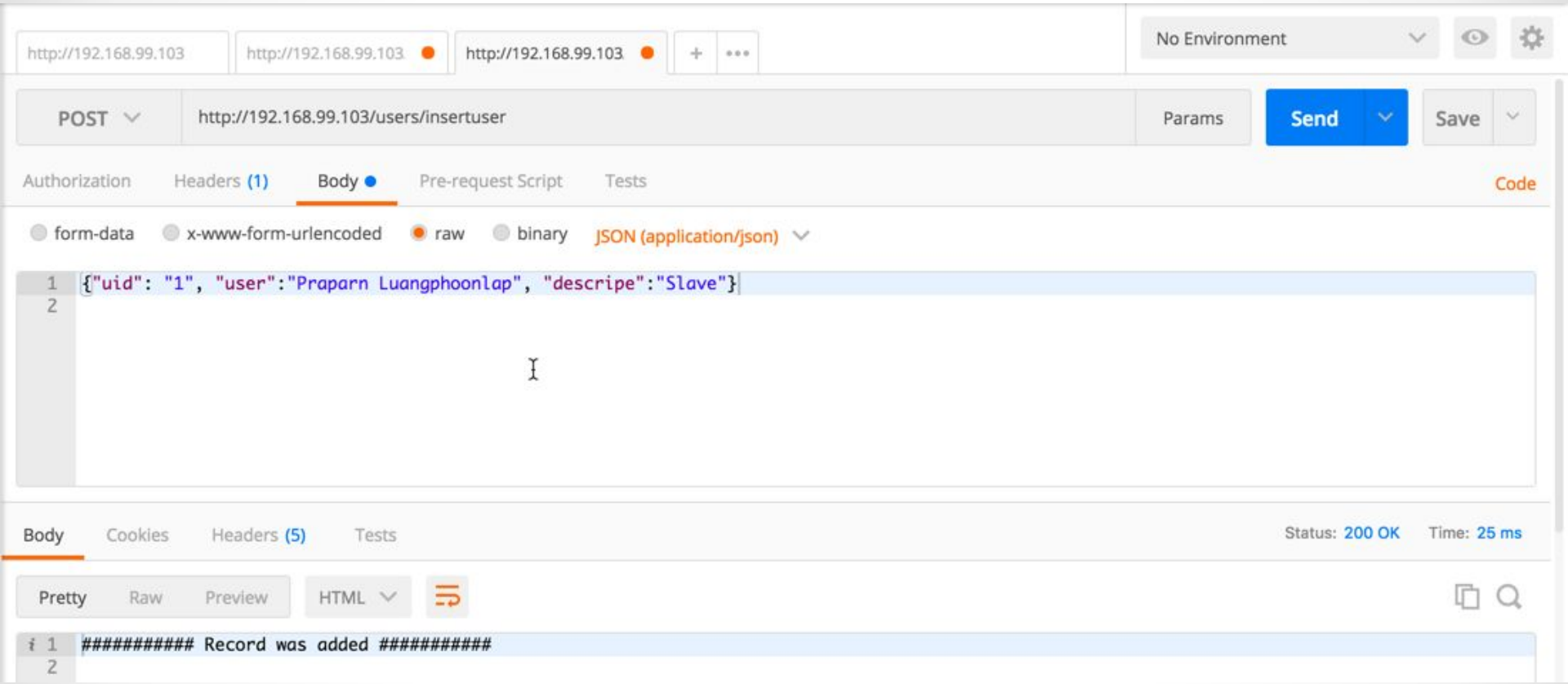
Workshop: DockerFile

- Case for POSTMAN



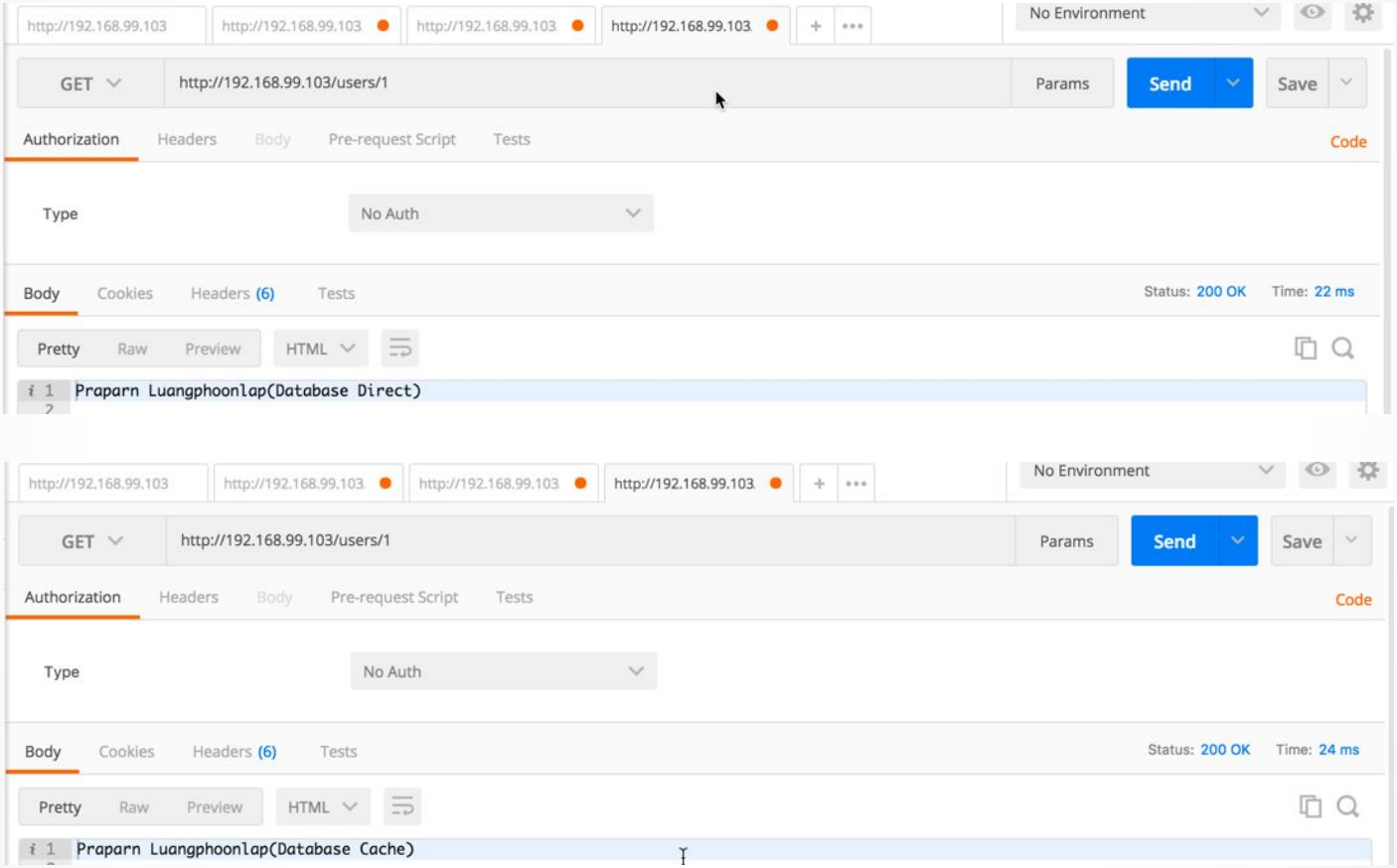
Workshop: DockerFile

- Case for POSTMAN



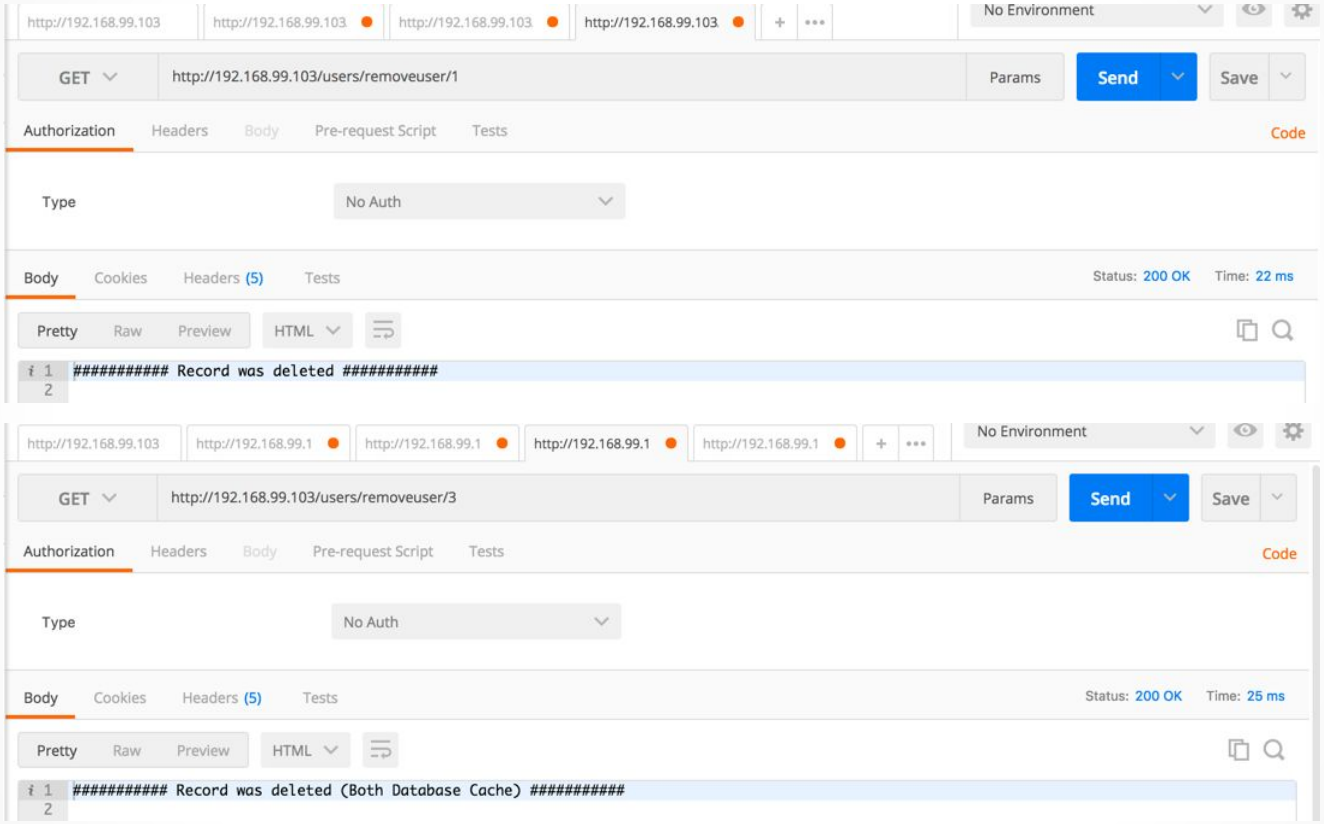
Workshop: DockerFile

- Case for POSTMAN



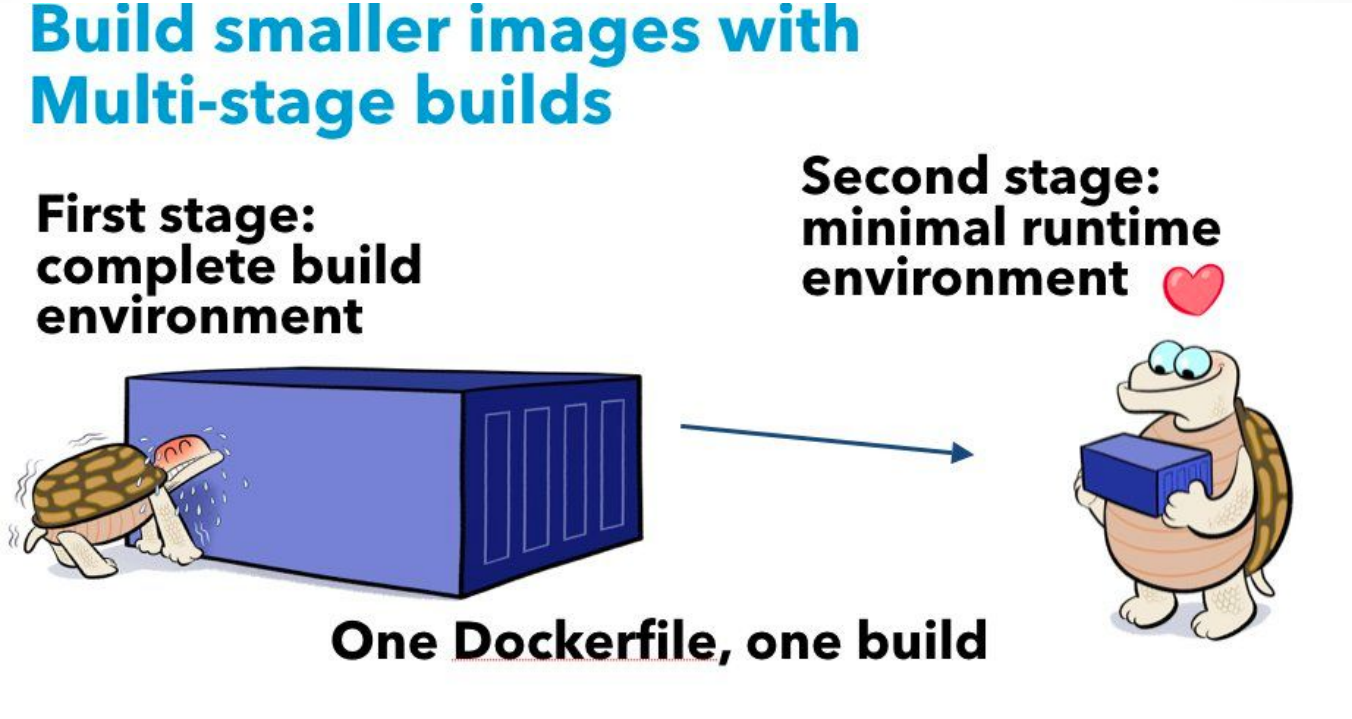
Workshop: DockerFile

- Case for POSTMAN



Dockerfile

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



docker

Dockerfile

Multi-state dockerfile:

```
1   ############################
2   # STEP 1 build executable binary
3   ############################
4   FROM golang:alpine AS builder
5   # Install git.
6   # Git is required for fetching the dependencies.
7   RUN apk update && apk add --no-cache git
8   WORKDIR $GOPATH/src/mypackage/myapp/
9   COPY . .
10  # Fetch dependencies.
11  # Using go get.
12  RUN go get -d -v
13  # Build the binary.
14  RUN go build -o /go/bin/hello
15  ############################
16  # STEP 2 build a small image
17  ############################
18  FROM scratch
19  # Copy our static executable.
20  COPY --from=builder /go/bin/hello /go/bin/hello
21  # Run the hello binary.
22  ENTRYPOINT ["/go/bin/hello"]
```

Output: Final Image 1 Unit

Compose

• • •

Compose

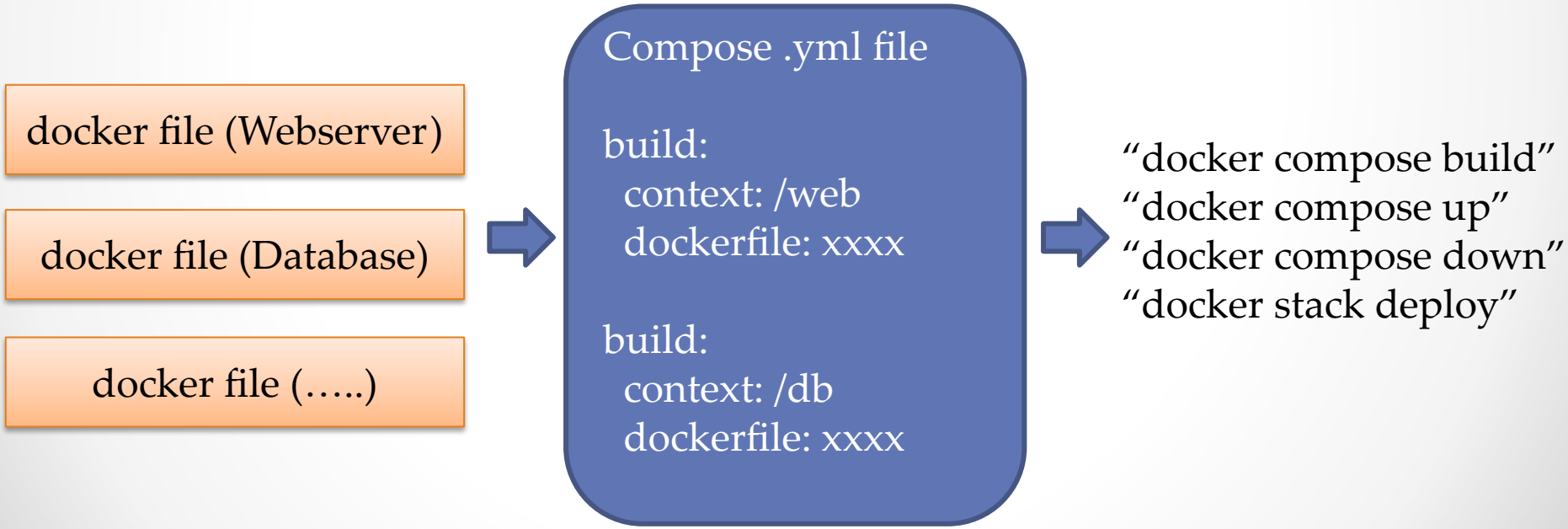
- Compose เป็นเครื่องมือที่ใช้ในการสร้าง system service ที่จะใช้ในการรันระบบงานทั้งระบบ ซึ่งตามปกติจะประกอบไปด้วยหลาย component อาทิเช่น
  - Web component service
  - Application component service
  - Database component service
  - Load balance component service
  - etc

- compose สามารถควบคุมการ start / stop / monitor การทำงานของ service ทั้งระบบเป็น single point

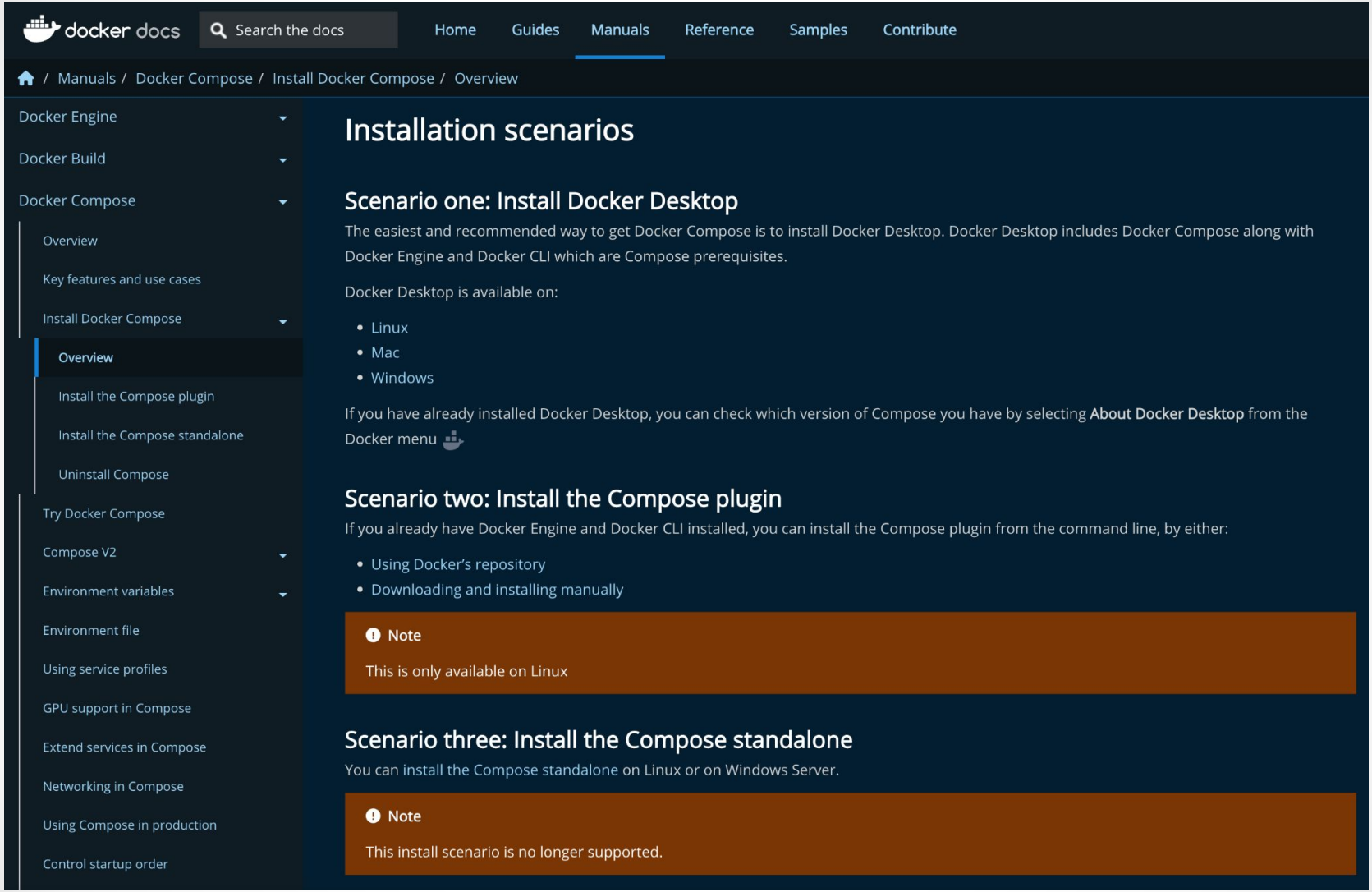
- เหมาะสำหรับการสร้าง development environment / test environment / automatic deploy to production (docker-machine / swam) (build one □ ship to everywhere)
- *No Support --ip now

Compose

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



Compose



Ref:https://docs.docker.com/compose/install/

Compose

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

Compose

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

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

Compose

Control startup and shutdown order in Compose

Estimated reading time: 2 minutes

You can control the order of service startup and shutdown with the depends_on option. Compose always starts and stops containers in dependency order, where dependencies are determined by `depends_on`, `links`, `volumes_from`, and `network_mode: "service:..."`.

However, for startup Compose does not wait until a container is "ready" (whatever that means for your particular application) - only until it's running. There's a good reason for this.

The problem of waiting for a database (for example) to be ready is really just a subset of a much larger problem of distributed systems. In production, your database could become unavailable or move hosts at any time. Your application needs to be resilient to these types of failures.

To handle this, design your application to attempt to re-establish a connection to the database after a failure. If the application retries the connection, it can eventually connect to the database.

The best solution is to perform this check in your application code, both at startup and whenever a connection is lost for any reason. However, if you don't need this level of resilience, you can work around the problem with a wrapper script:

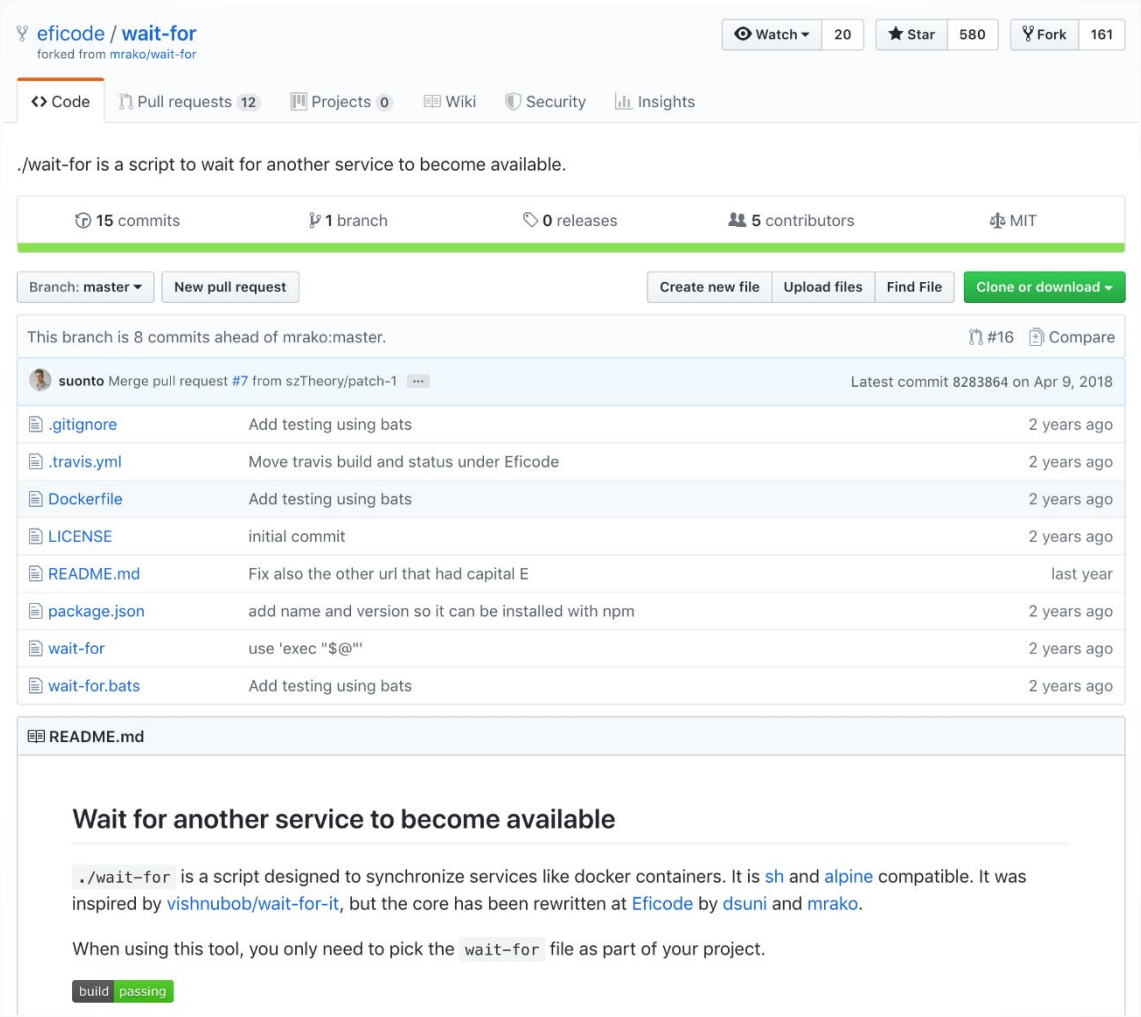
- Use a tool such as wait-for-it, dockerize, or sh-compatible wait-for. These are small wrapper scripts which you can include in your application's image to poll a given host and port until it's accepting TCP connections.

  For example, to use `wait-for-it.sh` or `wait-for` to wrap your service's command:

```
version: "2"
services:
  web:
    build: .
    ports:
      - "80:8000"
    depends_on:
      - "db"
    command: ["./wait-for-it.sh", "db:5432", "--", "python", "app.py"]
  db:
    image: postgres
```

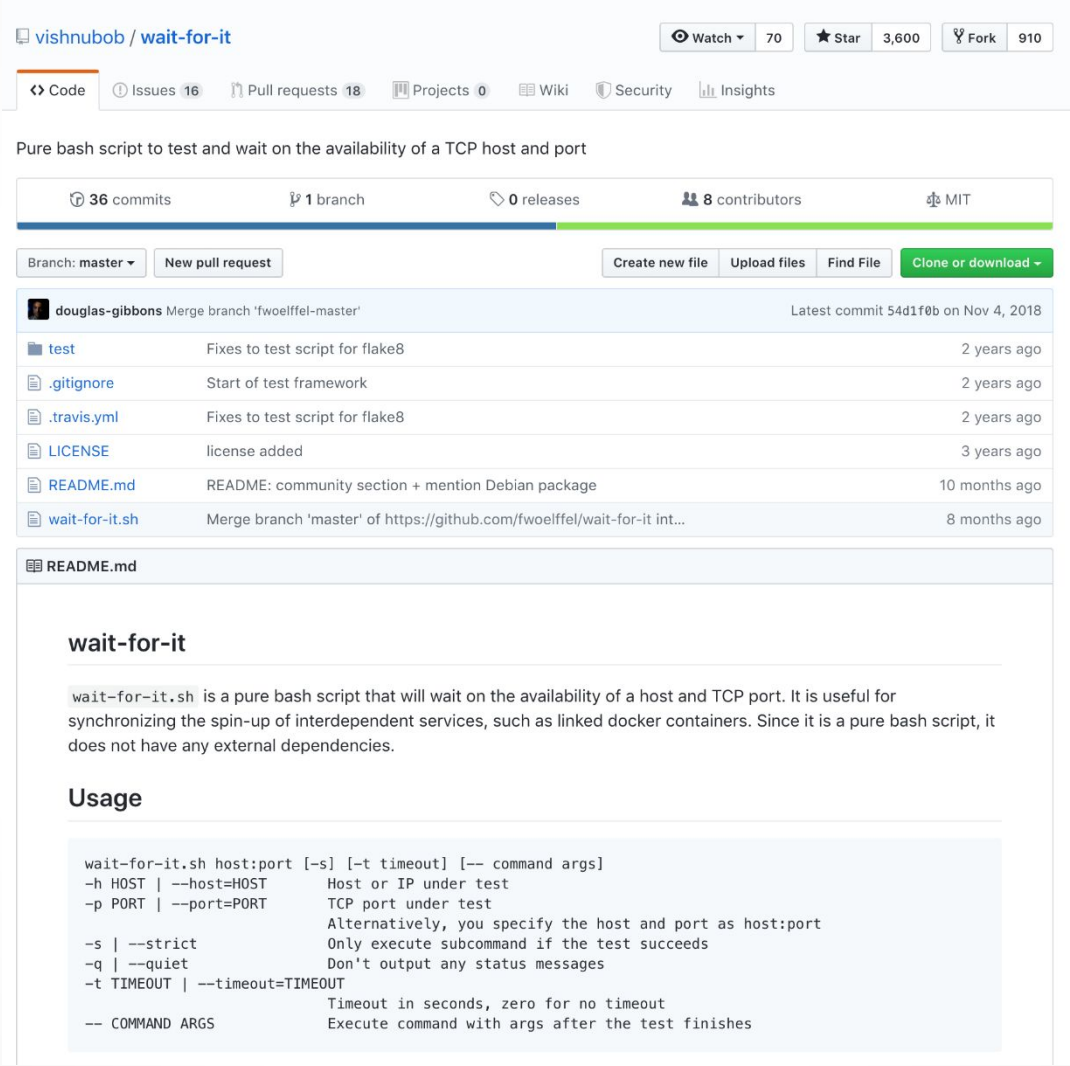
Ref: https://docs.docker.com/compose/startup-order/

Compose



Ref: https://github.com/Eficode/wait-for

Compose



vishnubob / wait-for-it

Watch 70 | Star 3,600 | Fork 910

Code | Issues 16 | Pull requests 18 | Projects 0 | Wiki | Security | Insights

Pure bash script to test and wait on the availability of a TCP host and port

36 commits | 1 branch | 0 releases | 8 contributors | MIT

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

douglas-gibbons Merge branch 'fwoelffel-master' — Latest commit 54d1f0b on Nov 4, 2018

test	Fixes to test script for flake8	2 years ago
.gitignore	Start of test framework	2 years ago
.travis.yml	Fixes to test script for flake8	2 years ago
LICENSE	license added	3 years ago
README.md	README: community section + mention Debian package	10 months ago
wait-for-it.sh	Merge branch 'master' of https://github.com/fwoelffel/wait-for-it int...	8 months ago

README.md

wait-for-it

`wait-for-it.sh` is a pure bash script that will wait on the availability of a host and TCP port. It is useful for synchronizing the spin-up of interdependent services, such as linked docker containers. Since it is a pure bash script, it does not have any external dependencies.

Usage

```
wait-for-it.sh host:port [-s] [-t timeout] [-- command args]
-h HOST | --host=HOST       Host or IP under test
-p PORT | --port=PORT       TCP port under test
                            Alternatively, you specify the host and port as host:port
-s | --strict               Only execute subcommand if the test succeeds
-q | --quiet                Don't output any status messages
-t TIMEOUT | --timeout=TIMEOUT
                            Timeout in seconds, zero for no timeout
-- COMMAND ARGS            Execute command with args after the test finishes
```

Ref: https://github.com/vishnubob/wait-for-it

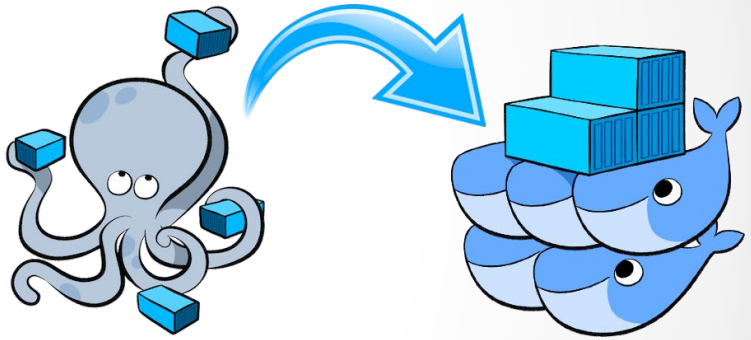
Compose

Workshop-2-1-DockerFile ▸ python_restfulset ▸ main.py

```python
from flask import Flask
from flask import Response
from flask import request
from redis import Redis
from datetime import datetime
import MySQLdb
import sys
import redis
import time
import hashlib
import os
import json
flagloop= 0
app = Flask(__name__)
startTime = datetime.now()
while flagloop == 0:
    try:
        db = MySQLdb.connect("maindb","root","password")
    except:
        time.sleep(2)
        continue
    else:
        flagloop= 1

CACHE_DB = redis.Redis(host=os.environ.get('REDIS_HOST', 'cachedb'), port=6379)
MAIN_DB = db.cursor()
@app.route('/')
def hello():
    return '<H1> Welcome Page from Container Python Lab </H1>Checkpoint Date/Time: ' + time.strftime("%c") +'\n'
@app.route('/init')
def init():
    MAIN_DB.execute("DROP DATABASE IF EXISTS ACCTABLE")
    MAIN_DB.execute("CREATE DATABASE ACCTABLE")
    MAIN_DB.execute("USE ACCTABLE")
    sql = """CREATE TABLE users (
         ID int,
         USER char(30),
         DESCRIPE char(250)
      )"""
    MAIN_DB.execute(sql)
    db.commit()
    return "##########   Database Create New Account Table Done  ##########\n"
```


Compose

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

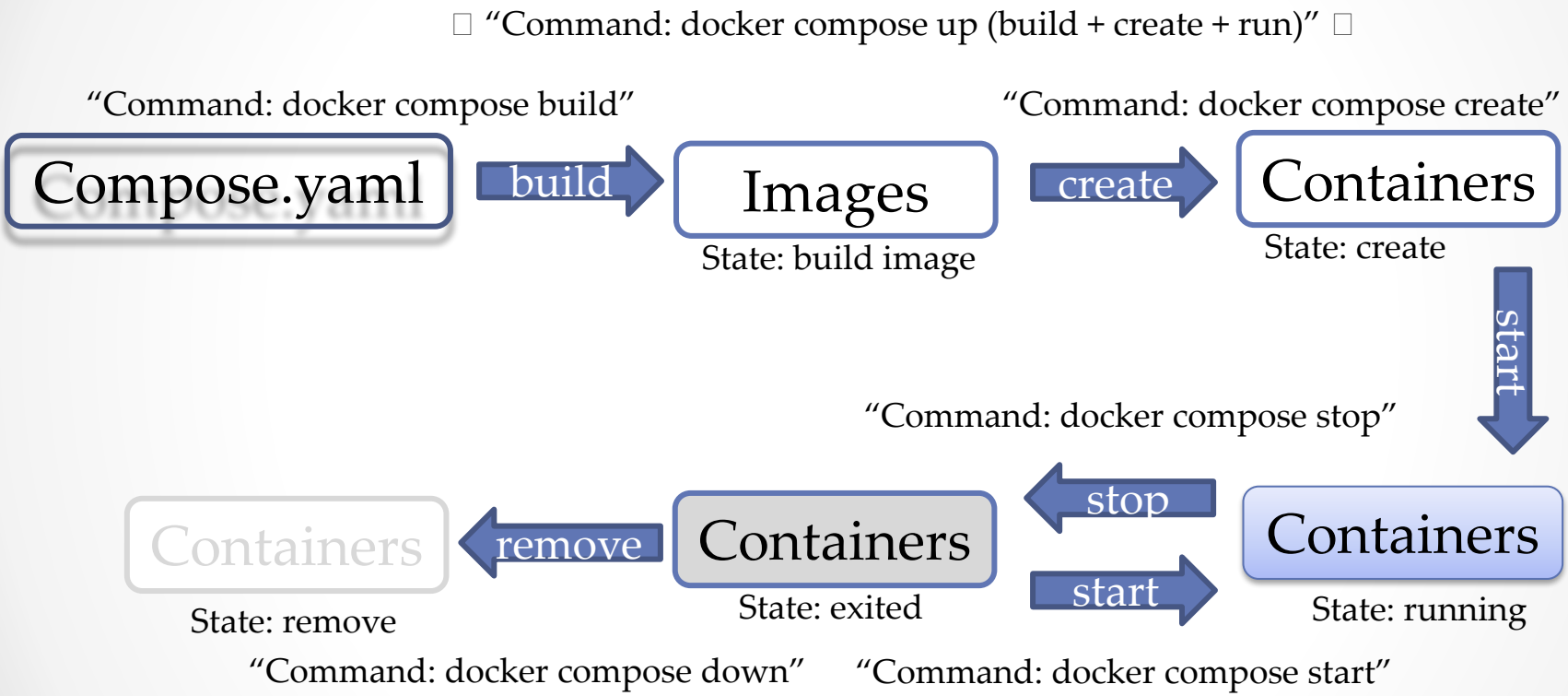


Compose

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

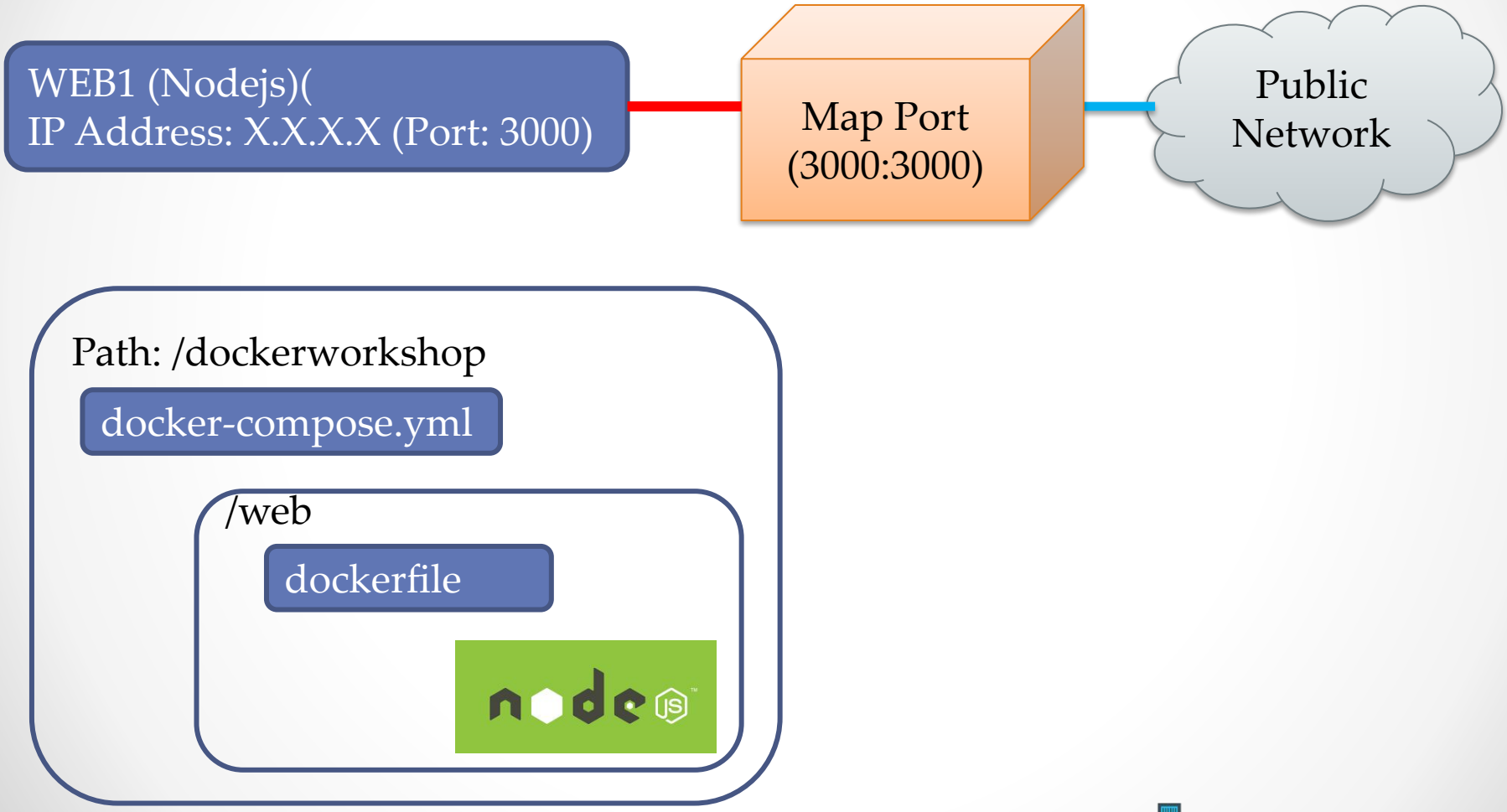
Compose

- Compose State:



Workshop: Compose

- Part1: composenodejs



Workshop: Compose

- dockerfile

```
FROM labdocker/alpine:3.13.6
LABEL maintainer="Praparn Lueangphoonlap (eva10409@gmail.com)"
LABEL Description="NodeJS/NGINX Build Container" Version="1.0"
ENV NODE_VERSION=v4.3.0 NPM_VERSION=2.14.12
RUN apk update && \
    apk add nodejs curl --no-cache && rm -rf /var/cache/apk/*
RUN mkdir /nodejs
COPY hello.js /nodejs/
CMD ["nginx","-c","/etc/nginx/nginx.conf"]
WORKDIR /nodejs
ENTRYPOINT ["node","hello.js"]
EXPOSE 3000
HEALTHCHECK --interval=15s --timeout=5s --start-period=10s --retries=3 \
    CMD curl -f http://localhost:3000/ || exit 1
```

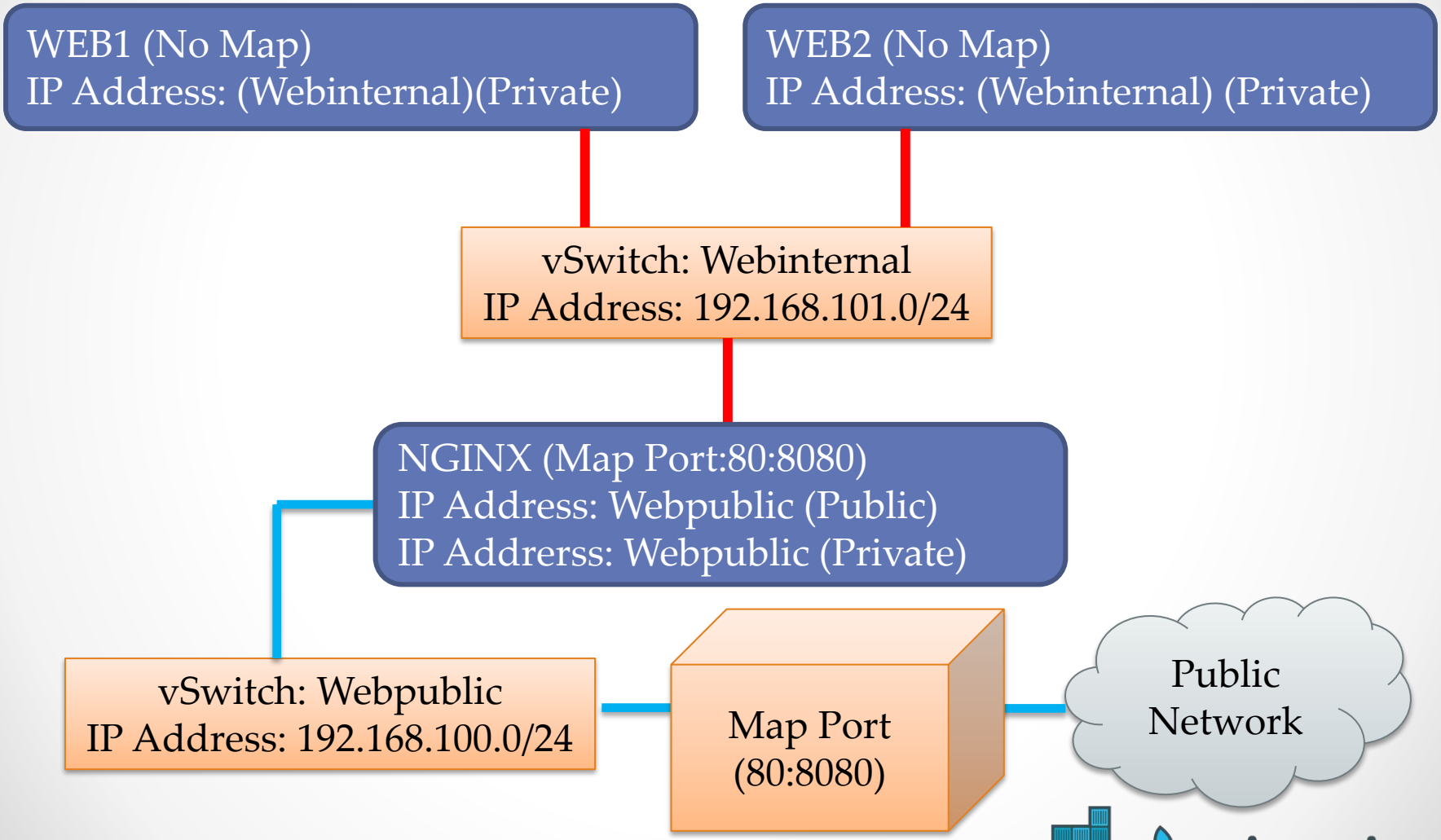

Workshop: Compose

- docker-compose.yml

```yaml
version: "3.4"
services:
  web:
    build:
      context: ./web
      dockerfile: dockerfile          # optional for specific dockerfile
    container_name: nodejs            # optional for specific container name
    command: ["node","hello.js"]      # optional for override command on dockerfile
    healthcheck:
      test: ["CMD", "curl", "-f", "http://localhost:3000"]  # option for health check and send curl for check http://localhost:3000
      interval: 30s                                         # interval health check
      timeout: 10s                                          # timeout for check
      retries: 3                                            # maximum retries
      start_period: 30s                                     # start period <news on 3.4>
    ports:
      - 3000:3000
```

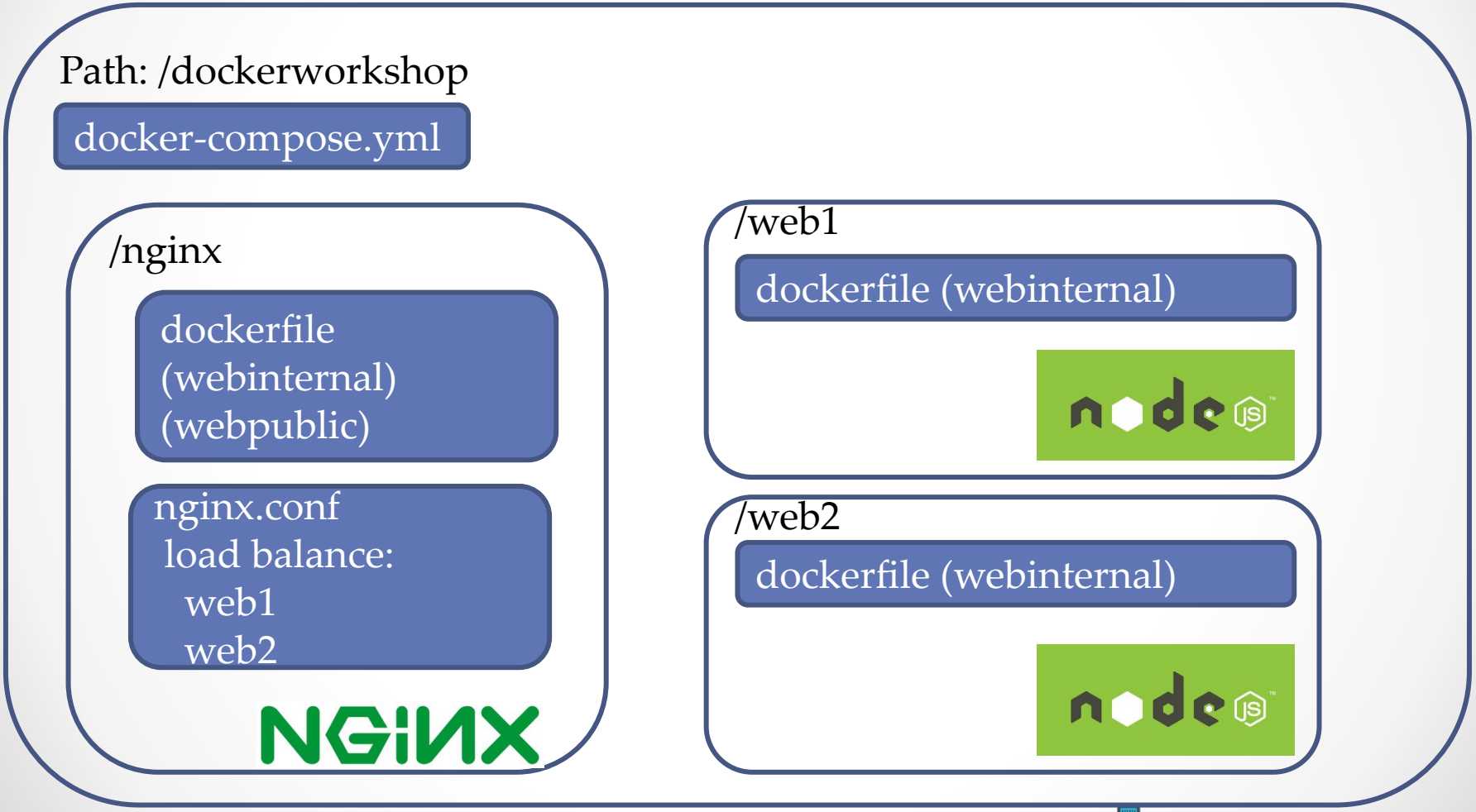

Workshop: Compose

- Part 2: composemultinodejs



Workshop: Compose

- Use case: multinodejs with nginx



docker

Workshop: Compose

- nginx: dockerfile

```
FROM labdocker/alpine:3.13.6
LABEL maintainer="Praparn Lueangphoonlap (eva10409@gmail.com)"
LABEL Description="NodeJS/NGINX Build Container" Version="1.0"
ENV NODE_VERSION=v10.14.2-r0 NPM_VERSION=v10.14.2-r0
RUN mkdir -p /run/nginx
RUN apk update && \
    apk add nginx curl --no-cache && \
    rm /etc/nginx/conf.d/default.conf \
    rm -rf /var/cache/apk/*
COPY nginx.conf /etc/nginx/nginx.conf
COPY labdocker.com.crt /etc/nginx/labdocker.com.crt
COPY labdocker.com.key /etc/nginx/labdocker.com.key
WORKDIR /usr/sbin
ENTRYPOINT ["nginx","-c","/etc/nginx/nginx.conf"]
EXPOSE 8080 8443
HEALTHCHECK --interval=15s --timeout=5s --start-period=10s --retries=3 \
    CMD curl -f http://localhost:8080/ || exit 1
```


Workshop: Compose

- Web1 & Web 2: dockerfile

```
FROM labdocker/alpine:3.13.6
LABEL maintainer="Praparn Lueangphoonlap (eva10409@gmail.com)"
LABEL Description="NodeJS/NGINX Build Container" Version="1.0"
ENV NODE_VERSION=v4.3.0 NPM_VERSION=2.14.12
RUN apk update && \
    apk add nodejs curl --no-cache && rm -rf /var/cache/apk/*
RUN mkdir /nodejs
COPY hello.js /nodejs/
WORKDIR /nodejs
ENTRYPOINT ["node","hello.js"]
EXPOSE 3000
HEALTHCHECK --interval=15s --timeout=5s --start-period=10s --retries=3 \
    CMD curl -f http://localhost:3000/ || exit 1
```


Workshop: Compose

- Web1: Hello.js

```
var http = require('http');

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

console.log('Server running at http://0.0.0.0:3000/');
```

- Web2: Hello.js

```
var http = require('http');

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

console.log('Server running at http://0.0.0.0:3000/');
```


Workshop: Compose

- docker-compose.yml

```yaml
version: '3.4'
services:
  nginx:
    build:
      context: ./nginx
      dockerfile: dockerfile          # optional for specific dockerfile
      cache_from:
        - alpine:latest
    container_name: nginx
    depends_on:
      - web1
      - web2
    healthcheck:
      test: ["CMD", "curl", "-f", "http://localhost:80"]  # option for health check and send curl for check http://localhost:3000
      interval: 30s                                       # interval health check
      timeout: 10s                                        # timeout for check
      retries: 3                                          # maximum retries
      start_period: 30s                                   # start period <news on 3.4>
    networks:
      webpublic:
        aliases:
        - nginx
      webinternal:
        aliases:
        - nginx
    ports:
      - "80:8080"
```


Workshop: Compose

- docker-compose.yml

```yaml
web1:
  build:
    context: ./web1
    dockerfile: dockerfile          # optional for specific dockerfile
    cache_from:
      - labdocker/alpineweb:latest
  container_name: web1
  healthcheck:
    test: ["CMD", "curl", "-f", "http://localhost:3000"]  # option for health check and send curl for check http://localhost:3000
    interval: 30s                                         # interval health check
    timeout: 10s                                          # timeout for check
    retries: 3                                            # maximum retries
    start_period: 30s                                     # start period <news on 3.4>
  networks:
    webinternal:
      aliases:
      - web1

web2:
  build:
    context: ./web2
    dockerfile: dockerfile          # optional for specific dockerfile
    cache_from:
      - labdocker/alpineweb:latest
  container_name: web2
  healthcheck:
    test: ["CMD", "curl", "-f", "http://localhost:3000"]  # option for health check and send curl for check http://localhost:3000
    interval: 30s                                         # interval health check
    timeout: 10s                                          # timeout for check
    retries: 3                                            # maximum retries
    start_period: 30s                                     # start period <news on 3.4>
  networks:
    webinternal:
      aliases:
      - web2
```


Workshop: Compose

- docker-compose.yml

```
networks:
    webpublic:
      driver: bridge
      ipam:
        driver: default
        config:
        - subnet: 192.168.100.0/24
    webinternal:
      driver: bridge
      ipam:
        driver: default
        config:
        - subnet: 192.168.101.0/24
```


Docker Security

• • •

Docker Security

- เพื่อให้การรันงานบน docker container มีความปลอดภัยสูง docker มีการพิจารณาเรื่อง security ในการทำงานแบ่งออกได้ดังนี้

- Control Group (CG Group) / Name Space
  - Docker container run in separate namespace (Process independence, Not release / touch with other container)
  - Separate network stack
  - CG group will create for separate resource (CPU, Memory, I/O) and limit for protect single container run process and make host fail (denied-of-service)

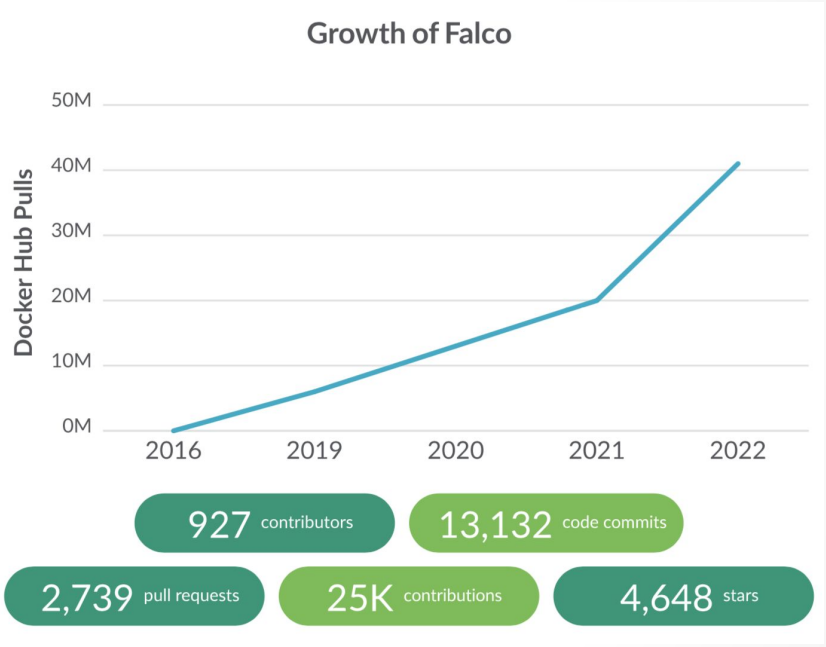
- Docker daemon operation (root privilege)
  - Beaware for allow user who operate with docker daemon
  - Some operation quite risk for security (docker pull, docker run –v with share host path etc)
  - Docker API should be aware for RESTFUL connection with secure method

Docker Security

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

Docker Security

- Docker Image Scanning
  - How can you make sure that your image is not vulnerability



Docker Security

- Docker Bench for Security
  - Docker container / script for check best practice to use docker in production (Ref: CIS Test)

README.md

Docker Bench for Security

```
# ------------------------------------------------------------------------------
# Docker Bench for Security v1.3.5
#
# Docker, Inc. (c) 2015-
#
# Checks for dozens of common best-practices around deploying Docker containers in production.
# Inspired by the CIS Docker Benchmark v1.2.0.
# ------------------------------------------------------------------------------

Initializing Tue Nov  5 10:27:42 UTC 2019


[INFO] 1 - Host Configuration

[INFO] 1.1 - General Configuration
[NOTE] 1.1.1  - Ensure the container host has been Hardened
[INFO] 1.1.2  - Ensure Docker is up to date
[INFO]        * Using 19.03.4, verify is it up to date as deemed necessary
[INFO]        * Your operating system vendor may provide support and security maintenance for Docker

[INFO] 1.2 - Linux Hosts Specific Configuration
[WARN] 1.2.1 - Ensure a separate partition for containers has been created
[INFO] 1.2.2  - Ensure only trusted users are allowed to control Docker daemon
[INFO]        * docker:x:998:
[WARN] 1.2.3  - Ensure auditing is configured for the Docker daemon
[WARN] 1.2.4  - Ensure auditing is configured for Docker files and directories - /var/lib/docker
[WARN] 1.2.5  - Ensure auditing is configured for Docker files and directories - /etc/docker
[WARN] 1.2.6  - Ensure auditing is configured for Docker files and directories - docker.service
[WARN] 1.2.7  - Ensure auditing is configured for Docker files and directories - docker.socket
[WARN] 1.2.8  - Ensure auditing is configured for Docker files and directories - /etc/default/docker
[INFO] 1.2.9  - Ensure auditing is configured for Docker files and directories - /etc/sysconfig/docker
[INFO]        * File not found
[INFO] 1.2.10  - Ensure auditing is configured for Docker files and directories - /etc/docker/daemon.json
[INFO]         * File not found
[WARN] 1.2.11  - Ensure auditing is configured for Docker files and directories - /usr/bin/containerd
[INFO] 1.2.12  - Ensure auditing is configured for Docker files and directories - /usr/sbin/runc
[INFO]         * File not found


[INFO] 2 - Docker daemon configuration
[WARN] 2.1  - Ensure network traffic is restricted between containers on the default bridge
[PASS] 2.2  - Ensure the logging level is set to 'info'
[PASS] 2.3  - Ensure Docker is allowed to make changes to iptables
```

The Docker Bench for Security is a script that checks for dozens of common best-practices around deploying Docker containers in production. The tests are all automated, and are inspired by the CIS Docker Benchmark v1.2.0.

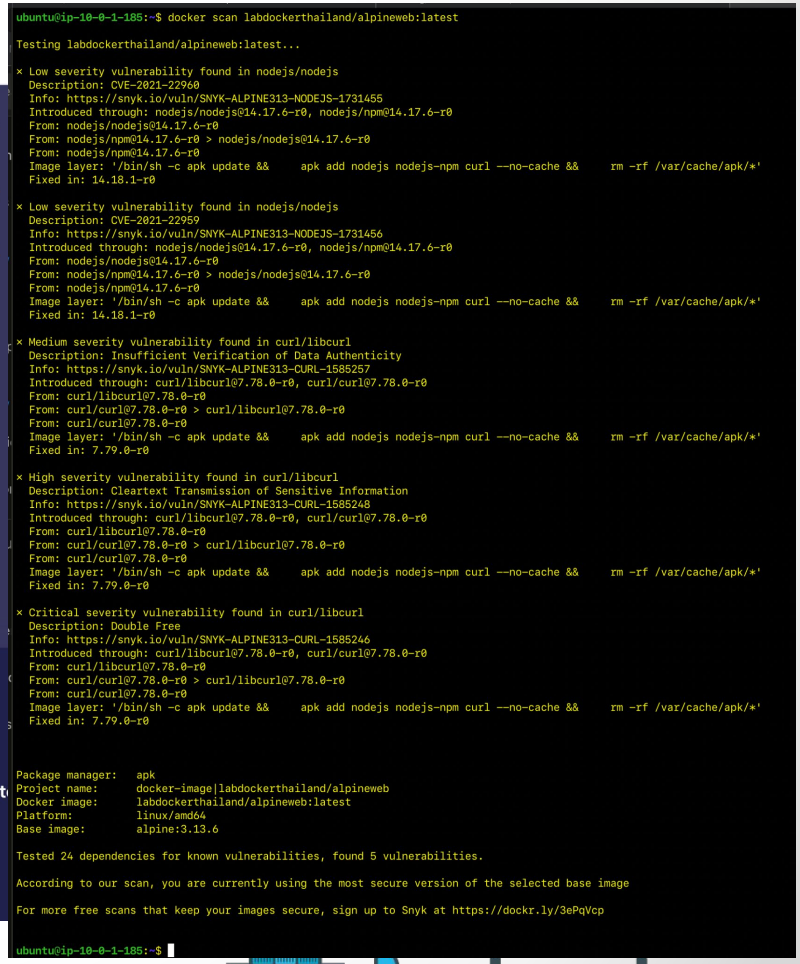
```
ubuntu@ip-10-0-1-234:~/docker-bench-security$ docker run -it --net host --pid host --cap-add audit_control \
>     -e DOCKER_CONTENT_TRUST=$DOCKER_CONTENT_TRUST \
>     -v /var/lib:/var/lib:ro \
>     -v /var/run/docker.sock:/var/run/docker.sock:ro \
>     -v /usr/lib/systemd:/usr/lib/systemd:ro \
>     -v /etc:/etc:ro --label docker_bench_security \
>     docker-bench-security
# ------------------------------------------------------------------------------
# Docker Bench for Security v1.3.5
#
# Docker, Inc. (c) 2015-
#
# Checks for dozens of common best-practices around deploying Docker containers in production.
# Inspired by the CIS Docker Benchmark v1.2.0.
# ------------------------------------------------------------------------------

Initializing Tue Mar  3 12:24:34 UTC 2020


[INFO] 1 - Host Configuration
[INFO] 1.1 - General Configuration
[NOTE] 1.1.1  - Ensure the container host has been Hardened
[INFO] 1.1.2  - Ensure Docker is up to date
[INFO]        * Using 19.03.6, verify is it up to date as deemed necessary
[INFO]        * Your operating system vendor may provide support and security maintenance for Docker

[INFO] 1.2 - Linux Hosts Specific Configuration
[WARN] 1.2.1 - Ensure a separate partition for containers has been created
[INFO] 1.2.2  - Ensure only trusted users are allowed to control Docker daemon
[INFO]        * docker:x:1001:ubuntu,1001
[WARN] 1.2.3  - Ensure auditing is configured for the Docker daemon
[WARN] 1.2.4  - Ensure auditing is configured for Docker files and directories - /var/lib/docker
[WARN] 1.2.5  - Ensure auditing is configured for Docker files and directories - /etc/docker
[INFO] 1.2.6  - Ensure auditing is configured for Docker files and directories - docker.service
[INFO]        * File not found
[INFO] 1.2.7  - Ensure auditing is configured for Docker files and directories - docker.socket
[INFO]        * File not found
[WARN] 1.2.8  - Ensure auditing is configured for Docker files and directories - /etc/default/docker
[INFO] 1.2.9  - Ensure auditing is configured for Docker files and directories - /etc/sysconfig/docker
[INFO]        * File not found
[WARN] 1.2.10  - Ensure auditing is configured for Docker files and directories - /etc/docker/daemon.json
[INFO] 1.2.11  - Ensure auditing is configured for Docker files and directories - /usr/bin/containerd
[INFO]         * File not found
[INFO] 1.2.12  - Ensure auditing is configured for Docker files and directories - /usr/sbin/runc
[INFO]         * File not found
```

Ref: https://github.com/docker/docker-bench-security

Docker: The Next-Gen of Virtualization

Docker Security

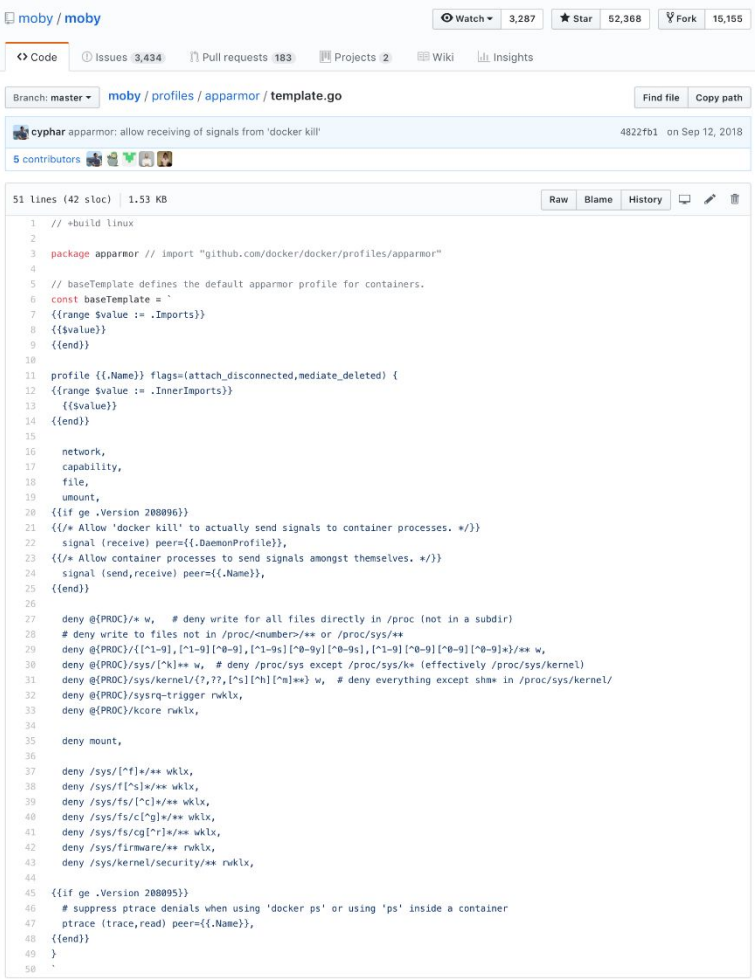
- Synk for docker image scanning (built-in)
  - Scan in each layer of docker image and compare with know vulnerability



Docker Security

- Default apparmor in docker (1.13.0 and above) apply on all docker contaiener

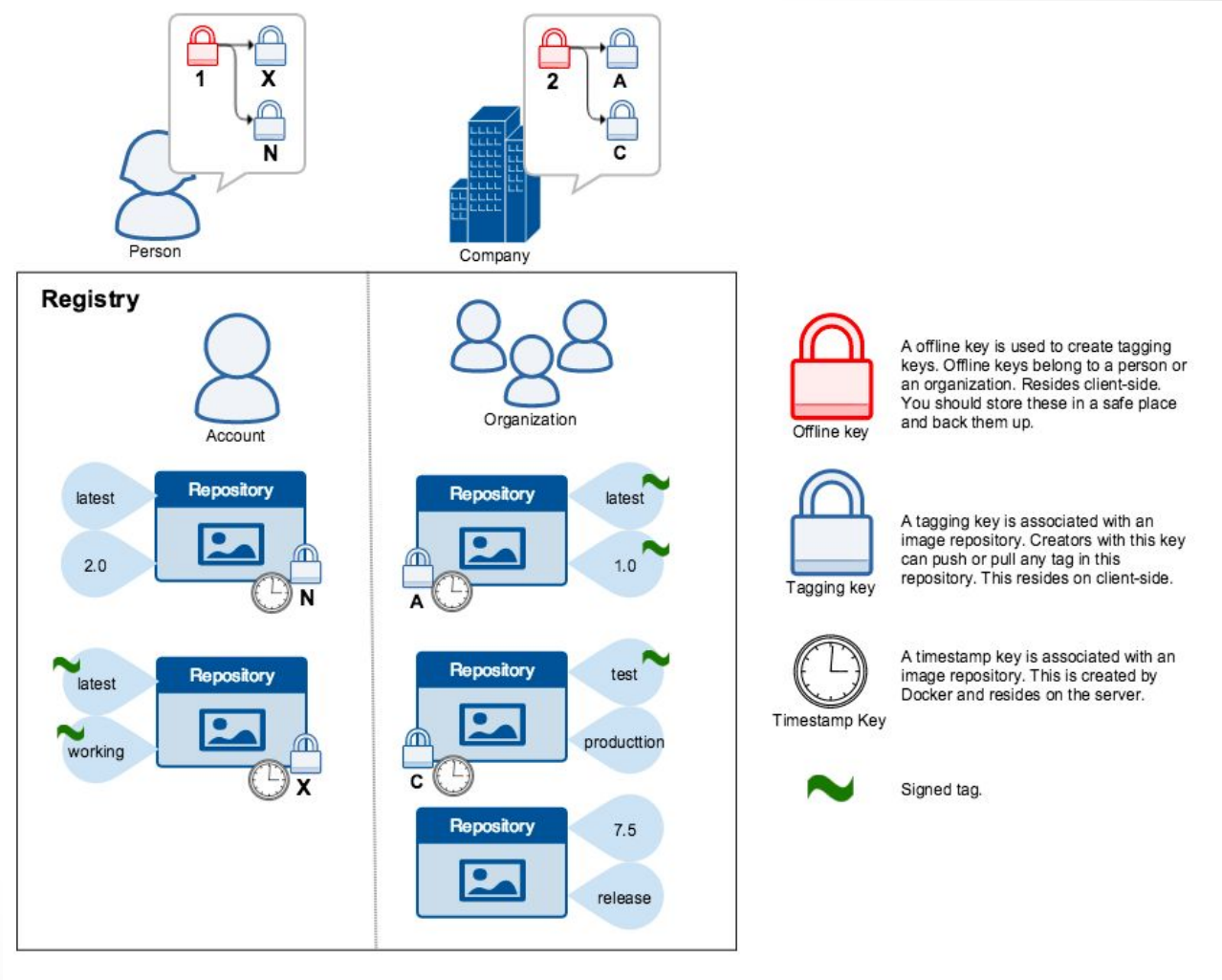
```
[ubuntu@ip-10-0-1-104:~$ docker container exec -it sectestdefault sh
[/ # touch /tmp/testcreatefile
[/ # ls -hl /tmp
total 0
-rw-r--r--    1 root     root           0 Mar  3 07:33 testcreatefile
[/ # touch /proc/testprohibitfile
touch: /proc/testprohibitfile: No such file or directory
[/ # ls -hl /proc
total 0
dr-xr-xr-x    9 root     root           0 Mar  3 07:33 1
dr-xr-xr-x    9 root     root           0 Mar  3 07:34 14
dr-xr-xr-x    9 root     root           0 Mar  3 07:33 6
drwxrwxrwt    2 root     root          40 Mar  3 07:33 acpi
-r--r--r--    1 root     root           0 Mar  3 07:34 buddyinfo
dr-xr-xr-x    4 root     root           0 Mar  3 07:33 bus
-r--r--r--    1 root     root           0 Mar  3 07:34 cgroups
-r--r--r--    1 root     root           0 Mar  3 07:34 cmdline
-r--r--r--    1 root     root           0 Mar  3 07:34 consoles
-r--r--r--    1 root     root           0 Mar  3 07:34 cpuinfo
-r--r--r--    1 root     root           0 Mar  3 07:34 crypto
-r--r--r--    1 root     root           0 Mar  3 07:34 devices
-r--r--r--    1 root     root           0 Mar  3 07:34 diskstats
-r--r--r--    1 root     root           0 Mar  3 07:34 dma
dr-xr-xr-x    2 root     root           0 Mar  3 07:34 driver
-r--r--r--    1 root     root           0 Mar  3 07:34 execdomains
-r--r--r--    1 root     root           0 Mar  3 07:34 fb
-r--r--r--    1 root     root           0 Mar  3 07:34 filesystems
dr-xr-xr-x    6 root     root           0 Mar  3 07:33 fs
-r--r--r--    1 root     root           0 Mar  3 07:34 interrupts
-r--r--r--    1 root     root           0 Mar  3 07:34 iomem
-r--r--r--    1 root     root           0 Mar  3 07:34 ioports
dr-xr-xr-x   23 root     root           0 Mar  3 07:33 irq
-r--r--r--    1 root     root           0 Mar  3 07:34 kallsyms
crw-rw-rw-    1 root     root        1,   3 Mar  3 07:33 kcore
-r--r--r--    1 root     root           0 Mar  3 07:34 key-users
crw-rw-rw-    1 root     root        1,   3 Mar  3 07:33 keys
[/ # ls -hl /proc | grep test
[/ # cp /tmp/testcreatefile /proc/testprohibitfile
cp: can't create '/proc/testprohibitfile': No such file or directory
[/ # ls -hl /proc | grep test
[/ # exit
[ubuntu@ip-10-0-1-104:~$ docker container stop sectestdefault
sectestdefault
```



Ref: https://github.com/moby/moby/tree/master/profiles/apparmor

Docker Security

- Image Content Trust



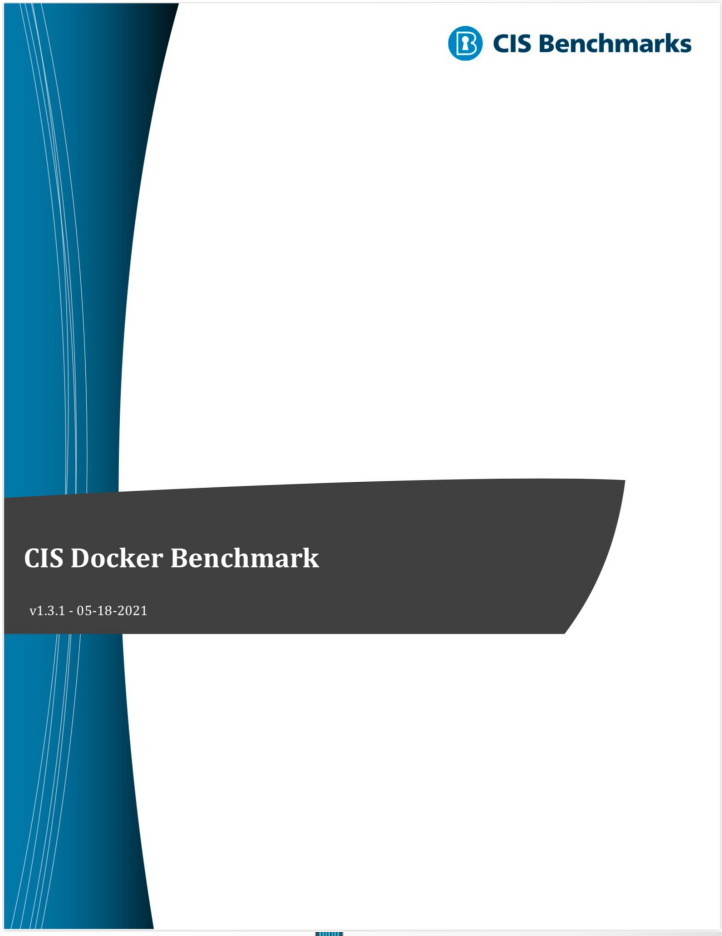
Docker Security

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

Workshop: Docker Security

- Docker Bench for Security
- Purpose: ทำการตรวจสอบช่องโหว่บน image ก่อนใช้งานใน production environment

```
ubuntu@ip-10-0-1-234:~/docker-bench-security$ docker run -it --net host --pid host --cap-add audit_control \
>     -e DOCKER_CONTENT_TRUST=$DOCKER_CONTENT_TRUST \
>     -v /var/lib:/var/lib:ro \
>     -v /var/run/docker.sock:/var/run/docker.sock:ro \
>     -v /usr/lib/systemd:/usr/lib/systemd:ro \
>     -v /etc:/etc:ro --label docker_bench_security \
>     docker-bench-security
# ------------------------------------------------------------------------------
# Docker Bench for Security v1.3.5
#
# Docker, Inc. (c) 2015-
#
# Checks for dozens of common best-practices around deploying Docker containers in production.
# Inspired by the CIS Docker Benchmark v1.2.0.
# ------------------------------------------------------------------------------

Initializing Tue Mar  3 12:24:34 UTC 2020


[INFO] 1 - Host Configuration
[
[INFO] 1.1 - General Configuration
[NOTE] 1.1.1  - Ensure the container host has been Hardened
[INFO] 1.1.2  - Ensure Docker is up to date
[INFO]        * Using 19.03.6, verify is it up to date as deemed necessary
[INFO]        * Your operating system vendor may provide support and security maintenance for Docker

[INFO] 1.2 - Linux Hosts Specific Configuration
[WARN] 1.2.1 - Ensure a separate partition for containers has been created
[INFO] 1.2.2  - Ensure only trusted users are allowed to control Docker daemon
[INFO]        * docker:x:1001:ubuntu,1001
[WARN] 1.2.3  - Ensure auditing is configured for the Docker daemon
[WARN] 1.2.4  - Ensure auditing is configured for Docker files and directories - /var/lib/docker
[WARN] 1.2.5  - Ensure auditing is configured for Docker files and directories - /etc/docker
[INFO] 1.2.6  - Ensure auditing is configured for Docker files and directories - docker.service
[INFO]        * File not found
[INFO] 1.2.7  - Ensure auditing is configured for Docker files and directories - docker.socket
[INFO]        * File not found
[WARN] 1.2.8  - Ensure auditing is configured for Docker files and directories - /etc/default/docker
[INFO] 1.2.9  - Ensure auditing is configured for Docker files and directories - /etc/sysconfig/docker
[INFO]        * File not found
[WARN] 1.2.10  - Ensure auditing is configured for Docker files and directories - /etc/docker/daemon.json
[INFO] 1.2.11  - Ensure auditing is configured for Docker files and directories - /usr/bin/containerd
[INFO]         * File not found
[INFO] 1.2.12  - Ensure auditing is configured for Docker files and directories - /usr/sbin/runc
[INFO]         * File not found
```



docker

Workshop: Docker Security

- Docker Bench for Security
- Purpose: ทำการตรวจสอบช่องโหว่บน image ก่อนใช้งานใน production environment

```
ubuntu@ip-10-0-1-234:~/docker-bench-security$ docker run -it --net host --pid host --cap-add audit_control \
>     -e DOCKER_CONTENT_TRUST=$DOCKER_CONTENT_TRUST \
>     -v /var/lib:/var/lib:ro \
>     -v /var/run/docker.sock:/var/run/docker.sock:ro \
>     -v /usr/lib/systemd:/usr/lib/systemd:ro \
>     -v /etc:/etc:ro --label docker_bench_security \
>     docker-bench-security
# ------------------------------------------------------------------------------
# Docker Bench for Security v1.3.5
#
# Docker, Inc. (c) 2015-
#
# Checks for dozens of common best-practices around deploying Docker containers in production.
# Inspired by the CIS Docker Benchmark v1.2.0.
# ------------------------------------------------------------------------------

Initializing Tue Mar  3 12:24:34 UTC 2020


[INFO] 1 - Host Configuration
[INFO] 1.1 - General Configuration
[NOTE] 1.1.1  - Ensure the container host has been Hardened
[INFO] 1.1.2  - Ensure Docker is up to date
[INFO]        * Using 19.03.6, verify is it up to date as deemed necessary
[INFO]        * Your operating system vendor may provide support and security maintenance for Docker

[INFO] 1.2 - Linux Hosts Specific Configuration
[WARN] 1.2.1 - Ensure a separate partition for containers has been created
[INFO] 1.2.2  - Ensure only trusted users are allowed to control Docker daemon
[INFO]        * docker:x:1001:ubuntu,1001
[WARN] 1.2.3  - Ensure auditing is configured for the Docker daemon
[WARN] 1.2.4  - Ensure auditing is configured for Docker files and directories - /var/lib/docker
[WARN] 1.2.5  - Ensure auditing is configured for Docker files and directories - /etc/docker
[INFO] 1.2.6  - Ensure auditing is configured for Docker files and directories - docker.service
[INFO]        * File not found
[INFO] 1.2.7  - Ensure auditing is configured for Docker files and directories - docker.socket
[INFO]        * File not found
[WARN] 1.2.8  - Ensure auditing is configured for Docker files and directories - /etc/default/docker
[INFO] 1.2.9  - Ensure auditing is configured for Docker files and directories - /etc/sysconfig/docker
[INFO]        * File not found
[WARN] 1.2.10  - Ensure auditing is configured for Docker files and directories - /etc/docker/daemon.json
[INFO] 1.2.11  - Ensure auditing is configured for Docker files and directories - /usr/bin/containerd
[INFO]         * File not found
[INFO] 1.2.12  - Ensure auditing is configured for Docker files and directories - /usr/sbin/runc
[INFO]         * File not found
```

1.1 Linux Hosts Specific Configuration

This section contains recommendations that securing Linux Hosts running Docker Containers.

1.1.1 Ensure a separate partition for containers has been created (Automated)

Profile Applicability:

- Level 1 - Linux Host OS

Description:

All Docker containers and their data and metadata is stored under `/var/lib/docker` directory. By default, `/var/lib/docker` should be mounted under either the `/` or `/var` partitions dependent on how the Linux operating system in use is configured.

Rationale:

Docker depends on `/var/lib/docker` as the default directory where all Docker related files, including the images, are stored. This directory could fill up quickly causing both Docker and the host to become unusable. For this reason, you should create a separate partition (logical volume) for storing Docker files.

Impact:

None.

Audit:

At the Docker host execute one of the below commands:

```
grep '/var/lib/docker\s' /proc/mounts
```

This should return the partition details for the `/var/lib/docker` mountpoint.

```
mountpoint -- "$(docker info -f '{{ .DockerRootDir }}')"
```

This should return whether the configured root directory is a mount point.

Workshop: Docker Security

- Docker Bench for Security
- Purpose: ทำการตรวจสอบช่องโหว่บน image ก่อนใช้งานใน production environment

```
ubuntu@ip-10-0-1-234:~/docker-bench-security$ docker run -it --net host --pid host --cap-add audit_control \
>     -e DOCKER_CONTENT_TRUST=$DOCKER_CONTENT_TRUST \
>     -v /var/lib:/var/lib:ro \
>     -v /var/run/docker.sock:/var/run/docker.sock:ro \
>     -v /usr/lib/systemd:/usr/lib/systemd:ro \
>     -v /etc:/etc:ro --label docker_bench_security \
>     docker-bench-security
# ------------------------------------------------------------------------------
# Docker Bench for Security v1.3.5
#
# Docker, Inc. (c) 2015-
#
# Checks for dozens of common best-practices around deploying Docker containers in production.
# Inspired by the CIS Docker Benchmark v1.2.0.
# ------------------------------------------------------------------------------

Initializing Tue Mar  3 12:24:34 UTC 2020


[INFO] 1 - Host Configuration

[INFO] 1.1 - General Configuration
[NOTE] 1.1.1  - Ensure the container host has been Hardened
[INFO] 1.1.2  - Ensure Docker is up to date
[INFO]        * Using 19.03.6, verify is it up to date as deemed necessary
[INFO]        * Your operating system vendor may provide support and security maintenance for Docker

[INFO] 1.2 - Linux Hosts Specific Configuration
[WARN] 1.2.1 - Ensure a separate partition for containers has been created
[INFO] 1.2.2  - Ensure only trusted users are allowed to control Docker daemon
[INFO]        * docker:x:1001:ubuntu,1001
[WARN] 1.2.3  - Ensure auditing is configured for the Docker daemon
[WARN] 1.2.4  - Ensure auditing is configured for Docker files and directories - /var/lib/docker
[WARN] 1.2.5  - Ensure auditing is configured for Docker files and directories - /etc/docker
[INFO] 1.2.6  - Ensure auditing is configured for Docker files and directories - docker.service
[INFO]        * File not found
[INFO] 1.2.7  - Ensure auditing is configured for Docker files and directories - docker.socket
[INFO]        * File not found
[WARN] 1.2.8  - Ensure auditing is configured for Docker files and directories - /etc/default/docker
[INFO] 1.2.9  - Ensure auditing is configured for Docker files and directories - /etc/sysconfig/docker
[INFO]        * File not found
[WARN] 1.2.10  - Ensure auditing is configured for Docker files and directories - /etc/docker/daemon.json
[INFO] 1.2.11  - Ensure auditing is configured for Docker files and directories - /usr/bin/containerd
[INFO]         * File not found
[INFO] 1.2.12  - Ensure auditing is configured for Docker files and directories - /usr/sbin/runc
[INFO]         * File not found
```

1.6 Ensure auditing is configured for Docker files and directories - /var/lib/docker (Scored)

Profile Applicability:

- Level 1 - Linux Host OS

Description:

Audit `/var/lib/docker`.

Rationale:

Apart from auditing your regular Linux file system and system calls, audit all Docker related files and directories. Docker daemon runs with `root` privileges. Its behavior depends on some key files and directories. `/var/lib/docker` is one such directory. It holds all the information about containers. It must be audited.

Audit:

Verify that there is an audit rule corresponding to `/var/lib/docker` directory.

For example, execute below command:

```
auditctl -l | grep /var/lib/docker
```

This should list a rule for `/var/lib/docker` directory.

Remediation:

Add a rule for `/var/lib/docker` directory.

For example,

Add the line as below in `/etc/audit/audit.rules` file:

```
-w /var/lib/docker -k docker
```

Then, restart the audit daemon. For example,

```
service auditd restart
```


Workshop: Docker Security

- Docker Bench for Security
- Purpose: ทำการตรวจสอบช่องโหว่บน image ก่อนใช้งานใน production environment

```
ubuntu@ip-10-0-1-234:~/docker-bench-security$ docker run -it --net host --pid host --cap-add audit_control \
>     -e DOCKER_CONTENT_TRUST=$DOCKER_CONTENT_TRUST \
>     -v /var/lib:/var/lib:ro \
>     -v /var/run/docker.sock:/var/run/docker.sock:ro \
>     -v /usr/lib/systemd:/usr/lib/systemd:ro \
>     -v /etc:/etc:ro --label docker_bench_security \
>     docker-bench-security
# ------------------------------------------------------------------------------
# Docker Bench for Security v1.3.5
#
# Docker, Inc. (c) 2015-
#
# Checks for dozens of common best-practices around deploying Docker containers in production.
# Inspired by the CIS Docker Benchmark v1.2.0.
# ------------------------------------------------------------------------------

Initializing Tue Mar  3 12:24:34 UTC 2020


[INFO] 1 - Host Configuration

[INFO] 1.1 - General Configuration
[NOTE] 1.1.1  - Ensure the container host has been Hardened
[INFO] 1.1.2  - Ensure Docker is up to date
[INFO]        * Using 19.03.6, verify is it up to date as deemed necessary
[INFO]        * Your operating system vendor may provide support and security maintenance for Docker

[INFO] 1.2 - Linux Hosts Specific Configuration
[WARN] 1.2.1 - Ensure a separate partition for containers has been created
[INFO] 1.2.2  - Ensure only trusted users are allowed to control Docker daemon
[INFO]        * docker:x:1001:ubuntu,1001
[WARN] 1.2.3  - Ensure auditing is configured for the Docker daemon
[WARN] 1.2.4  - Ensure auditing is configured for Docker files and directories - /var/lib/docker
[WARN] 1.2.5  - Ensure auditing is configured for Docker files and directories - /etc/docker
[INFO] 1.2.6  - Ensure auditing is configured for Docker files and directories - docker.service
[INFO]        * File not found
[INFO] 1.2.7  - Ensure auditing is configured for Docker files and directories - docker.socket
[INFO]        * File not found
[WARN] 1.2.8  - Ensure auditing is configured for Docker files and directories - /etc/default/docker
[INFO] 1.2.9  - Ensure auditing is configured for Docker files and directories - /etc/sysconfig/docker
[INFO]        * File not found
[WARN] 1.2.10  - Ensure auditing is configured for Docker files and directories - /etc/docker/daemon.json
[INFO] 1.2.11  - Ensure auditing is configured for Docker files and directories - /usr/bin/containerd
[INFO]         * File not found
[INFO] 1.2.12  - Ensure auditing is configured for Docker files and directories - /usr/sbin/runc
[INFO]         * File not found
```

```
ubuntu@ip-10-0-1-234:~$ sudo more /etc/audit/rules.d/audit.rules
## First rule - delete all
-D

## Increase the buffers to survive stress events.
## Make this bigger for busy systems
-b 8192

## This determine how long to wait in burst of events
--backlog_wait_time 0

## Set failure mode to syslog
-f 1

-w /usr/bin/docker -p wa
-w /var/lib/docker -p wa
-w /etc/docker -p wa
-w /lib/systemd/system/docker.service -p wa
-w /lib/systemd/system/docker.socket -p wa
-w /etc/default/docker -p wa
-w /etc/docker/daemon.json -p wa
-w /usr/bin/docker-containerd -p wa
-w /usr/bin/docker-runc -p wa
-w /usr/bin/docker -k docker
ubuntu@ip-10-0-1-234:~$
```


Workshop: Docker Security

- Docker Bench for Security
- Purpose: ทำการตรวจสอบช่องโหว่บน image ก่อนใช้งานใน production environment

```
ubuntu@ip-10-0-1-234:~$ docker run -it --net host --pid host --cap-add audit_control     -e DOCKER_CONTENT_TRUST=$DOCKER_CONTENT_TRUST
     -v /var/lib:/var/lib:ro     -v /var/run/docker.sock:/var/run/docker.sock:ro     -v /usr/lib/systemd:/usr/lib/systemd:ro     -v /e
tc:/etc:ro --label docker_bench_security     docker-bench-security
# ------------------------------------------------------------------------
# Docker Bench for Security v1.3.5
#
# Docker, Inc. (c) 2015-
#
# Checks for dozens of common best-practices around deploying Docker containers in production.
# Inspired by the CIS Docker Benchmark v1.2.0.
# ------------------------------------------------------------------------

Initializing Tue Mar  3 14:42:09 UTC 2020


[INFO] 1 - Host Configuration

[INFO] 1.1 - General Configuration
[NOTE] 1.1.1  - Ensure the container host has been Hardened
[INFO] 1.1.2  - Ensure Docker is up to date
[INFO]        * Using 19.03.6, verify is it up to date as deemed necessary
[INFO]        * Your operating system vendor may provide support and security maintenance for Docker

[INFO] 1.2 - Linux Hosts Specific Configuration
[WARN] 1.2.1 - Ensure a separate partition for containers has been created
[INFO] 1.2.2  - Ensure only trusted users are allowed to control Docker daemon
[INFO]        * docker:x:1001:ubuntu,1001
[WARN] 1.2.3  - Ensure auditing is configured for the Docker daemon
[PASS] 1.2.4  - Ensure auditing is configured for Docker files and directories - /var/lib/docker
[PASS] 1.2.5  - Ensure auditing is configured for Docker files and directories - /etc/docker
[INFO] 1.2.6  - Ensure auditing is configured for Docker files and directories - docker.service
[INFO]        * File not found
[INFO] 1.2.7  - Ensure auditing is configured for Docker files and directories - docker.socket
[INFO]        * File not found
[PASS] 1.2.8  - Ensure auditing is configured for Docker files and directories - /etc/default/docker
[INFO] 1.2.9  - Ensure auditing is configured for Docker files and directories - /etc/sysconfig/docker
[INFO]        * File not found
[PASS] 1.2.10  - Ensure auditing is configured for Docker files and directories - /etc/docker/daemon.json
[INFO] 1.2.11  - Ensure auditing is configured for Docker files and directories - /usr/bin/containerd
[INFO]         * File not found
[INFO] 1.2.12  - Ensure auditing is configured for Docker files and directories - /usr/sbin/runc
[INFO]         * File not found
```


Workshop: Docker Security

- Apparmor
- Purpose: ระงับการทำงานบางอย่างบน container ที่ไม่จำเป็นเพื่อลดความเสี่ยงในการใช้งานเกินขอบเขตของ application

```
ubuntu@ip-10-0-1-104:~$ docker container run -dt --name sectestrestrict --security-opt "apparmor=restrict-apparmor" labdocker/alpine:latest sh
6db2a1be7b9d1b3f15409e49bc1a0e1a9373cc9b8740a017ff8fb163283b8d1b
ubuntu@ip-10-0-1-104:~$ docker container exec -it sectestdefault sh
Error: No such container: sectestdefault
ubuntu@ip-10-0-1-104:~$ docker container exec -it sectestrestrict sh
/ # ping 1.1.1.1
PING 1.1.1.1 (1.1.1.1): 56 data bytes
ping: can't create raw socket: Permission denied
/ # apk update
ERROR: Unable to lock database: Permission denied
ERROR: Failed to open apk database: Permission denied
/ # apk add curl
ERROR: Unable to lock database: Permission denied
ERROR: Failed to open apk database: Permission denied
/ # curl https://www.google.com
sh: curl: not found
/ # touch /tmp/testcreatefile1
touch: /tmp/testcreatefile1: Permission denied
/ # touch /home/testcreatefile2
touch: /home/testcreatefile2: Permission denied
/ # touch /etc/testcreatefile3
touch: /etc/testcreatefile3: Permission denied
/ # touch /proc/testcreatefile4
touch: /proc/testcreatefile4: No such file or directory
/ # cp /tmp/testcreatefile /proc/testprohibitfile
cp: can't stat '/tmp/testcreatefile': No such file or directory
/ # ls -hl /proc | grep test
/ # exit
ubuntu@ip-10-0-1-104:~$ docker container stop sectestrestrict && docker container rm sectestrestrict
sectestrestrict
sectestrestrict
ubuntu@ip-10-0-1-104:~$
```


Workshop: Docker Security

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

```
root@labdocker:~# docker push praparn/alpine:sign
The push refers to a repository [docker.io/praparn/alpine]
6102f0d2ad33: Layer already exists
sign: digest: sha256:65242e8220a341cec40628caaea77eb4acd2fc252329aa853526fde15a4a1d85 size: 528
Signing and pushing trust metadata
You are about to create a new root signing key passphrase. This passphrase
will be used to protect the most sensitive key in your signing system. Please
choose a long, complex passphrase and be careful to keep the password and the
key file itself secure and backed up. It is highly recommended that you use a
password manager to generate the passphrase and keep it safe. There will be no
way to recover this key. You can find the key in your config directory.
Enter passphrase for new root key with ID ca6ab4b:
Repeat passphrase for new root key with ID ca6ab4b:
Enter passphrase for new repository key with ID 68d88cd (docker.io/praparn/alpine):
Repeat passphrase for new repository key with ID 68d88cd (docker.io/praparn/alpine):
```

Registry

• • •

Registry

- Registry เป็น container แบบหนึ่งที่ใช้สำหรับจัดเก็บ image ที่สร้างขึ้นไว้บน private server โดยไม่จำเป็นต้องผ่าน public internet (hub.docker.com) ทำให้สามารถจัดเก็บ image ภายในองค์กรได้อย่างปลอดภัย

- Registry เป็นพื้นที่สำหรับจัดเก็บ image ที่ถูกสร้างขึ้นโดยแบ่งออกตาม image name และ tag version ที่กำหนดไว้

  Ex: labdocker/alpineweb:latest

- กรณีการใช้งาน registry ระหว่าง Server ด้วยกันสามารถกำหนด TLS security
  ระหว่างกันได้

- สามารถเลือก storage backend อื่นๆในการจัดเก็บ image ได้เช่น
- Etc: s3 (aws), azure, gcs (google cloud) etc
- มี option gc (garbage collect) ด้วย option
- “bin/registry garbage-collect [--dry-run] config.yml”

docker

Workshop: Registry

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

```
docker image pull registry:2.8.0 (registry version 2.0)
```

- Start container เพื่อเริ่มใช้งาน

```
docker container run -d -p 5000:5000 \
--restart=always --name registry \
 -e REGISTRY_STORAGE_DELETE_ENABLED=true \
--mount type=bind,source=/home/docker,target=/var/lib/registry \
registry:2.8.0
```

- ดำเนินการ tag image และทดสอบ push image docker เข้า registry

```
docker image tag labdocker/alpine:3.13.6 \
localhost:5000/alpine:3.13.6
```

```
docker push localhost:5000/alpine:3.13.6
```

- *ต้องบันทึก digest image ไว้เพื่อใช้อ้างอิงภายหลัง

Workshop: Registry

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

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

- ตรวจสอบ revision image tag ที่จัดเก็บไว้ในแต่ละ image

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

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

- สามารถทำการลบ image ที่จัดเก็บไว้โดยอ้างอิงจาก digest

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


Workshop: Registry

- ตรวจสอบ image ทาง physical file จาก docker-machine

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

- ลบ image file ออกจาก docker-machine

```
docker image rm localhost:5000/alpine:3.13.6
```

- ทำการดึง image ใหม่มาจาก registry

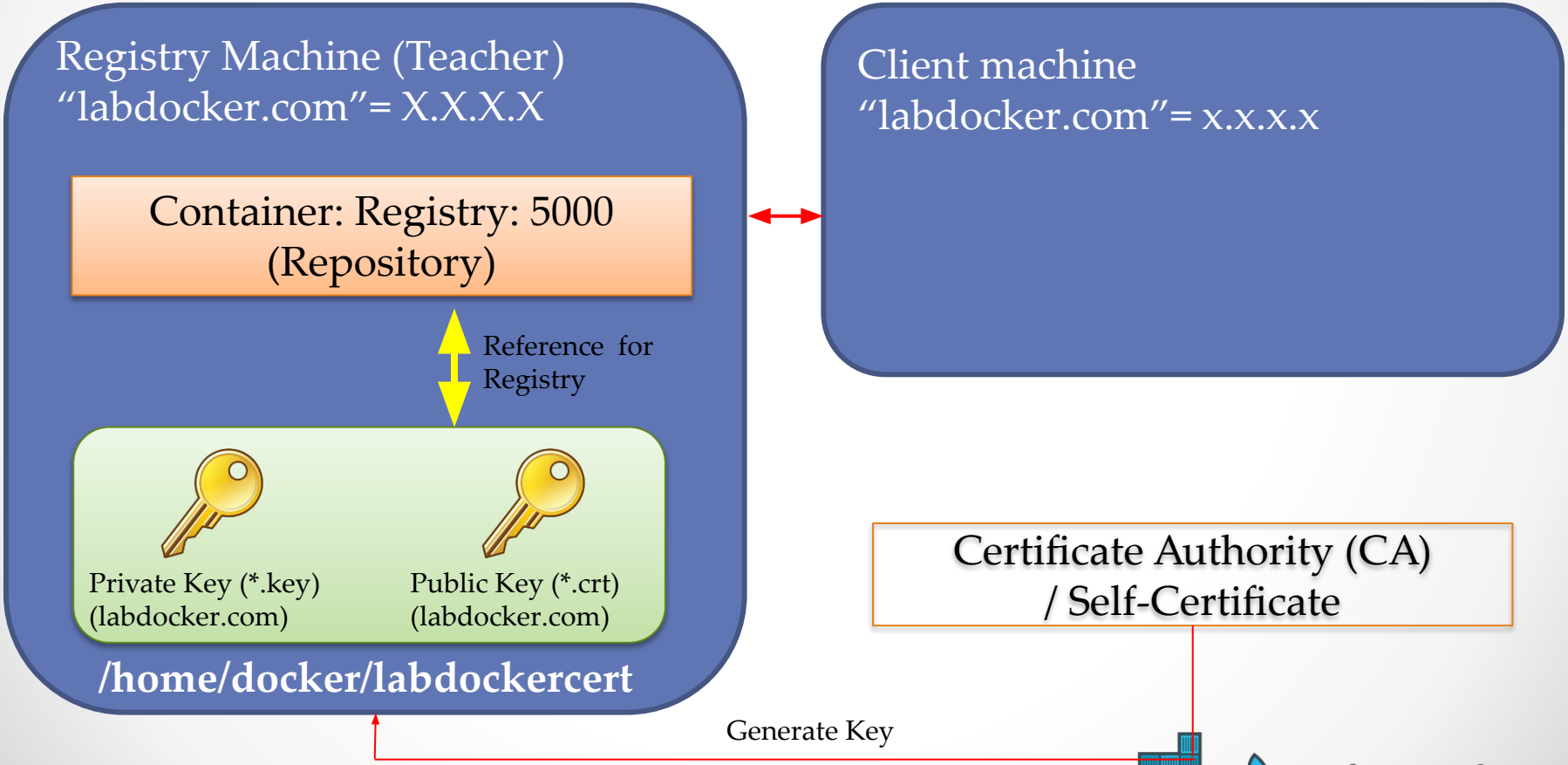
```
docker image pull localhost:5000/alpine:3.13.6
```

- ทำการลบ container registry เพื่อยกเลิกการใช้งาน

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

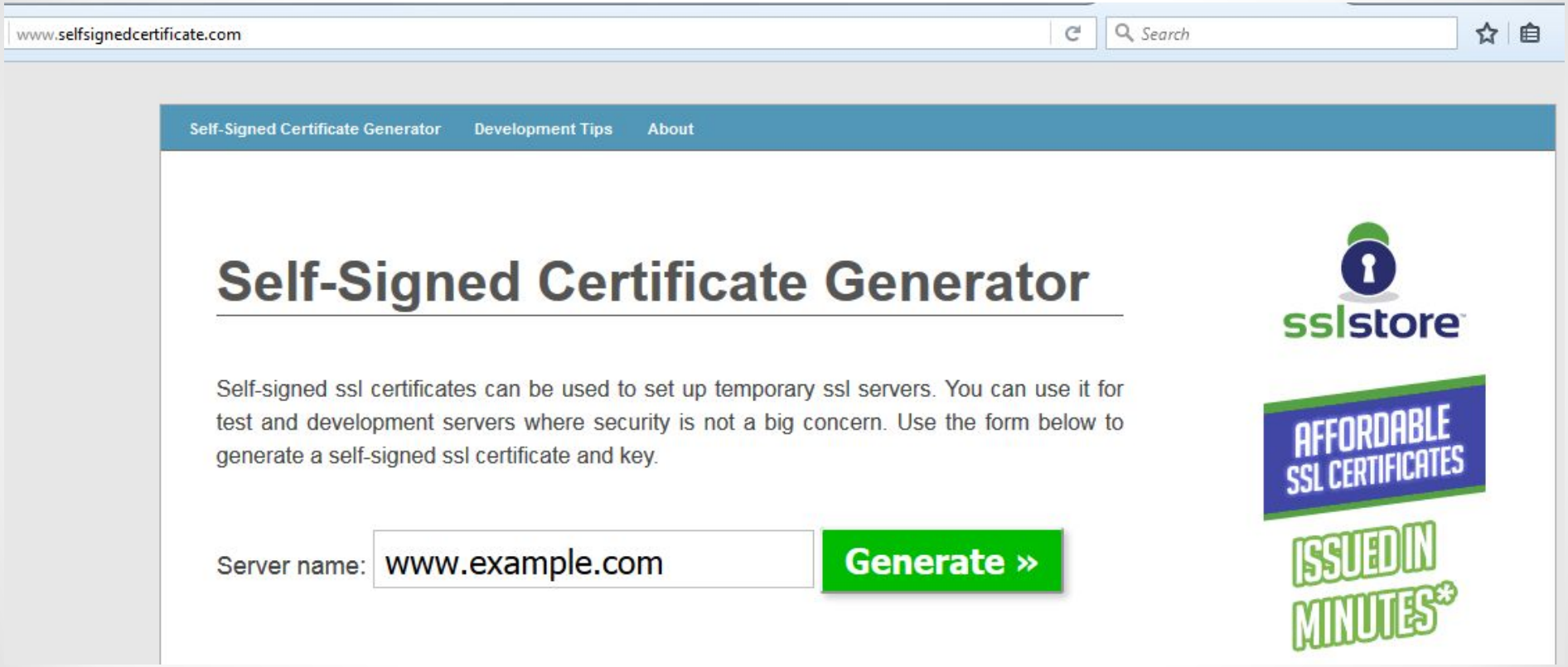

Registry on difference machine

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



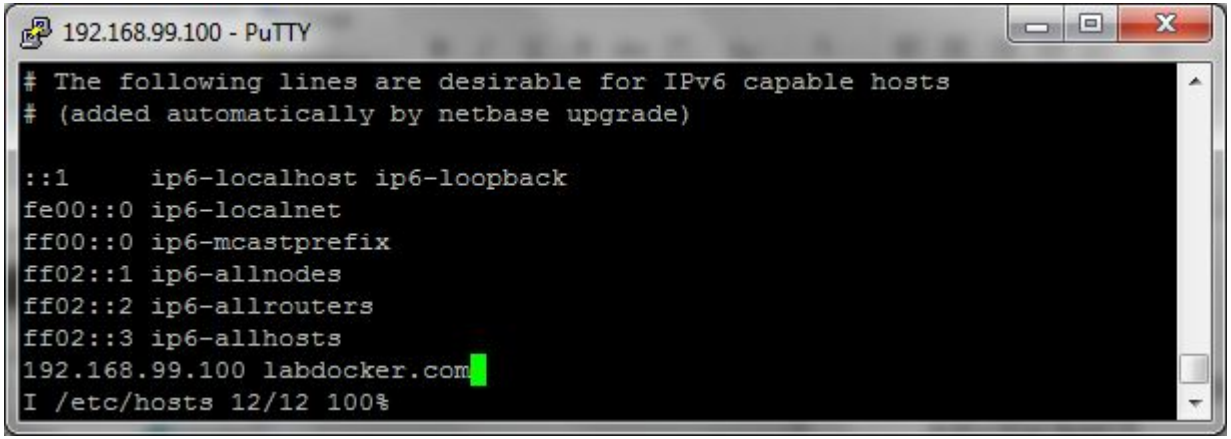
Registry on difference machine

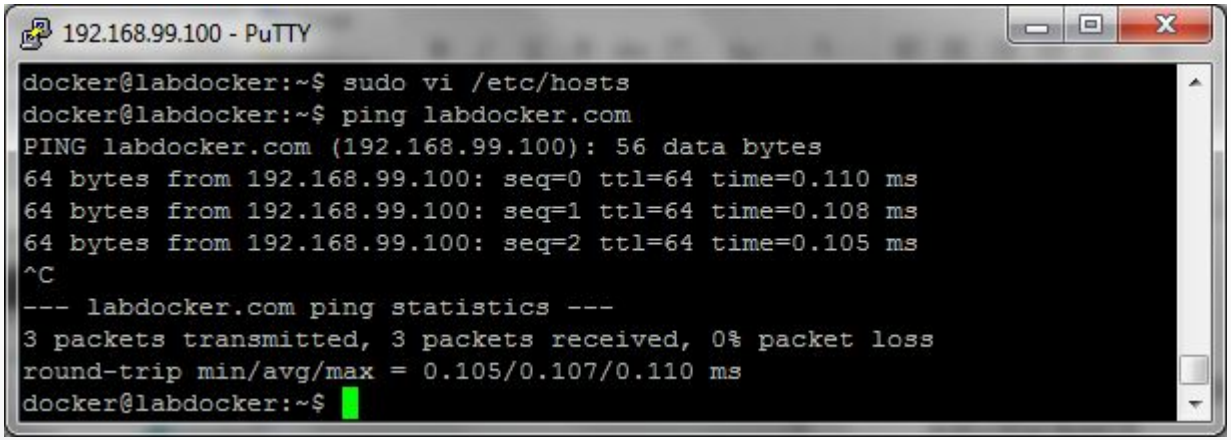
- การสร้าง certificate จาก CA / Self-Signed



Registry on difference machine

- เพิ่ม domain “labdocker.com” ลงใน /etc/hosts (registry, all machine)





Registry on difference machine

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

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

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


Registry on difference machine

- เริ่มใช้งาน registry โดย start container ดังนี้ (labdocker)

```
docker container run -d -p 5000:5000 \
      --restart=always --name registrylab \
      --mount type=bind,source=$(pwd),target=/var/lib/registry \
      --mount type=bind,source=$(pwd),target=/certs  \
      --mount type=bind,source=$(pwd),target=/auth\
      -e "REGISTRY_AUTH=htpasswd" \
      -e "REGISTRY_AUTH_HTPASSWD_REALM=Registry Realm" \
      -e REGISTRY_AUTH_HTPASSWD_PATH=/auth/htpasswd \
      -e REGISTRY_STORAGE_DELETE_ENABLED=true \
      -e REGISTRY_HTTP_TLS_CERTIFICATE=/certs/labdocker.com.crt \
      -e REGISTRY_HTTP_TLS_KEY=/certs/labdocker.com.key \
      registry:2.8.0
```

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

```
docker image tag labdocker/alpine:3.13.6 labdocker.com:5000/alpine:1.0
docker login -u <username> -p <password>
docker image push labdocker.com:5000/alpine:1.0
```


Registry on difference machine

- ในการตรวจสอบ registry image file สำหรับ registry ที่มีการ enable TLS เรียบบร้อยแล้วต้องเพิ่มพารามิเตอร์ “--cacert “ เพื่อระบุ certificate สำหรับการใช้งานและเปลี่ยน protocol เป็น https

- การ List รายการ image ที่จัดเก็บไว้บน

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

- ตรวจสอบ revision image tag ที่จัดเก็บไว้ในแต่ละ image

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

- สามารถทำการลบ image ที่จัดเก็บไว้โดยอ้างอิงจาก digest

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


Registry on difference machine

- ดึง image ผ่าน labdocker 2

```
[ubuntu@ip-10-0-1-239:~/docker-workshop-112018-cloud/Workshop-2-3-Registry$ docker login labdocker.com:5000
[Username: docker
[Password:
WARNING! Your password will be stored unencrypted in /home/ubuntu/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

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

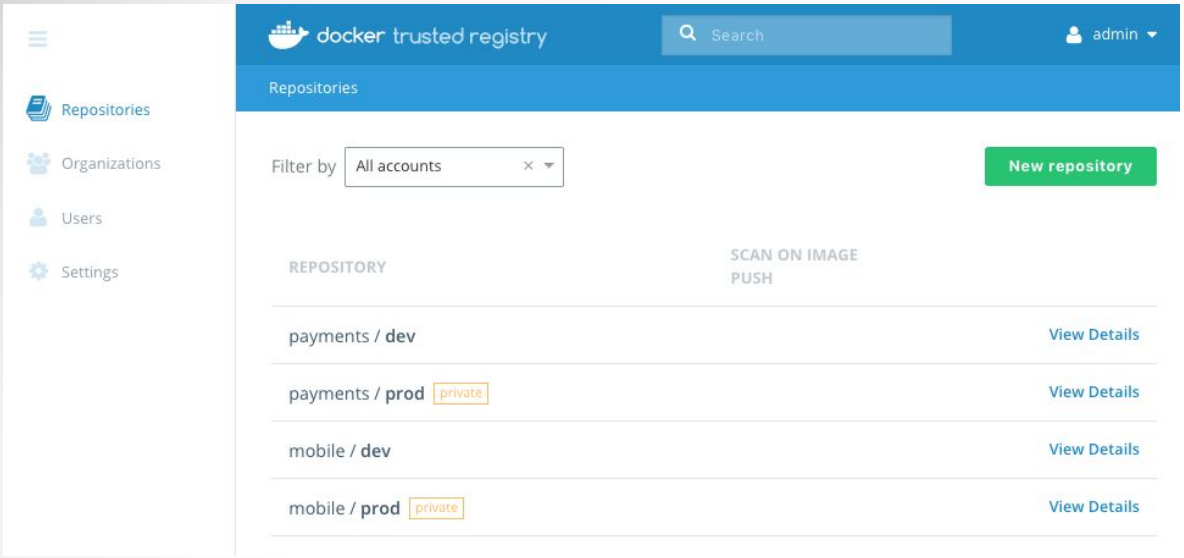
- Optional: สำหรับเพิ่มรันเว็บเพื่อตรวจสอบ Container

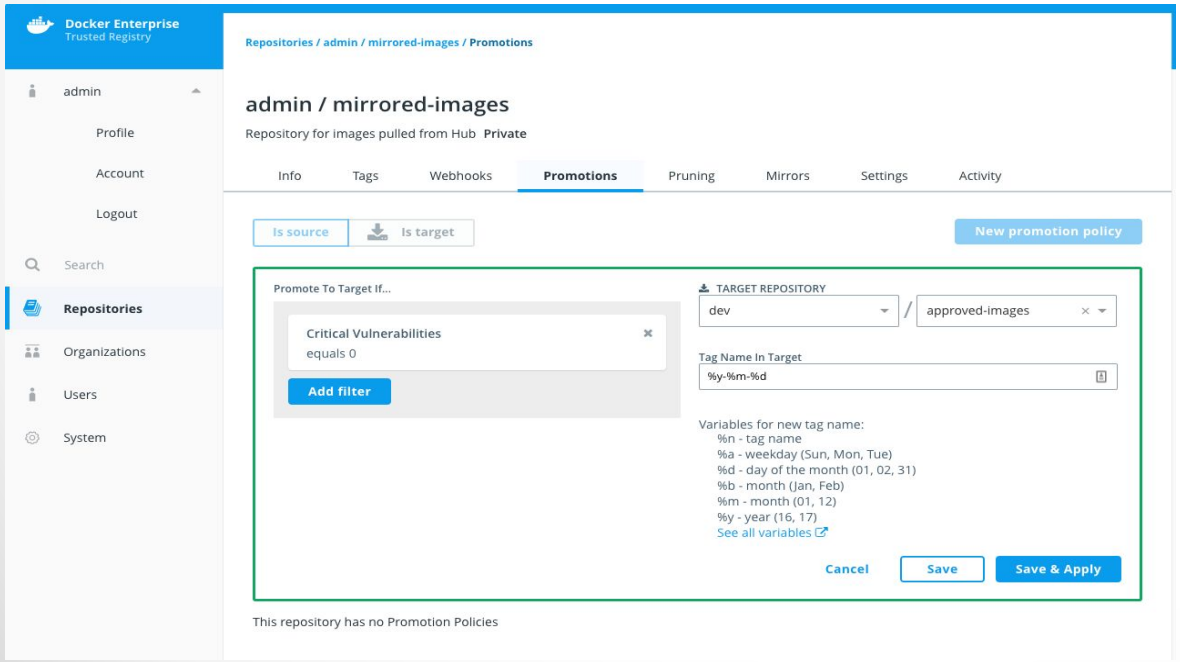
```
-v `pwd`/auth:/auth \
 -e "REGISTRY_AUTH=htpasswd" \
 -e "REGISTRY_AUTH_HTPASSWD_REALM=Registry
Realm" \
```

- ทำการลบ container registry เพื่อยกเลิกการใช้งาน

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

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





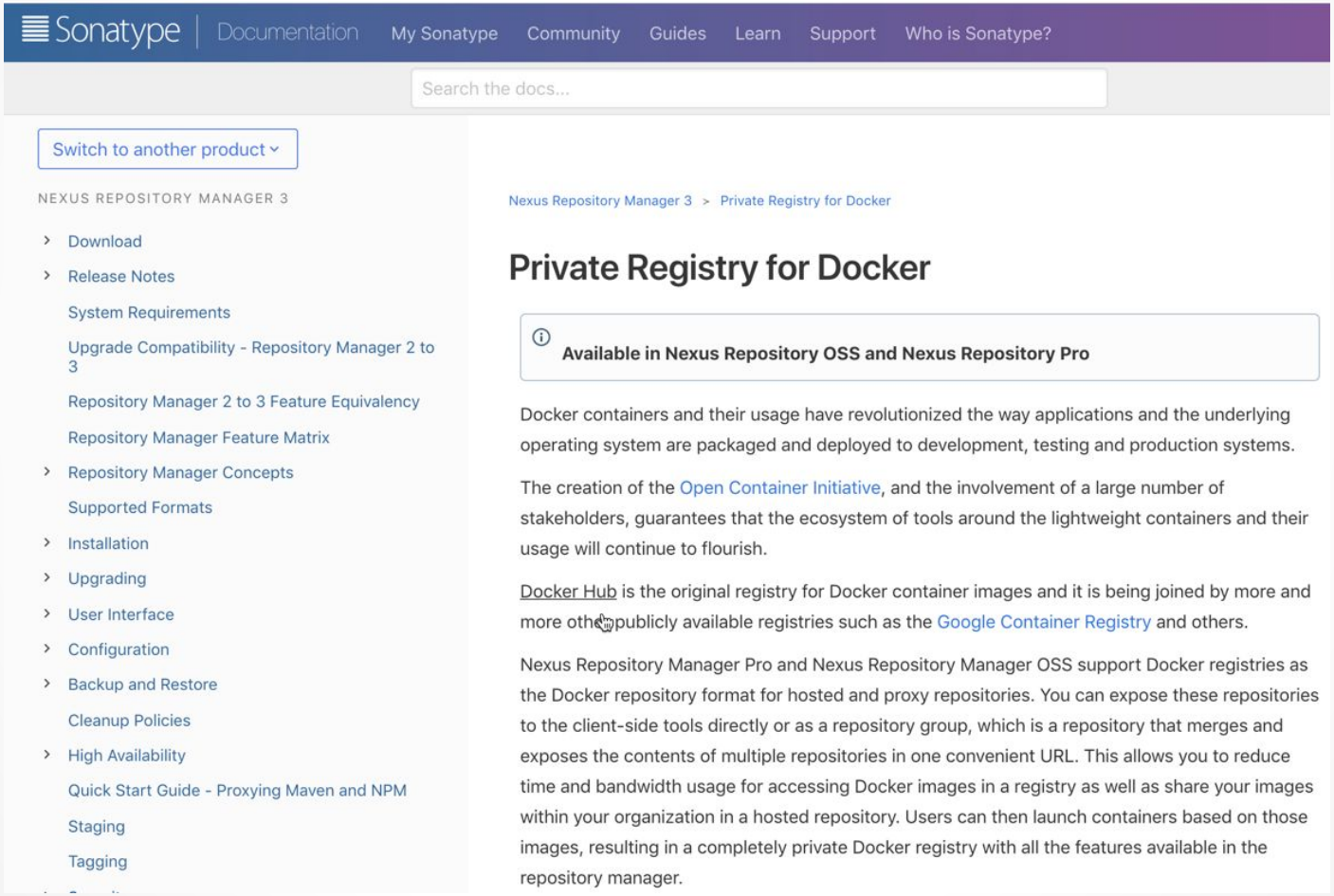
Docker Trust Registry (DTR)

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

docker

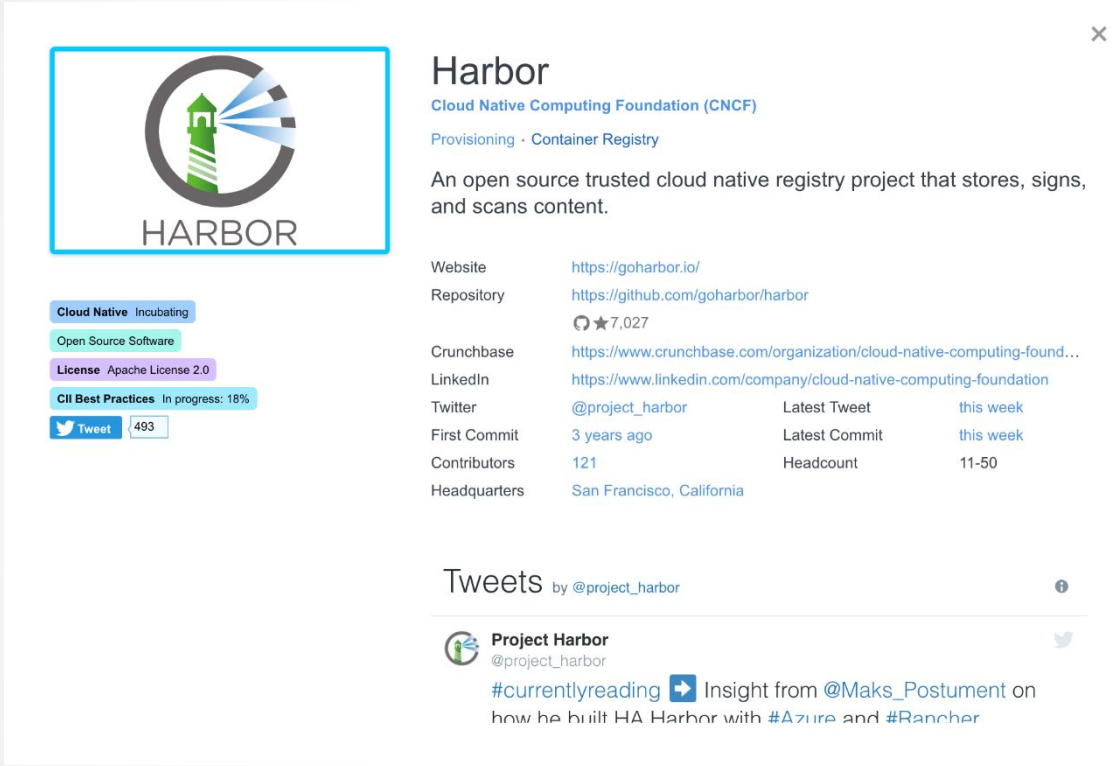
Artifact Registry

- Sonatype



Artifact Registry

- Harbor CNCF Registry (Incubator)



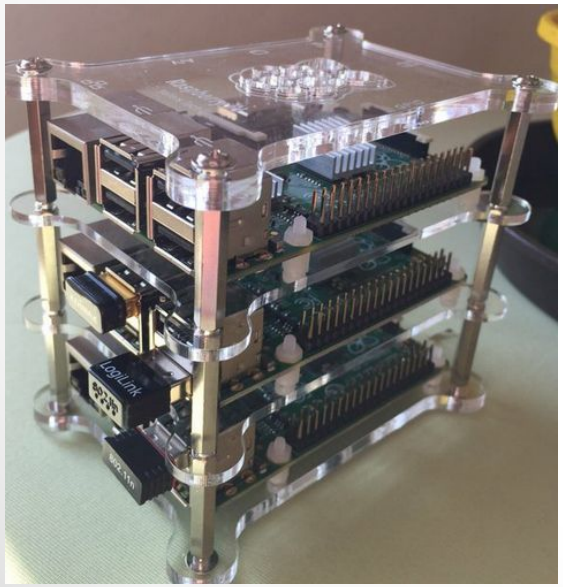
docker

Swarm

• • •

Swarm: Conceptual

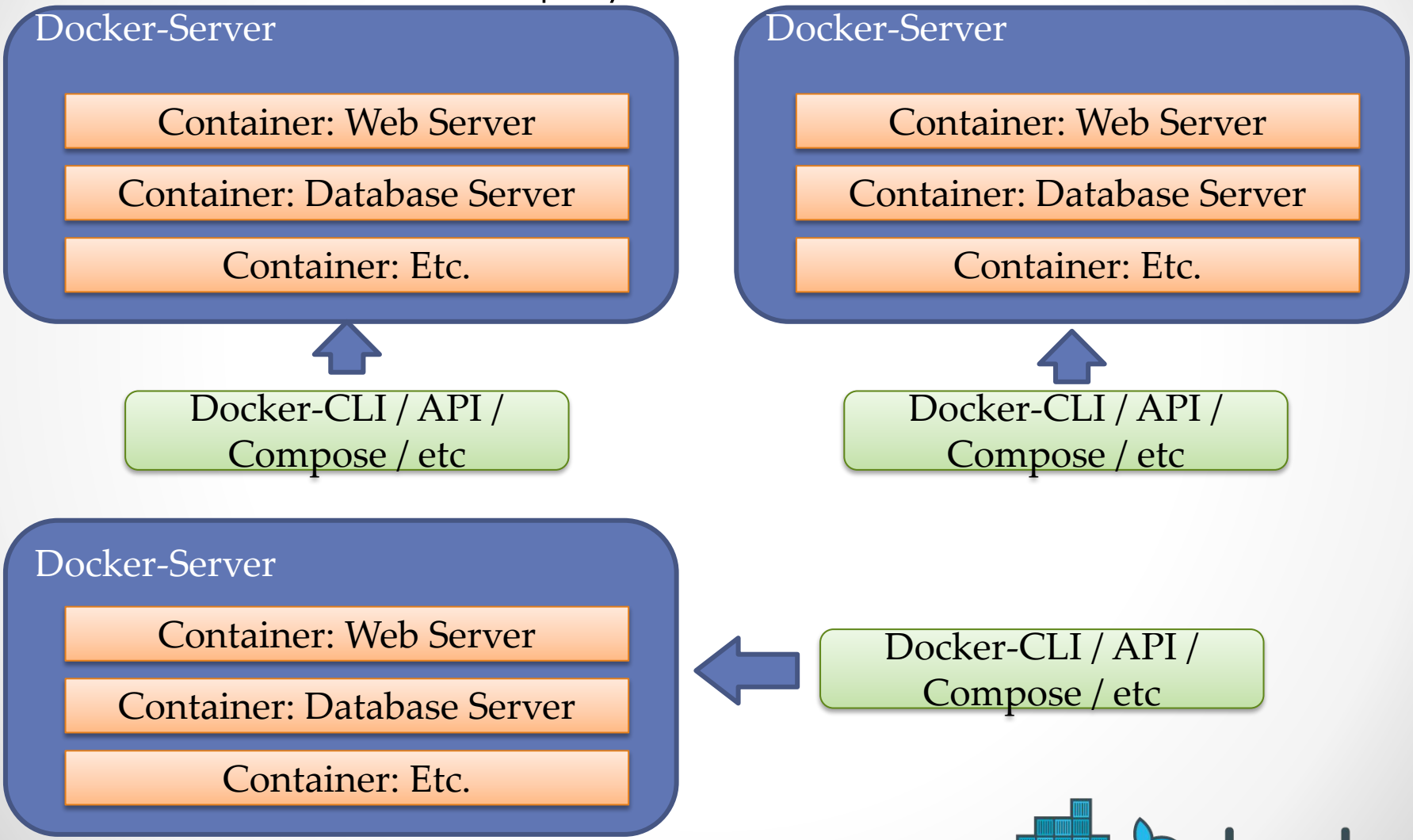
- Swarm ถือเป็น native tool ในการจัดการ docker-engine หลายๆ เครื่องให้รวมเป็น resource pool เดียวกัน เพื่อให้ง่ายต่อการบริหารจัดการ
- เพิ่มขีดความสามารถในการทำงานร่วมกันของ docker-engine
- Hybrid Swan Cluster (Windows / Linux)
- Fail Over / High Availability (HA)
- Micro Service



docker

Swarm: Conceptual

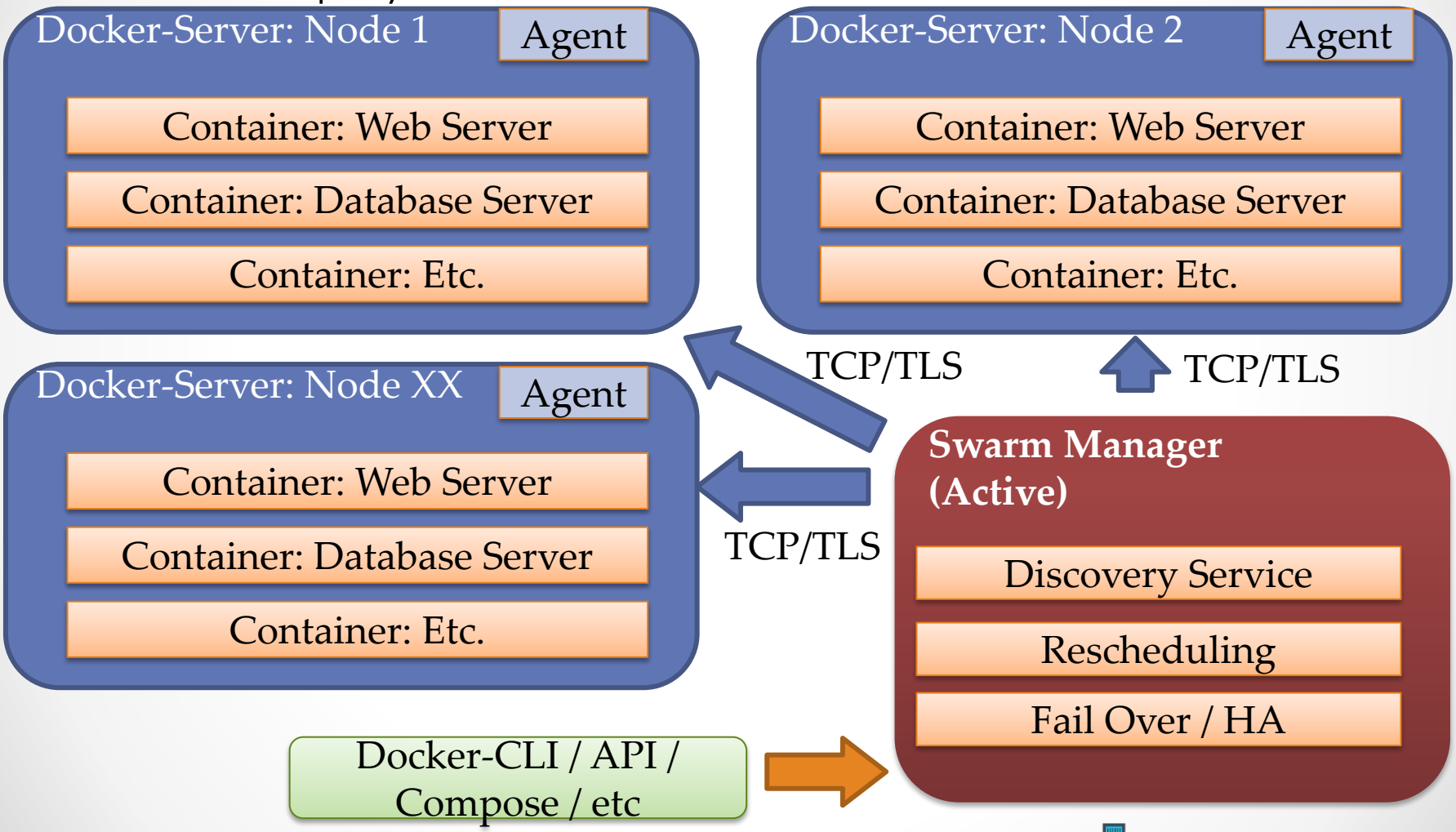
- Traditional Docker Deployment



docker

Swarm: Conceptual

- Swarm Deployment

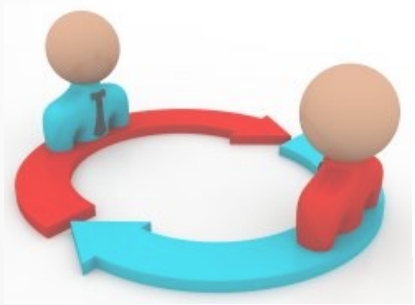


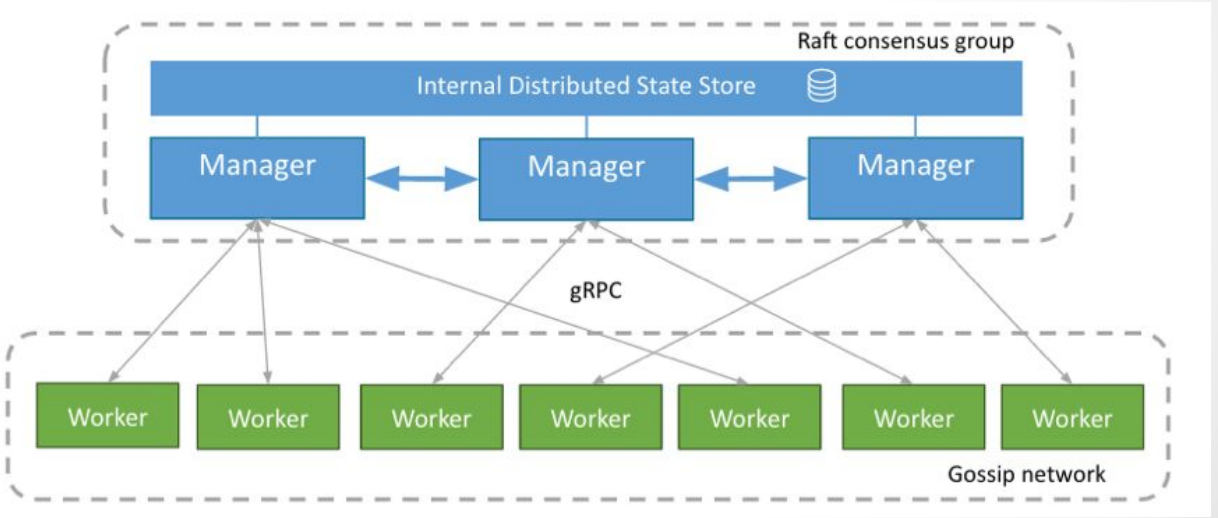
docker

Swarm Mode Architecture

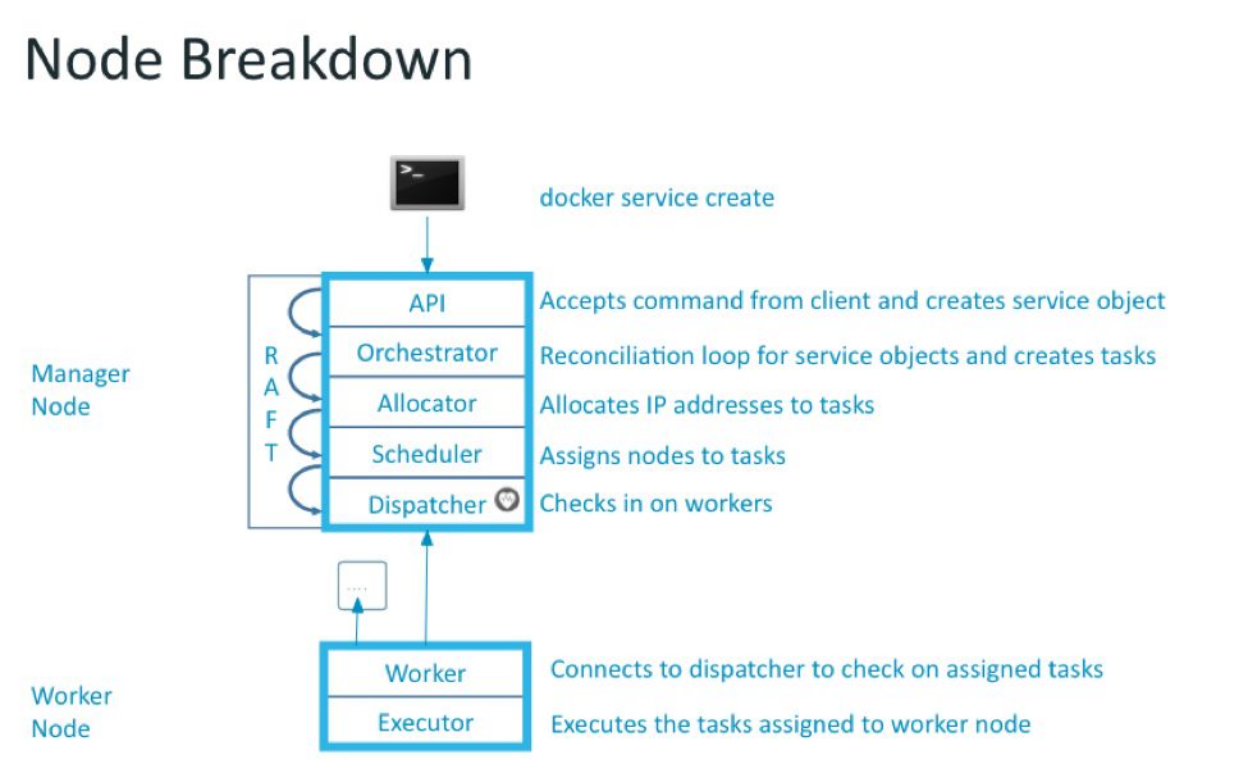
- Swarm Mode (build-in swarm) เป็นฟังก์ชั่นที่มาพร้อม docker-engine ตั้งแต่แรก และสามารถบริหารจัดการ swarm ได้ด้วยตัวเองรวมถึง build-in TLS certificate เพื่อใช้ทำงานระหว่าง node ทันที (Security On the Box)

- Swarm Role
  - Manager Node
  - Worker Node

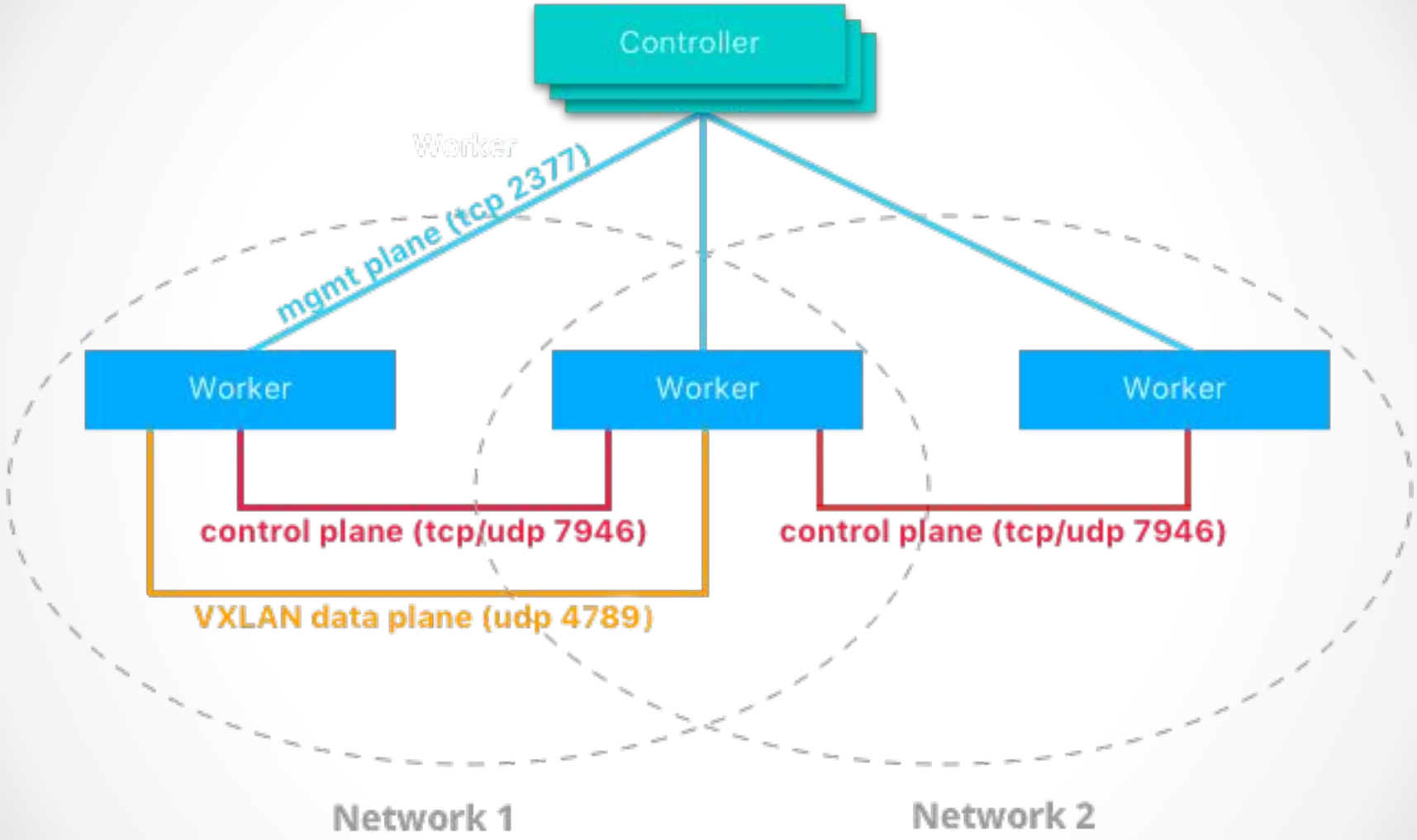




Swarm Mode Architecture

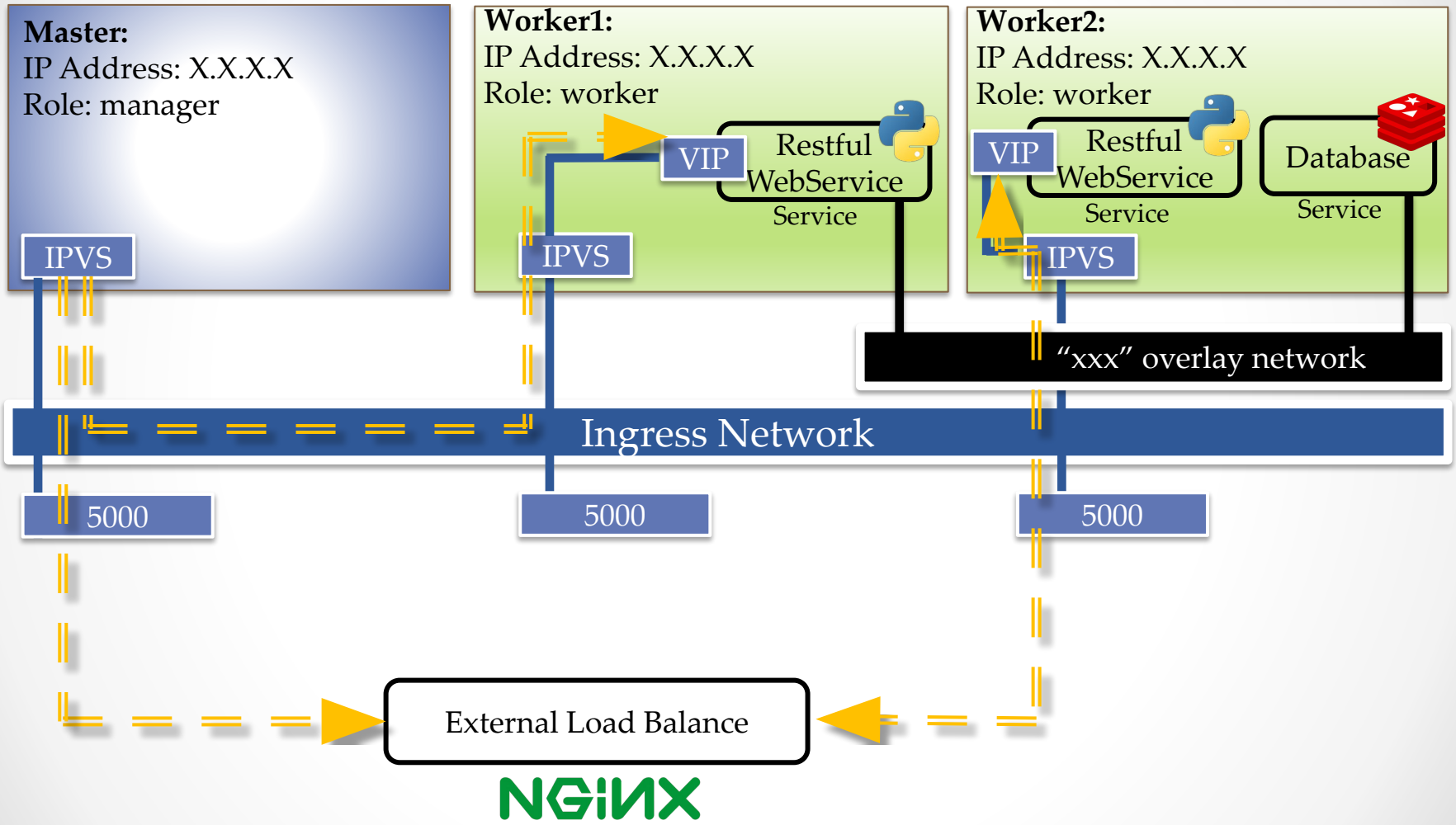


Swarm Mode Architecture



docker

Swarm Mode Architecture



docker

Swarm Mode Architecture

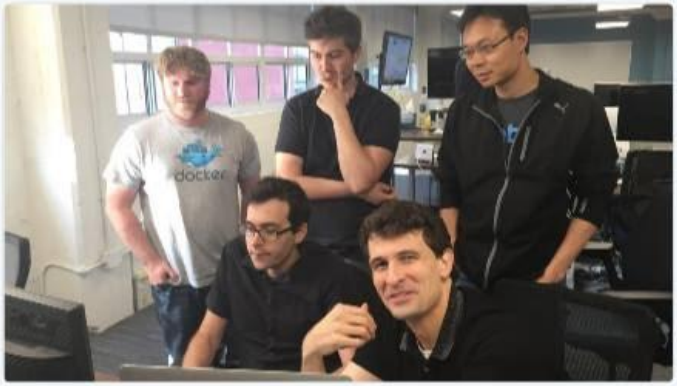
https://blog.docker.com/2016/07/docker-built-in-orchestration-ready-for-production-docker-1-12-goes-ga/

All
Engineering
Curated
Docker Weekly

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

Nathan LeClaire @upthecyberpunks 48m

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



3

Nathan LeClaire
@upthecyberpunks

@chanwit @dongluochen

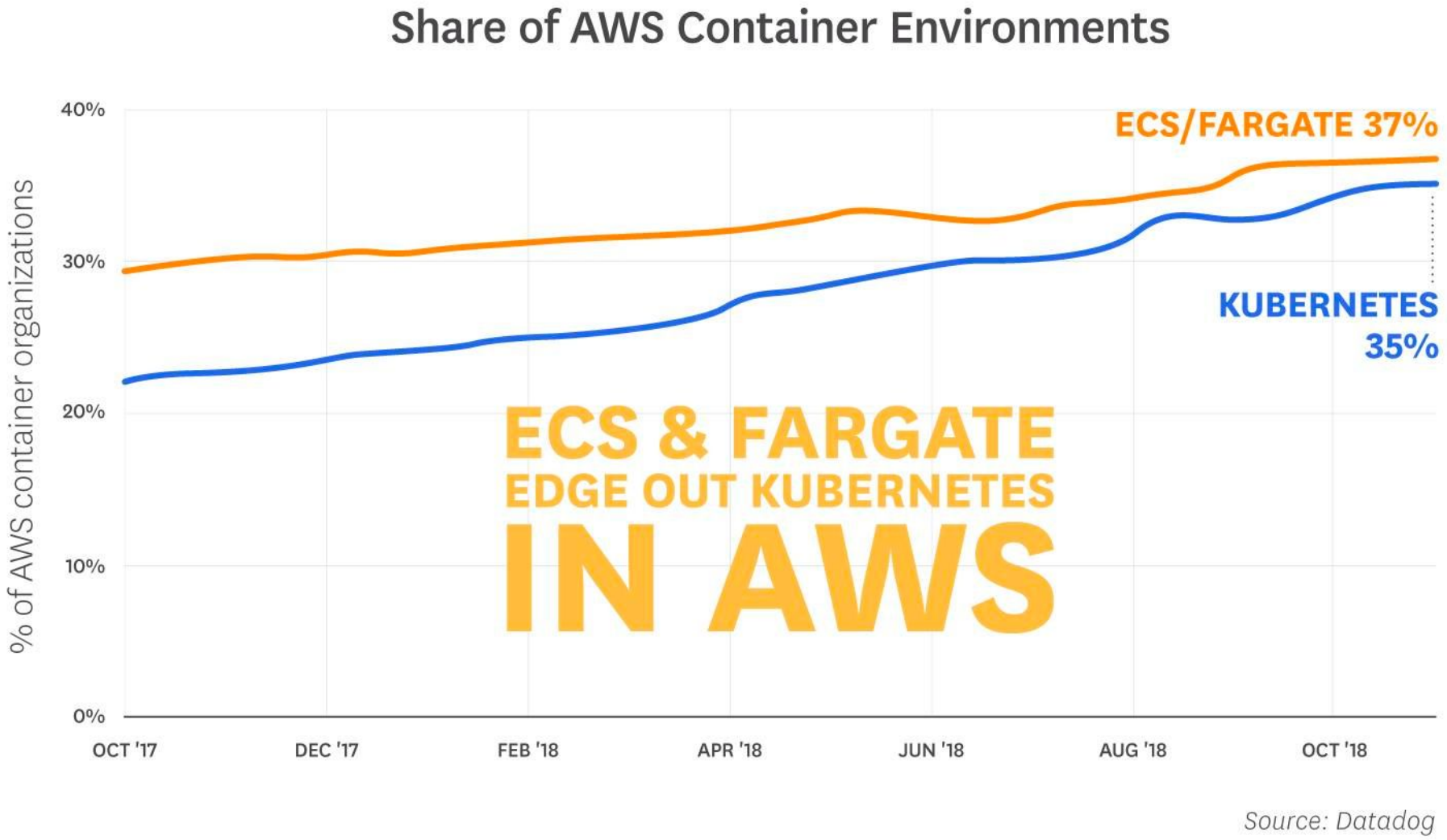
Jul 30, 2016, 7:56 AM

docker

K8S vs Docker what is the difference ?

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

K8S vs Docker what is the difference ?



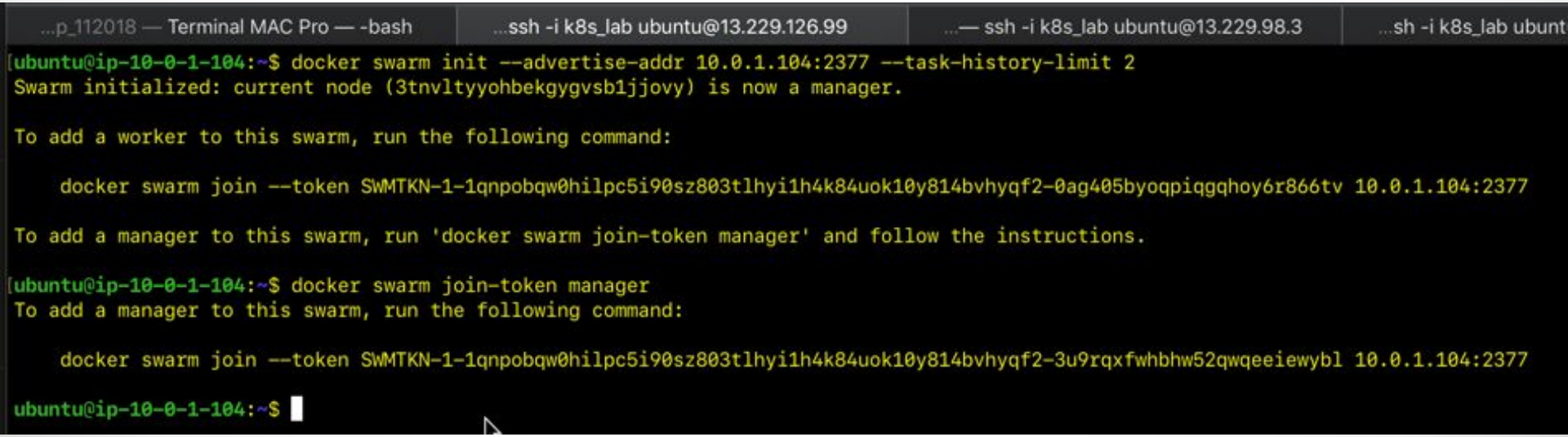
docker

Swarm Init/Join

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

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

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



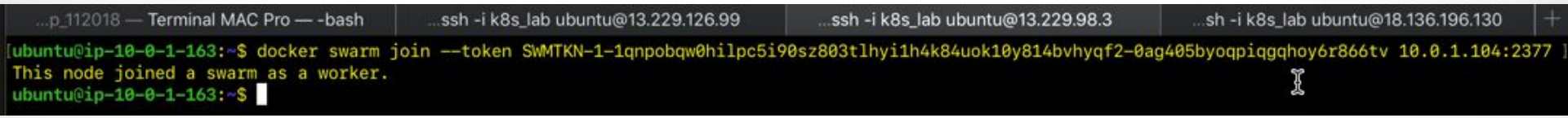
Swarm Init/Join

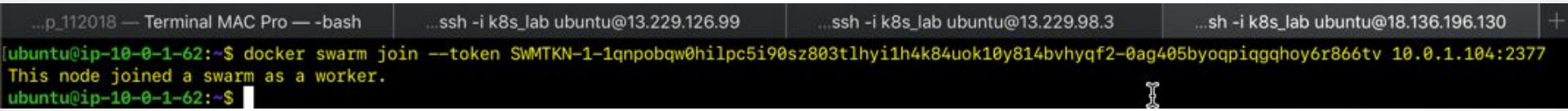
- Option:
  - --cert-expiry duration: กำหนดระยะเวลาในการ expire certificate (default: 2160 hours = 90 Days)
  - --dispatcher-heartbeat duration: กำหนดช่วงเวลาในการ check heartbeat ภายใน swarm (Default: 5 seconds)
  - --external-ca value: กำหนดให้ใช้งาน CA certificate ภายนอกเพื่อทำ trust node ภายใน swarm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

Swarm Init/Join

- ทำการ join node เข้าไปยัง swarm cluster ด้วยคำสั่ง

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





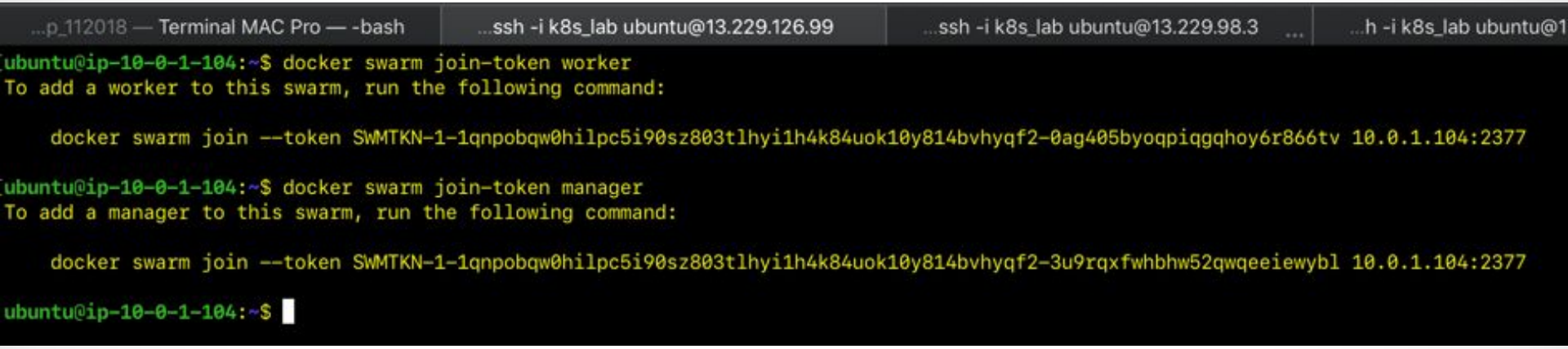
Swarm Init/Join

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

Swarm Init/Join

- ตรวจสอบ Token ที่ใช้ในการ Join Swarm (Worker/Manager)

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



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

Swarm Init/Join

- ตรวจสอบ node ที่ทำงานภายใต้ swarm cluster ด้วยคำสั่ง docker node ls

```
docker node ls
```

Swarm Init/Join

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

Workshop: Swarm Mode

- ทำการตรวจสอบเครื่องใน LAB ที่ได้รับมอบหมายแต่ละกลุ่มซึ่ง

Name: Training_DockerZerotoHero_StudentG1_1
(IP: 10.0.1.X), Role: Master
Name: Training_DockerZerotoHero_StudentG1_2
(IP: 10.0.1.X), Role: Worker1
Name: Training_DockerZerotoHero_StudentG1_3
(IP: 10.0.1.X), Role: Worker2

- เริ่ม initial swarm ที่เครื่อง Master

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


Workshop: Swarm Mode

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

```
docker swarm join --token
SWMTKN-1-1qnpobqw0hilpc5i90sz803tlhyi1h4k84uok10y81
4bvhyqf2-0ag405byoqpiqgqhoy6r866tv 10.0.1.104:2377
```

- ตรวจสอบสถานะของ swarm หลังจาก join node เข้าไปเรียบร้อยแล้ว

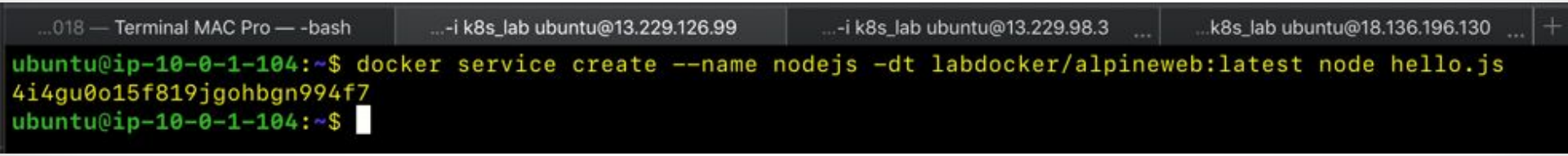
```
docker node ls
```

Swarm Service

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

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

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



Swarm Service

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

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

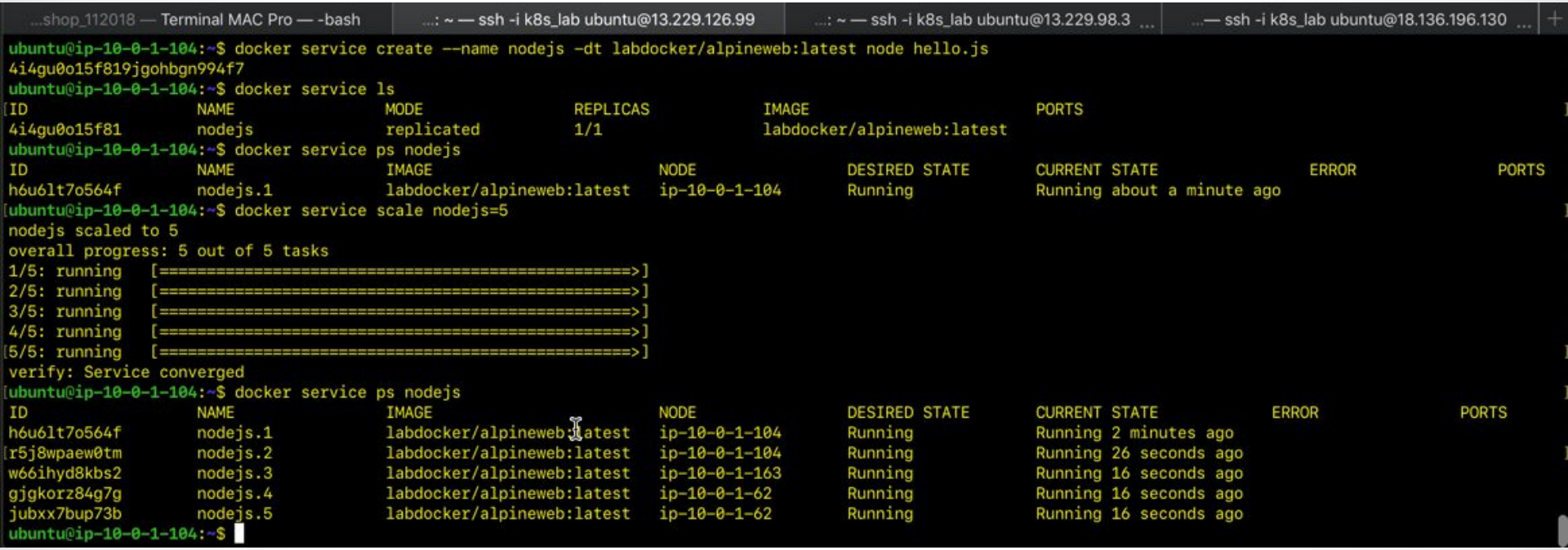
Swarm Service

```
    --name string                         Service name
    --network network                     Network attachments
    --no-healthcheck                      Disable any container-specified HEALTHCHECK
    --no-resolve-image                    Do not query the registry to resolve image digest and supported platforms
    --placement-pref pref                 Add a placement preference
-p, --publish port                        Publish a port as a node port
-q, --quiet                               Suppress progress output
    --read-only                           Mount the container's root filesystem as read only
    --replicas uint                       Number of tasks
    --reserve-cpu decimal                 Reserve CPUs
    --reserve-memory bytes                Reserve Memory
    --restart-condition string            Restart when condition is met ("none"|"on-failure"|"any") (default "any")
    --restart-delay duration              Delay between restart attempts (ns|us|ms|s|m|h) (default 5s)
    --restart-max-attempts uint           Maximum number of restarts before giving up
    --restart-window duration             Window used to evaluate the restart policy (ns|us|ms|s|m|h)
    --rollback-delay duration             Delay between task rollbacks (ns|us|ms|s|m|h) (default 0s)
    --rollback-failure-action string      Action on rollback failure ("pause"|"continue") (default "pause")
    --rollback-max-failure-ratio float    Failure rate to tolerate during a rollback (default 0)
    --rollback-monitor duration           Duration after each task rollback to monitor for failure (ns|us|ms|s|m|h) (default 5s)
    --rollback-order string               Rollback order ("start-first"|"stop-first") (default "stop-first")
    --rollback-parallelism uint           Maximum number of tasks rolled back simultaneously (0 to roll back all at once) (default 1)
    --secret secret                       Specify secrets to expose to the service
    --stop-grace-period duration          Time to wait before force killing a container (ns|us|ms|s|m|h) (default 10s)
    --stop-signal string                  Signal to stop the container
-t, --tty                                 Allocate a pseudo-TTY
    --update-delay duration               Delay between updates (ns|us|ms|s|m|h) (default 0s)
    --update-failure-action string        Action on update failure ("pause"|"continue"|"rollback") (default "pause")
    --update-max-failure-ratio float      Failure rate to tolerate during an update (default 0)
    --update-monitor duration             Duration after each task update to monitor for failure (ns|us|ms|s|m|h) (default 5s)
    --update-order string                 Update order ("start-first"|"stop-first") (default "stop-first")
    --update-parallelism uint             Maximum number of tasks updated simultaneously (0 to update all at once) (default 1)
-u, --user string                         Username or UID (format: <name|uid>[:<group|gid>])
    --with-registry-auth                  Send registry authentication details to swarm agents
-w, --workdir string                      Working directory inside the container
```

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

Swarm Service

docker service scale nodejs=5



Swarm Service

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

docker service inspect nodejs | more

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

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


Swarm Service

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

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


Swarm Service

docker service ps nodejs

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

docker service rm nodejs

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


Swarm Service

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

Update a service

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

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

Swarm Service

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

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

Swarm Service

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

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

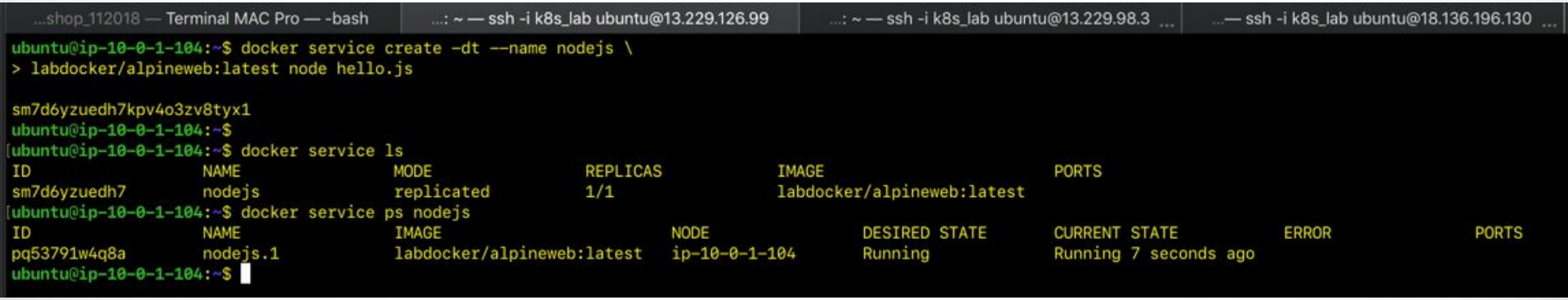
Workshop: Swarm Service

- สร้าง service nodejs เพื่อเริ่มทำงานบน swarm

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

- ตรวจสอบสถานะของ service หลังจากการสร้าง

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



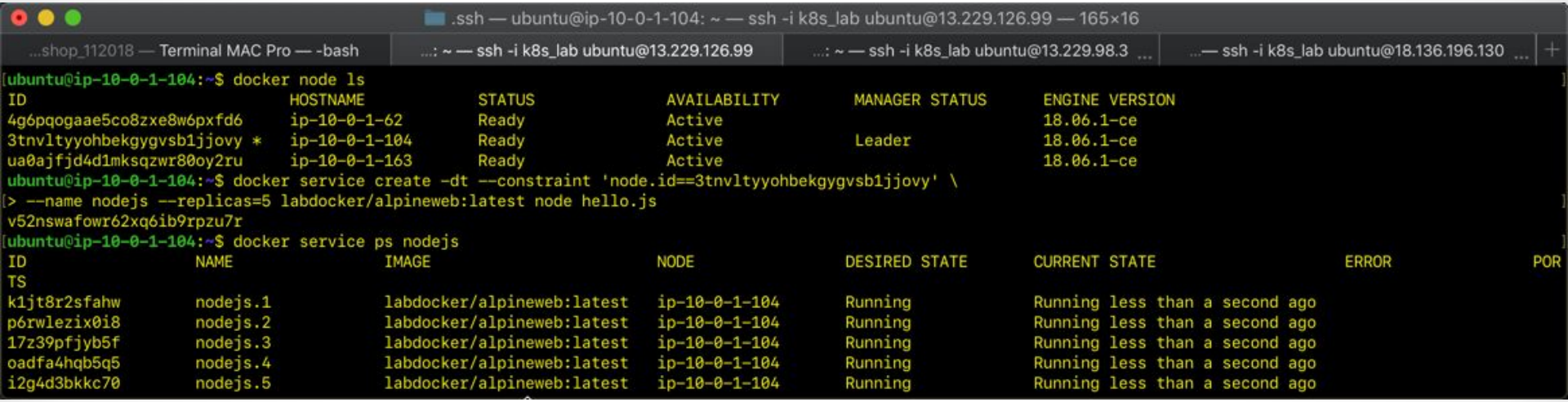
docker

Orchestrator Assignment

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

Orchestrator Assignment

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



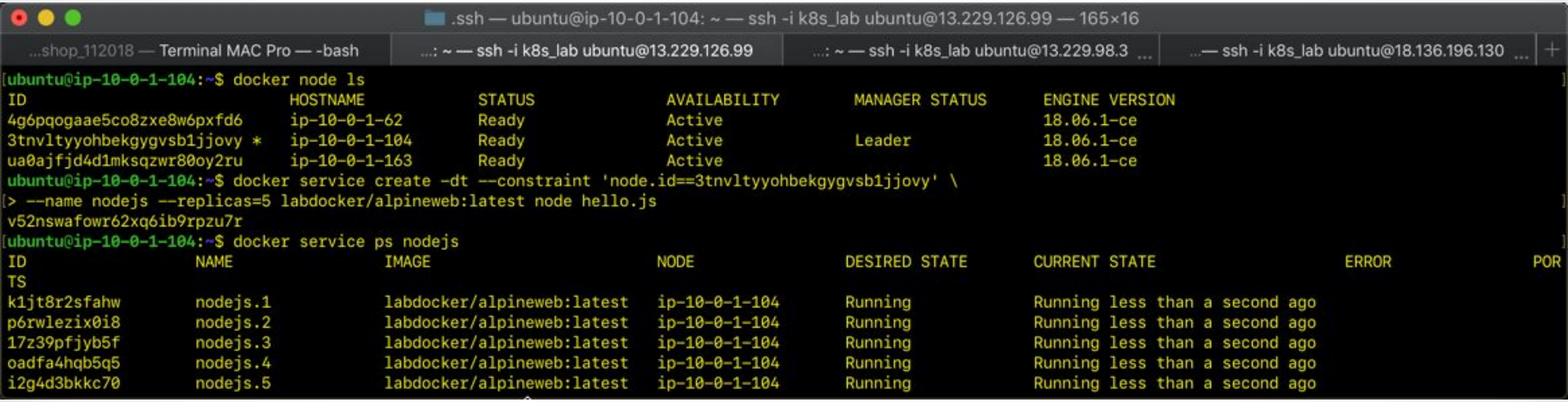
Orchestrator Assignment

- Remove constraint/Add constraint
  - node.hostname

Workshop: node constrain

- สร้าง service nodejs เพื่อเริ่มทำงานบน swarm

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



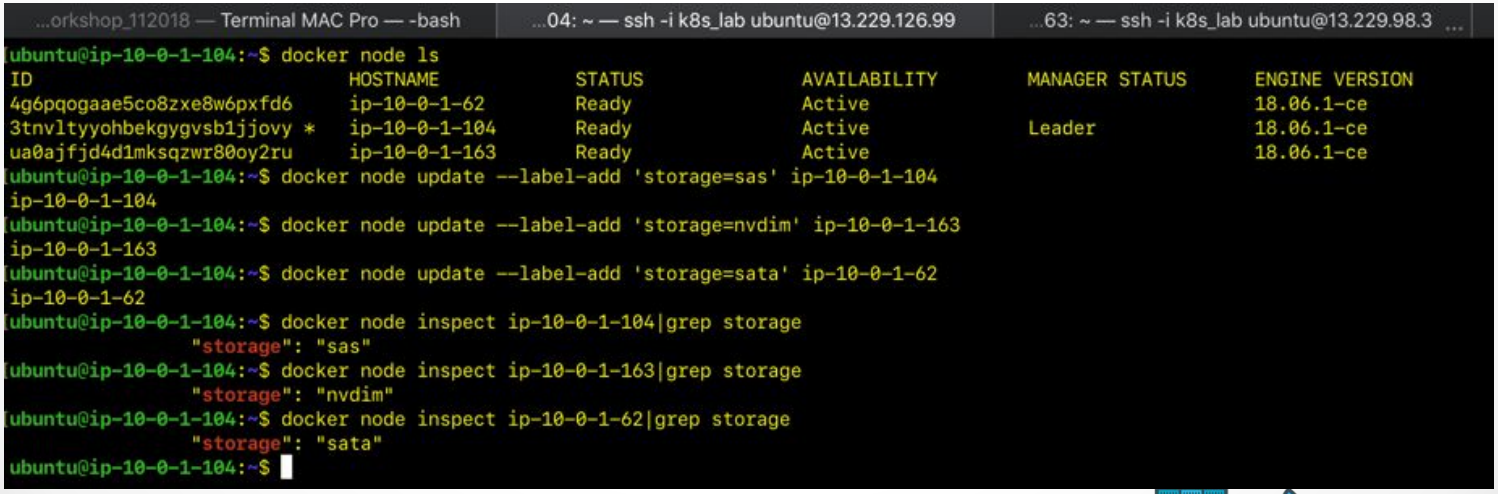
docker

Orchestrator Assignment

- node constrain (custom label)

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

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



Orchestrator Assignment

- custom label

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

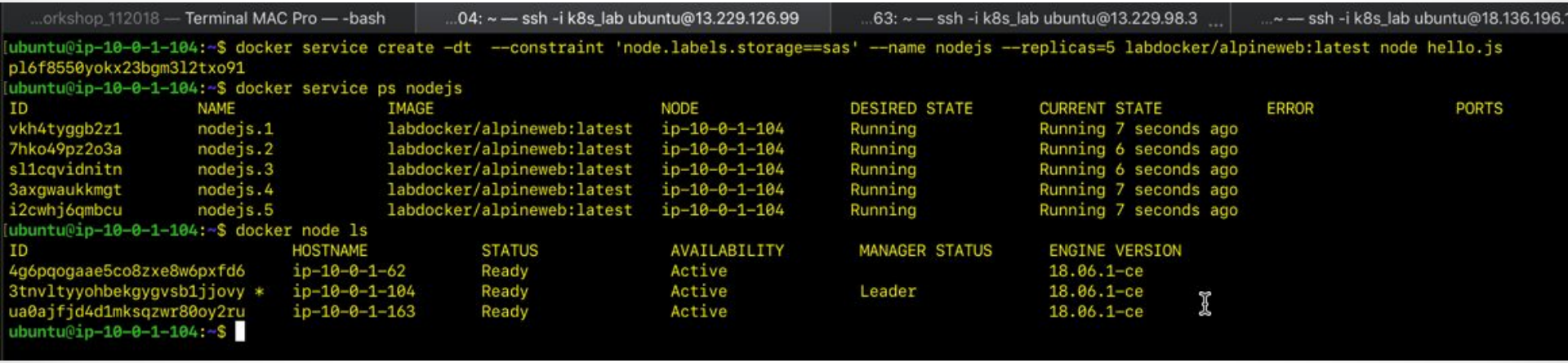
docker service create -dt --constraint 'node.labels.storage==sas' --name nodejs --replicas=5 labdocker/alpineweb:latest node hello.js

```
...orkshop_112018 — Terminal MAC Pro — -bash   ...04: ~ — ssh -i k8s_lab ubuntu@13.229.126.99   ...63: ~ — ssh -i k8s_lab ubuntu@13.229.98.3   ...~ — ssh -i k8s_lab ubuntu@18.136
ubuntu@ip-10-0-1-104:~$ docker service create -dt  --constraint 'node.labels.storage==sas' --name nodejs --replicas=5 labdocker/alpineweb:latest node hello.js
pl6f8550yokx23bgm3l2txo91
ubuntu@ip-10-0-1-104:~$ docker service ps nodejs
ID                  NAME                IMAGE                          NODE                DESIRED STATE       CURRENT STATE           ERROR               PORTS
vkh4tyggb2z1        nodejs.1            labdocker/alpineweb:latest     ip-10-0-1-104       Running             Running 7 seconds ago
7hko49pz2o3a        nodejs.2            labdocker/alpineweb:latest     ip-10-0-1-104       Running             Running 6 seconds ago
sl1cqvidnitn        nodejs.3            labdocker/alpineweb:latest     ip-10-0-1-104       Running             Running 6 seconds ago
3axgwaukkmgt        nodejs.4            labdocker/alpineweb:latest     ip-10-0-1-104       Running             Running 7 seconds ago
i2cwhj6qmbcu        nodejs.5            labdocker/alpineweb:latest     ip-10-0-1-104       Running             Running 7 seconds ago
ubuntu@ip-10-0-1-104:~$
```

Workshop: custom label

- สร้าง docker service โดยระบุ constraint จาก custom label ที่สร้างขึ้น

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

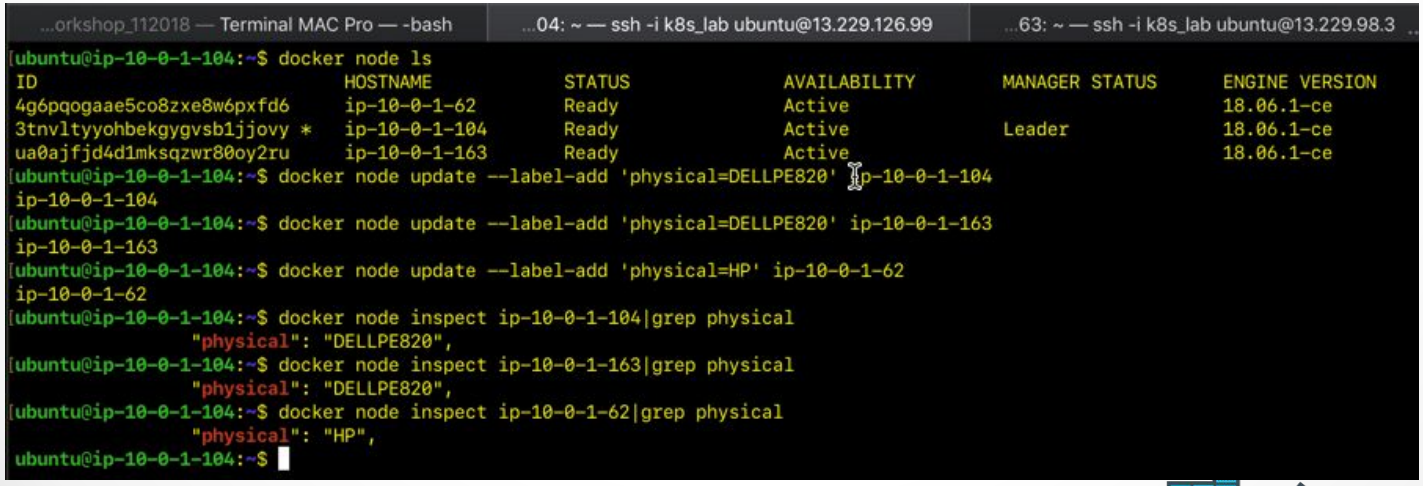


Orchestrator Assignment

- service placement preference (--placement-pref)

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

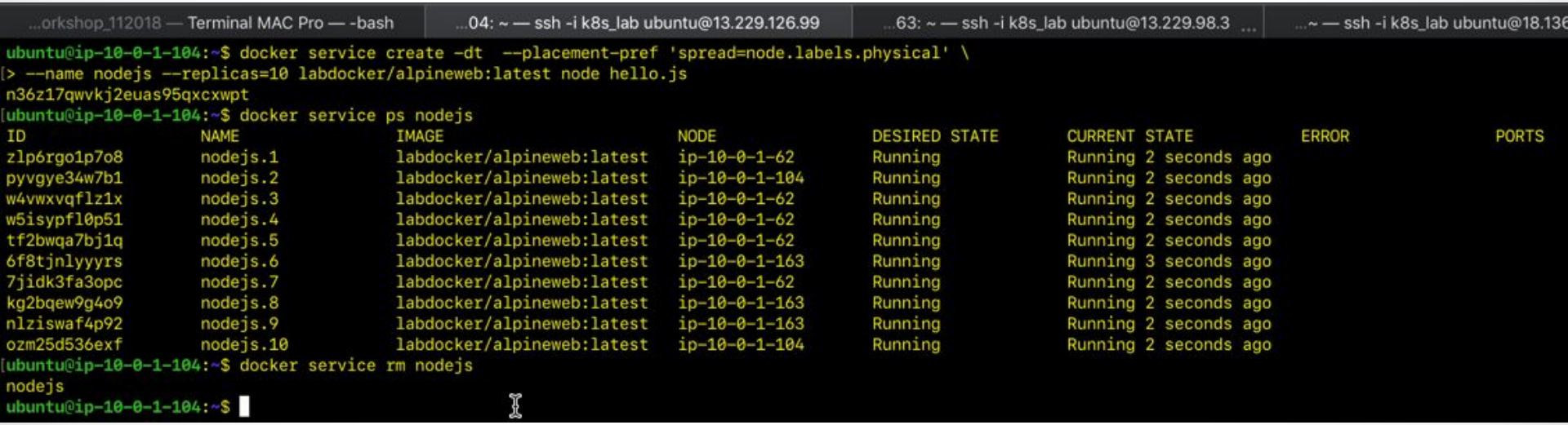
```
docker node inspect swarm-mng|grep physical
docker node inspect swarm-node1|grep physical
docker node inspect swarm-node2|grep physical
```



Orchestrator Assignment

- service placement preference

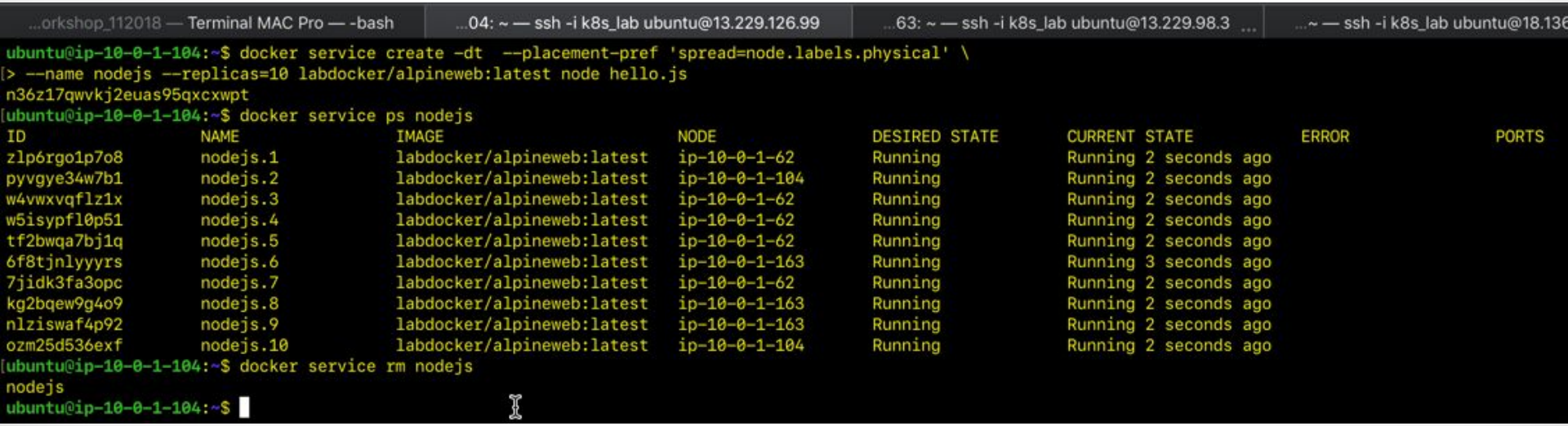
```
docker service create -dt  --placement-pref 'spread=node.labels.physical' \
--name nodejs --replicas=10 labdocker/alpineweb:latest node hello.js
```



Workshop: Orchestrator

- สร้าง docker service โดยระบุใน swarm กระจายภายใต้ label

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



Config and Secret

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

Config and Secret

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

```
echo “config value” | docker config create <config name> -
```

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

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

- Apply config to service

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

```
docker service <rm/ --config-rm> <config name>
```


Config and Secret

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

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

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

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

- Apply config to service

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

```
docker service <rm /update --secret-rm> source=<secret name>
```


Workshop: Config and Secret

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

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

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

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

```
...orkshop_112018 — Terminal MAC Pro — -bash   ...04: ~ — ssh -i k8s_lab ubuntu@13.229.126.99   ...63: ~ — ssh -i k8s_lab ubuntu@13.229.98.3 ...
[ubuntu@ip-10-0-1-104:~$ docker config create nginx.conf ~/docker_workshop_112018/Workshop-2-4-Swarm/nginx.conf
bni17w6tmbks39l1ri3ryvvxz
[ubuntu@ip-10-0-1-104:~$ docker config ls
ID                          NAME                CREATED             UPDATED
bni17w6tmbks39l1ri3ryvvxz   nginx.conf          3 seconds ago       3 seconds ago
ubuntu@ip-10-0-1-104:~$ docker secret create labdocker.com.crt ~/docker_workshop_112018/Workshop-2-4-Swarm/labdocker.com.crt
pt2dhhjiqroqd7gb6ig9bv62n
ubuntu@ip-10-0-1-104:~$ docker secret create labdocker.com.key ~/docker_workshop_112018/Workshop-2-4-Swarm/labdocker.com.key
nu11fusm2zbl815112s8pe6nb
[ubuntu@ip-10-0-1-104:~$ docker secret ls
ID                          NAME                DRIVER              CREATED                 UPDATED
pt2dhhjiqroqd7gb6ig9bv62n   labdocker.com.crt                       Less than a second ago   Less than a second ago
nu11fusm2zbl815112s8pe6nb   labdocker.com.key                       Less than a second ago   Less than a second ago
ubuntu@ip-10-0-1-104:~$ docker service create -dt \
```

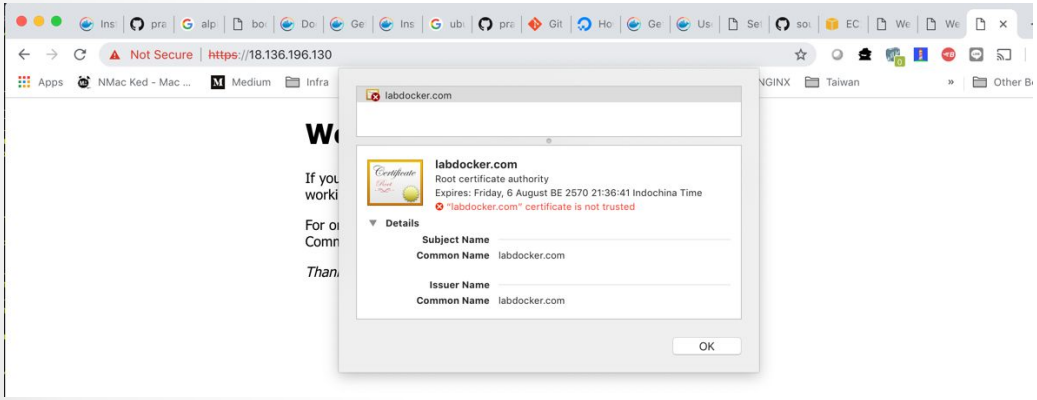

Workshop: Config and Secret

- nginx.conf

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


Workshop: Config and Secret

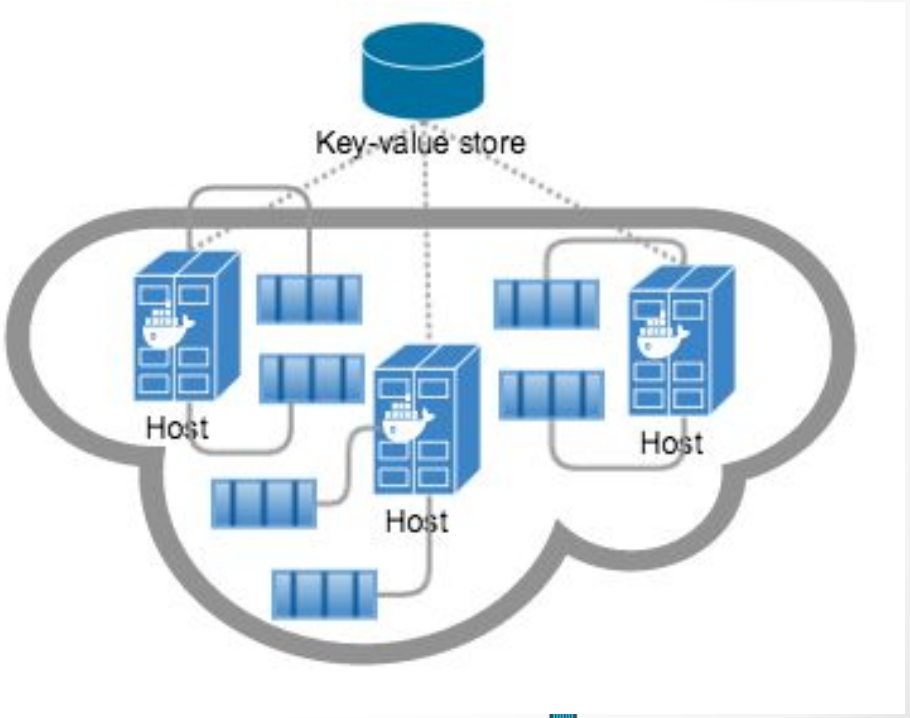
```
docker service -dt create \
--name nginx \
--secret labdocker.com.key \
--secret labdocker.com.crt \
--config source=nginx.conf,target=/etc/nginx/nginx.conf \
-p 443:443 labdocker/nginx:http2
```



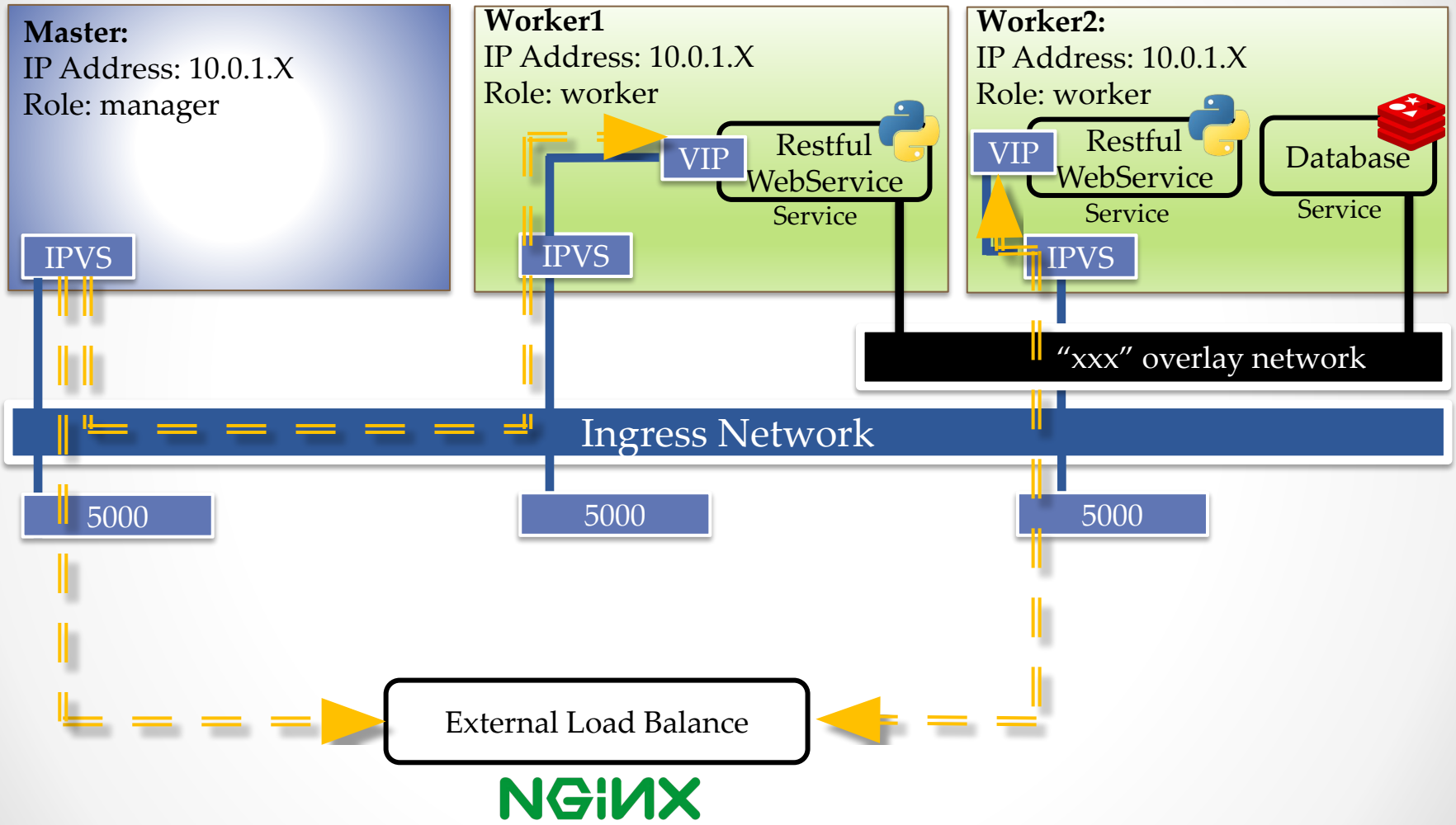

```
praparns-MacBook-Pro:docker_workshop_112018 praparn$ curl -k https://18.136.196.130
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
praparns-MacBook-Pro:docker_workshop_112018 praparn$
```


Overlay and Ingress Network

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

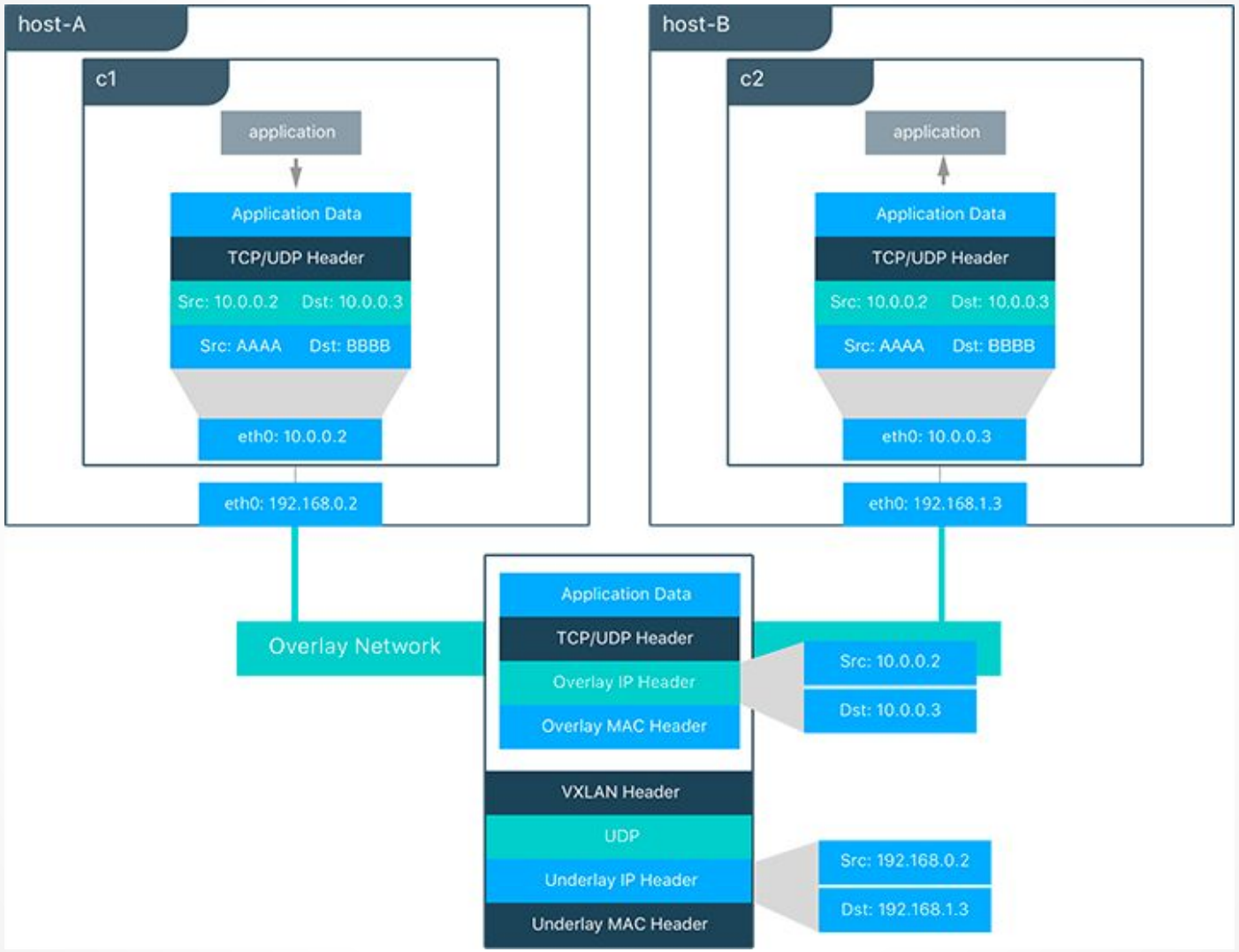


Overlay and Ingress Network



docker

Overlay and Ingress Network



docker

Workshop: Overlay and Ingress

- สร้าง overlay network บน swarm manager

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

Workshop: Overlay and Ingress

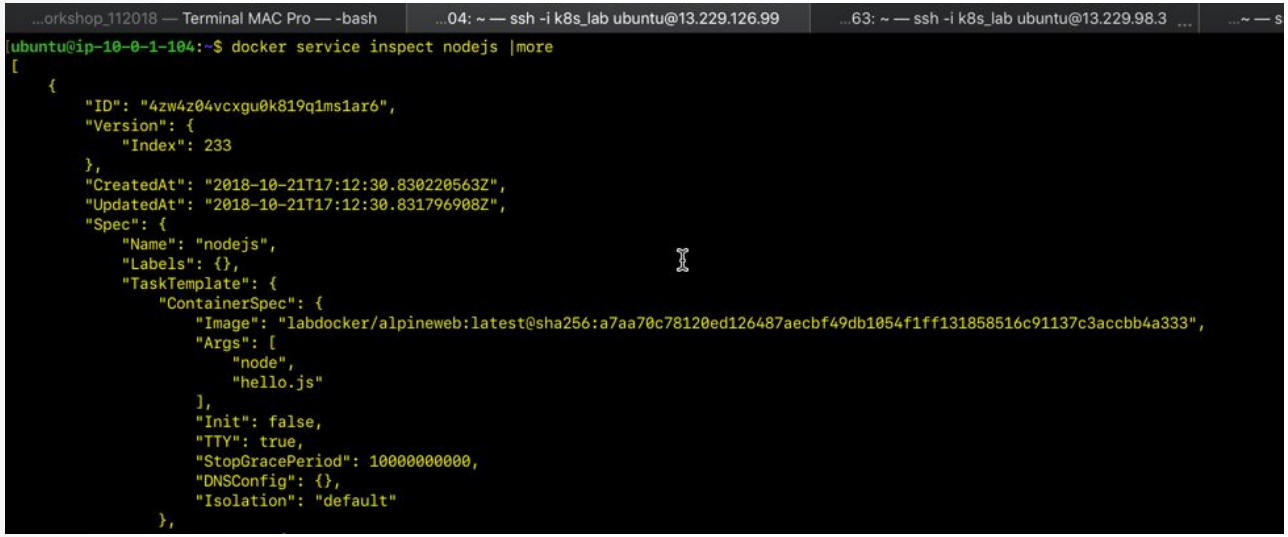
- สร้าง new service เพื่อใช้งาน swarm network

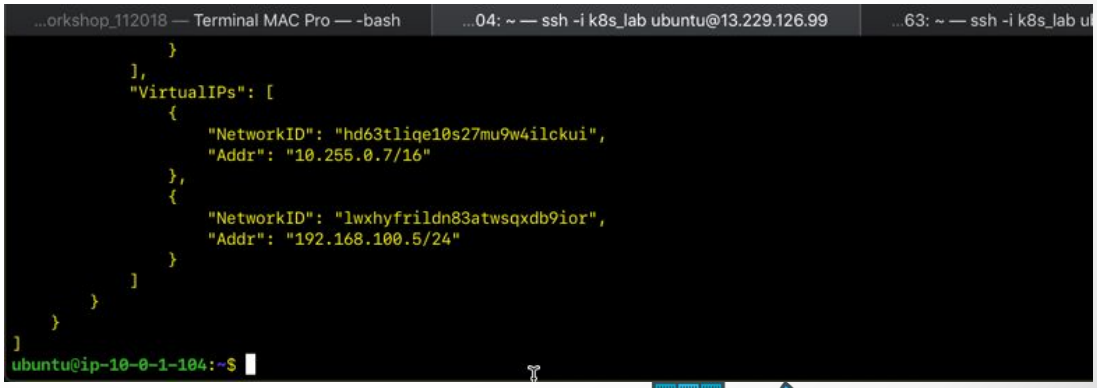
```
docker service -dt create --name nodejs --replicas=2
--network-add swarmnet -p 3000:3000
labdocker/alpineweb:latest node hello.js
```

Workshop: Overlay and Ingress

- ตรวจสอบ IP Address ของ service และทดสอบ Ping ข้าม node

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





Workshop: Overlay and Ingress

- ตรวจสอบ IP Address ของ service และทดสอบ Ping ข้าม node

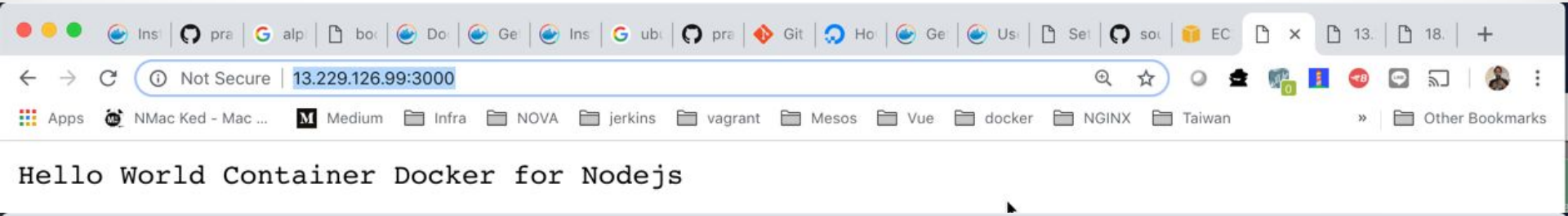
```
[ubuntu@ip-10-0-1-104:~$ docker ps
CONTAINER ID        IMAGE                          COMMAND                  CREATED             STATUS              PORTS               NAMES
bfc3f7a2e264        labdocker/alpineweb:latest     "node hello.js node …"   4 minutes ago       Up 4 minutes        3000/tcp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

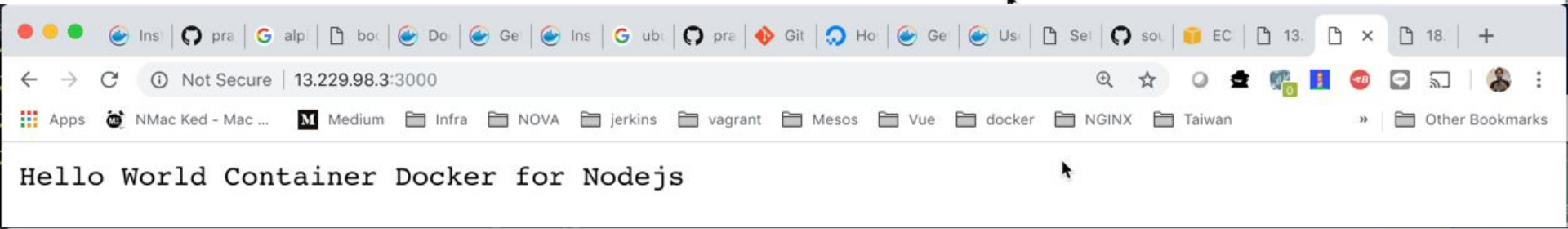
```
[ubuntu@ip-10-0-1-163:~$ docker ps
CONTAINER ID        IMAGE                          COMMAND                  CREATED             STATUS              PORTS               NAMES
cf75d3a7074b        labdocker/alpineweb:latest     "node hello.js node …"   7 minutes ago       Up 7 minutes        3000/tcp            nodejs.2.51tldbd8z00yaq761ac6ozpy0
[ubuntu@ip-10-0-1-163:~$ docker exec -it nodejs.2.51tldbd8z00yaq761ac6ozpy0 ping 192.168.100.5
PING 192.168.100.5 (192.168.100.5): 56 data bytes
64 bytes from 192.168.100.5: seq=0 ttl=64 time=0.100 ms
64 bytes from 192.168.100.5: seq=1 ttl=64 time=0.062 ms
^C
--- 192.168.100.5 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 0.062/0.081/0.100 ms
[ubuntu@ip-10-0-1-163:~$ docker exec -it nodejs.2.51tldbd8z00yaq761ac6ozpy0 ping 192.168.100.6
PING 192.168.100.6 (192.168.100.6): 56 data bytes
64 bytes from 192.168.100.6: seq=0 ttl=64 time=0.244 ms
64 bytes from 192.168.100.6: seq=1 ttl=64 time=0.237 ms
^C
--- 192.168.100.6 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 0.237/0.240/0.244 ms
ubuntu@ip-10-0-1-163:~$
```

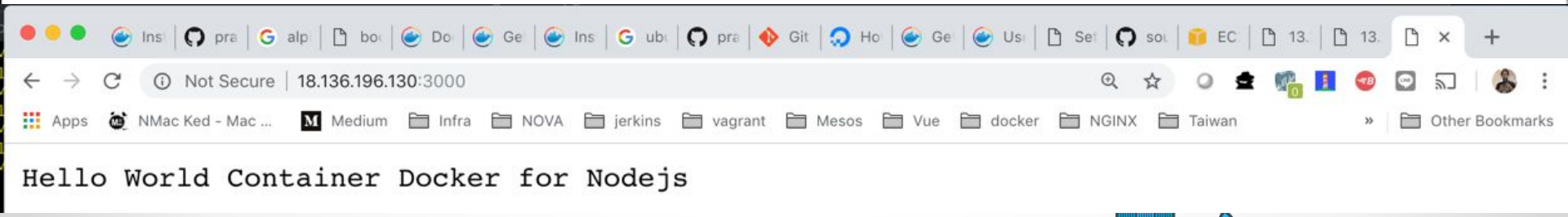

Workshop: Overlay and Ingress

- เรียกใช้บริการผ่าน http port ด้วย browser/curl

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



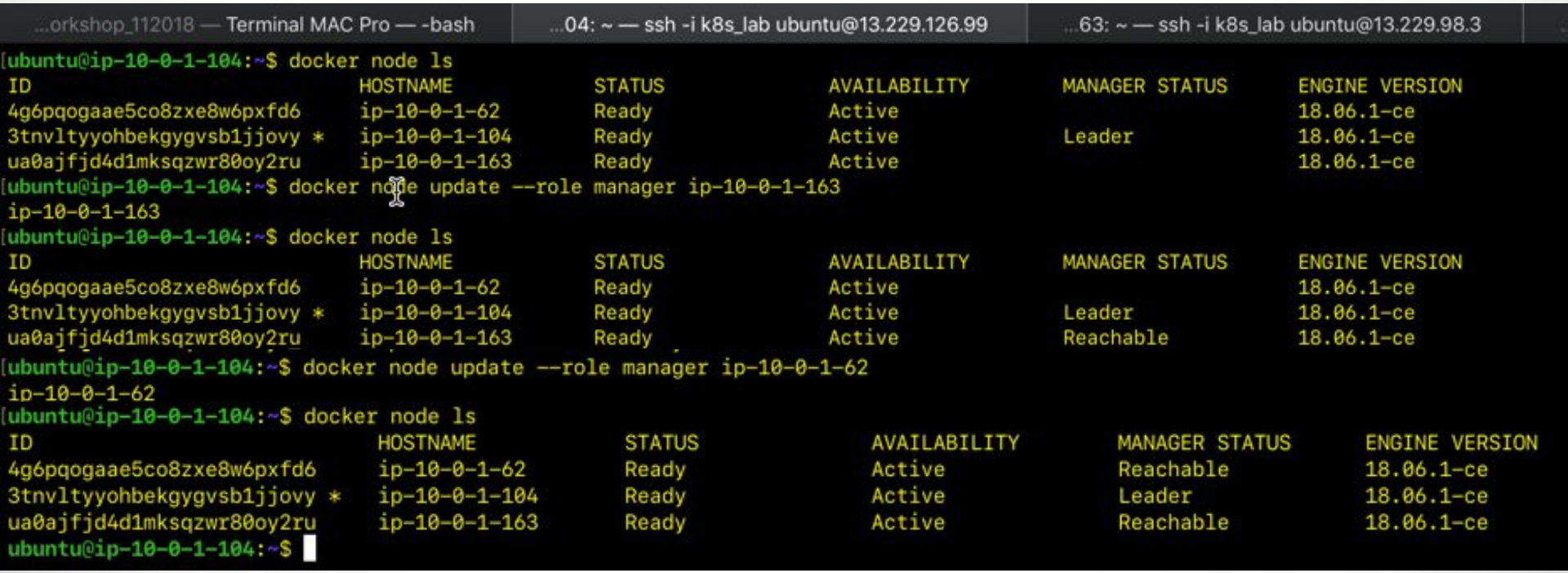




HA Manager Role

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

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



HA Manager Role

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

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

  - Other will take role for manager within 5 min.

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


HA Manager Role

- Remove
  - Update node to worker2 and set "Drain" to node

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

  - Leave swarm on node "labdocker"

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

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


HA Manager Role

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

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

```
...orkshop_112018 — Terminal MAC Pro — -bash | ...4: ~ — ssh -i k8s_lab ubuntu@13.229.224.202 | ...63: ~ — ssh -i k8s_lab ubuntu@13.229.98.3 | ...~ — ssh -i k8s_lab ubuntu@18.136.196
ubuntu@ip-10-0-1-62:~$ docker swarm leave
Node left the swarm.
ubuntu@ip-10-0-1-62:~$ docker swarm join --token SWMTKN-1-1qnpobqw0hilpc5i90sz803tlhyi1h4k84uok10y814bvhyqf2-3u9rqxfwhbhw52qwqeeiewybl 10.0.1.104:2377
This node joined a swarm as a manager.
ubuntu@ip-10-0-1-62:~$ docker node ls
ID                            HOSTNAME            STATUS              AVAILABILITY        MANAGER STATUS      ENGINE VERSION
mydkcrgwkurxyjjb1kf14odl7 *   ip-10-0-1-62        Ready               Active              Reachable           18.06.1-ce
3tnvltyyohbekgygvsb1jjovy     ip-10-0-1-104       Ready               Active              Reachable           18.06.1-ce
ua0ajfjd4d1mksqzwr80oy2ru     ip-10-0-1-163       Ready               Active              Leader              18.06.1-ce
ubuntu@ip-10-0-1-62:~$
```

  - ตรวจสอบค่า token เพื่อ join swarm

```
ubuntu@ip-10-0-1-62:~$ docker swarm join-token manager
To add a manager to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-1qnpobqw0hilpc5i90sz803tlhyi1h4k84uok10y814bvhyqf2-3u9rqxfwhbhw52qwqeeiewybl 10.0.1.62:2377

ubuntu@ip-10-0-1-62:~$
```


Workshop: HA Manager Role

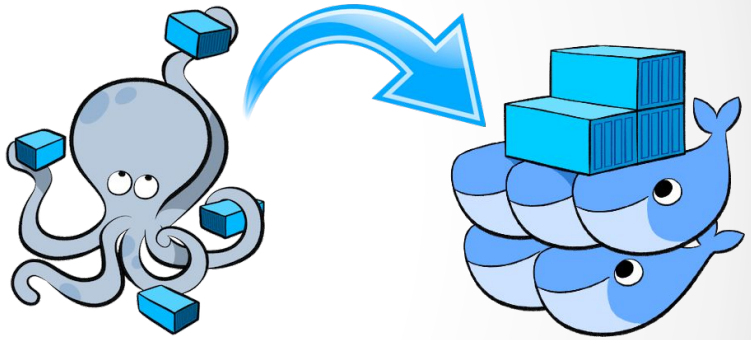
- ทำการ join new manager เข้าสู่ swarm cluster/update/drain

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

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

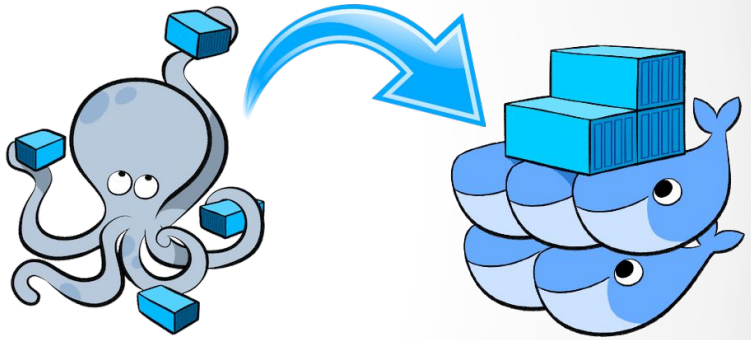

Docker Stack Deploy (Compose)

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



Docker Stack Deploy (Compose)

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



Docker Stack Deploy (Compose)

- Compose syntax: (docker stack deploy)

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

```
docker stack ls
```

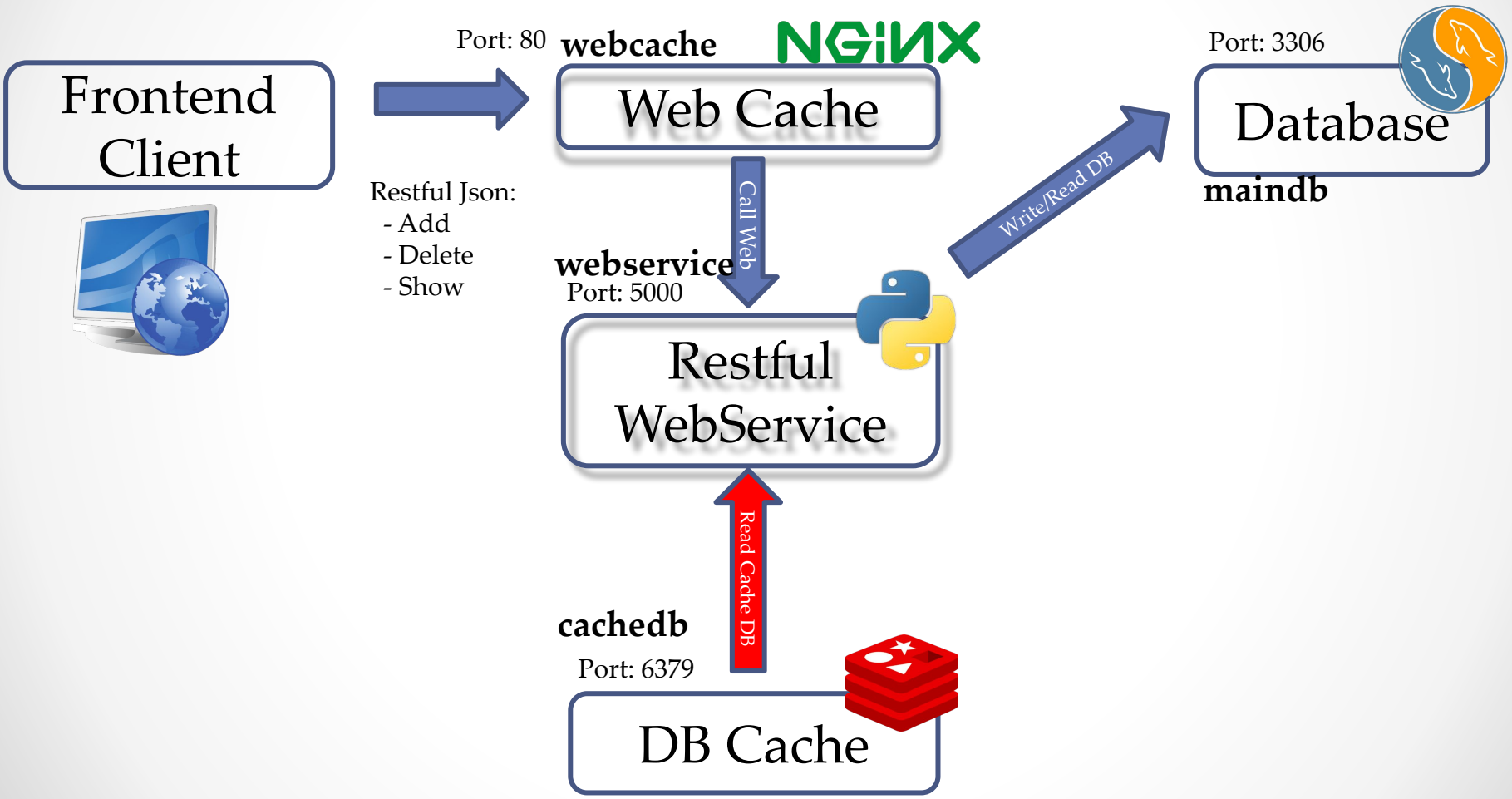
```
docker stack ps <stack name>
```

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

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

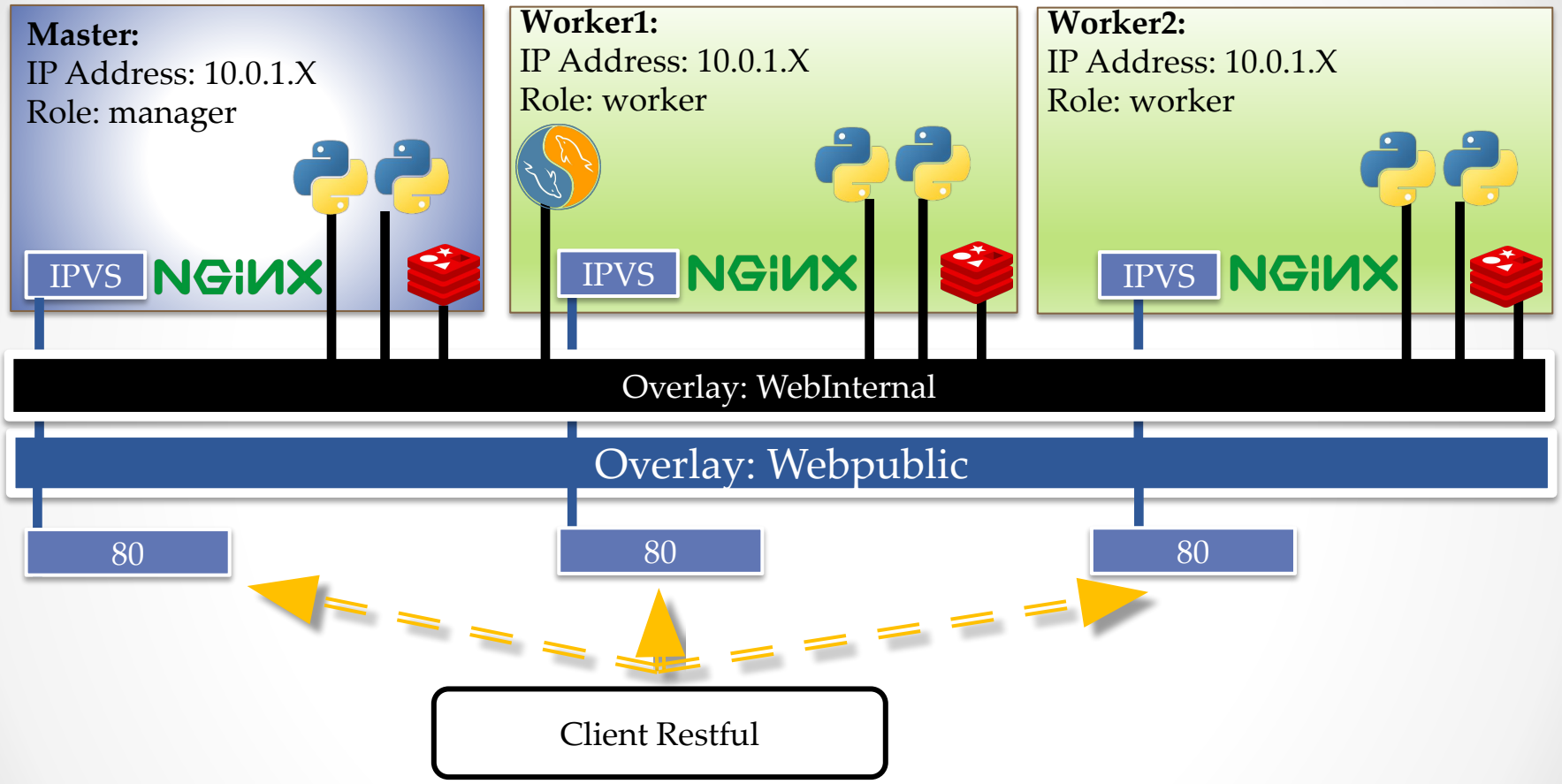

Workshop: Swarm Compose

- Optimize for WorkLoad



docker

Workshop: Swarm Compose



docker

Workshop: Swarm Compose

```yaml
version: '3.3'
services:
  webcache:
    image: labdocker/cluster:webcache
    container_name: nginx
    deploy:
      mode: global
      update_config:
        parallelism: 1
        delay: 10s
      restart_policy:
        condition: on-failure
        delay: 30s
        max_attempts: 5
        window: 60s
      endpoint_mode: vip
    depends_on:
      - webservice
      - cachedb
      - maindb
    networks:
      webpublic:
        aliases:
        - webcache
      webinternal:
        aliases:
        - webcache
    ports:
      - "80:80"

```

```yaml
  webservice:
    image: labdocker/cluster:webservice
    container_name: webservice
    deploy:
      mode: replicated
      replicas: 6
      update_config:
        parallelism: 2
        delay: 10s
      restart_policy:
        condition: on-failure
        delay: 30s
        max_attempts: 3
        window: 60s
      endpoint_mode: vip
    depends_on:
      - cachedb
      - maindb
    networks:
      webinternal:
        aliases:
        - webservice
    ports:
      - "5000:5000"
```


Workshop: Swarm Compose

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

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


Workshop: Swarm Compose

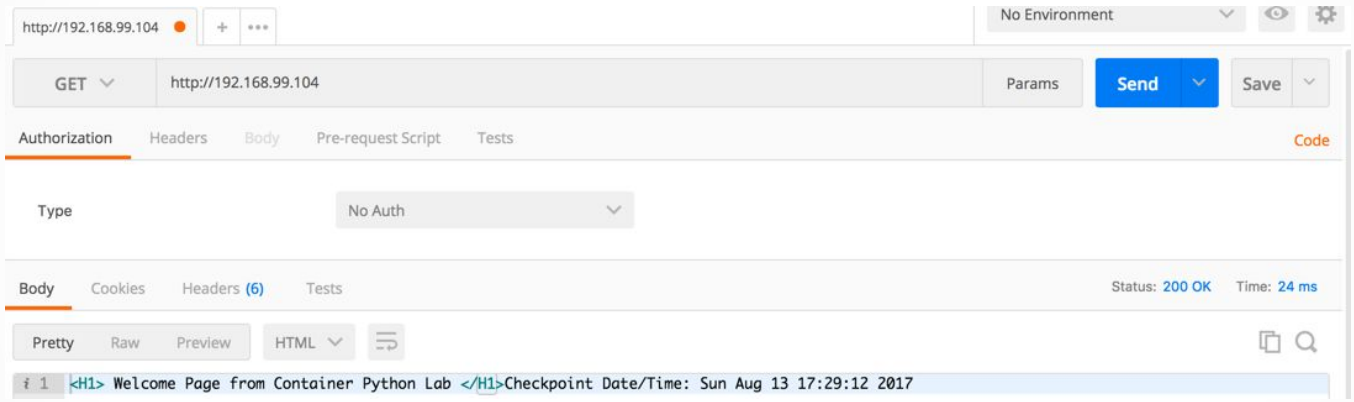
```
ubuntu@ip-10-0-1-104:~/docker_workshop_112018/Workshop-2-4-Swarm/python_restfulset$ docker stack deploy -c docker-compose_swarm.yml webservice
Ignoring deprecated options:

container_name: Setting the container name is not supported.

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


Workshop: Swarm Compose

```
docker@swarm-mng:/Share_DockerToolbox/python_restfulset$ docker stack ps webservice
ID                  NAME                                    IMAGE                          NODE                DESIRED STATE       CURRENT STATE
       ERROR                       PORTS
mcj0ux4x25uv        webservice_webcache.wwzx60rvl5res39v0lxir2gfh   labdocker/cluster:webcache     swarm-node2         Running             Running 46 seconds ag
o
ol223n2c2htp        webservice_webcache.0k95omm28xfef22thummpz3uf   labdocker/cluster:webcache     swarm-mng           Running             Running 49 seconds ag
o
fj0qu565u8mt        webservice_webcache.q814dhi3bzl4sze4zmjpiy3qa   labdocker/cluster:webcache     swarm-node1         Running             Running 50 seconds ag
o
kw8oprgn8b3v        webservice_webcache.wwzx60rvl5res39v0lxir2gfh   labdocker/cluster:webcache     swarm-node2         Shutdown            Failed about a minute
 ago    "task: non-zero exit (1)"
a4foun5hzks3        webservice_webcache.0k95omm28xfef22thummpz3uf   labdocker/cluster:webcache     swarm-mng           Shutdown            Failed about a minute
 ago    "task: non-zero exit (1)"
horm2noczrvt        webservice_webcache.q814dhi3bzl4sze4zmjpiy3qa   labdocker/cluster:webcache     swarm-node1         Shutdown            Failed about a minute
 ago    "task: non-zero exit (1)"
vhutisinb7np        webservice_webcache.wwzx60rvl5res39v0lxir2gfh   labdocker/cluster:webcache     swarm-node2         Shutdown            Failed about a minute
 ago    "task: non-zero exit (1)"
ktjzx8thiq6k        webservice_webcache.q814dhi3bzl4sze4zmjpiy3qa   labdocker/cluster:webcache     swarm-node1         Shutdown            Failed about a minute
 ago    "task: non-zero exit (1)"
7siaoo4hnvmh        webservice_webcache.0k95omm28xfef22thummpz3uf   labdocker/cluster:webcache     swarm-mng           Shutdown            Failed about a minute
 ago    "task: non-zero exit (1)"
axdh6t2ua971        webservice_cachedb.wwzx60rvl5res39v0lxir2gfh    labdocker/redis:latest         swarm-node2         Running             Running about a minut
e ago
t8jc8aopm6wu        webservice_cachedb.q814dhi3bzl4sze4zmjpiy3qa    labdocker/redis:latest         swarm-node1         Running             Running about a minut
e ago
32d2fao3cjcu        webservice_cachedb.0k95omm28xfef22thummpz3uf    labdocker/redis:latest         swarm-mng           Running             Running about a minut
e ago
mtbhxlkuwjg7        webservice_webservice.1                 labdocker/cluster:webservice   swarm-node1         Running             Running 39 seconds ag
o
nw2jky8drn42         \_ webservice_webservice.1             labdocker/cluster:webservice   swarm-mng           Shutdown            Failed about a minute
 ago    "task: non-zero exit (1)"
ivhofu5u9kxe        webservice_maindb.1                     labdocker/mysql:latest         swarm-node2         Running             Running about a minut
```

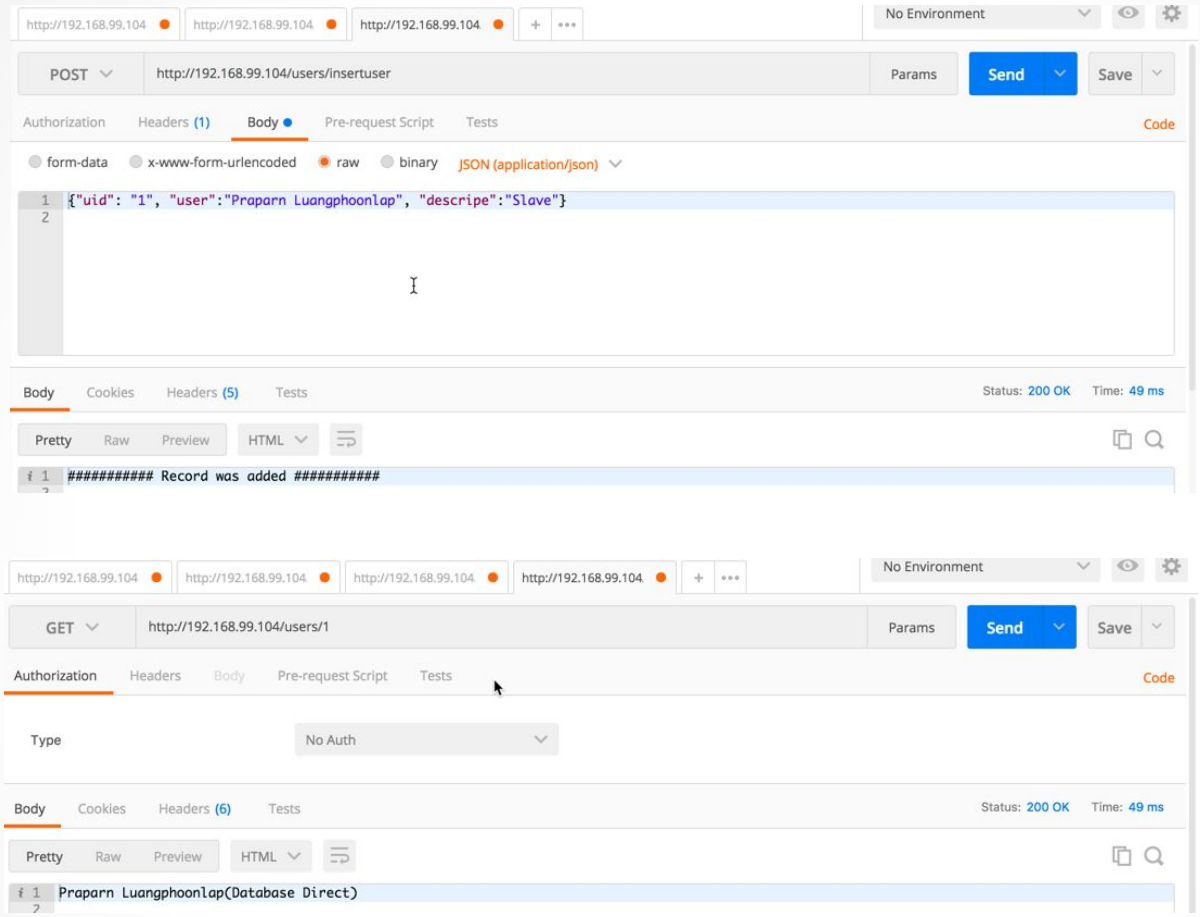

Workshop: Swarm Compose

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

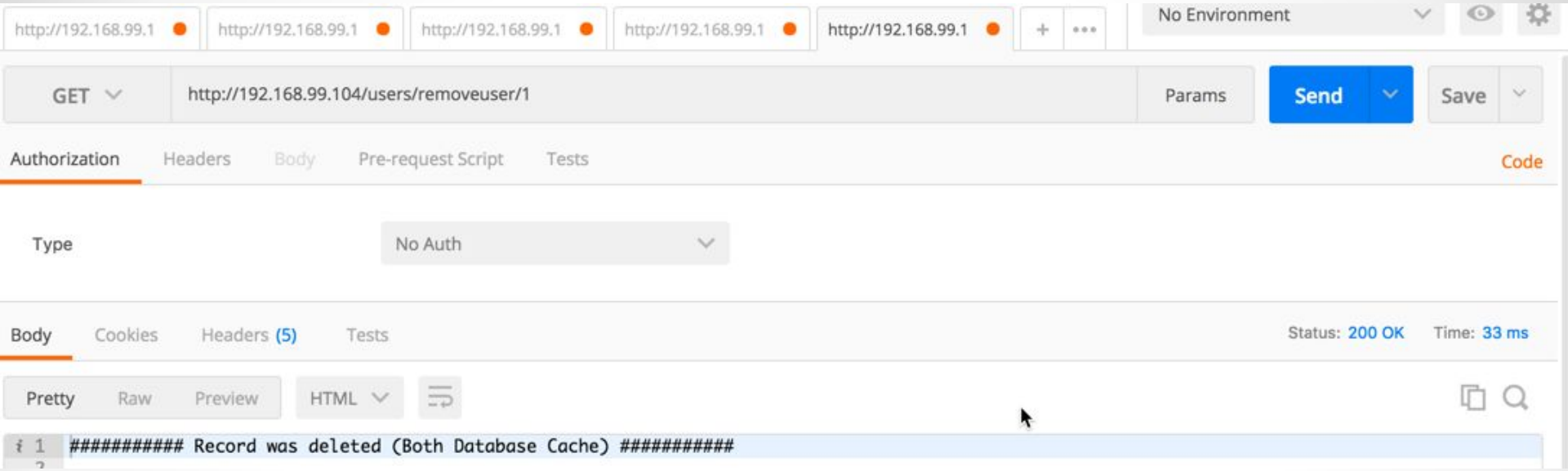


docker

Workshop: Swarm Compose



Workshop: Swarm Compose



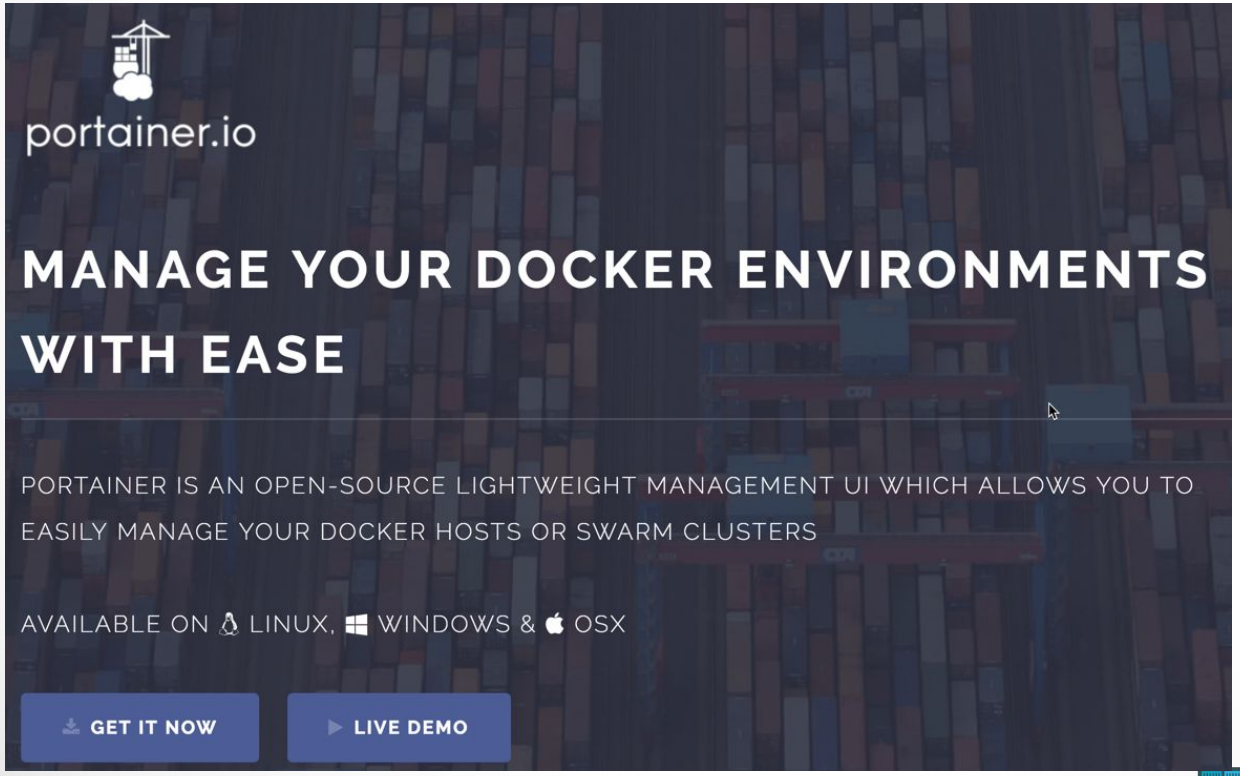
docker

Portainer for Docker

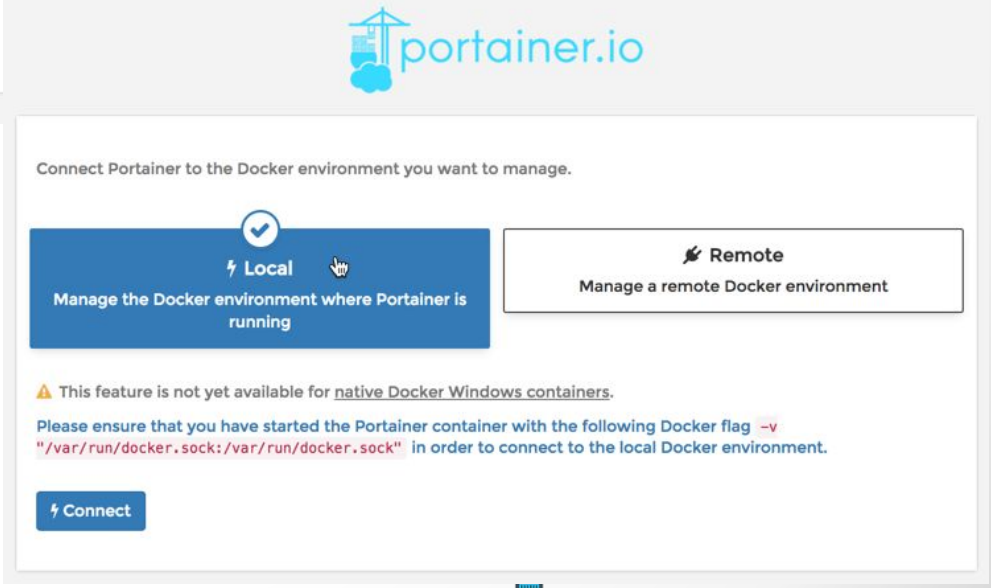
• • •

portainer for Docker

- portainer.io เป็น open source management ที่สามารถใช้บริหารจัดการ docker-machine ผ่านหน้า GUI web ได้อย่างมีประสิทธิภาพ แบบ On premise
- การใช้งาน portainer จะใช้การบริหารจัดการผ่านหน้าเว็บ (x.x.x.x:9443)

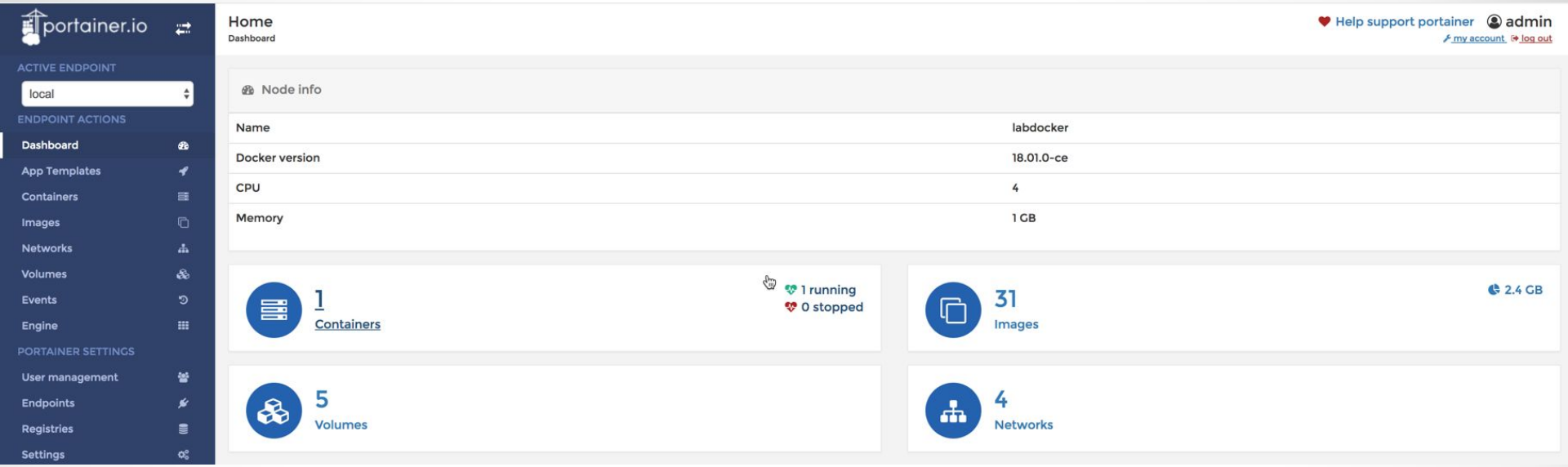


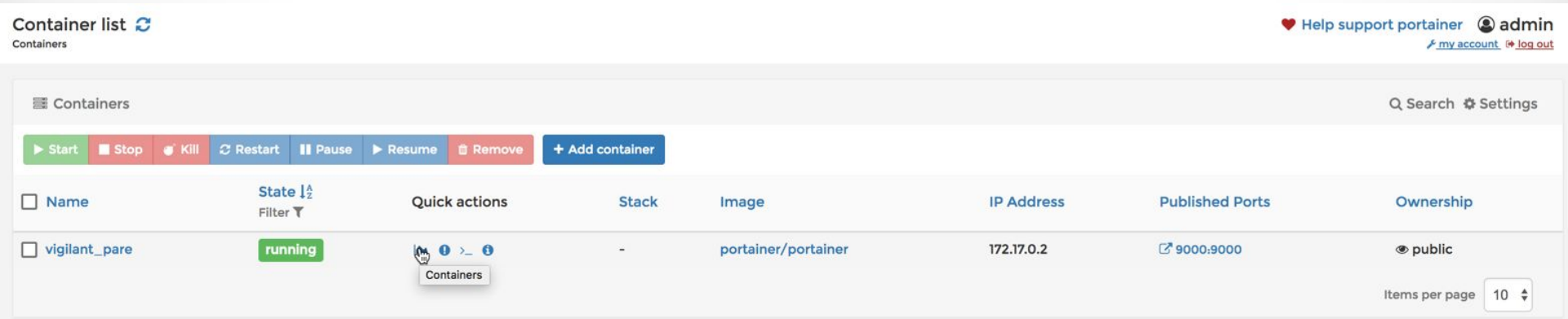
portainer for Docker



docker

portainer for Docker

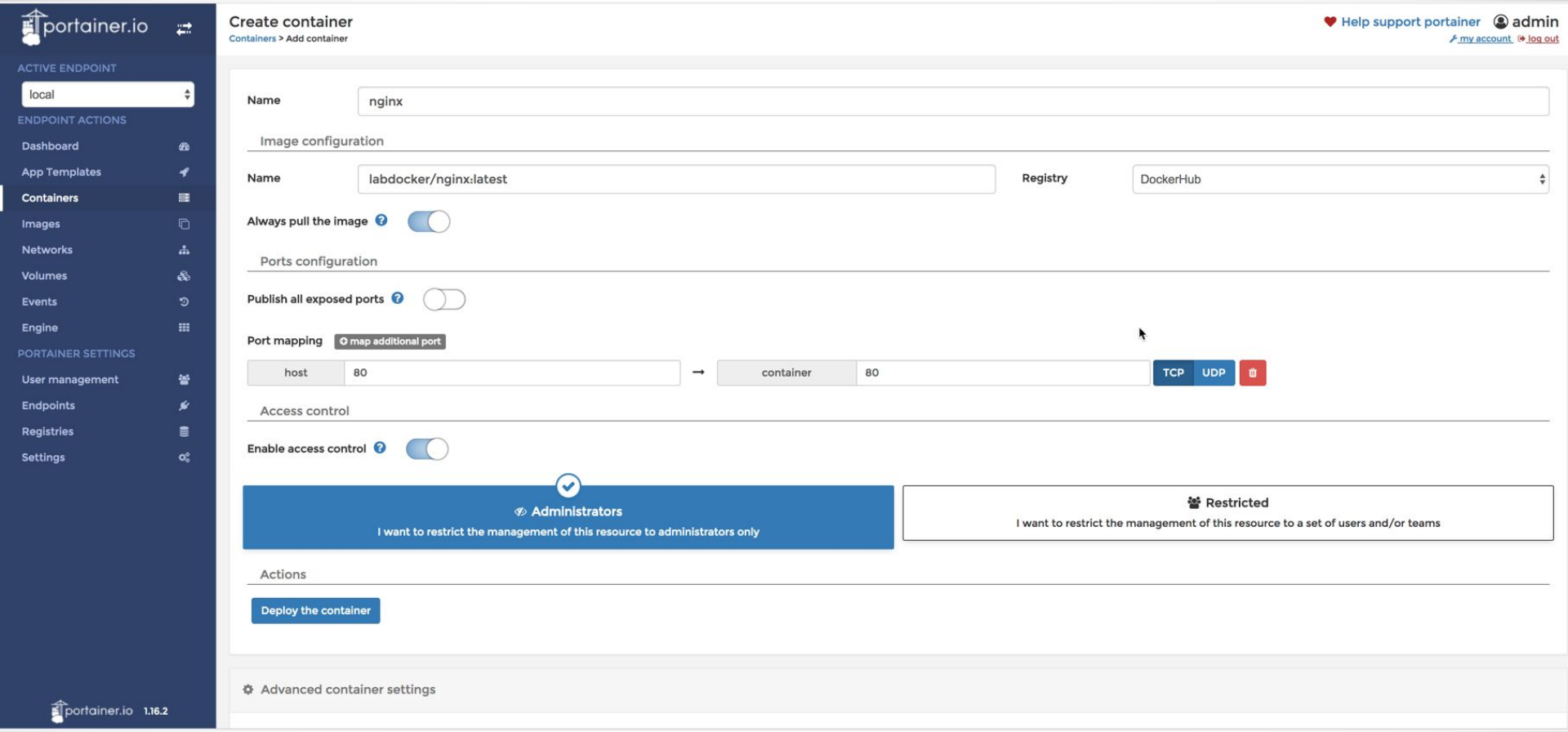




Docker: The Next-Gen of Virtualization

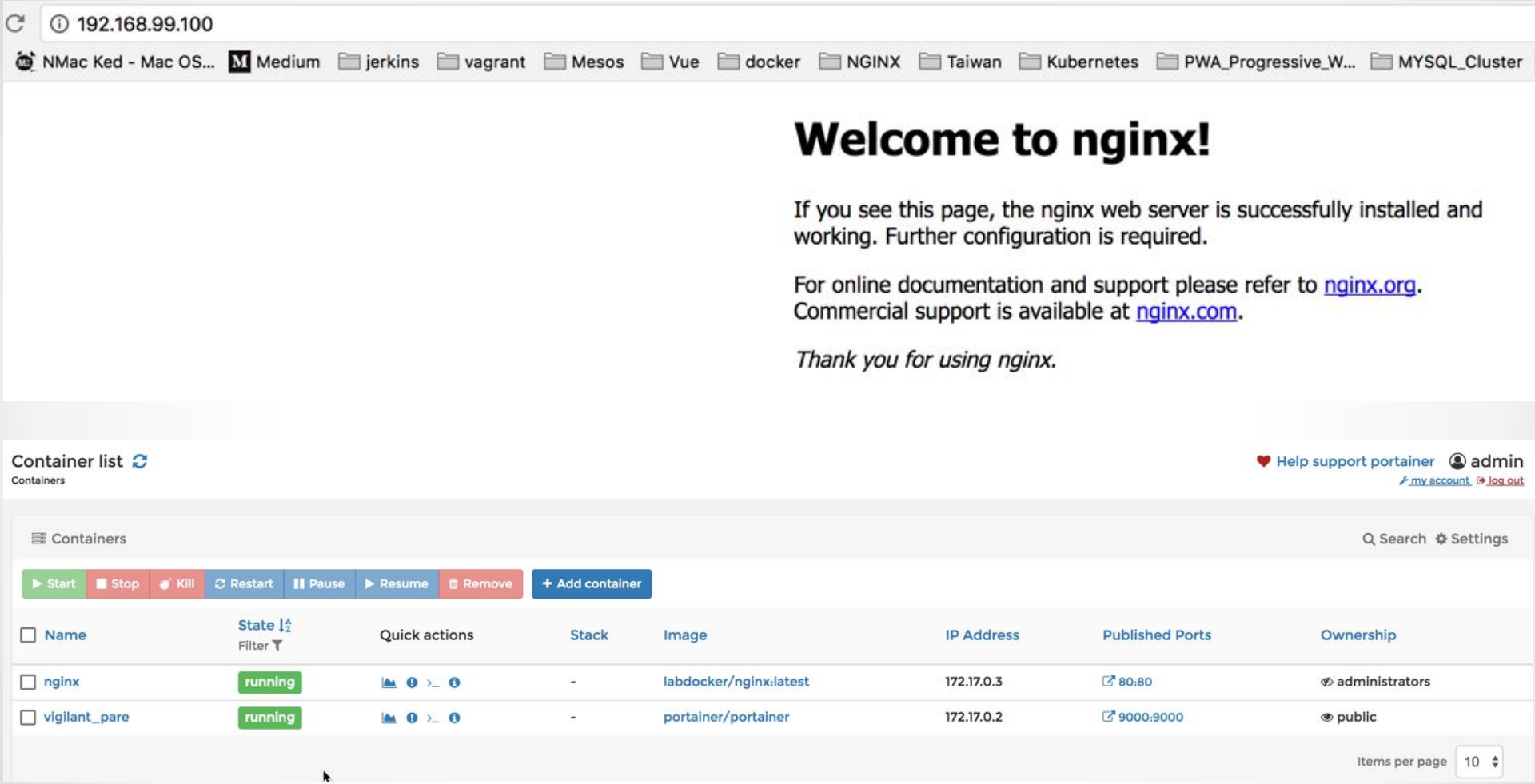
docker

portainer for Docker

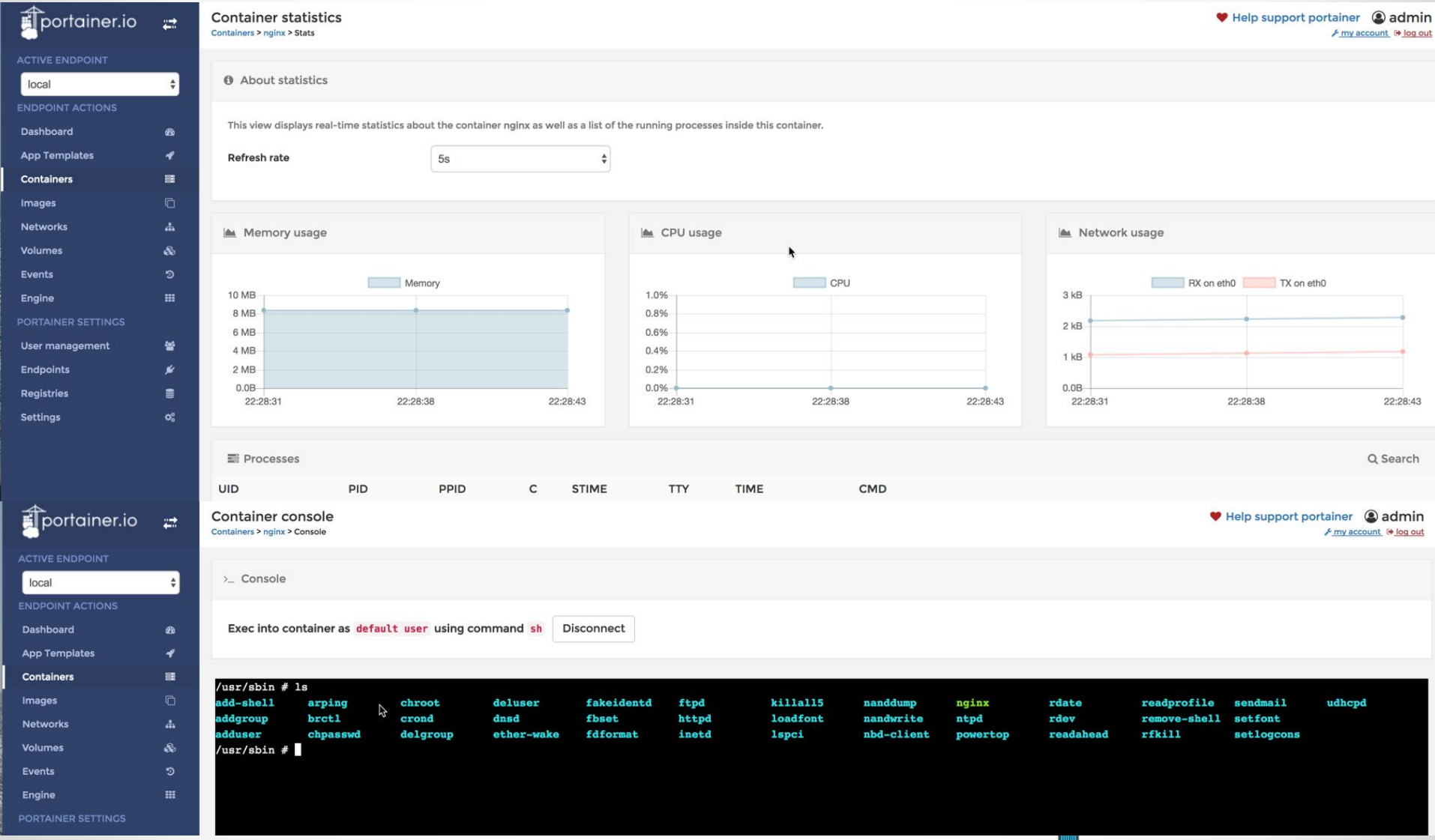


docker

portainer for Docker



portainer for Docker



Docker: The Next-Gen of Virtualization

docker

Recapture for Part 2

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

By Praparn Luengphoonlap
Email: eva10409@gmail.com

docker