# Docker & Kubernetes

Ghulam Mustafa Raza,

Expert Cloud Native Engineer

# Big Data Tech. Lab in SK telecom

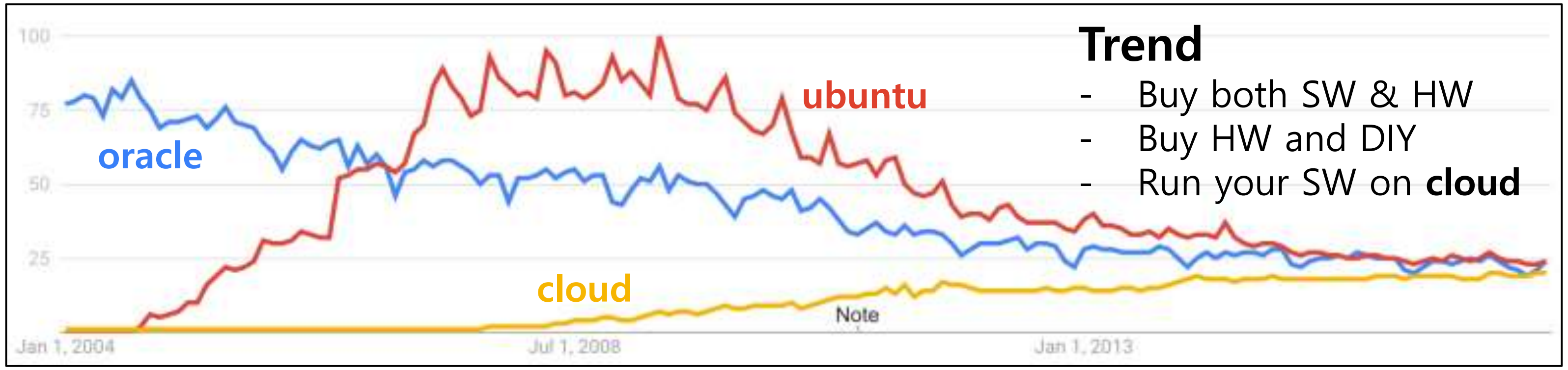
- Discovery Group
- Predictive Maintenance Group
- Manufacturing Solution Group
  - Groups making own solutions

- **Technology and Architecture Leading Group**
  - Big data processing engine
  - Advanced analytics algorithms
  - **Systematize service deployment and service operation on cluster**
    - **Docker**
    - **Kubernetes**

# Prepare for an era of **cloud** with **Docker** and **Kubernetes**

* technology trend in USA (2004-2017)

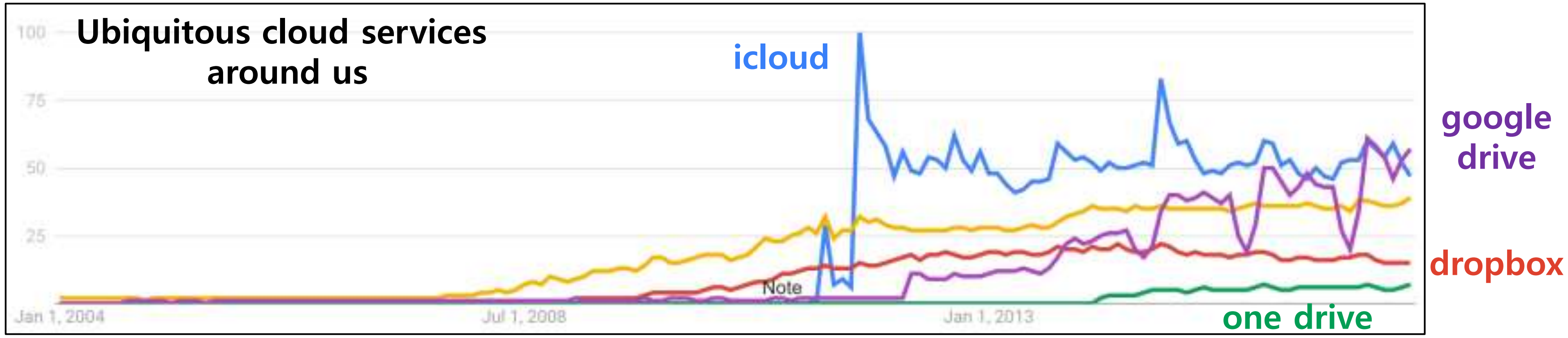
**Major technologies**



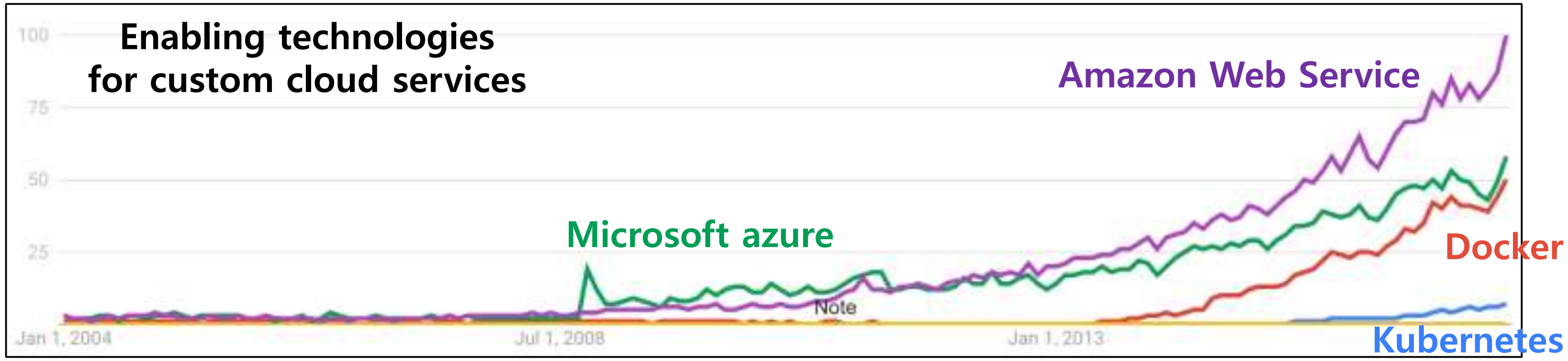
**Trend**

- Buy both SW & HW
- Buy HW and DIY
- Run your SW on **cloud**

**Cloud services** for users

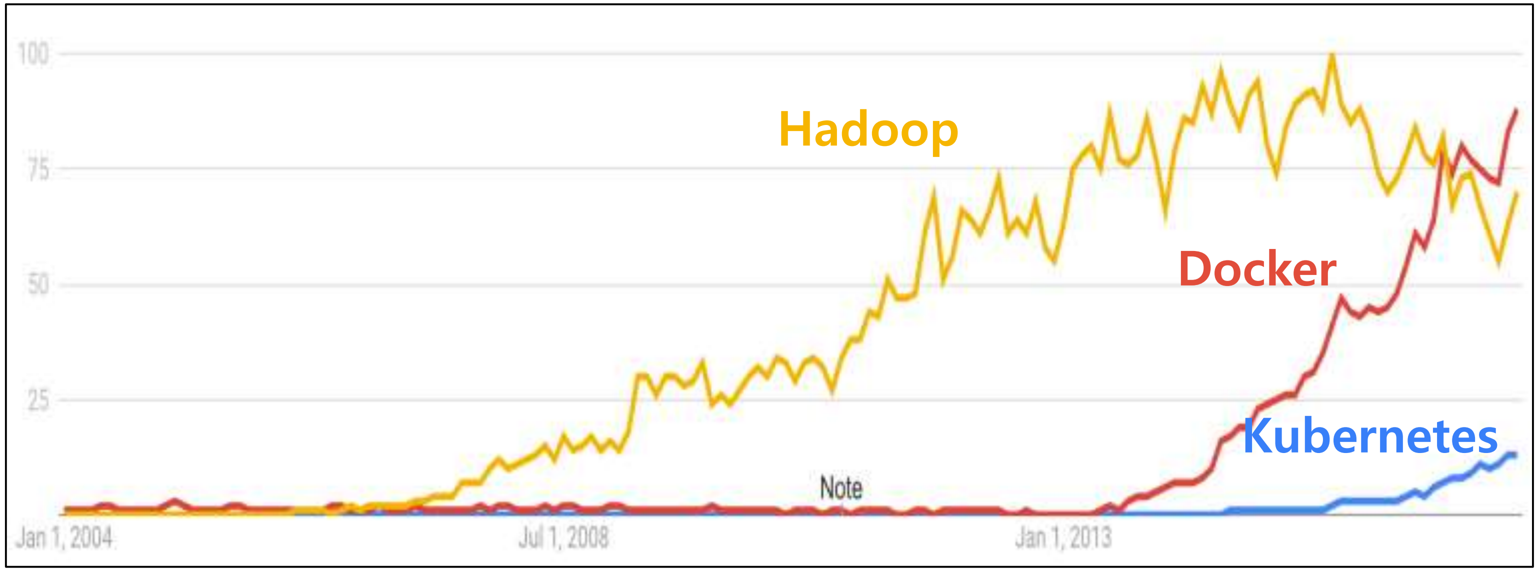

**Ubiquitous cloud services around us**

**Cloud technologies** for service providers

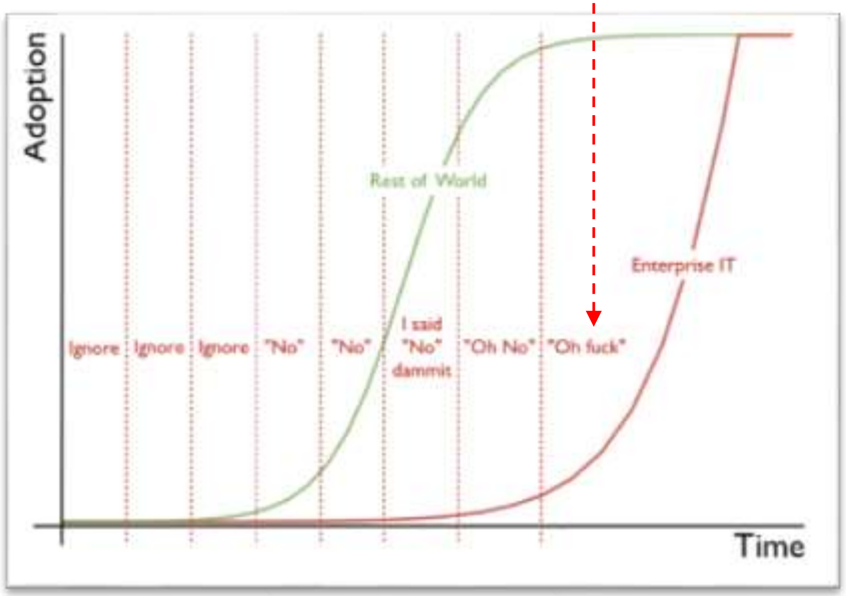

**Enabling technologies for custom cloud services**

# Overview & Conclusion

- **Docker** to build ***portable*** software
  - Build your software upon **Docker**
  - Then distribute it anywhere (even on MS Azure and Amazon Web Service)
- **Kubernetes** to orchestrate multiple **Docker** instances
- Start using **Docker** and **Kubernetes** before too late!
  - Google has been using container technologies more than 10 years



**Popularity of Docker and Kubernetes**



**The Enterprise IT Adoption Cycle**

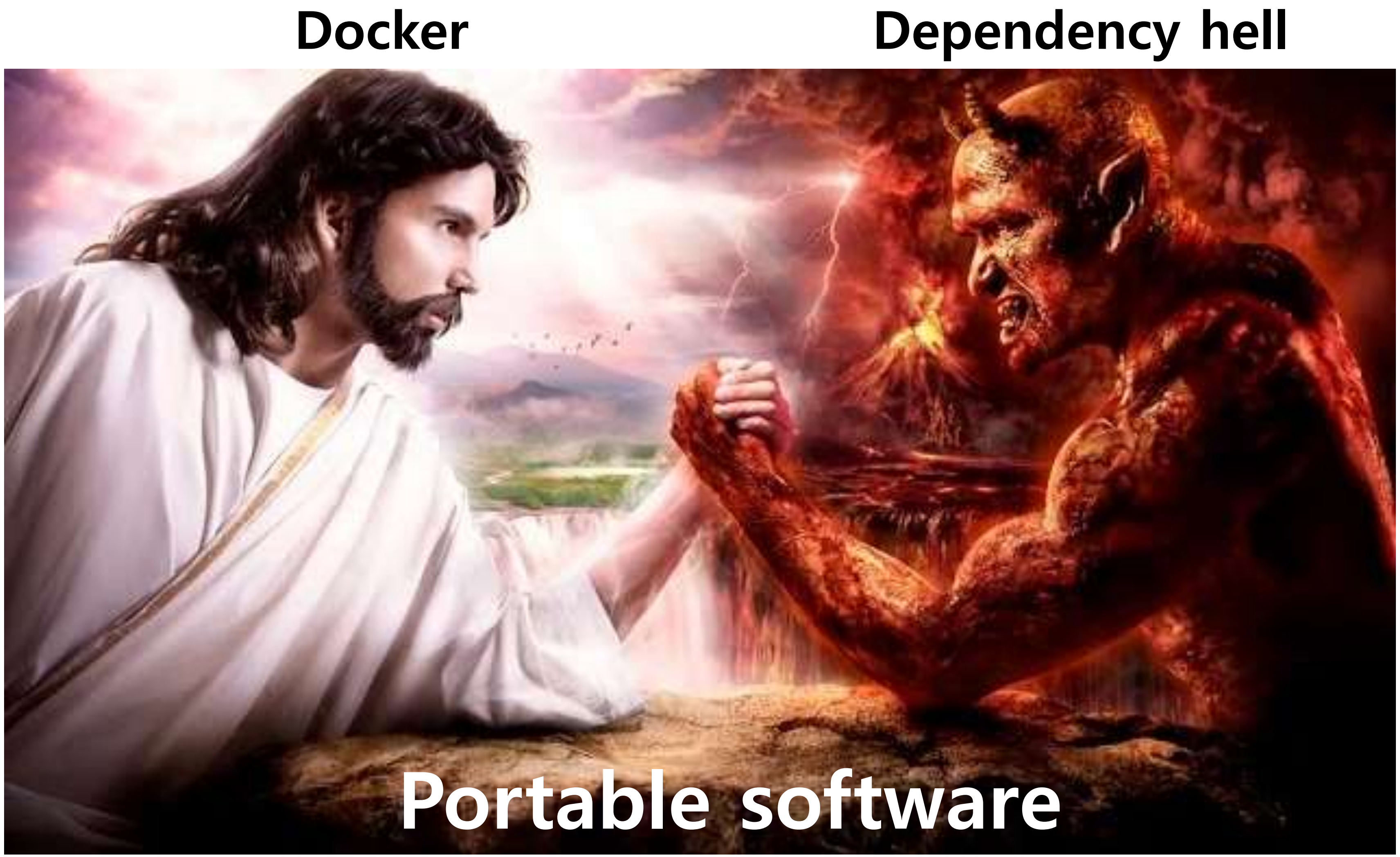
# Docker

Motivation

Enabling technologies for Docker

How to use Docker

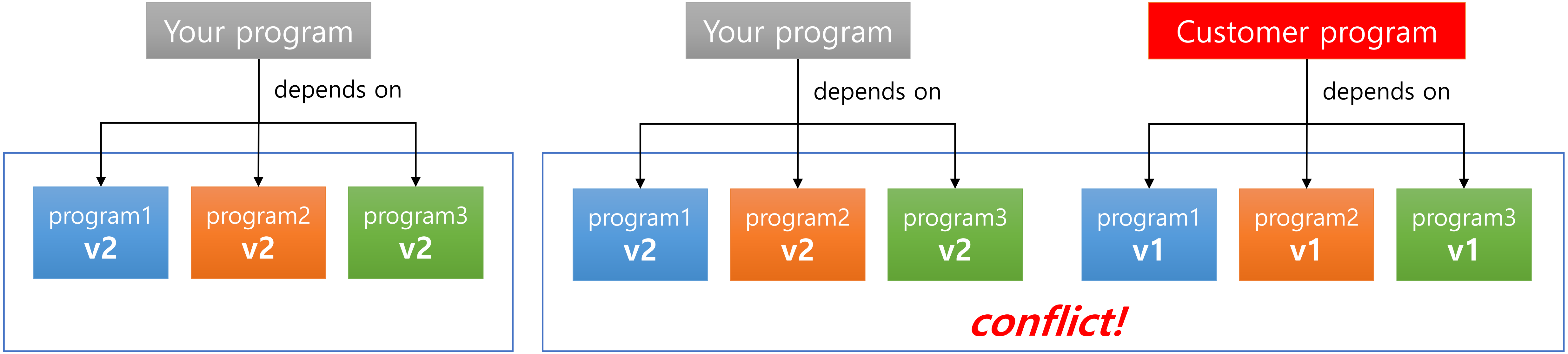
## **Docker** came to save us from the **dependency hell**



# Dependency hell

**Development**
environment

**Production**
environment



Your program

depends on

program1 **v2**

program2 **v2**

program3 **v2**

Your program

depends on

program1 **v2**

program2 **v2**

program3 **v2**

Customer program

depends on

program1 **v1**

program2 **v1**

program3 **v1**

***conflict!***

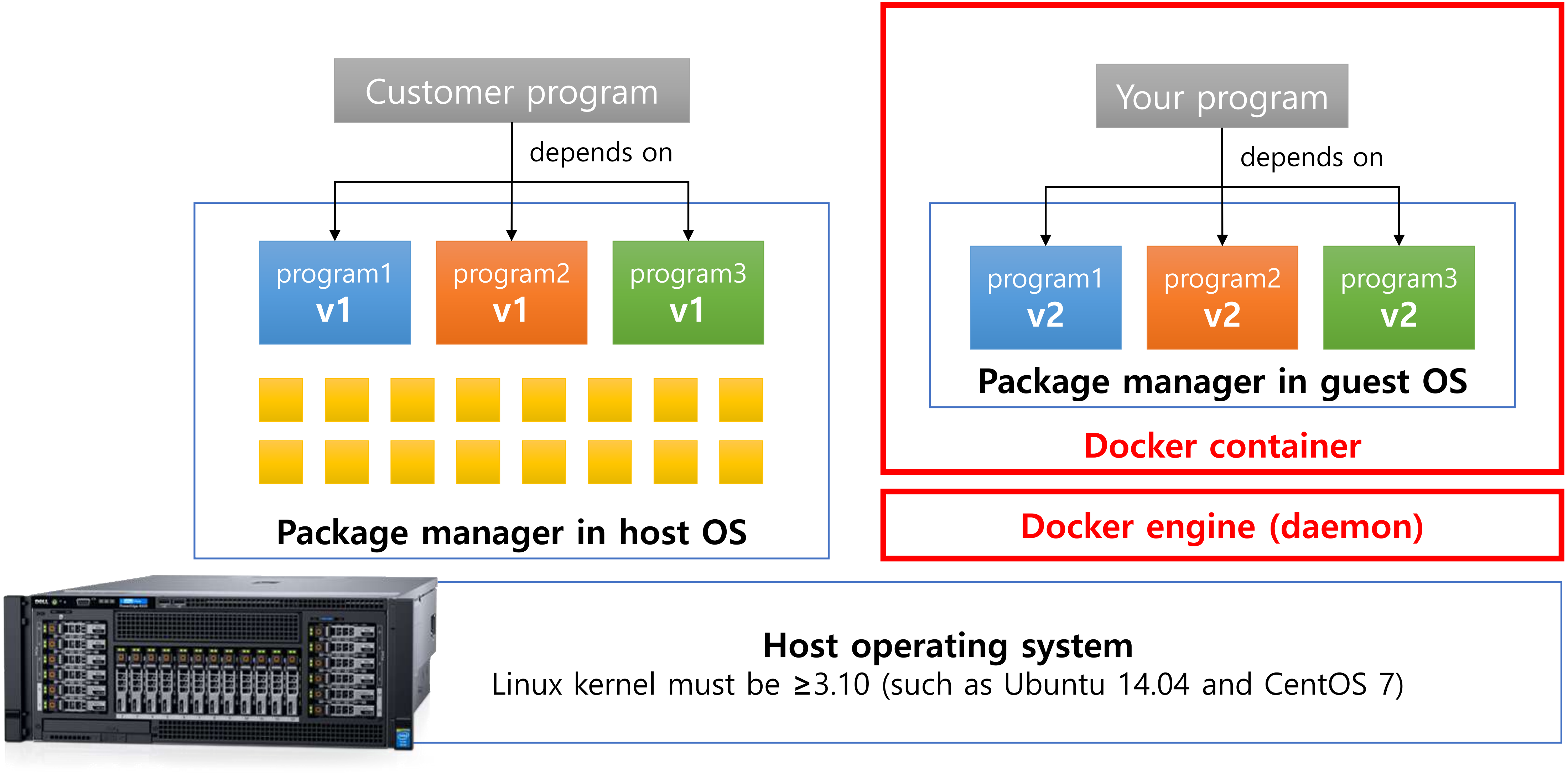
**Package manager**

**Package manager**

# Few choices left to you

1. Convince your customer (a.k.a. 甲)
2. Install all the dependencies manually (without the package manager)
3. Modify your program to make it depend v1

# Use **Docker** for isolating your application



Customer program

depends on

program1 **v1**

program2 **v1**

program3 **v1**

**Package manager in host OS**

Your program

depends on

program1 **v2**

program2 **v2**

program3 **v2**

**Package manager in guest OS**

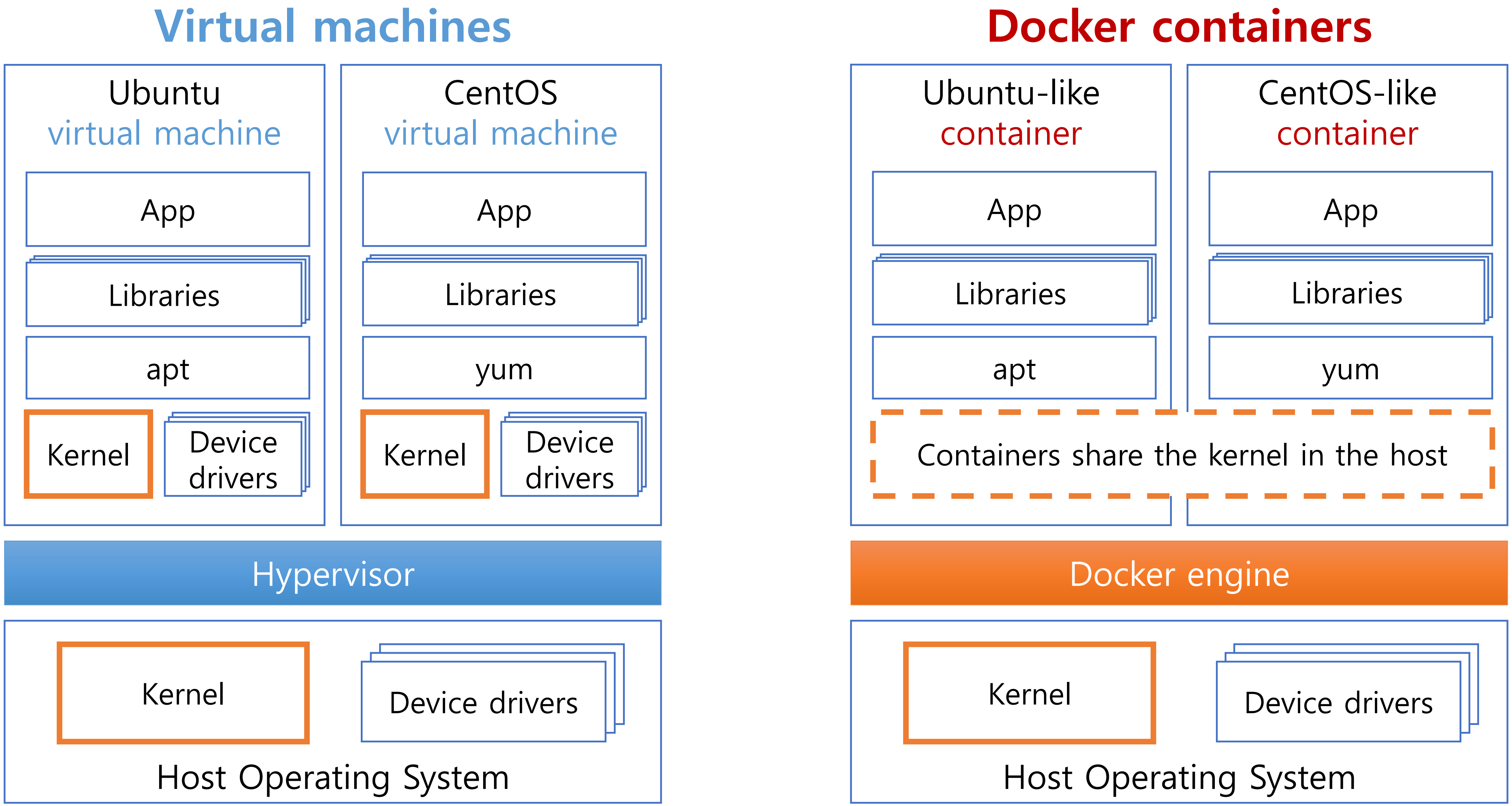
**Docker container**

**Docker engine (daemon)**

**Host operating system**

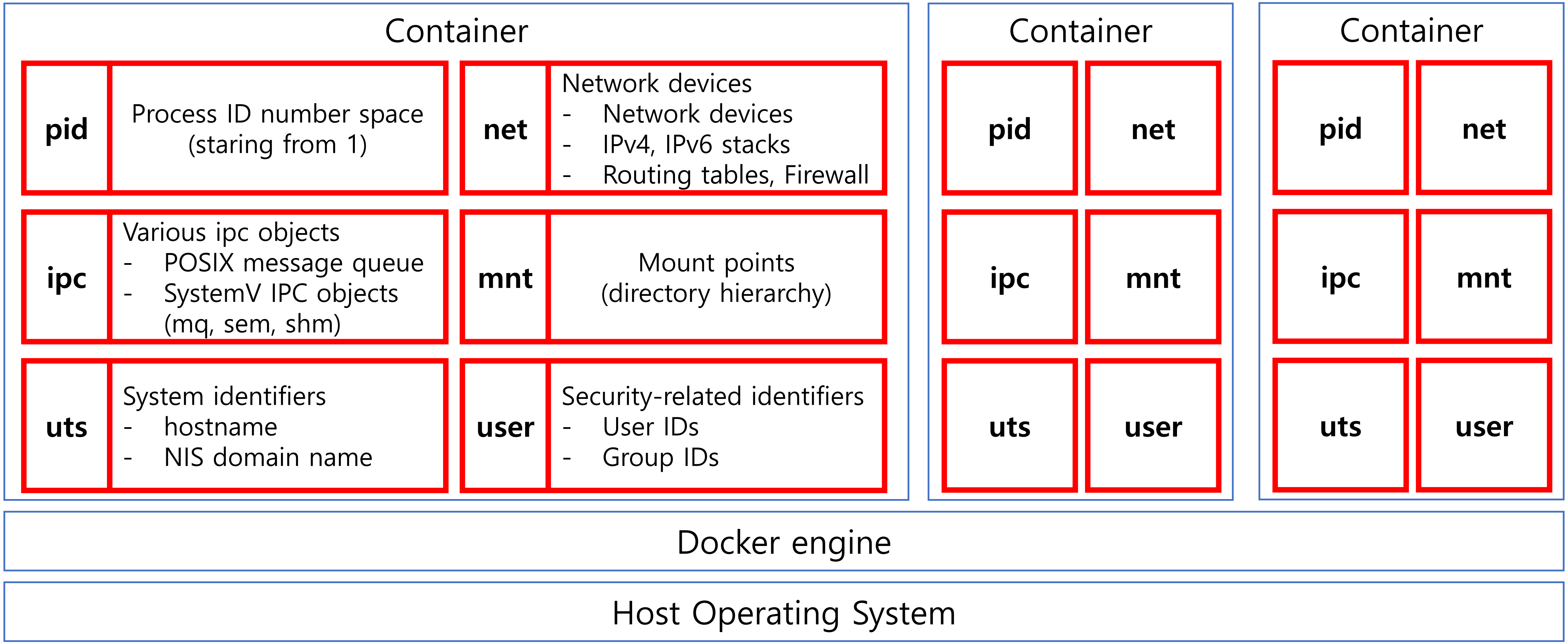
Linux kernel must be ≥3.10 (such as Ubuntu 14.04 and CentOS 7)

# **Virtual machines** and **docker containers**



# **Linux namespaces** – what makes **isolated environments** in a host OS

**Six namespaces** are enough to give ***an illusion of running inside a virtual machine***



Container

**pid** Process ID number space (staring from 1)

**net** Network devices
- Network devices
- IPv4, IPv6 stacks
- Routing tables, Firewall

**ipc** Various ipc objects
- POSIX message queue
- SystemV IPC objects (mq, sem, shm)

**mnt** Mount points (directory hierarchy)

**uts** System identifiers
- hostname
- NIS domain name

**user** Security-related identifiers
- User IDs
- Group IDs

Container

**pid** **net**

**ipc** **mnt**

**uts** **user**

Container

**pid** **net**

**ipc** **mnt**

**uts** **user**

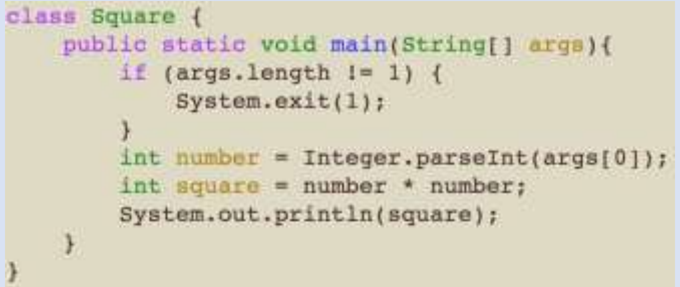
Docker engine

Host Operating System

# Analogy between **program** and **docker**

## Program

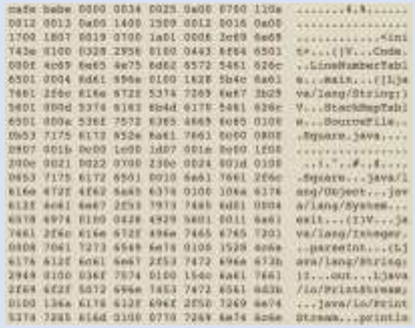
### Source code



```
class Square {
    public static void main(String[] args){
        if (args.length != 1) {
            System.exit(1);
        }
        int number = Integer.parseInt(args[0]);
        int square = number * number;
        System.out.println(square);
    }
}
```

compile

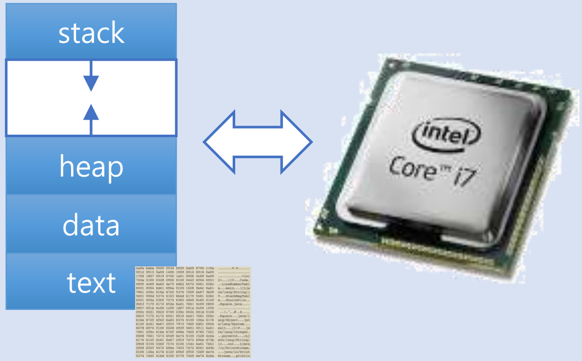
### Byte/machine code (read only)



```
cafe babe 0000 0034 0025 0a00 0700 110a  ........4.%.....
0012 0013 0a00 1400 1500 0012 0016 0a00  ................
1700 1800 0019 0700 1a01 0006 3c69 6e69  ............<ini
743e 0100 0328 2956 0100 0443 6f64 6501  t>...()V...Code.
000f 4c69 6e65 4e75 6d62 6572 5461 626c  ..LineNumberTabl
6501 0004 6d61 696e 0100 1628 5b4c 6a61  e...main...([Lja
7661 2f6c 616e 672f 5374 7269 6e67 3b29  va/lang/String;)
5601 000d 5374 6163 6b4d 6170 5461 626c  V...StackMapTabl
6501 000a 536f 7572 6365 4669 6c65 0100  e...SourceFile..
0b53 7175 6172 652e 6a61 7661 0c00 0800  .Square.java....
0907 001b 0c00 1c00 1d07 001e 0c00 1f00  ................
200c 0021 0022 0700 230c 0024 0025 0100   ..!."..#..$.%..
0653 7175 6172 6501 0010 6a61 7661 2f6c  .Square...java/l
616e 672f 4f62 6a65 6374 0100 106a 6176  ang/Object...jav
612f 6c61 6e67 2f53 7973 7465 6d01 0004  a/lang/System...
6578 6974 0100 0428 4929 5601 0011 6a61  exit...(I)V...ja
7661 2f6c 616e 672f 496e 7465 6765 7201  va/lang/Integer.
0008 7061 7273 6549 6e74 0100 1528 4c6a  ..parseInt...(Lj
6176 612f 6c61 6e67 2f53 7472 696e 673b  ava/lang/String;
2949 0100 036f 7574 0100 154c 6a61 7661  )I...out...Ljava
2f69 6f2f 5072 696e 7453 7472 6561 6d3b  /io/PrintStream;
0100 136a 6176 612f 696f 2f50 7269 6e74  ...java/io/Print
5374 7265 616d 0100 0770 7269 6e74 6c6e  Stream...println
```

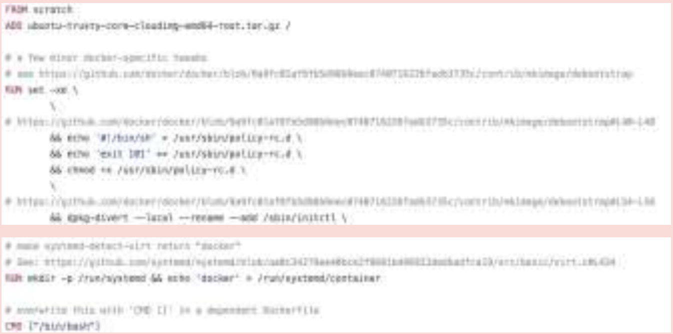
execute

### Process (read only)



## Docker

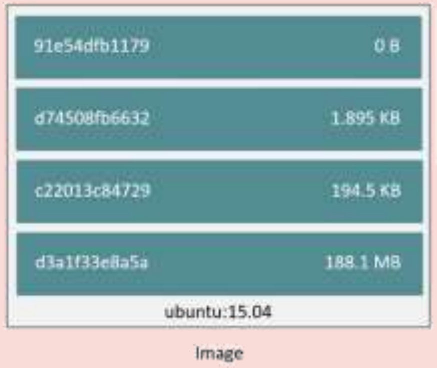
### Dockerfile



```
FROM scratch
ADD ubuntu-trusty-core-cloudimg-amd64-root.tar.gz /

# a few minor docker-specific tweaks
# see https://github.com/docker/docker/blob/9a9fc01af8fb5d98b8eec0740716226fadb3735c/contrib/mkimage/debootstrap
RUN set -xe \
        \
# https://github.com/docker/docker/blob/9a9fc01af8fb5d98b8eec0740716226fadb3735c/contrib/mkimage/debootstrap#L40-L48
        && echo '#!/bin/sh' > /usr/sbin/policy-rc.d \
        && echo 'exit 101' >> /usr/sbin/policy-rc.d \
        && chmod +x /usr/sbin/policy-rc.d \
        \
# https://github.com/docker/docker/blob/9a9fc01af8fb5d98b8eec0740716226fadb3735c/contrib/mkimage/debootstrap#L54-L56
        && dpkg-divert --local --rename --add /sbin/initctl \

# make systemd-detect-virt return "docker"
# See: https://github.com/systemd/systemd/blob/aa0c34279ee40bce2f9681b496922dedbadfca19/src/basic/virt.c#L434
RUN mkdir -p /run/systemd && echo 'docker' > /run/systemd/container

# overwrite this with 'CMD []' in a dependent Dockerfile
CMD ["/bin/bash"]
```

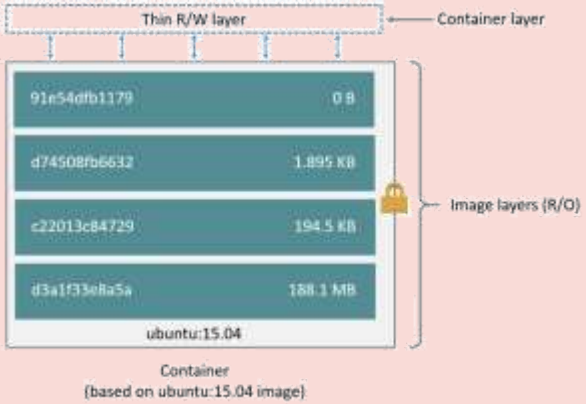
build

### Docker image (read-only **layers**)



run

### Docker container (read-only **layers** + **writable layer**)



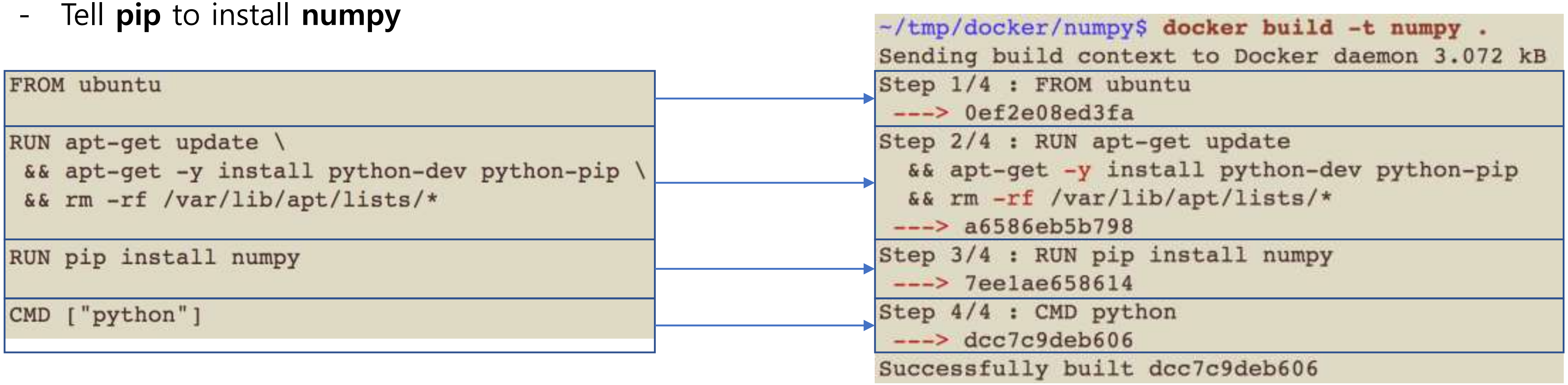
# How to define **an image** and run **a container** from it?

## 1) Write Dockerfile

- Specify to install **python** with **pip** on **ubuntu**
- Tell **pip** to install **numpy**

## 2) Build an image from Dockerfile

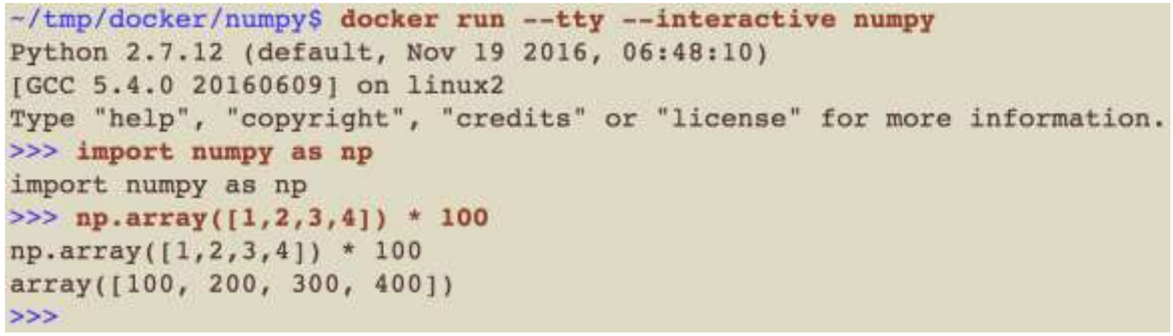
- Execute each line of Dockerfile to build **an image**



```
FROM ubuntu

RUN apt-get update \
 && apt-get -y install python-dev python-pip \
 && rm -rf /var/lib/apt/lists/*

RUN pip install numpy

CMD ["python"]
```

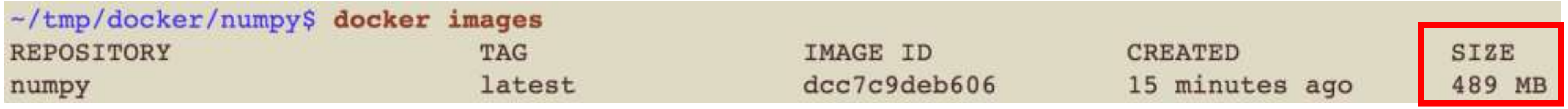
```
~/tmp/docker/numpy$ docker build -t numpy .
Sending build context to Docker daemon 3.072 kB
Step 1/4 : FROM ubuntu
 ---> 0ef2e08ed3fa
Step 2/4 : RUN apt-get update
   && apt-get -y install python-dev python-pip
   && rm -rf /var/lib/apt/lists/*
 ---> a6586eb5b798
Step 3/4 : RUN pip install numpy
 ---> 7ee1ae658614
Step 4/4 : CMD python
 ---> dcc7c9deb606
Successfully built dcc7c9deb606
```

## 3) Execute a Docker container from the image

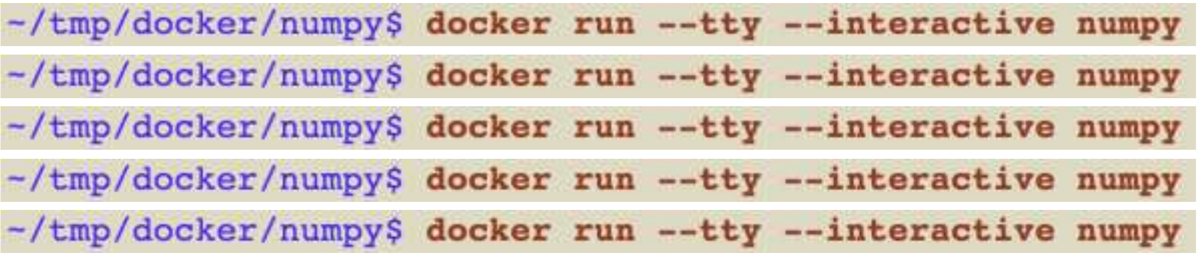


```
~/tmp/docker/numpy$ docker run --tty --interactive numpy
Python 2.7.12 (default, Nov 19 2016, 06:48:10)
[GCC 5.4.0 20160609] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy as np
import numpy as np
>>> np.array([1,2,3,4]) * 100
np.array([1,2,3,4]) * 100
array([100, 200, 300, 400])
>>>
```

# 1 to N relationship between **image** and **container**

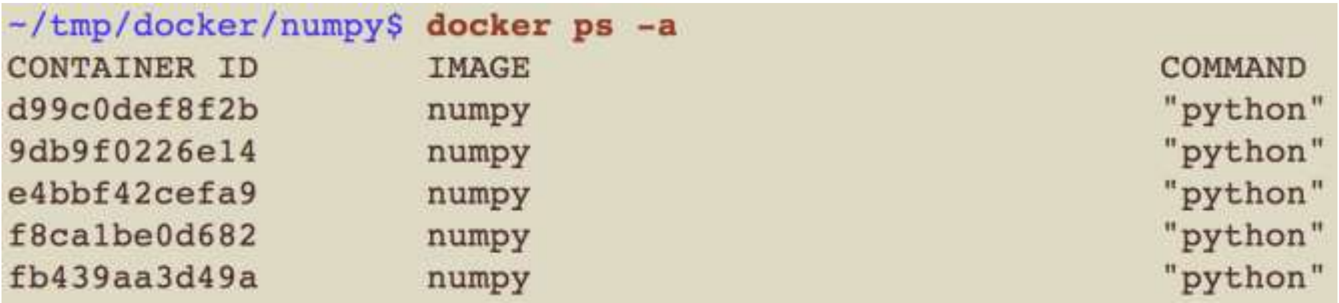


```
~/tmp/docker/numpy$ docker images
REPOSITORY                  TAG                 IMAGE ID            CREATED             SIZE
numpy                       latest              dcc7c9deb606        15 minutes ago      489 MB
```



```
~/tmp/docker/numpy$ docker run --tty --interactive numpy
~/tmp/docker/numpy$ docker run --tty --interactive numpy
~/tmp/docker/numpy$ docker run --tty --interactive numpy
~/tmp/docker/numpy$ docker run --tty --interactive numpy
~/tmp/docker/numpy$ docker run --tty --interactive numpy
```

Execute **five containers** from **an image**

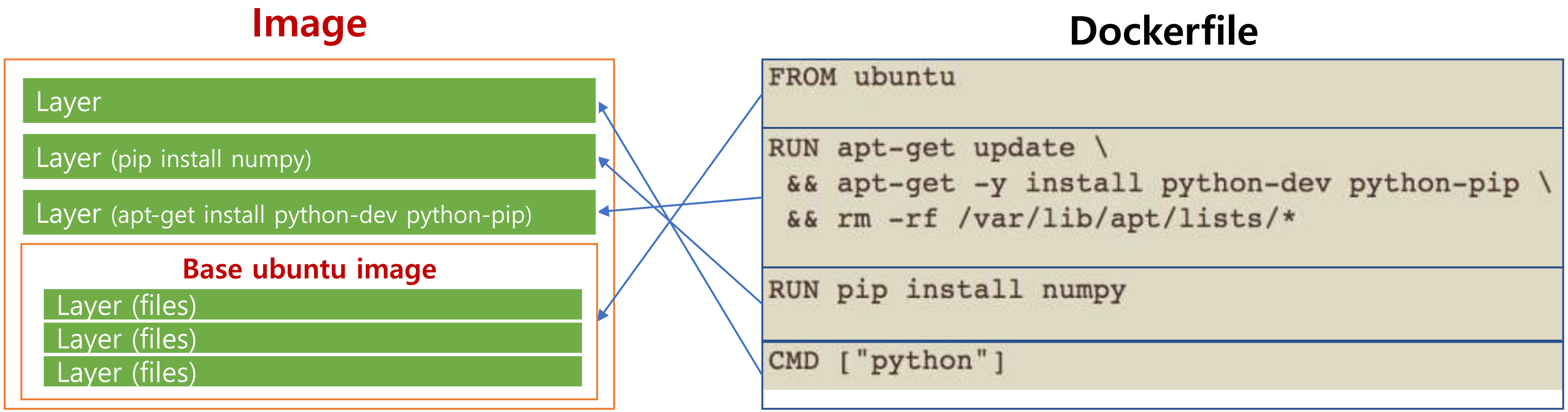


```
~/tmp/docker/numpy$ docker ps -a
CONTAINER ID         IMAGE                                    COMMAND
d99c0def8f2b         numpy                                    "python"
9db9f0226e14         numpy                                    "python"
e4bbf42cefa9         numpy                                    "python"
f8ca1be0d682         numpy                                    "python"
fb439aa3d49a         numpy                                    "python"
```

**Q)** Five containers take up 2,445MB (=489MB*5) in the host?

**A)** No due to **image layering & sharing**

# **Images** consists of **layers** each of which is **a set of files**



**Image**

Layer

Layer (pip install numpy)

Layer (apt-get install python-dev python-pip)

**Base ubuntu image**

Layer (files)

Layer (files)

Layer (files)

**Dockerfile**

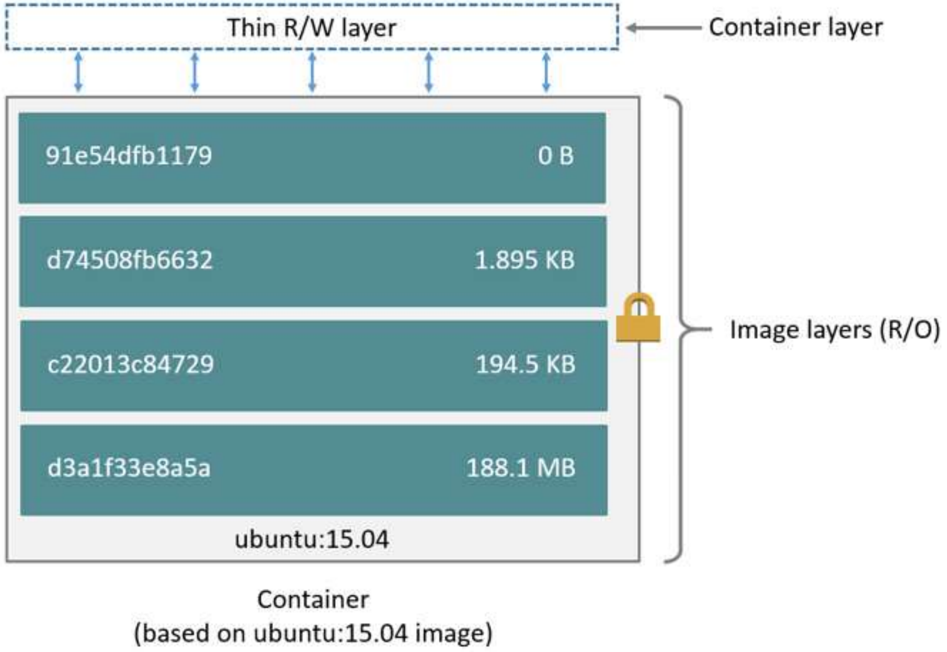
```
FROM ubuntu

RUN apt-get update \
 && apt-get -y install python-dev python-pip \
 && rm -rf /var/lib/apt/lists/*

RUN pip install numpy

CMD ["python"]
```

- **Instructions** (FROM, RUN, CMD, etc) create **layers**
  - **Base images** (imported by "FROM") also consist of layers
- If a file exists in multiple layers, the one in the upper layer is seen

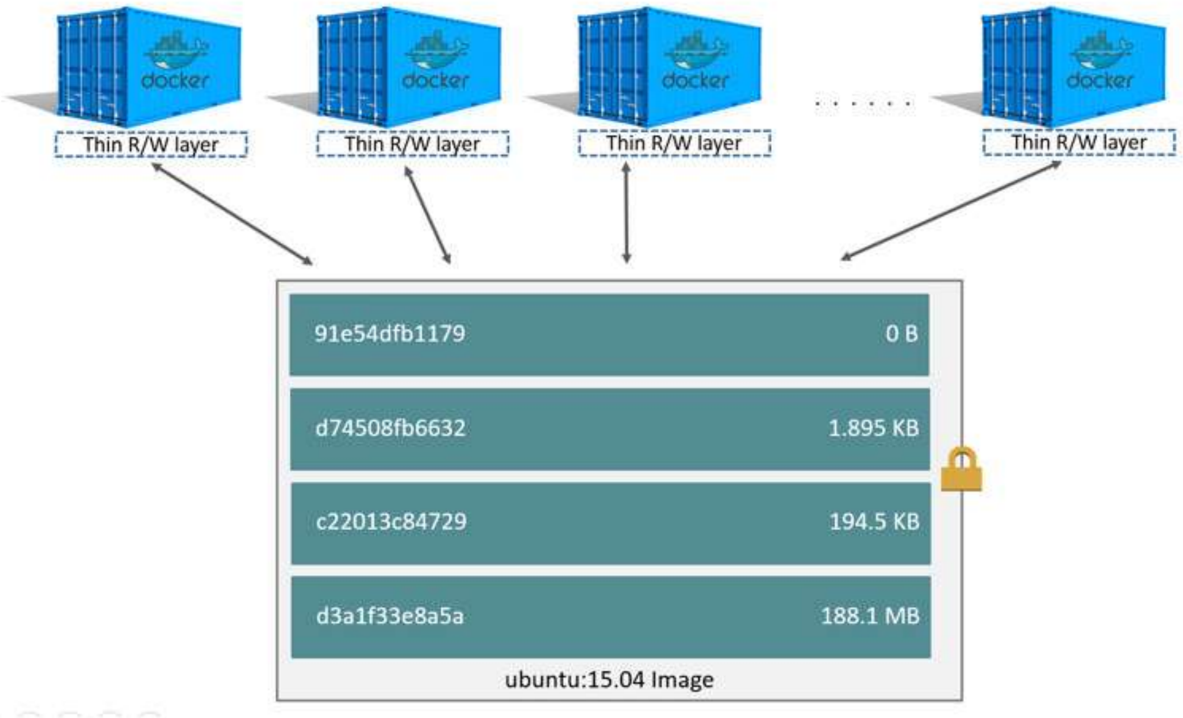
# Docker container

- **A container** is just **a thin read/write layer**
  - base images are not copied to containers

- Copy-On-Write (**COW**)
  - When **a file in the base image** is modified,
    - **copy** the file to the R/W layer
    - and **then modify** the copied file



# Image sharing between containers

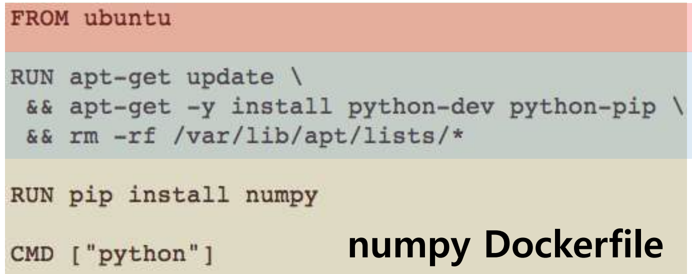


**ubuntu:15.04 image (~188MB)** does not copied to all containers

# Layer sharing between images

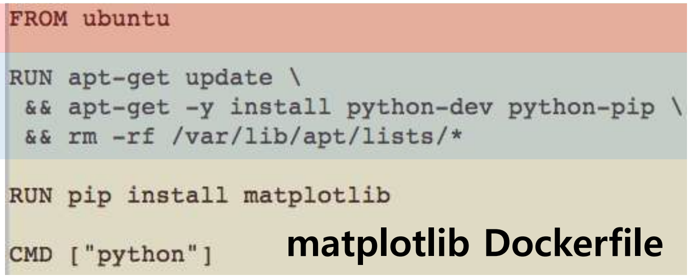
If multiple Dockerfiles

1. start from the same base image
2. share a sequence of instructions (one RUN instruction in a below example)



```
FROM ubuntu

RUN apt-get update \
 && apt-get -y install python-dev python-pip \
 && rm -rf /var/lib/apt/lists/*

RUN pip install numpy

CMD ["python"]
```

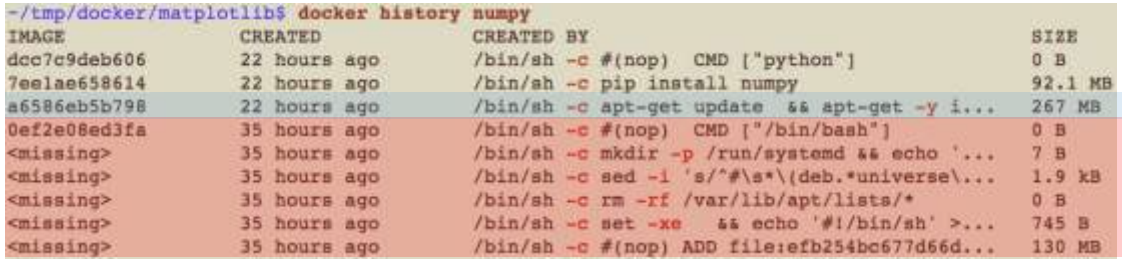
**numpy Dockerfile**



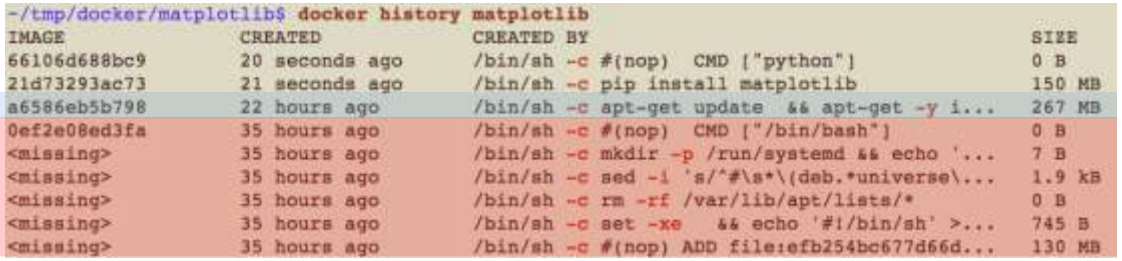
```
FROM ubuntu

RUN apt-get update \
 && apt-get -y install python-dev python-pip \
 && rm -rf /var/lib/apt/lists/*

RUN pip install matplotlib

CMD ["python"]
```

**matplotlib Dockerfile**

, then docker engine automatically reuses existing layers

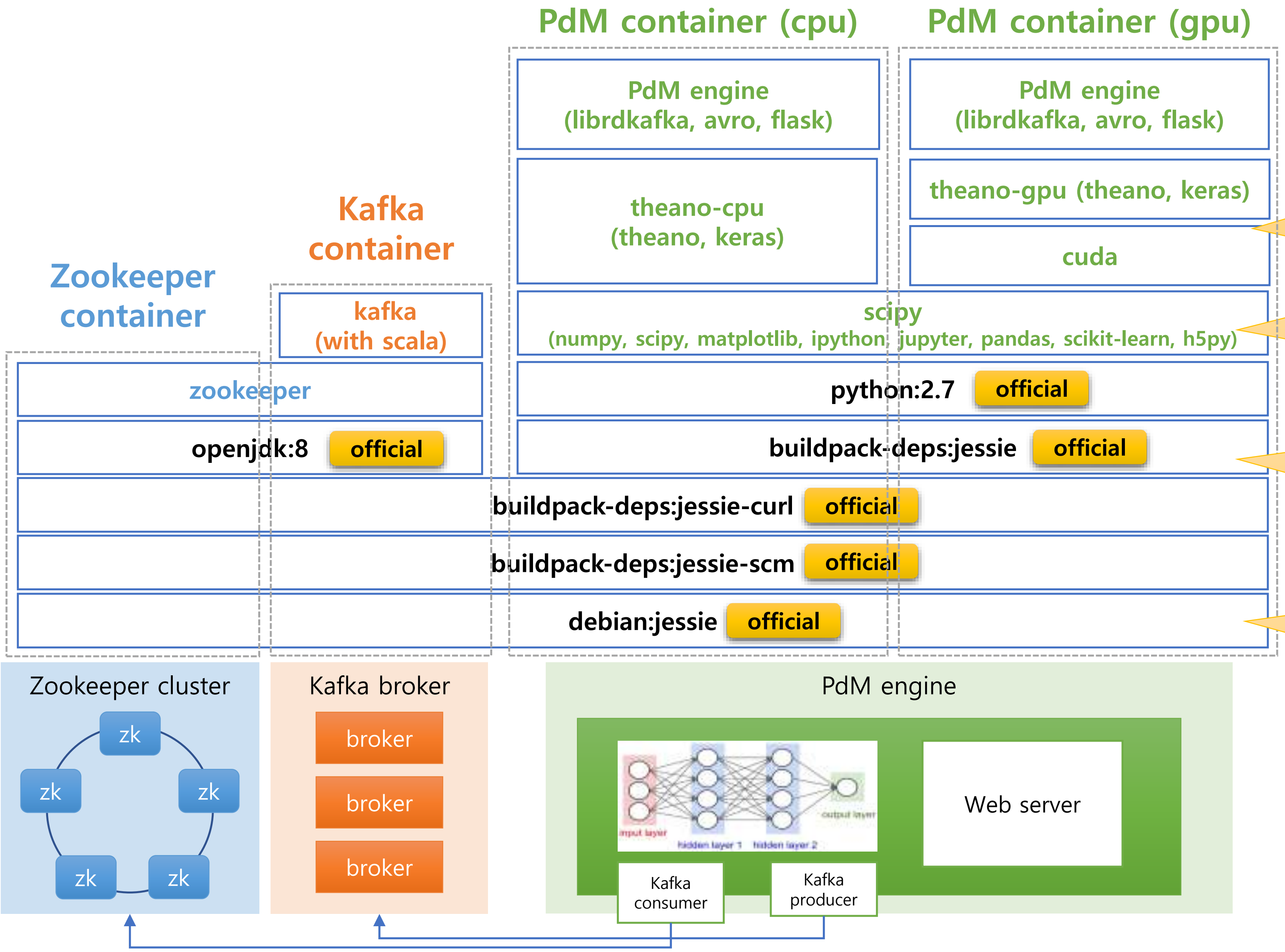


```
~/tmp/docker/matplotlib$ docker history numpy
IMAGE               CREATED             CREATED BY                                      SIZE
dcc7c9deb606        22 hours ago        /bin/sh -c #(nop)  CMD ["python"]               0 B
7ee1ae658614        22 hours ago        /bin/sh -c pip install numpy                    92.1 MB
a6586eb5b798        22 hours ago        /bin/sh -c apt-get update  && apt-get -y i...   267 MB
0ef2e08ed3fa        35 hours ago        /bin/sh -c #(nop)  CMD ["/bin/bash"]            0 B
<missing>           35 hours ago        /bin/sh -c mkdir -p /run/systemd && echo '...   7 B
<missing>           35 hours ago        /bin/sh -c sed -i 's/^#\s*\(deb.*universe\...   1.9 kB
<missing>           35 hours ago        /bin/sh -c rm -rf /var/lib/apt/lists/*          0 B
<missing>           35 hours ago        /bin/sh -c set -xe   && echo '#!/bin/sh' >...   745 B
<missing>           35 hours ago        /bin/sh -c #(nop) ADD file:efb254bc677d66d...   130 MB
```



```
~/tmp/docker/matplotlib$ docker history matplotlib
IMAGE               CREATED             CREATED BY                                      SIZE
66106d688bc9        20 seconds ago      /bin/sh -c #(nop)  CMD ["python"]               0 B
21d73293ac73        21 seconds ago      /bin/sh -c pip install matplotlib               150 MB
a6586eb5b798        22 hours ago        /bin/sh -c apt-get update  && apt-get -y i...   267 MB
0ef2e08ed3fa        35 hours ago        /bin/sh -c #(nop)  CMD ["/bin/bash"]            0 B
<missing>           35 hours ago        /bin/sh -c mkdir -p /run/systemd && echo '...   7 B
<missing>           35 hours ago        /bin/sh -c sed -i 's/^#\s*\(deb.*universe\...   1.9 kB
<missing>           35 hours ago        /bin/sh -c rm -rf /var/lib/apt/lists/*          0 B
<missing>           35 hours ago        /bin/sh -c set -xe   && echo '#!/bin/sh' >...   745 B
<missing>           35 hours ago        /bin/sh -c #(nop) ADD file:efb254bc677d66d...   130 MB
```

# Example of stacking docker images



PdM container (cpu)

PdM engine
(librdkafka, avro, flask)

theano-cpu
(theano, keras)

PdM container (gpu)

PdM engine
(librdkafka, avro, flask)

theano-gpu (theano, keras)

cuda

Kafka container

kafka
(with scala)

Zookeeper container

zookeeper

openjdk:8 official

scipy
(numpy, scipy, matplotlib, ipython, jupyter, pandas, scikit-learn, h5py)

python:2.7 official

buildpack-deps:jessie official

buildpack-deps:jessie-curl official

buildpack-deps:jessie-scm official

debian:jessie official

Zookeeper cluster

zk zk zk zk zk

Kafka broker

broker

broker

broker

PdM engine

Web server

Kafka consumer

Kafka producer

**theano** compiles its expression graphs into CPU/GPU instructions

**scipy** libraries has nothing to do with GPU, so share it

**buildpack-deps** contains essential tools to download/compile softwares

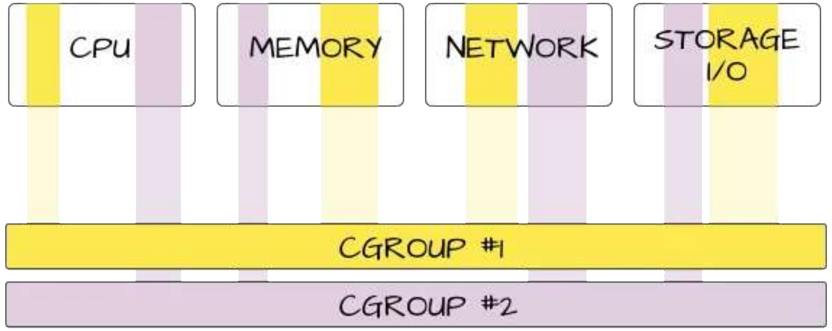
**jessie** is the latest, stable Debian release

# Enabling technologies for docker (wrap-up)

- **Linux namespaces** (covered)
  - To isolate system resources
    - pid, net, ipc, mnt, uts, user
  - It makes a secure & isolate environment (like a VM)

- **Advanced multi-layer unification File System** (covered)
  - Image layering & sharing

- **Linux control groups** (not covered)
  - To track, limit, and isolate resources
    - CPU, memory, network, and IO



* https://mairin.wordpress.com/2011/05/13/ideas-for-a-cgroups-ui/

# Docker topics not covered here

- How to install **Docker engine**
- What are the docker instructions other than **FROM**, **RUN**, and **CMD**
  - ENV / ADD / ENTRYPOINT / LABEL / EXPOSE / COPY / VOLUME / WORKDIR / ONBUILD
- How to push **local Docker images** to docker hub
- How to pull **remote images** from docker hub
- …

**Consult with [https://docs.docker.com/engine/getstarted/](https://docs.docker.com/engine/getstarted/)**

# Kubernetes

Motivation

A motivating example

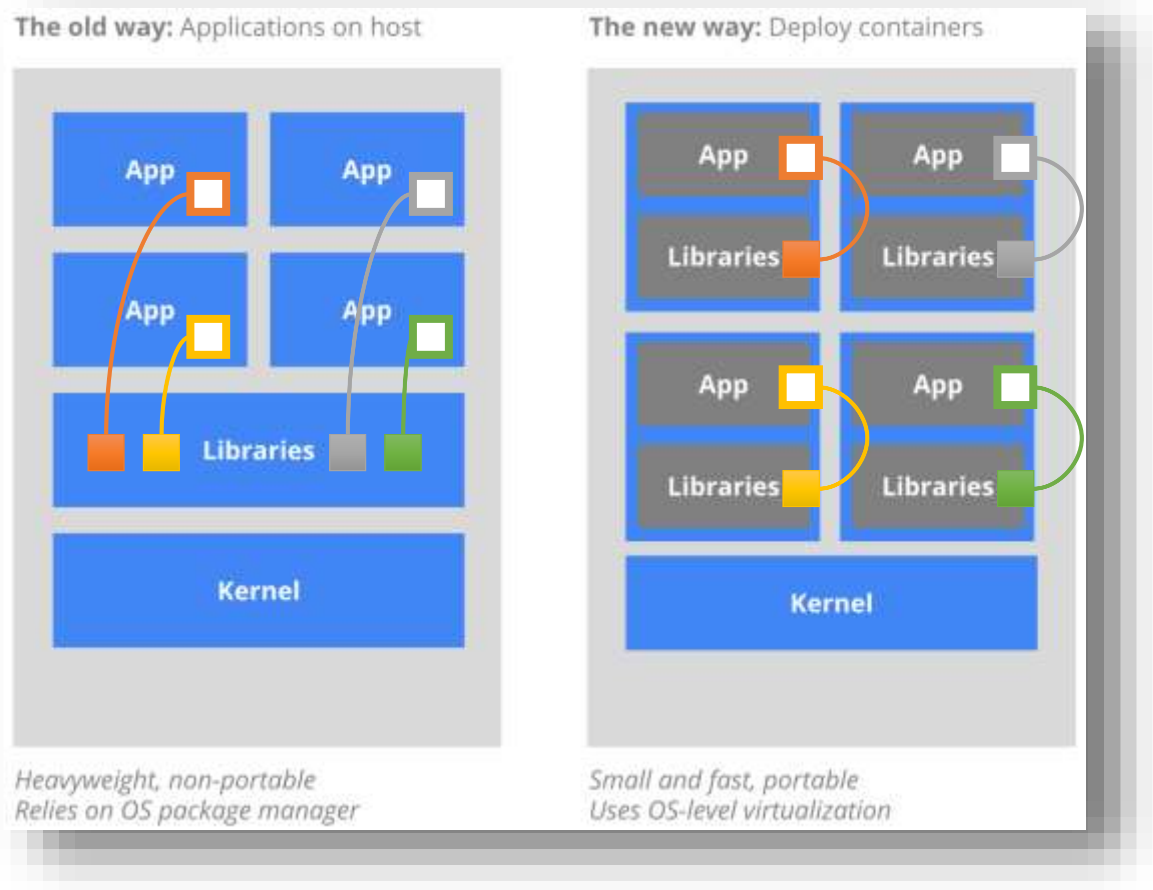
# Disclaimer

- The purpose of this section is to briefly explain Kubernetes **without details**

- For a detailed explanation with the exact Kubernetes terminology, see the following slide
  - https://www.slideshare.net/ssuser6bb12d/kubernetes-introduction-71846110

# What is Kubernetes for?

## Container-based virtualization + Container orchestration


The old way: Applications on host
App
App
App
App
Libraries
Kernel
Heavyweight, non-portable
Relies on OS package manager
The new way: Deploy containers
App
Libraries
App
Libraries
App
Libraries
App
Libraries
Kernel
Small and fast, portable
Uses OS-level virtualization

from the official site : https://kubernetes.io/docs/whatisk8s/

**To satisfy common needs in production**

**replicating application instances**
**naming and discovery**
**load balancing**
**horizontal auto-scaling**
co-locating helper processes
mounting storage systems
distributing secrets
application health checking
rolling updates
resource monitoring
log access and ingestion
…

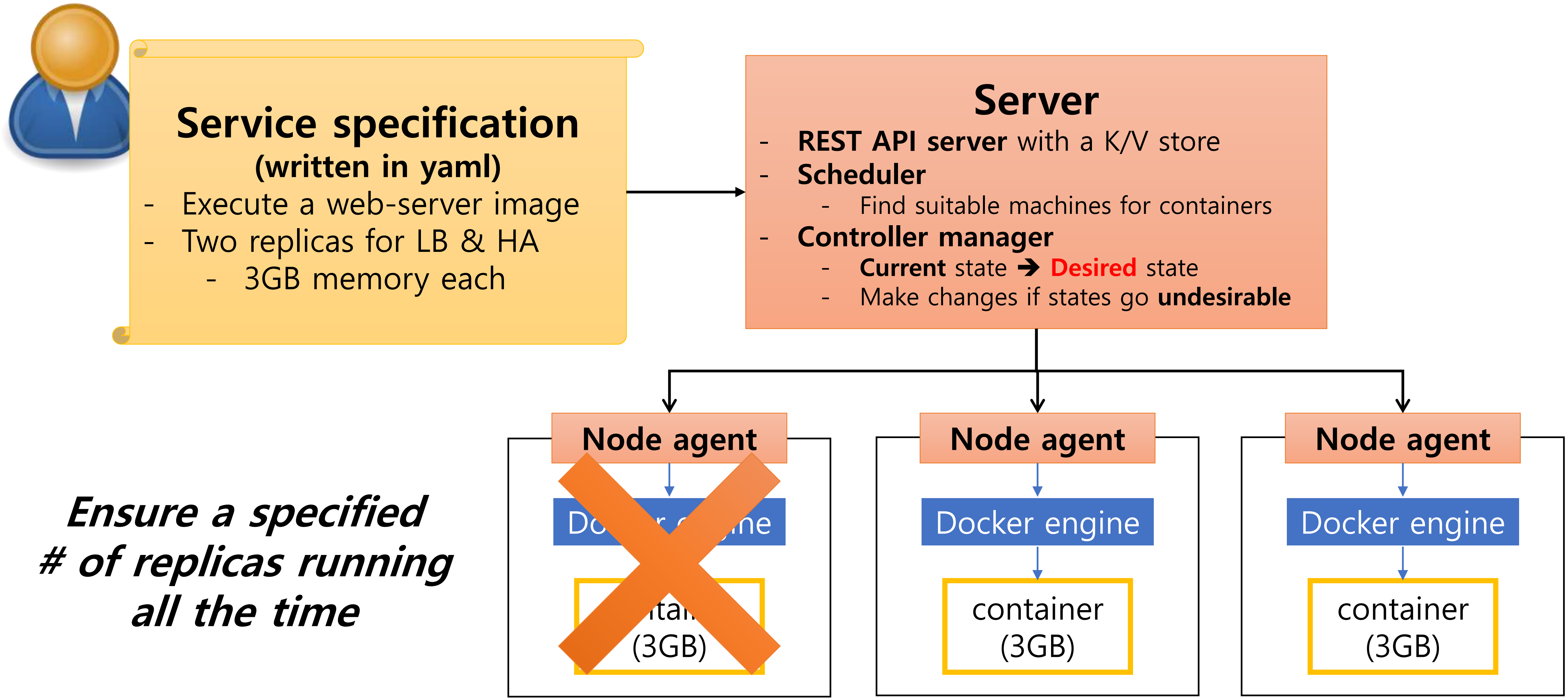
# Why **Docker** with **Kubernetes?**

- A mission of our group
    - **Systematize service deployment** and **service operation** on cluster
    - I believe that **systematizing smth.** is **to minimize human efforts on smth.**

- How to **minimize human efforts** on **service deployment**?
    - Make software portable using **a container technology**
        - **Docker** (chosen for its maturity and popularity)
        - **Rkt** from CoreOS (alternative)
    - Build images and run containers anywhere
        - Your laptop, servers, on-premise clusters, even cloud

- How to **minimize human efforts** on **service operation**?
    - Inform **a container orchestration runtime** of **service specification**
        - **Kubernetes** from Google (chosen for its **maturity** and **expressivity**)
        - **Docker swarm** from Docker
    - Define **your specification** and then the runtime operates your services as you wish

# Kubernetes architecture



**Service specification**
**(written in yaml)**

- Execute a web-server image
- Two replicas for LB & HA
  - 3GB memory each

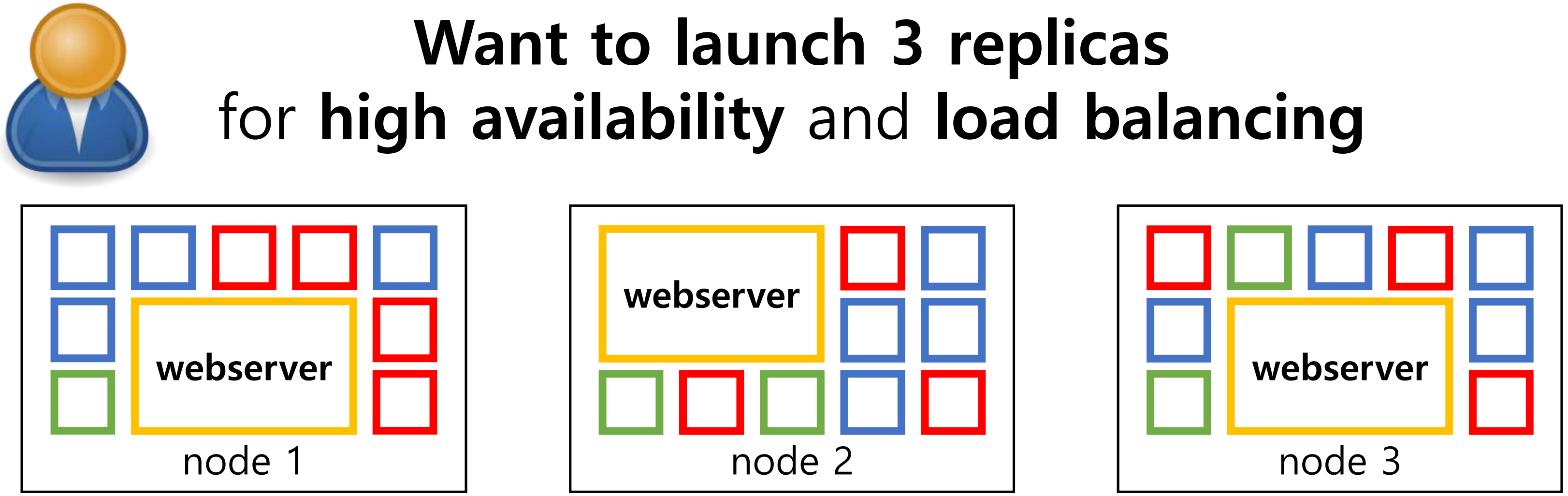
**Server**

- **REST API server** with a K/V store
- **Scheduler**
  - Find suitable machines for containers
- **Controller manager**
  - **Current** state ➔ **Desired** state
  - Make changes if states go **undesirable**

***Ensure a specified***
***# of replicas running***
***all the time***

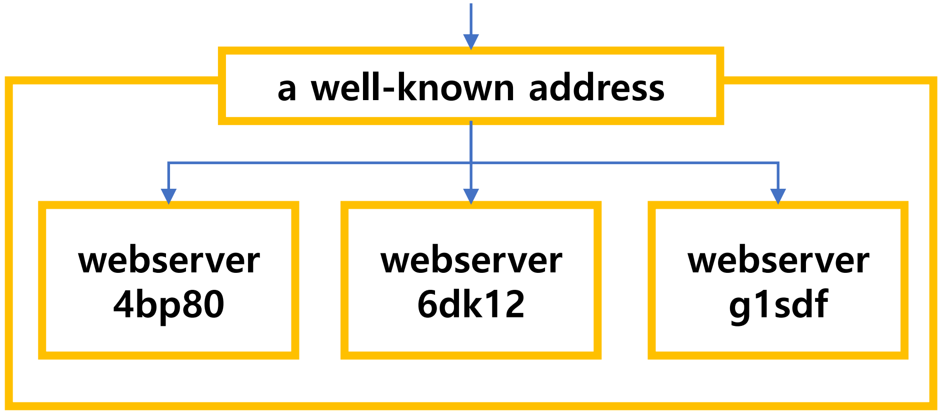
# Web server example

**Want to launch 3 replicas**
for **high availability** and **load balancing**



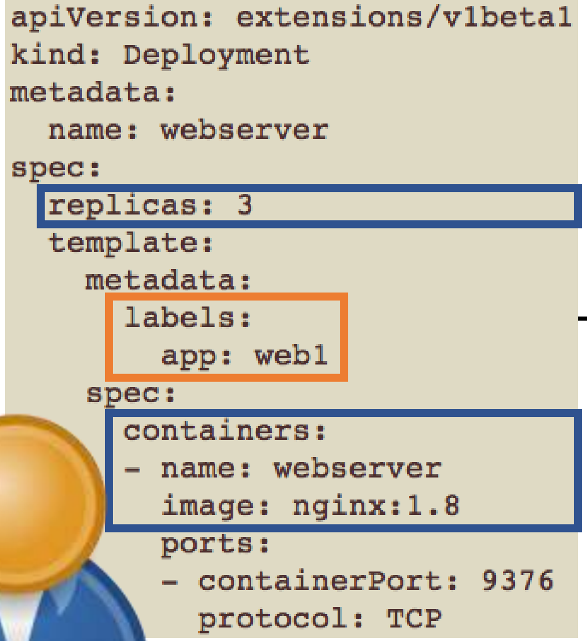
## How to achieve the followings?

- Users must be unaware of the replicas
- Traffic is evenly distributed to replicas



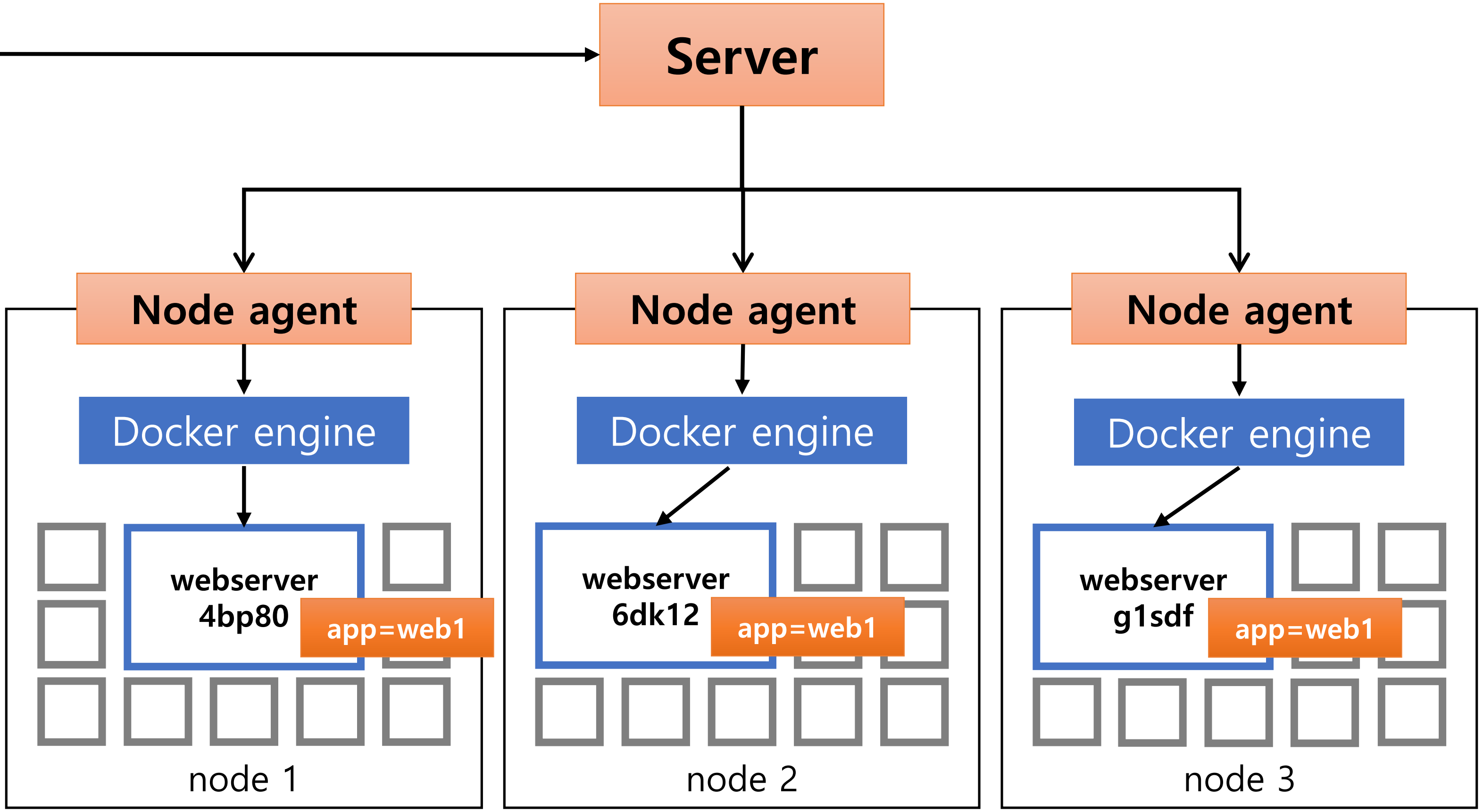
**It's a piece of cake with Kubernetes!**

# How to replicate your service instances



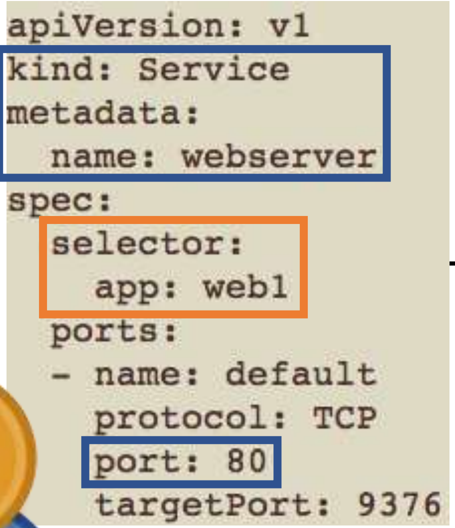
```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: webserver
spec:
  replicas: 3
  template:
    metadata:
      labels:
        app: web1
    spec:
      containers:
      - name: webserver
        image: nginx:1.8
        ports:
        - containerPort: 9376
          protocol: TCP
```

Specify **your Docker image** and **a replication factor** using Deployment

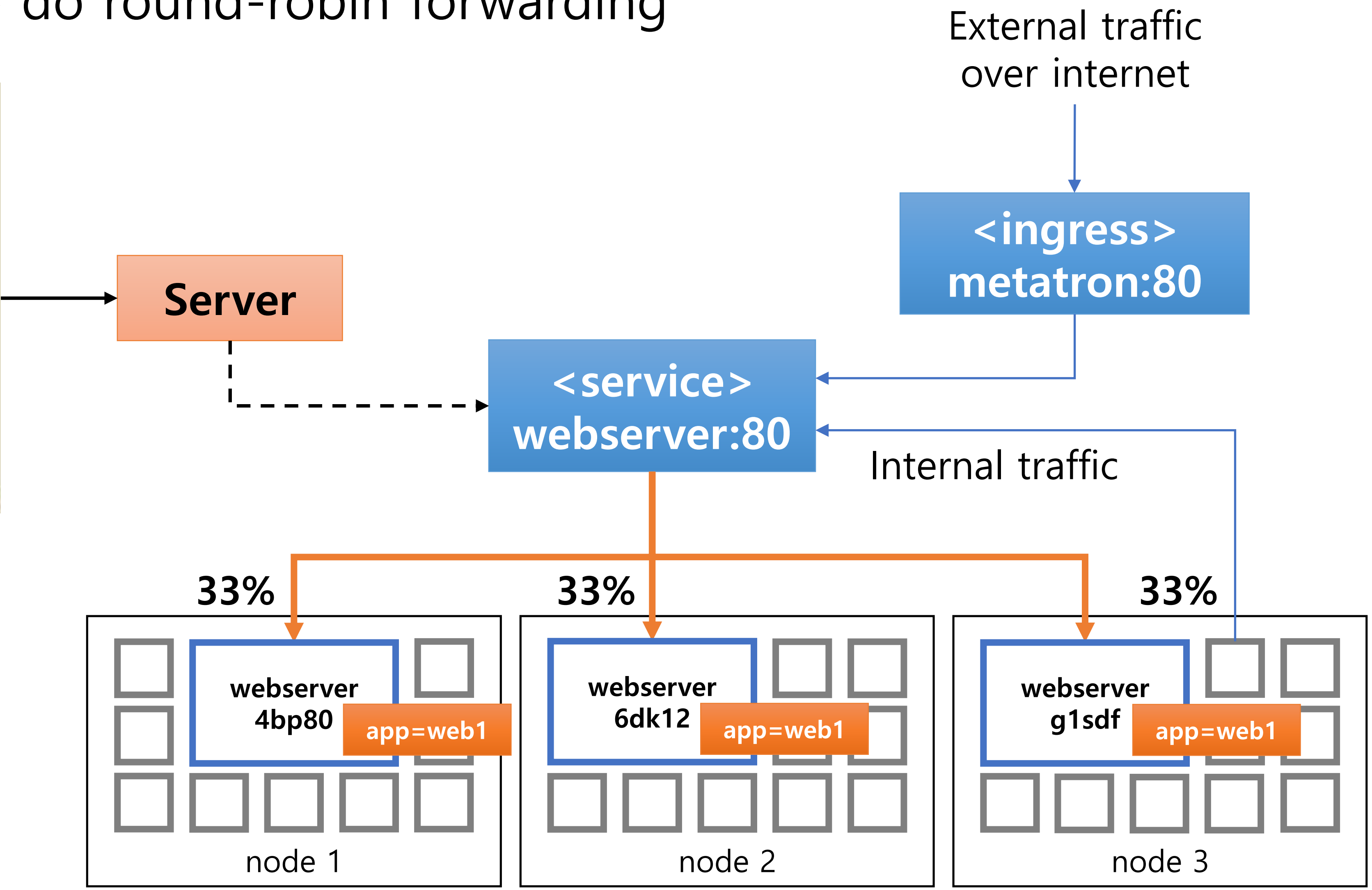


Specify **a common label** to group containers with different names

# Define **a service** to do round-robin forwarding



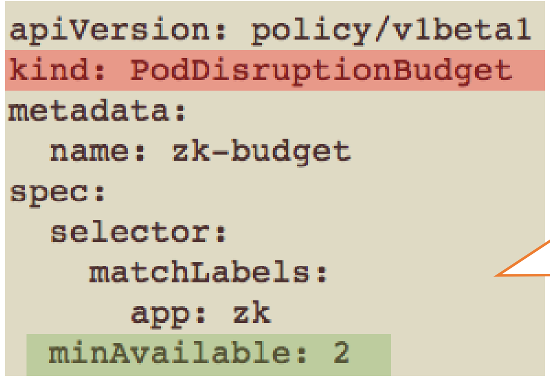
```
apiVersion: v1
kind: Service
metadata:
  name: webserver
spec:
  selector:
    app: web1
  ports:
  - name: default
    protocol: TCP
    port: 80
    targetPort: 9376
```



*Kubernetes runs **its own DNS server** for name resolution*

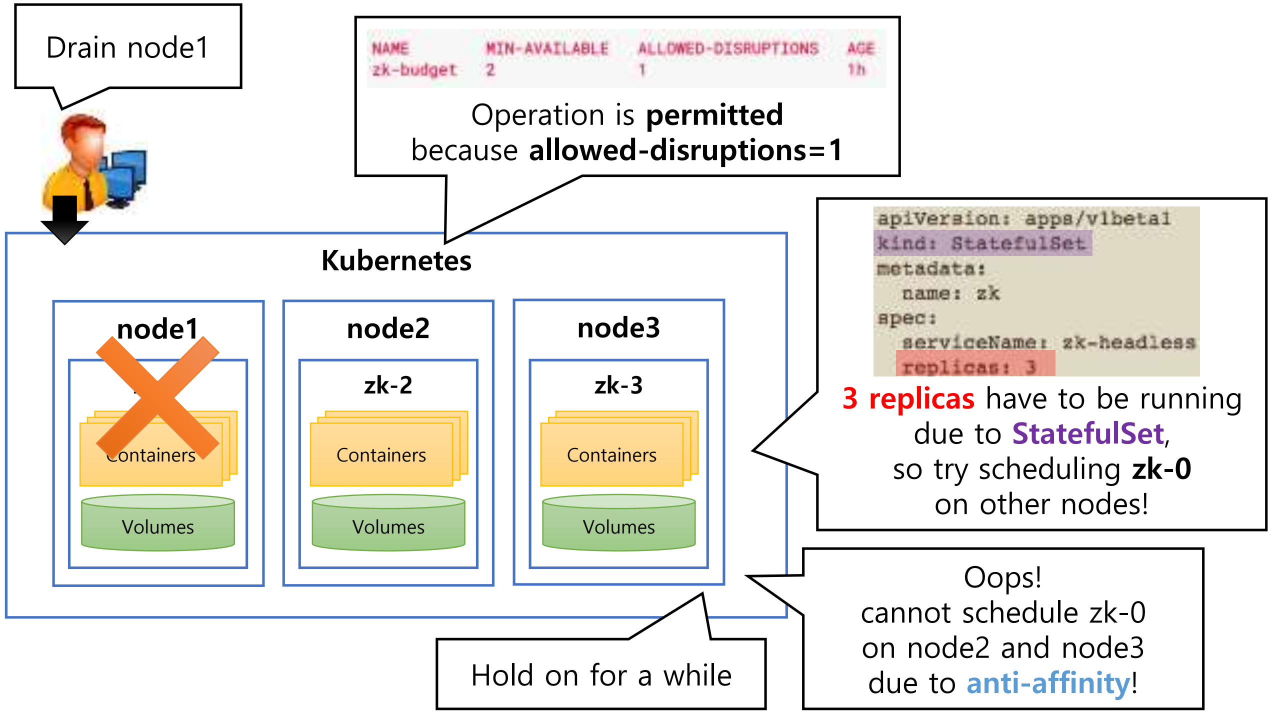
*Kubernetes **manipulates iptables** on each node **to proxy traffic***

# How to guarantee **a certain # of running containers** during maintenance



```
apiVersion: policy/v1beta1
kind: PodDisruptionBudget
metadata:
  name: zk-budget
spec:
  selector:
    matchLabels:
      app: zk
  minAvailable: 2
```

Define **disruption budget**
to specify requirement for
**the minimum available containers**



```
NAME        MIN-AVAILABLE   ALLOWED-DISRUPTIONS   AGE
zk-budget   2               1                     1h
```

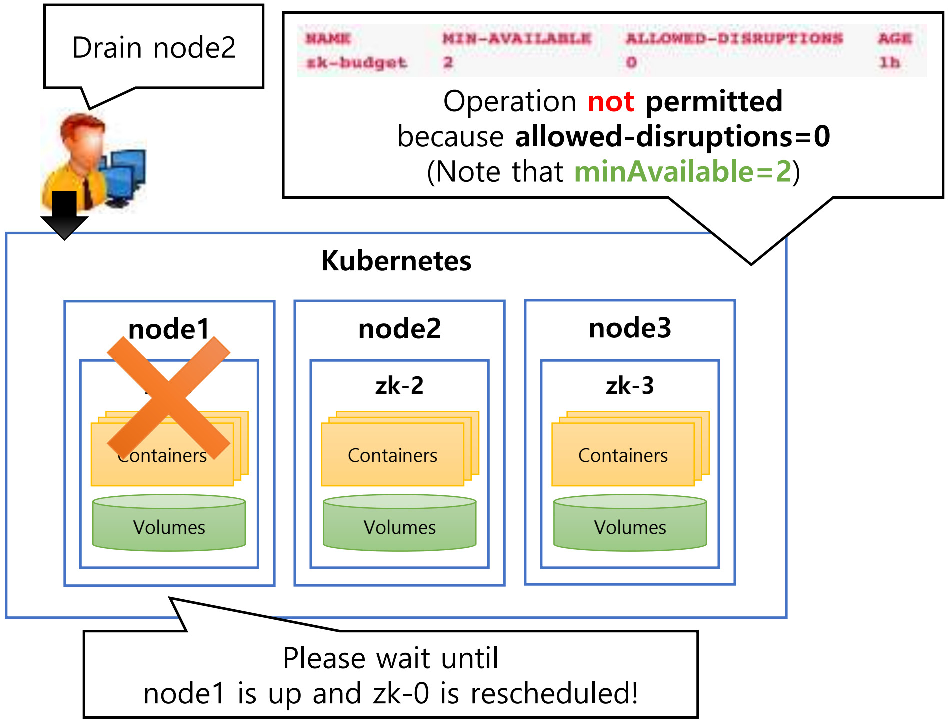
Operation is **permitted**
because **allowed-disruptions=1**

```
apiVersion: apps/v1beta1
kind: StatefulSet
metadata:
  name: zk
spec:
  serviceName: zk-headless
  replicas: 3
```

**3 replicas** have to be running
due to **StatefulSet**,
so try scheduling **zk-0**
on other nodes!

Oops!
cannot schedule zk-0
on node2 and node3
due to **anti-affinity**!

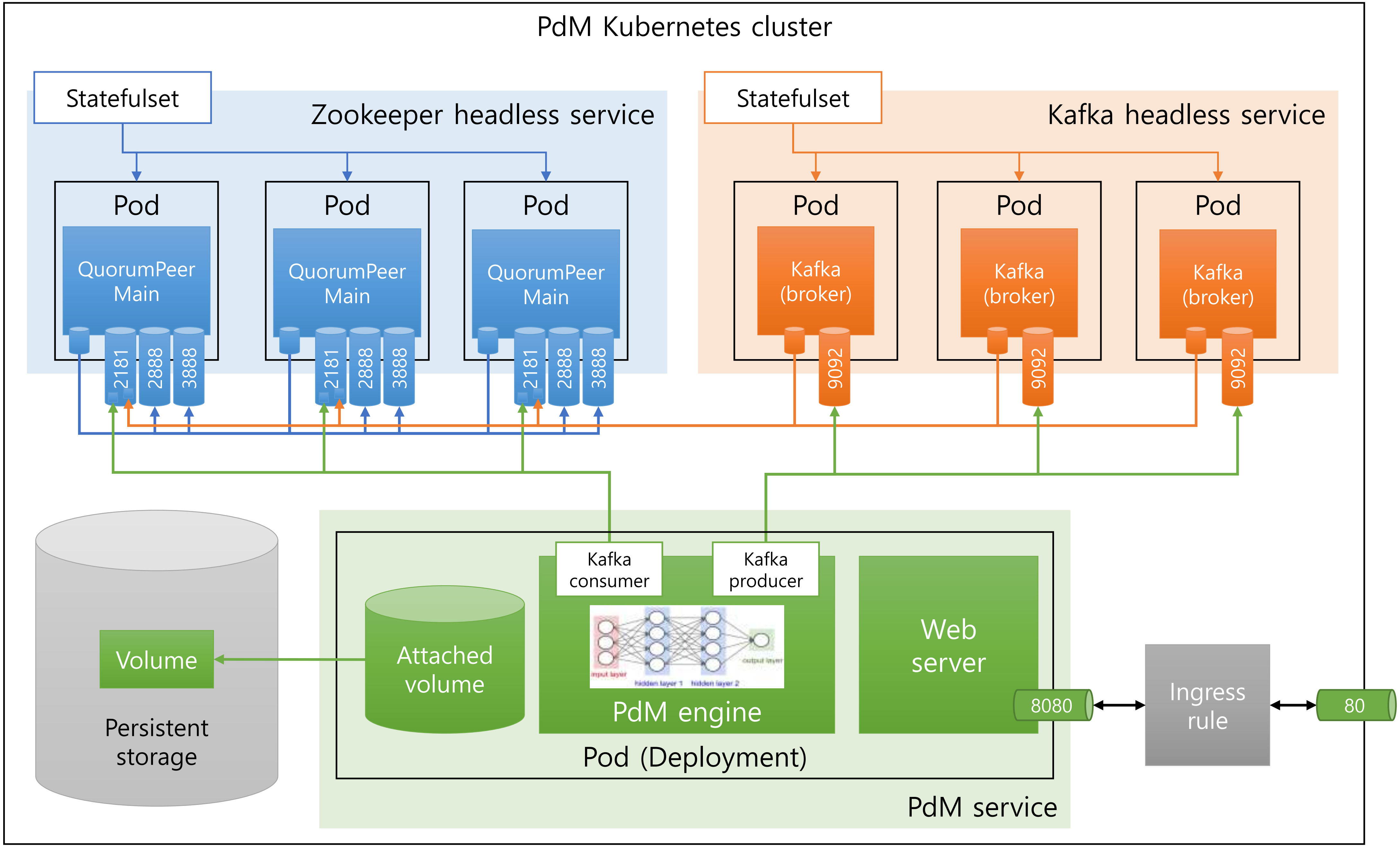
Hold on for a while



```
NAME        MIN-AVAILABLE   ALLOWED-DISRUPTIONS   AGE
zk-budget   2               0                     1h
```

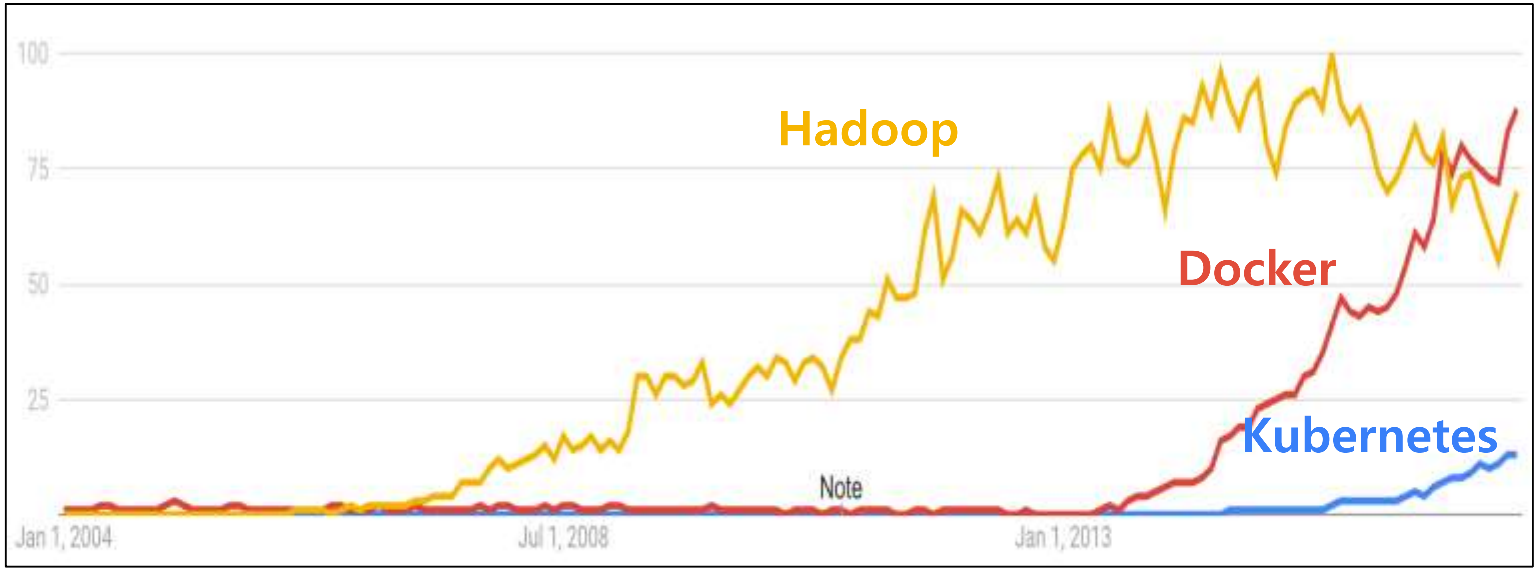
Operation **not permitted**
because **allowed-disruptions=0**
(Note that minAvailable=2)

Please wait until
node1 is up and zk-0 is rescheduled!

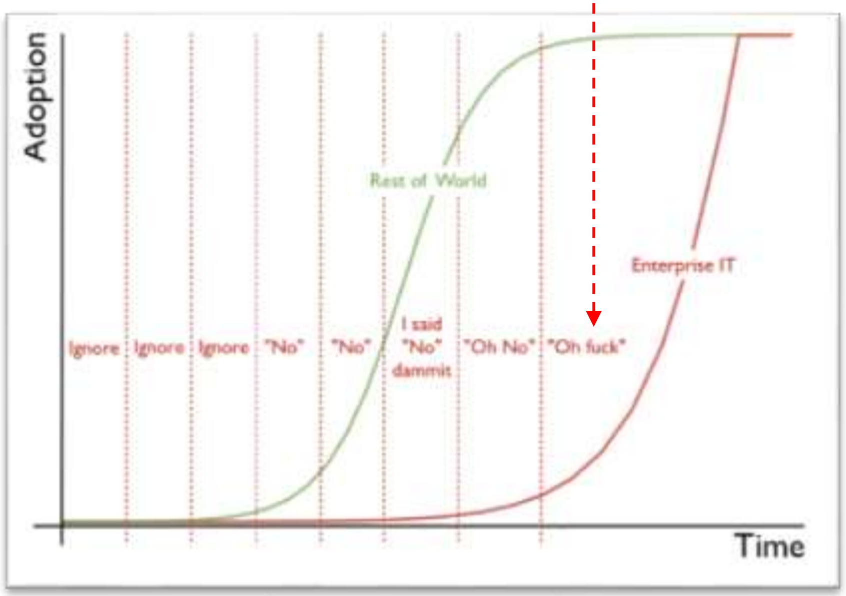

PdM Kubernetes cluster
Statefulset
Zookeeper headless service
Pod
QuorumPeer Main
2181
2888
3888
Pod
QuorumPeer Main
2181
2888
3888
Pod
QuorumPeer Main
2181
2888
3888
Statefulset
Kafka headless service
Pod
Kafka (broker)
9092
Pod
Kafka (broker)
9092
Pod
Kafka (broker)
9092
Volume
Persistent storage
Attached volume
Kafka consumer
Kafka producer
input layer
hidden layer 1
hidden layer 2
output layer
PdM engine
Web server
8080
Pod (Deployment)
PdM service
Ingress rule
80

# Overview & Conclusion

- **Docker** to build ***portable*** software
  - Build your software upon Docker
  - Then distribute it anywhere (even on MS Azure and AWS)
- **Kubernetes** to orchestrate multiple **Docker** instances
- Start using **Docker** and **Kubernetes** before too late!
  - Google has been using container technologies more than 10 years



**Popularity of Docker and Kubernetes**



**The Enterprise IT Adoption Cycle**

***the end***