Docker Penang Meetup # 1

@DockerPenang

docker

Docker Community

The Docker Meetup Community is…

200,000+
Meetup members around the world

200+
Active Local groups

81
countries

1000+
1000+ meetups per year

100K
Attendees per year

How to be part of Docker Community & its benefits ...

- Sign up page - **https://goo.gl/pjKMFS**
- Everyone is welcome as long as we comply with **Code of Conduct**
- Docker Docs - **https://docs.docker.com**
- Docker Forum - **https://forums.docker.com**
- Play with Docker Classroom - **http://training.play-with-docker.com**
- **DockerCon 2018** - [June 12-15, Moscone Center, San Francisco]

Docker Bday #5

March 19th - 25th

Goal: Give the community the opportunity to use and learn Docker through fun and engaging HOL with the help of mentors.

Mentors will be on hand to help to answer any one on one questions.

Docker Bday #5

Week of: March 19th - 25th

Guidelines:

- Host in-person workshop in March, ideally the week of March 19th -25th
- Announce through your chapter page on events.docker.com
- Recruit mentors
- All Docker advocacy groups can participate - all campus ambassadors etc.

Content: Play with Docker

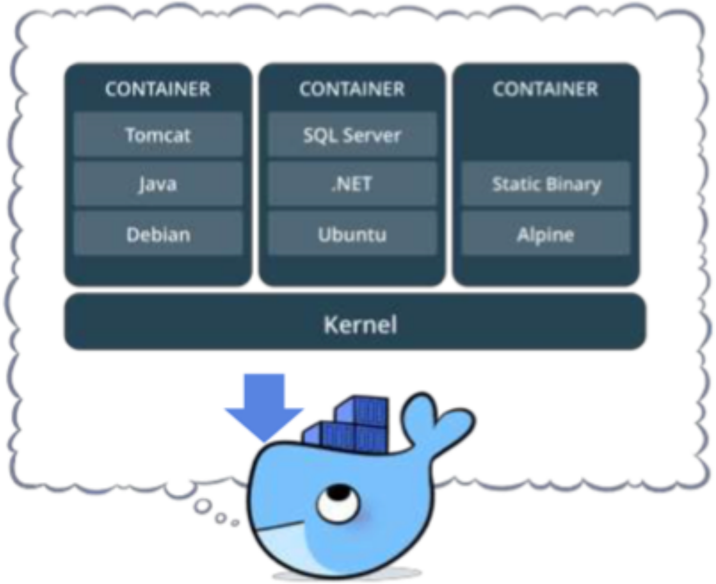
- Docker 101 Windows, Linux
- Getting started w/ d4m / d4w + K8s
- Orchestration - Kubernetes, Swarm

Introduction to Docker

The open platform to build, ship and run any applications anywhere

docker

What is Container?



OS-level virtualization solution

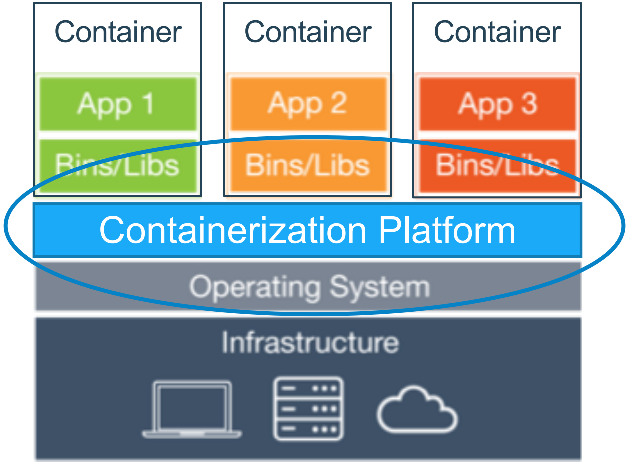
- Standardized packaging for software and dependencies.
- Provides a **lightweight** virtual environment that **groups and isolates a set of processes and resources** such as memory, CPU, disk, etc., from the host and any other containers.

Container vs Virtual Machine

Guests share same kernel

- Abstract application from OS

OS level virtualization



Container | Container | Container

App 1 | App 2 | App 3

Bins/Libs | Bins/Libs | Bins/Libs

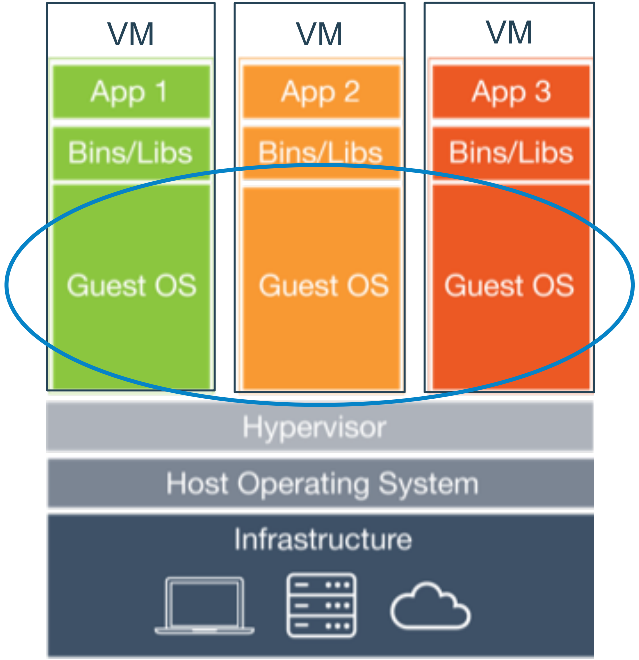
Containerization Platform

Operating System

Infrastructure

VS

Hardware level virtualization



VM | VM | VM

App 1 | App 2 | App 3

Bins/Libs | Bins/Libs | Bins/Libs

Guest OS | Guest OS | Guest OS

Hypervisor

Host Operating System

Infrastructure

Each guest has it own OS kernel

- Abstract OS from Hardware

Key Benefits for Container

Efficiency

Speed

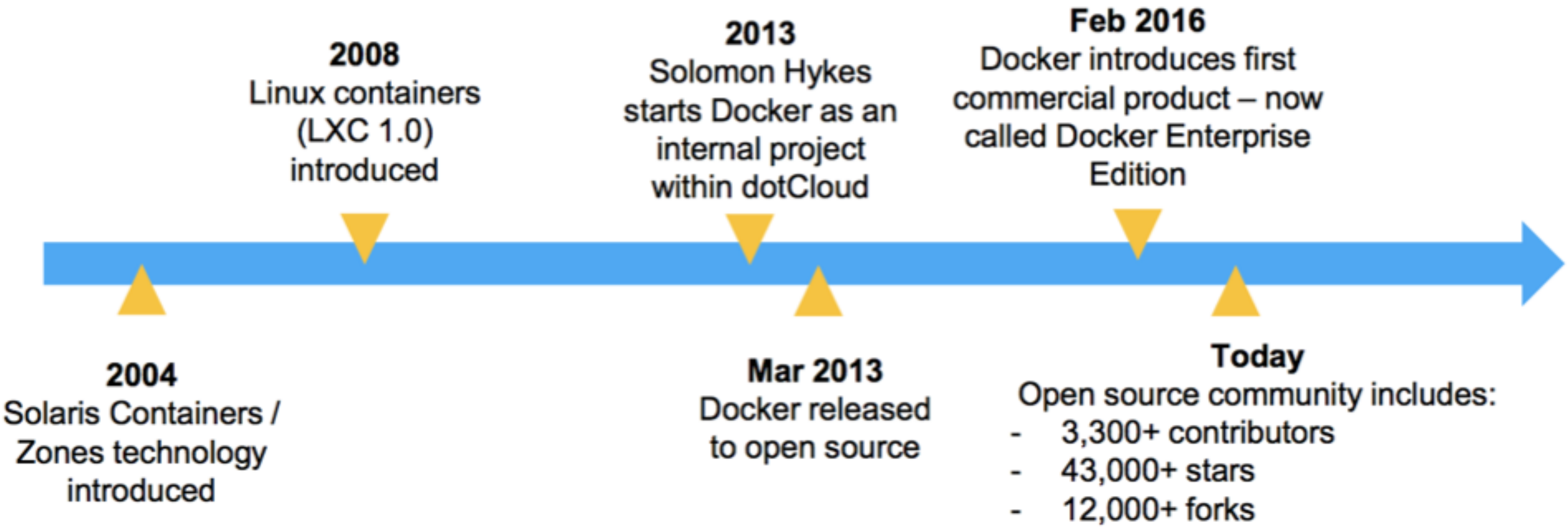
Portability

What is Docker?

“World’s leading **software containerization platform”**

Facilitate an **ecosystem** to **build, ship and run any applications, anywhere.**

History of Docker



2004
Solaris Containers / Zones technology introduced

2008
Linux containers (LXC 1.0) introduced

2013
Solomon Hykes starts Docker as an internal project within dotCloud

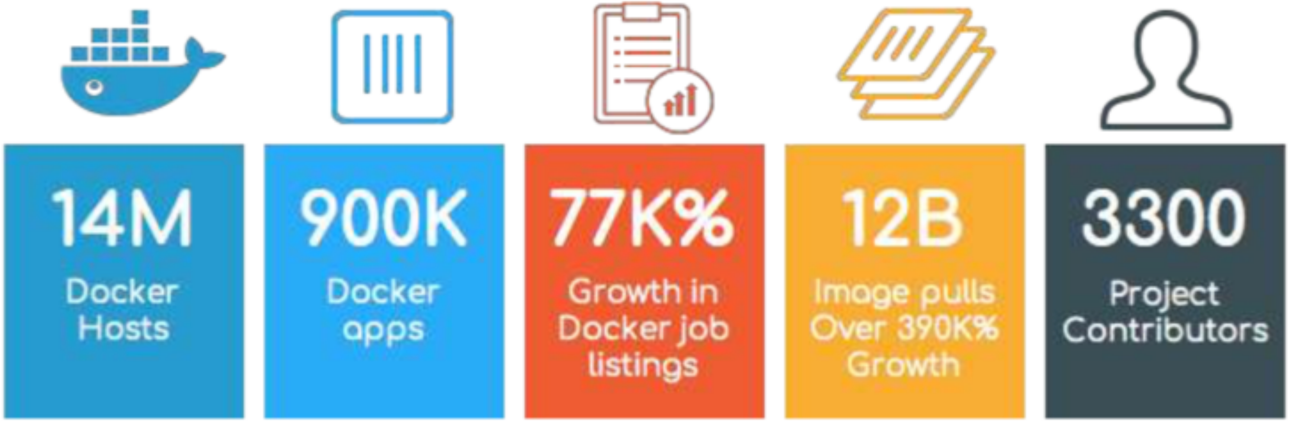
Mar 2013
Docker released to open source

Feb 2016
Docker introduces first commercial product – now called Docker Enterprise Edition

Today
Open source community includes:

- 3,300+ contributors
- 43,000+ stars
- 12,000+ forks

Incredible adoption in just 4 years



14M
Docker Hosts

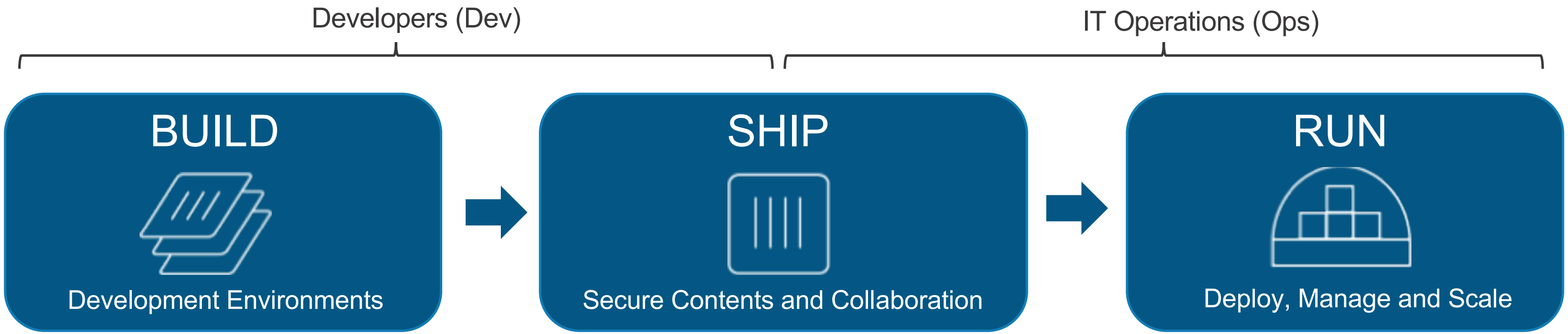
900K
Docker apps

77K%
Growth in Docker job listings

12B
Image pulls Over 390K% Growth

3300
Project Contributors

Why Docker?



Developers (Dev)

IT Operations (Ops)

BUILD

Development Environments

SHIP

Secure Contents and Collaboration

RUN

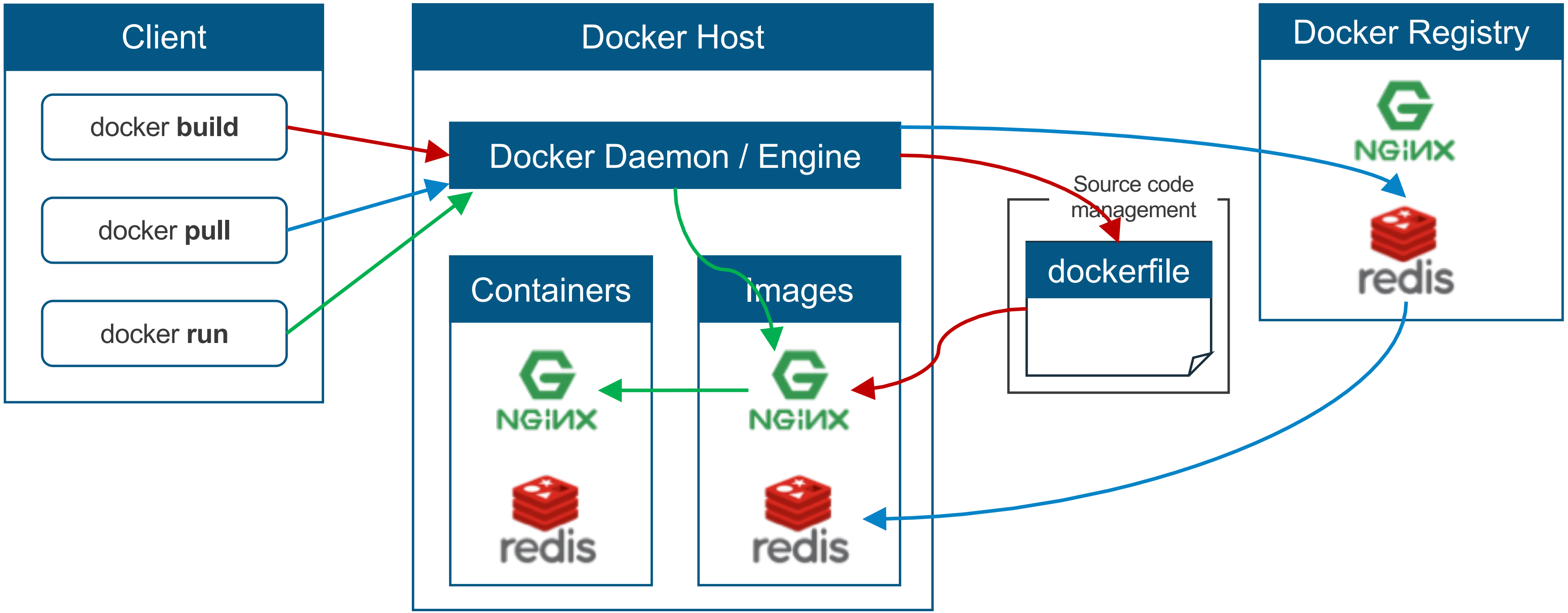
Deploy, Manage and Scale

- Speed up development environment setup
- Support use of different (best) technology stack

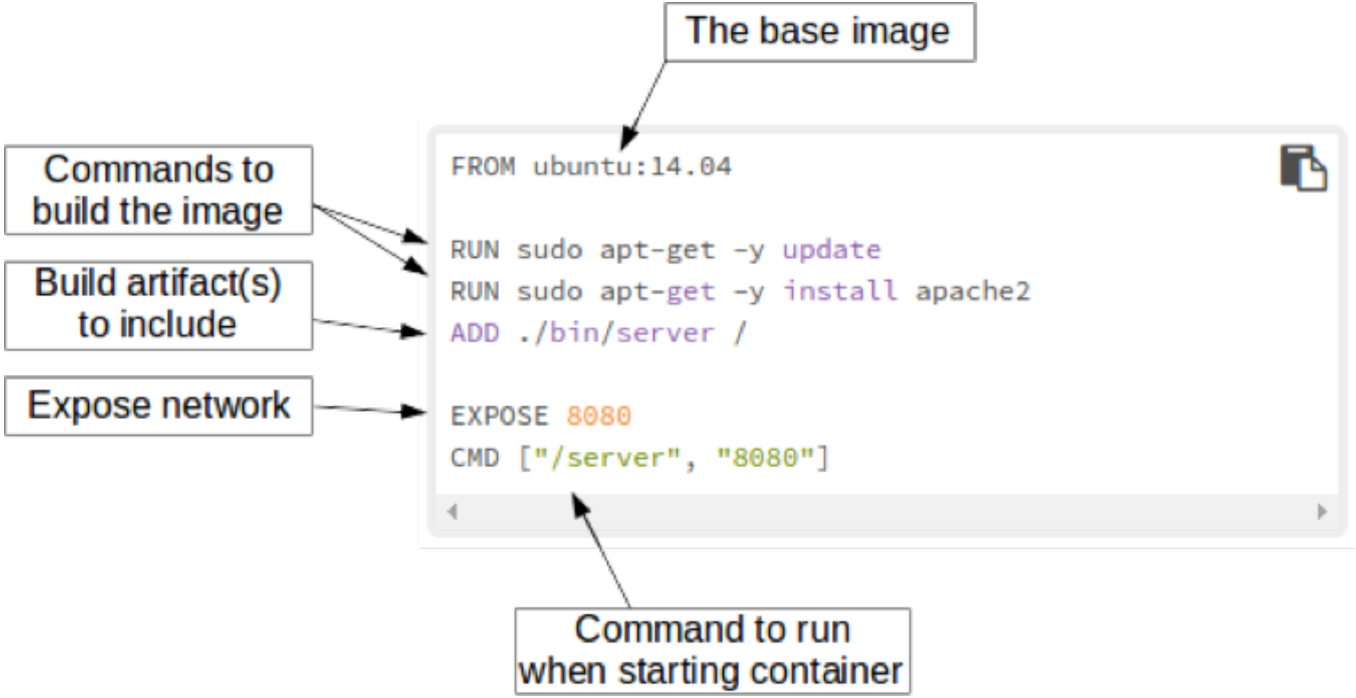
- Eliminate environment inconsistencies
- Share and distribute application become easier via Docker Hub

- Scale in seconds
- Allow rollback infrastructure changes that cause production issue quickly and easily

Core Components for Docker



Example of dockerfile



The base image

Commands to build the image

Build artifact(s) to include

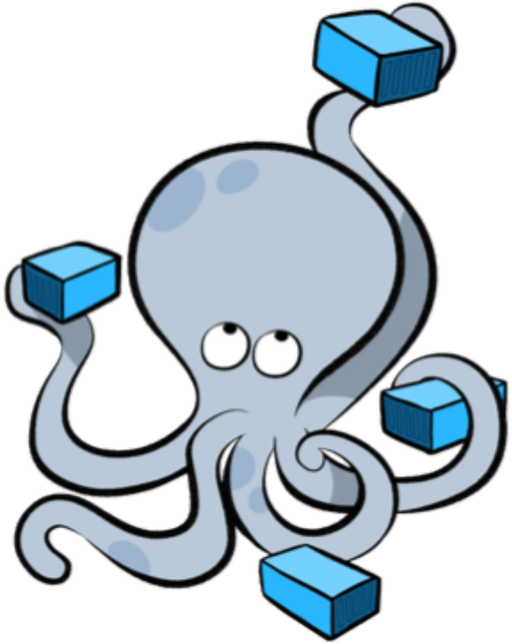
Expose network

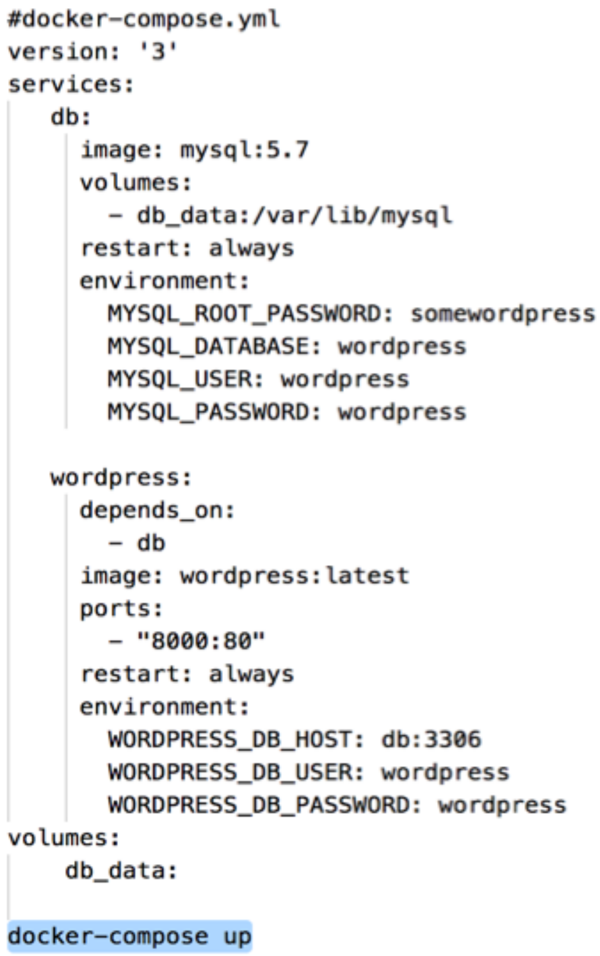
```
FROM ubuntu:14.04

RUN sudo apt-get -y update
RUN sudo apt-get -y install apache2
ADD ./bin/server /

EXPOSE 8080
CMD ["/server", "8080"]
```

Command to run when starting container

Docker Compose

Tool for defining and running
multi-container Docker applications.





```
#docker-compose.yml
version: '3'
services:
   db:
     image: mysql:5.7
     volumes:
       - db_data:/var/lib/mysql
     restart: always
     environment:
       MYSQL_ROOT_PASSWORD: somewordpress
       MYSQL_DATABASE: wordpress
       MYSQL_USER: wordpress
       MYSQL_PASSWORD: wordpress

   wordpress:
     depends_on:
       - db
     image: wordpress:latest
     ports:
       - "8000:80"
     restart: always
     environment:
       WORDPRESS_DB_HOST: db:3306
       WORDPRESS_DB_USER: wordpress
       WORDPRESS_DB_PASSWORD: wordpress
volumes:
    db_data:

docker-compose up
```

Docker Registry

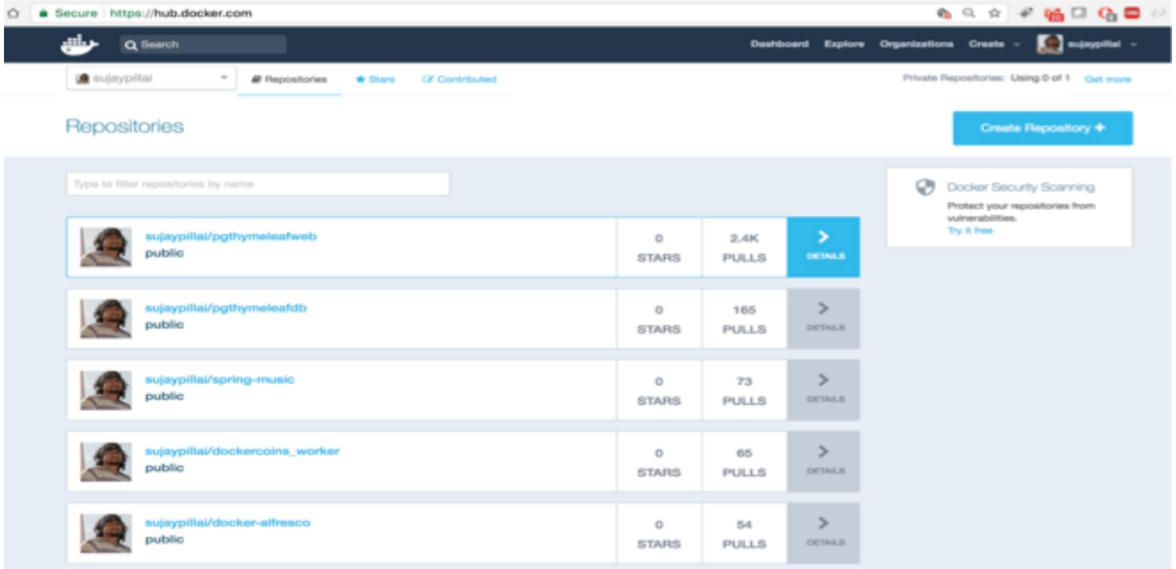
The Registry is a stateless, highly scalable server side application that stores and lets you distribute Docker images.

Why use it:

- Tightly control where your images are being stored
- Fully own your images distribution pipeline
- Integrate image storage and distribution tightly into your in-house development workflow

Docker Hub

A cloud-based registry service which allows you to link to code repositories, build your images and test them, stores manually pushed images, and links to Docker Cloud so you can deploy images to your hosts.



Private Repositories: Using 0 of 1

Repositories

Docker Security Scanning
Protect your repositories from vulnerabilities.

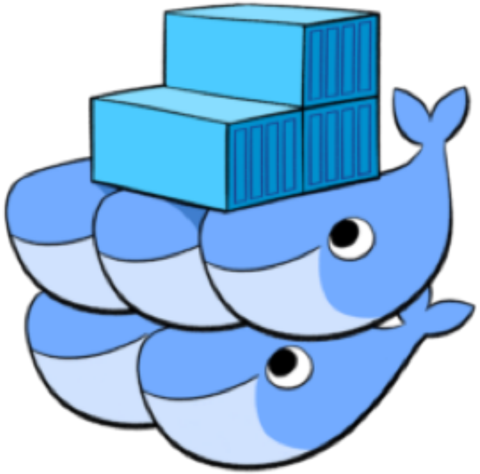
sujaypillai/pgthymeleafweb public	0 STARS	2.4K PULLS
sujaypillai/pgthymeleafdb public	0 STARS	165 PULLS
sujaypillai/spring-music public	0 STARS	73 PULLS
sujaypillai/dockercoins_worker public	0 STARS	65 PULLS
sujaypillai/docker-alfresco public	0 STARS	54 PULLS

Major Features:

- Image Repositories
- Automate Builds
- Webhooks
- Organizations
- Github & Bitbucket integration

Docker Swarm

A swarm is a group of machines that are running Docker and joined into a cluster.



Features:

- Cluster management integrated with Docker Engine
- Decentralized design
- Declarative service model
- Scaling
- Desired state reconciliation
- Multi-host networking
- Service discovery
- Load balancing
- Secure by default
- Rolling updates

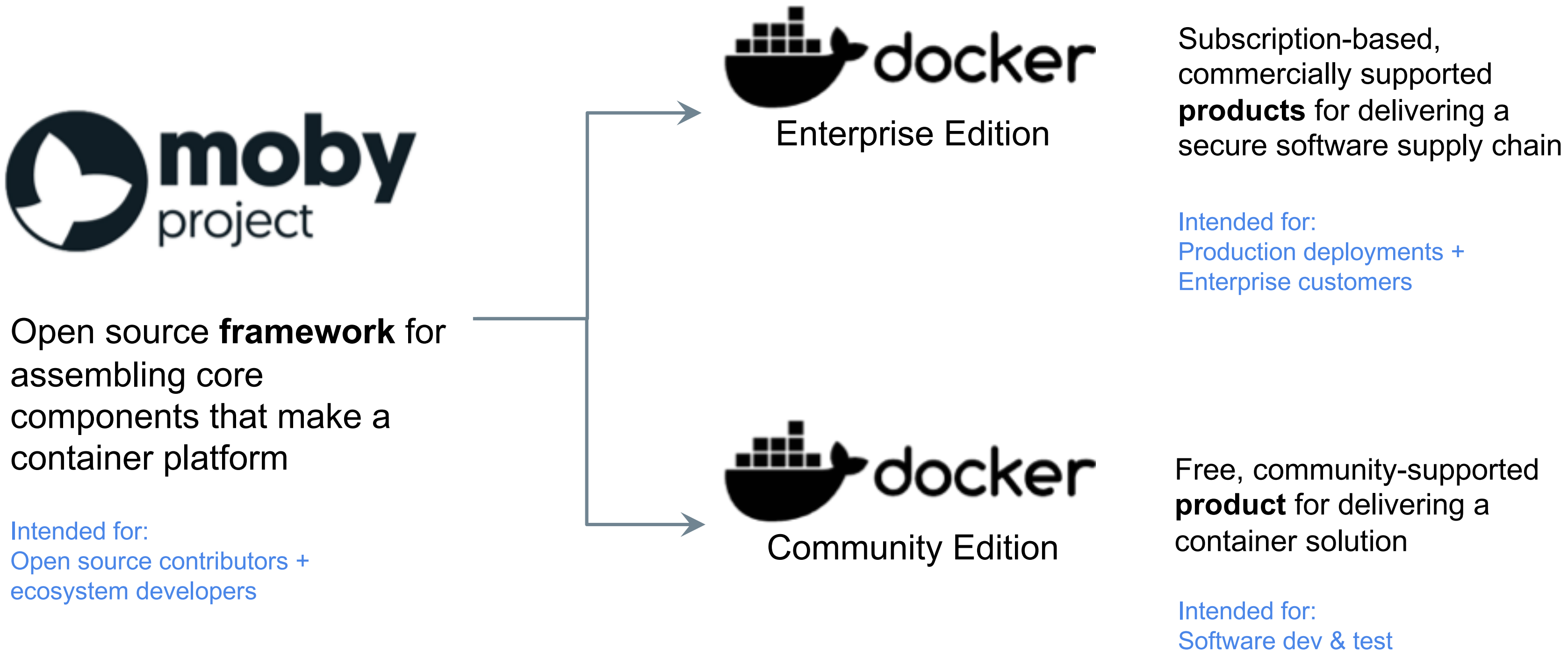
Docker Machine

Tool for provisioning and managing your Dockerized hosts

You can use Docker Machine to:

- Install and run Docker on Mac or Windows
- Provision and manage multiple remote Docker hosts
- Provision Swarm clusters

The Docker Family Tree



moby project

Open source **framework** for assembling core components that make a container platform

Intended for:
Open source contributors +
ecosystem developers

docker
Enterprise Edition

Subscription-based, commercially supported **products** for delivering a secure software supply chain

Intended for:
Production deployments +
Enterprise customers

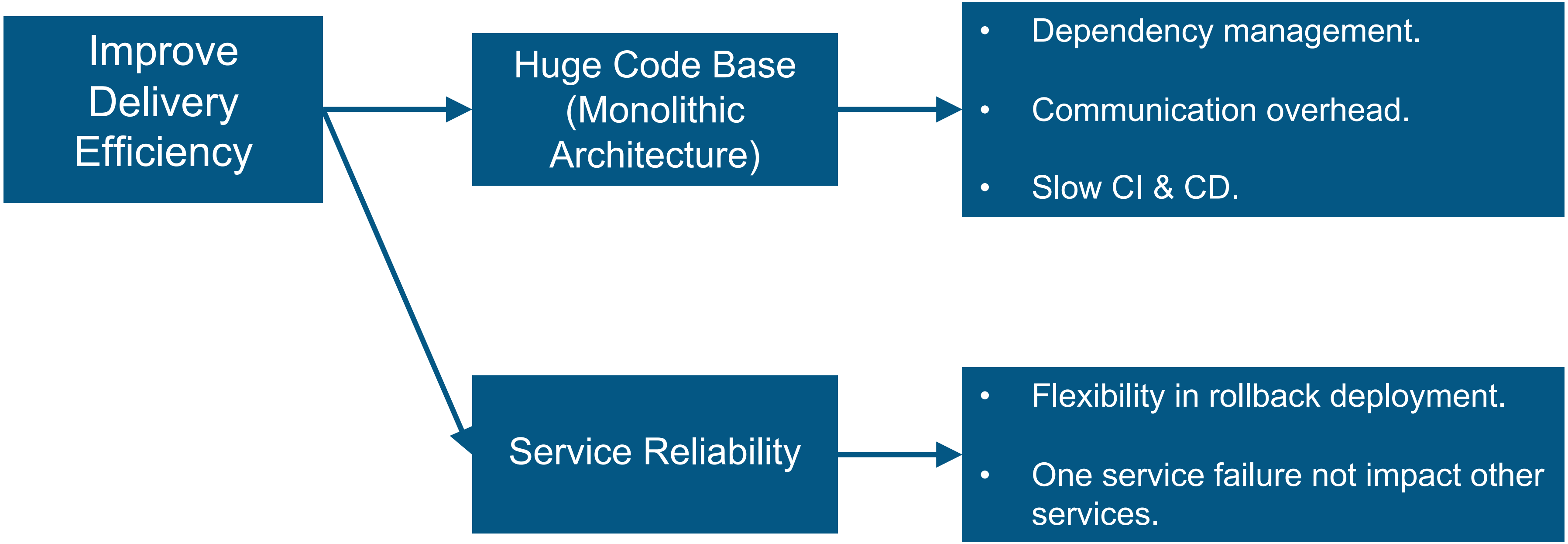
docker
Community Edition

Free, community-supported **product** for delivering a container solution

Intended for:
Software dev & test

Docker in Seek Asia

Why we make a move?



Improve Delivery Efficiency

Huge Code Base (Monolithic Architecture)

- Dependency management.
- Communication overhead.
- Slow CI & CD.

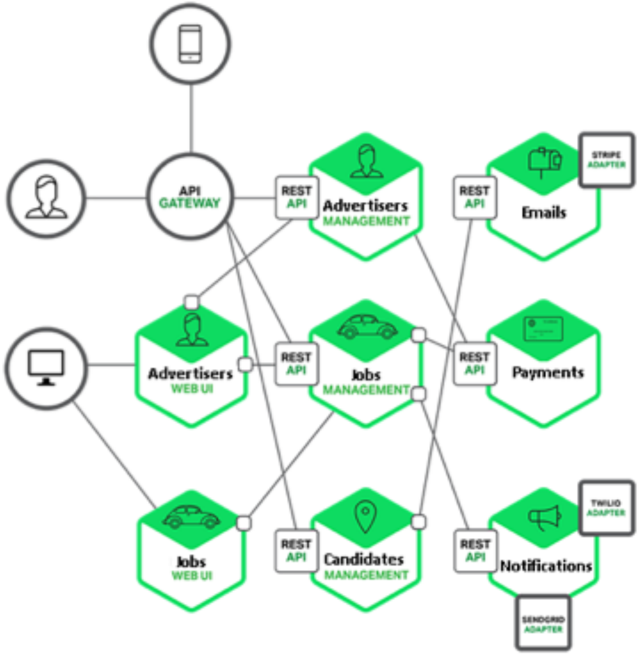
Service Reliability

- Flexibility in rollback deployment.
- One service failure not impact other services.

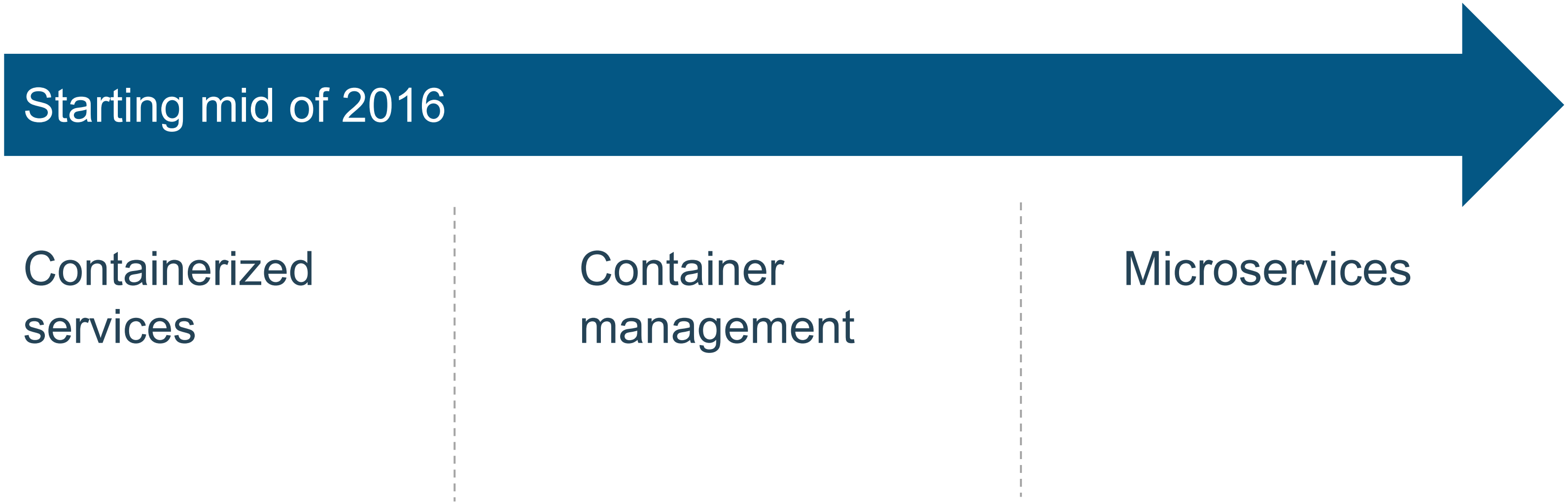
Microservices + Docker as the Solution

Moving to **Microservices Architecture** is the solution, which emphasize build decoupled services in small and independently.

However, we do not wish to maintain large amount of VMs which cause the overhead. **Docker** come in to the picture, which solve the environment consistency as well.



Technology Adoption Strategy



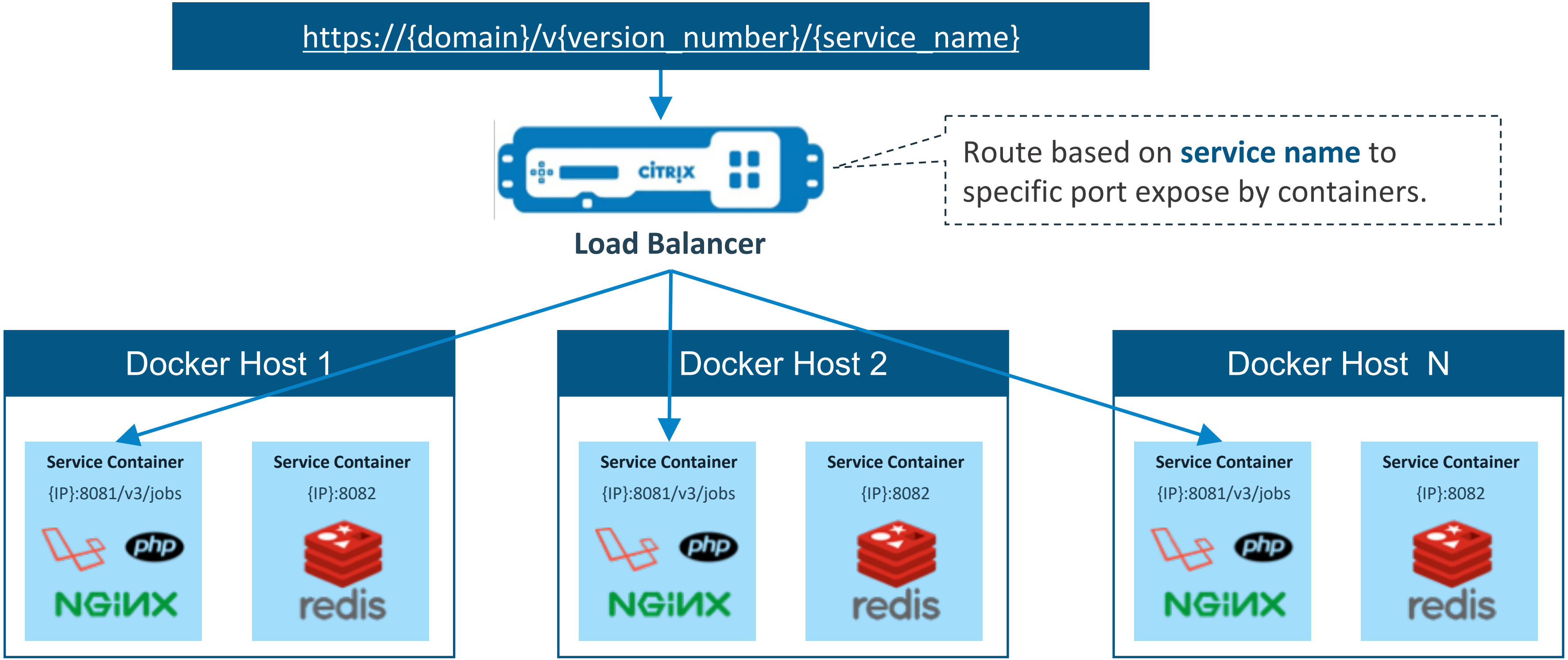
Starting mid of 2016

Containerized services

Container management

Microservices

The Big Picture



https://{domain}/v{version_number}/{service_name}

Load Balancer

Route based on **service name** to specific port expose by containers.

Docker Host 1

Service Container
{IP}:8081/v3/jobs

Service Container
{IP}:8082

Docker Host 2

Service Container
{IP}:8081/v3/jobs

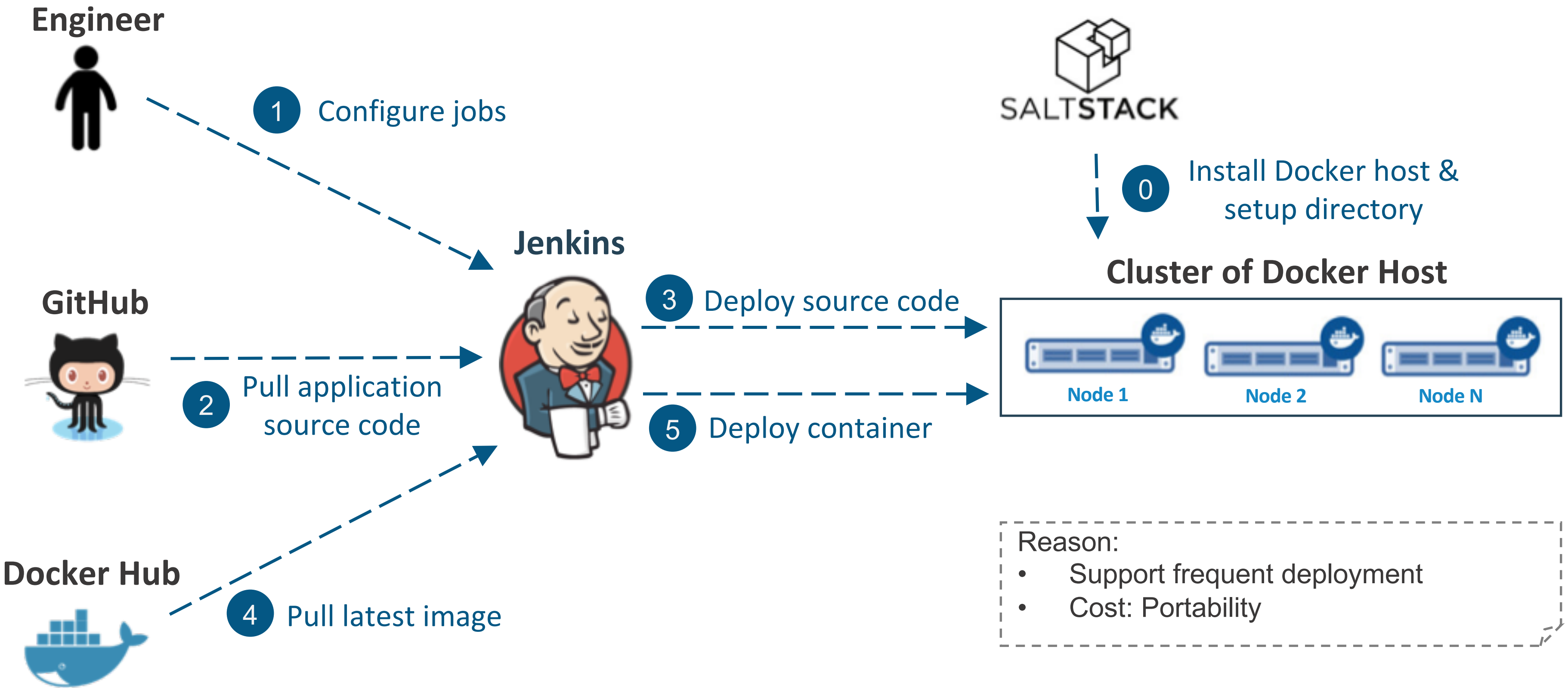
Service Container
{IP}:8082

Docker Host N

Service Container
{IP}:8081/v3/jobs

Service Container
{IP}:8082

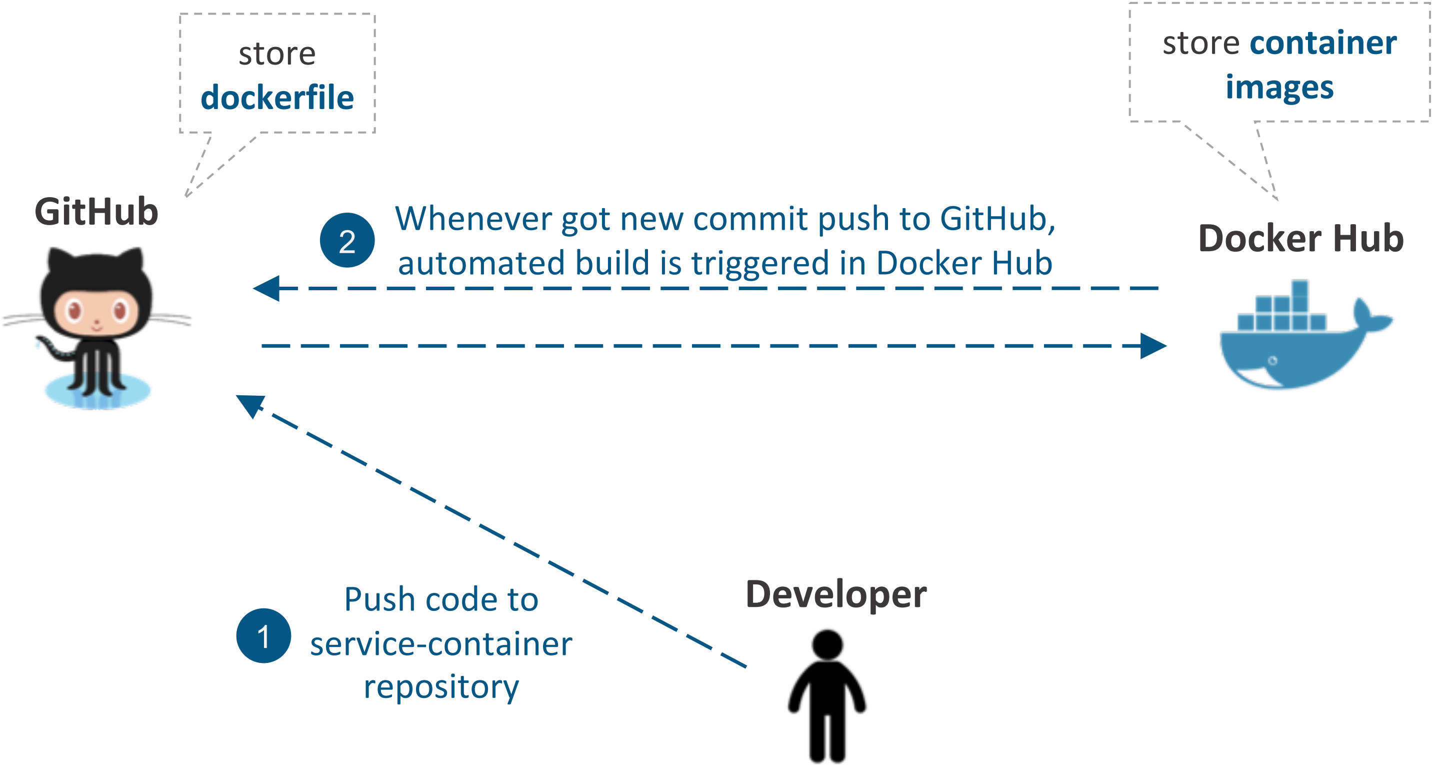
Production Deployment



Reason:

- Support frequent deployment
- Cost: Portability

Image built and store in Docker Hub



store **dockerfile**

store **container images**

GitHub

Docker Hub

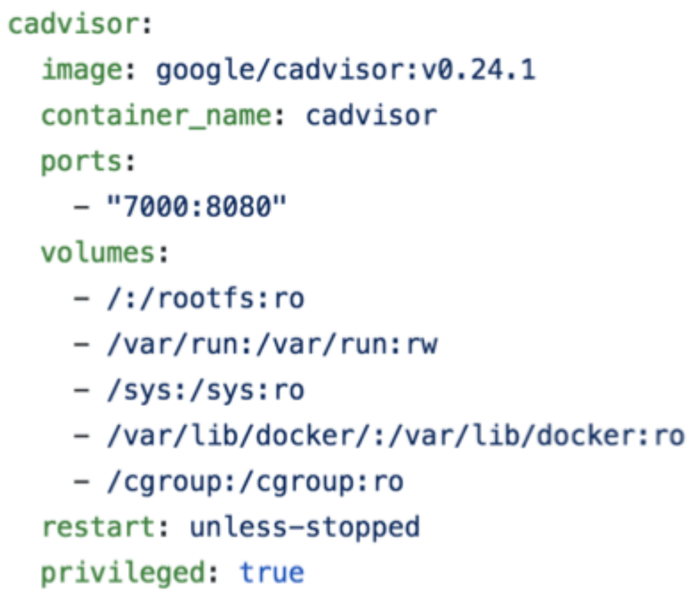
Developer

1 Push code to service-container repository

2 Whenever got new commit push to GitHub, automated build is triggered in Docker Hub

Use Docker Compose to run it

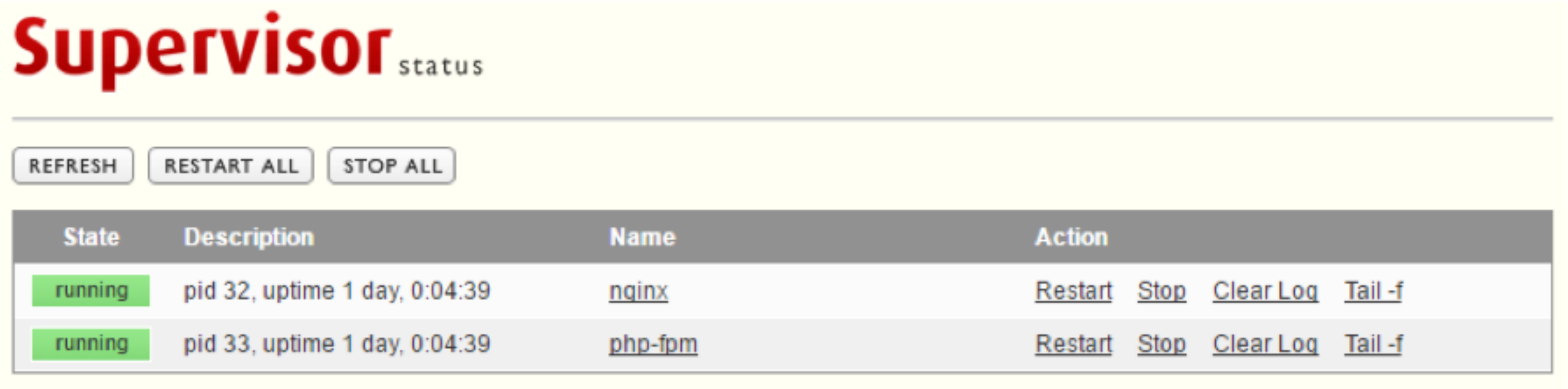
docker-compose.yml



```
cadvisor:
  image: google/cadvisor:v0.24.1
  container_name: cadvisor
  ports:
    - "7000:8080"
  volumes:
    - /:/rootfs:ro
    - /var/run:/var/run:rw
    - /sys:/sys:ro
    - /var/lib/docker/:/var/lib/docker:ro
    - /cgroup:/cgroup:ro
  restart: unless-stopped
  privileged: true
```

docker-compose up -d

Use Supervisor to manage Services

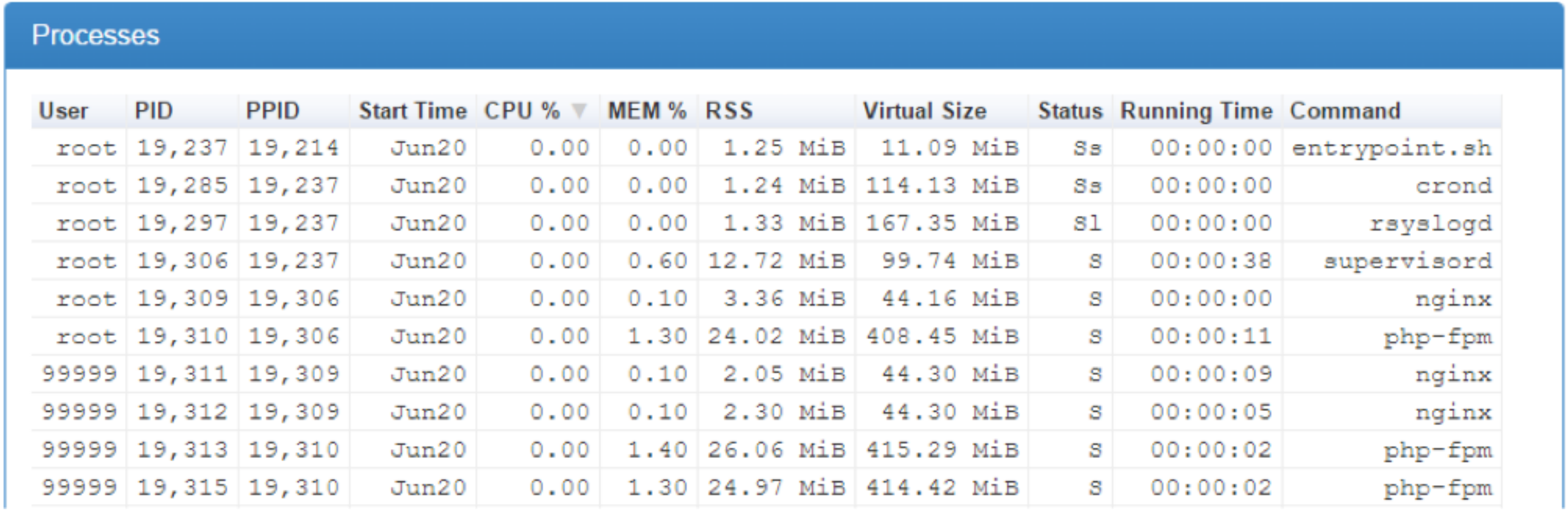


State	Description	Name	Action
running	pid 32, uptime 1 day, 0:04:39	nginx	Restart Stop Clear Log Tail -f
running	pid 33, uptime 1 day, 0:04:39	php-fpm	Restart Stop Clear Log Tail -f

- 1 microservice 1 container.
- Reduce complexity in managing multiple container during adoption period.

Use cAdvisor to monitor Docker

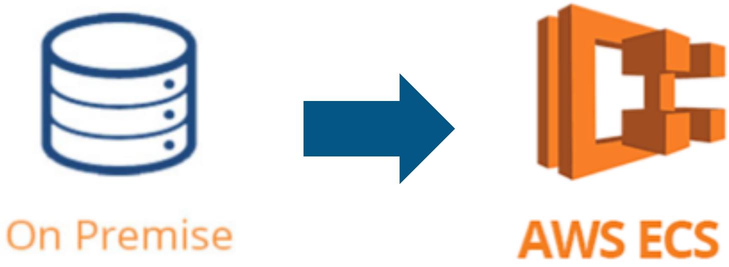
- Is a Docker container too.
- Will be running in each Docker Host.
- To monitor all running process inside each container in that Docker Host.



Processes

User	PID	PPID	Start Time	CPU % ▼	MEM %	RSS	Virtual Size	Status	Running Time	Command
root	19,237	19,214	Jun20	0.00	0.00	1.25 MiB	11.09 MiB	Ss	00:00:00	entrypoint.sh
root	19,285	19,237	Jun20	0.00	0.00	1.24 MiB	114.13 MiB	Ss	00:00:00	crond
root	19,297	19,237	Jun20	0.00	0.00	1.33 MiB	167.35 MiB	Sl	00:00:00	rsyslogd
root	19,306	19,237	Jun20	0.00	0.60	12.72 MiB	99.74 MiB	S	00:00:38	supervisord
root	19,309	19,306	Jun20	0.00	0.10	3.36 MiB	44.16 MiB	S	00:00:00	nginx
root	19,310	19,306	Jun20	0.00	1.30	24.02 MiB	408.45 MiB	S	00:00:11	php-fpm
99999	19,311	19,309	Jun20	0.00	0.10	2.05 MiB	44.30 MiB	S	00:00:09	nginx
99999	19,312	19,309	Jun20	0.00	0.10	2.30 MiB	44.30 MiB	S	00:00:05	nginx
99999	19,313	19,310	Jun20	0.00	1.40	26.06 MiB	415.29 MiB	S	00:00:02	php-fpm
99999	19,315	19,310	Jun20	0.00	1.30	24.97 MiB	414.42 MiB	S	00:00:02	php-fpm

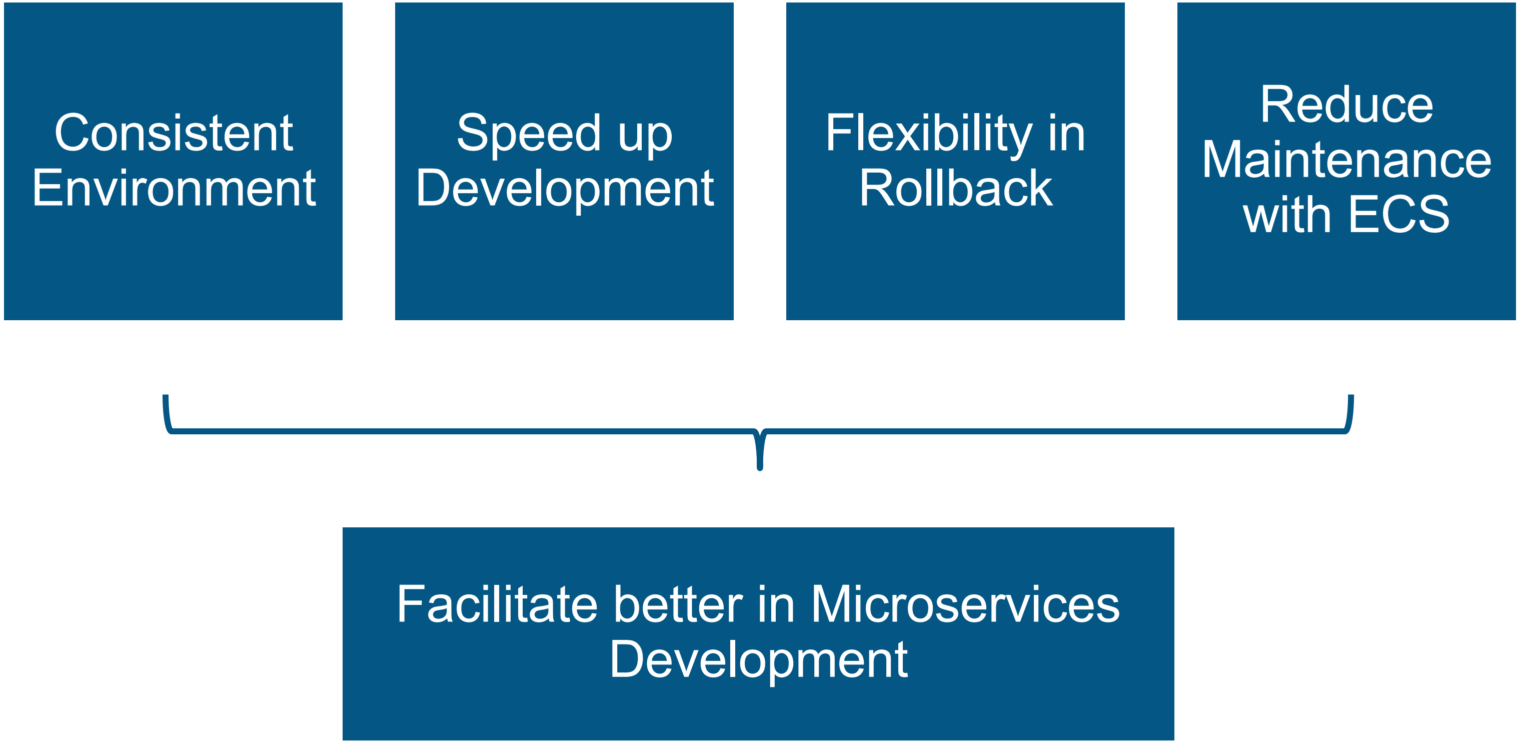
Better Container Management in Cloud



Use of AWS Elastic Container Service

- Pets -> Cattle
- Auto Scaling
- Security
- Save developer time

So, what we gain?



Consistent Environment

Speed up Development

Flexibility in Rollback

Reduce Maintenance with ECS

Facilitate better in Microservices Development

Docker at Jabil

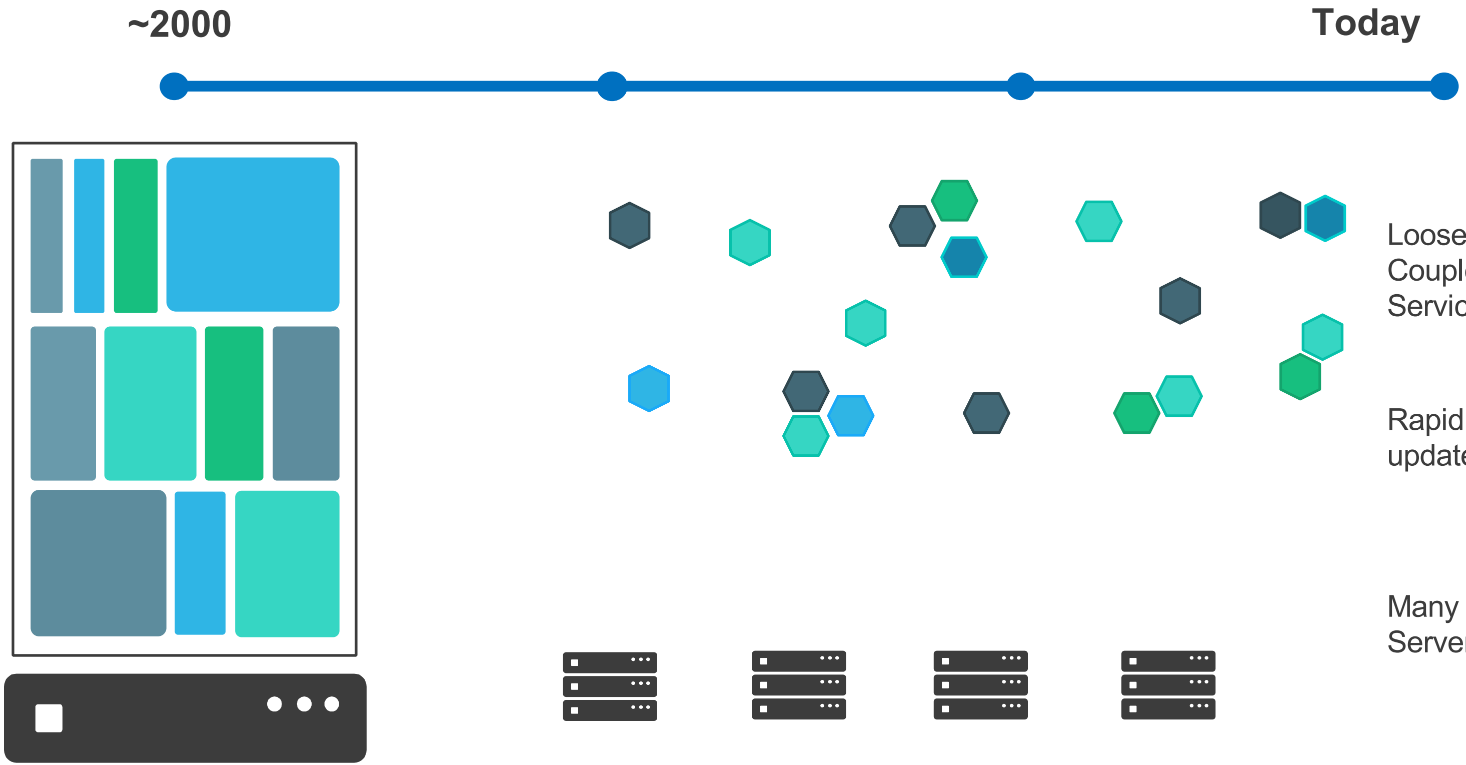
Our journey from CE to EE

Movement in the Cloud

- Migrate workloads to cloud
- Portability across environments
- Want to avoid cloud vendor lock-in

80% of Jabil workloads will be cloud-based in 3 years
and
25% of all IT Spend

Applications are transforming



Monolithic

Slow changing

Big Servers

Loosely Coupled Services

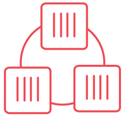
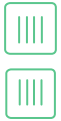
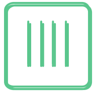
Rapidly updated

Many Small Servers or devices

The Myth of Bi-Modal IT

	MICROSERVICES	TRADITIONAL APPS
Cloud or New Infrastructure	You are either here..	▨
Old Infrastructure	▨	…or here

Enabling a Journey

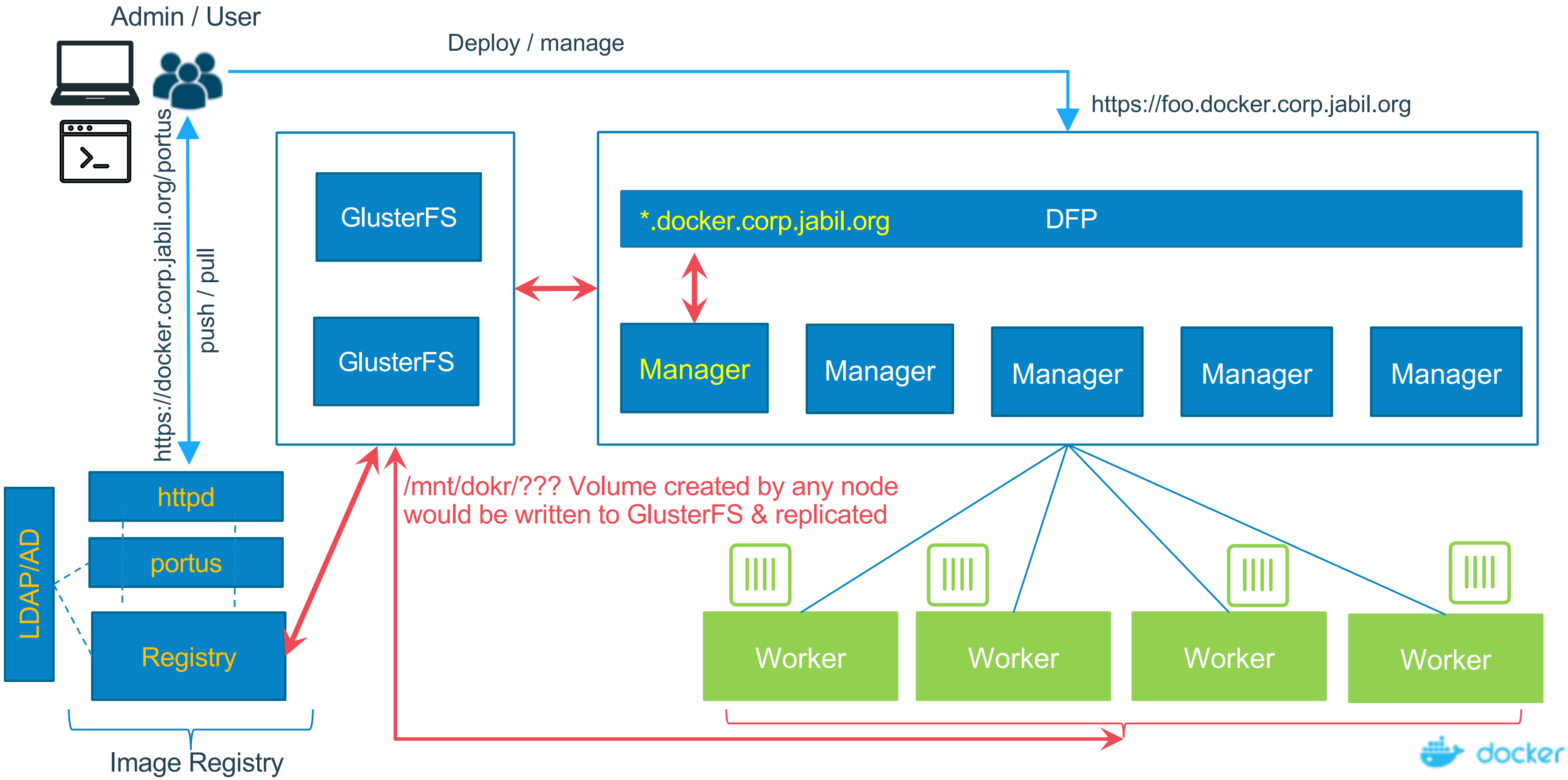
	MICROSERVICES	AGILE TRADITIONAL APPS	TRADITIONAL APPS
Cloud or New Infrastructure			
Old Infrastructure			

…that is past AND future proof

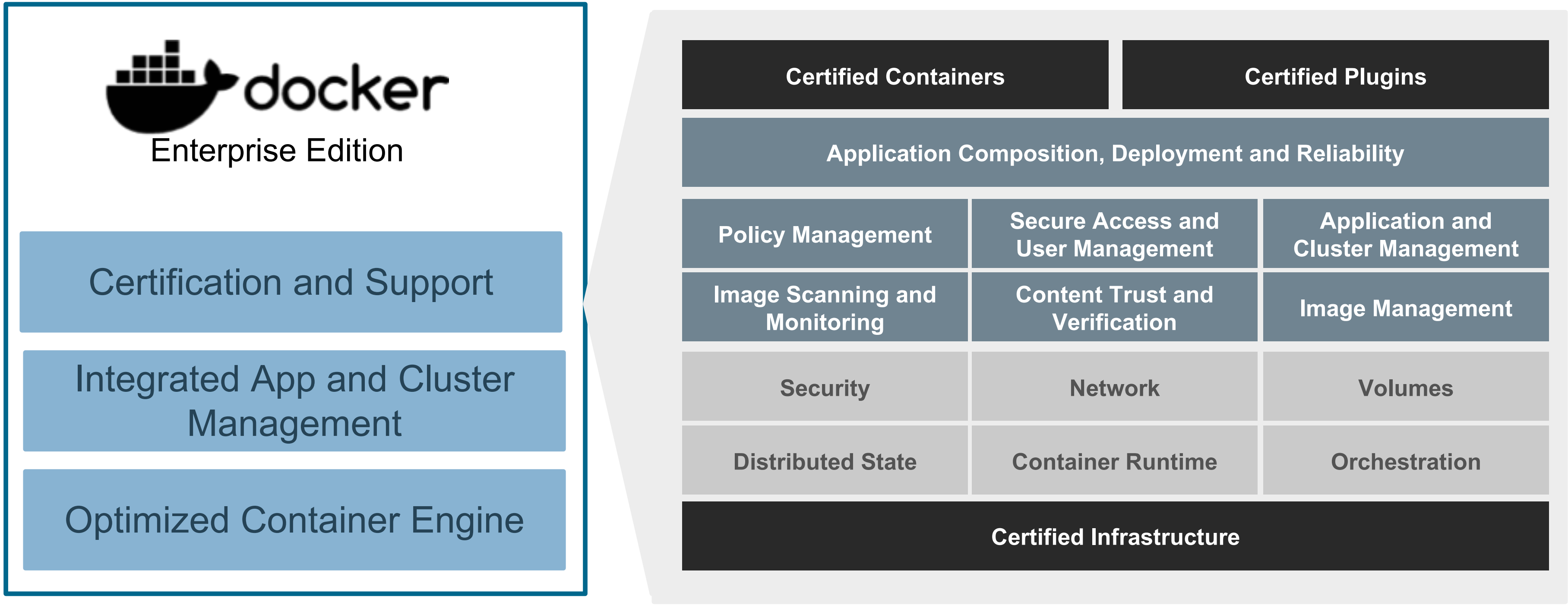
What we learned with CE

- Started with CE 1.13 and upgraded to 17.06.0-ce
- 9 node cluster (5 manager + 4 worker nodes)
- DFP – HAProxy + custom logic provides on-demand reconfiguration
- GlusterFS for storage (2 clustered servers)
- Standalone registry server & Portus as web frontend
- Portainer – Management solution for Docker
- Prometheus - Monitoring

Docker CE Architecture



Docker Enterprise Edition Capabilities



docker

Enterprise Edition

Certification and Support

Integrated App and Cluster Management

Optimized Container Engine

Certified Containers | Certified Plugins

Application Composition, Deployment and Reliability

Policy Management | Secure Access and User Management | Application and Cluster Management

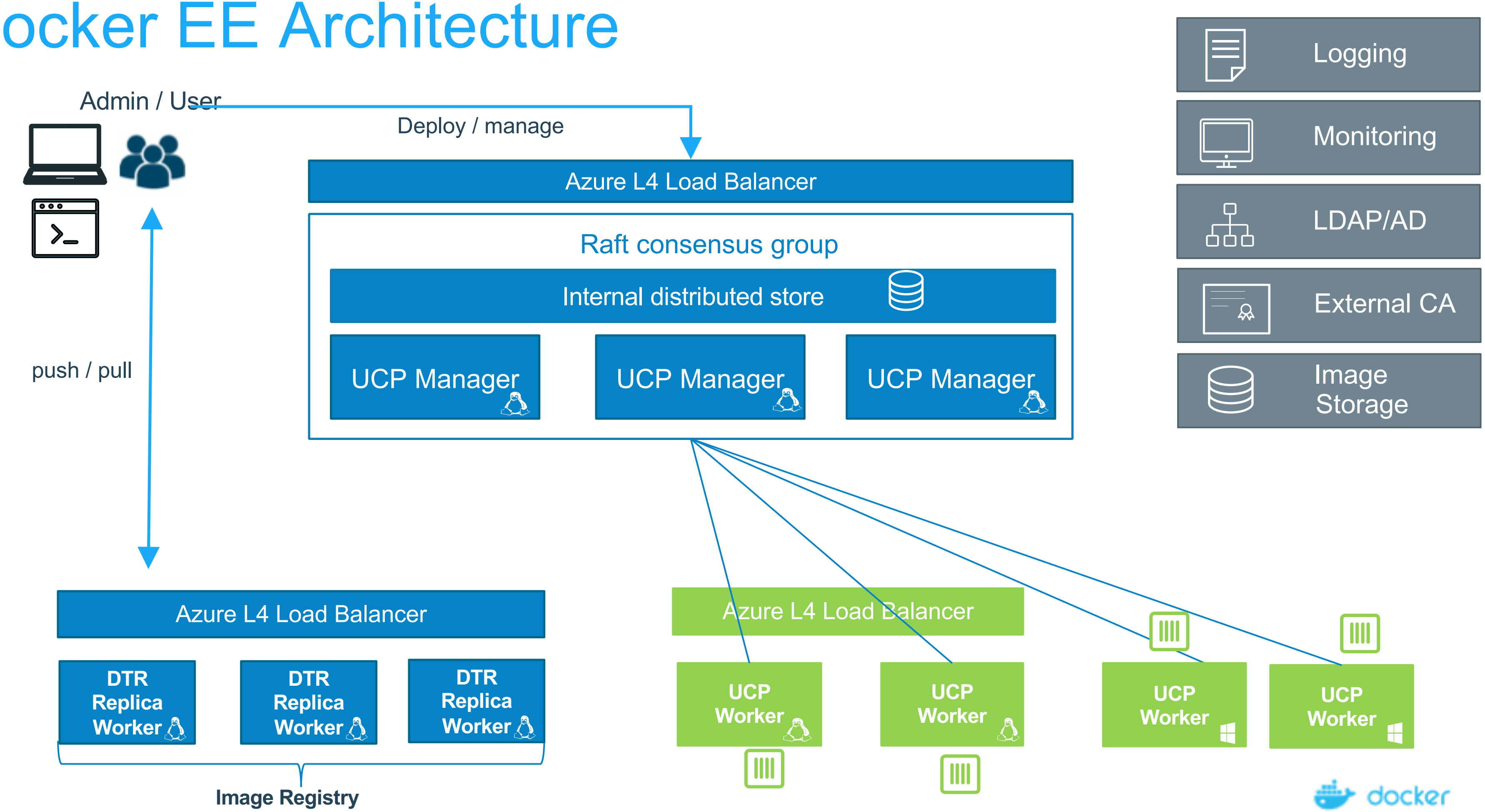
Image Scanning and Monitoring | Content Trust and Verification | Image Management

Security | Network | Volumes

Distributed State | Container Runtime | Orchestration

Certified Infrastructure

Docker EE Architecture



docker