Docker & Devops

BAOHUA YANG

baohyang@cn.ibm.com

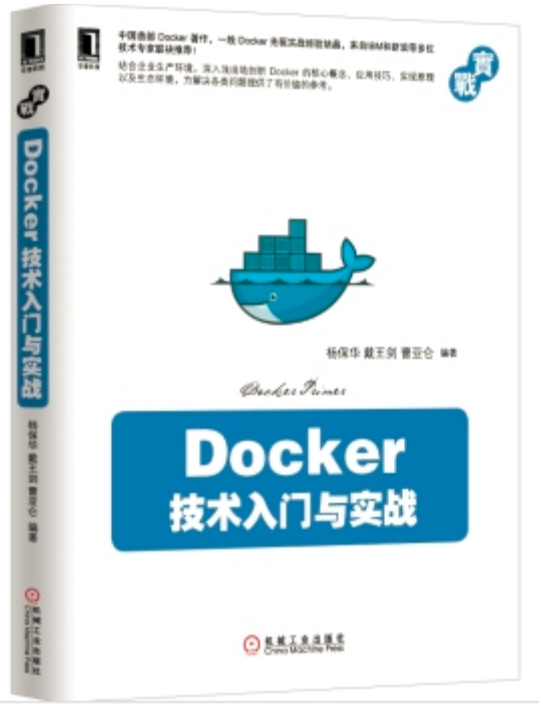
2015-05-23

Docker 从入门到实践

https://github.com/yeasy/docker_practice

Open-Source since August, 2014

Contributors:	20+
Commits:	260+
Stars:	1100+
Readers:	47, 000+
Views:	520, 000+



Who am I & Why I’ m Here

Baohua Yang

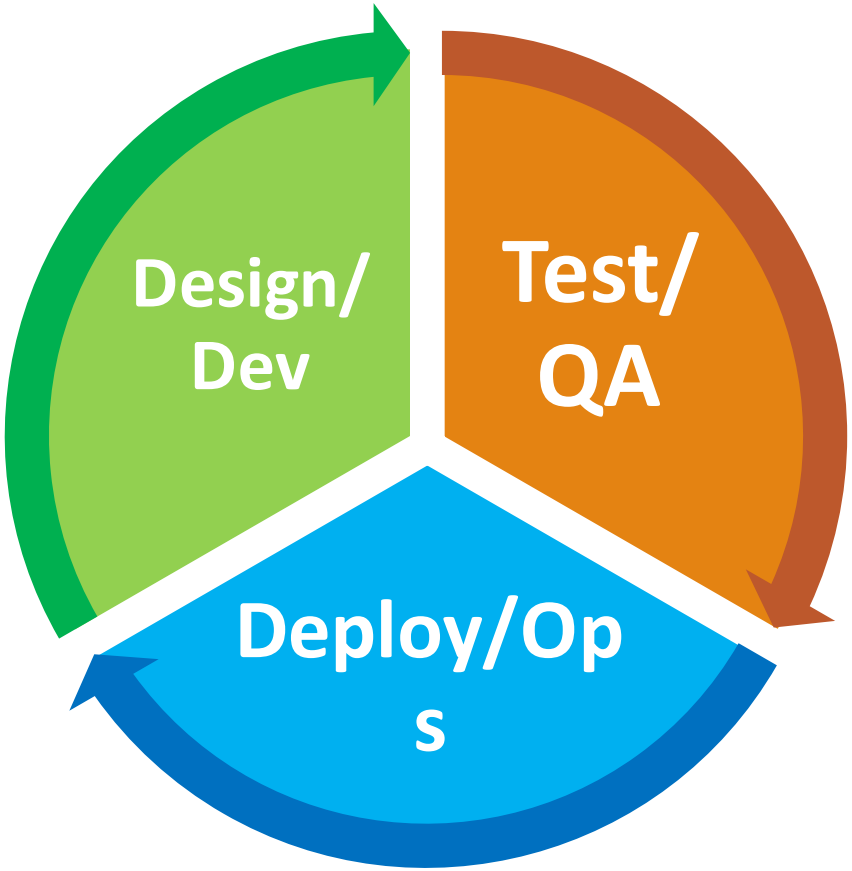
- Lead the research on Cloud Networking
- Design & write code for Cloud solutions & products
- Focus on key technologies in IaaS and PaaS
  - OpenStack
  - SDN
  - Docker
  - CF
  - IoT

Love emerging technology and open source!

Believe PaaS would be the next step!

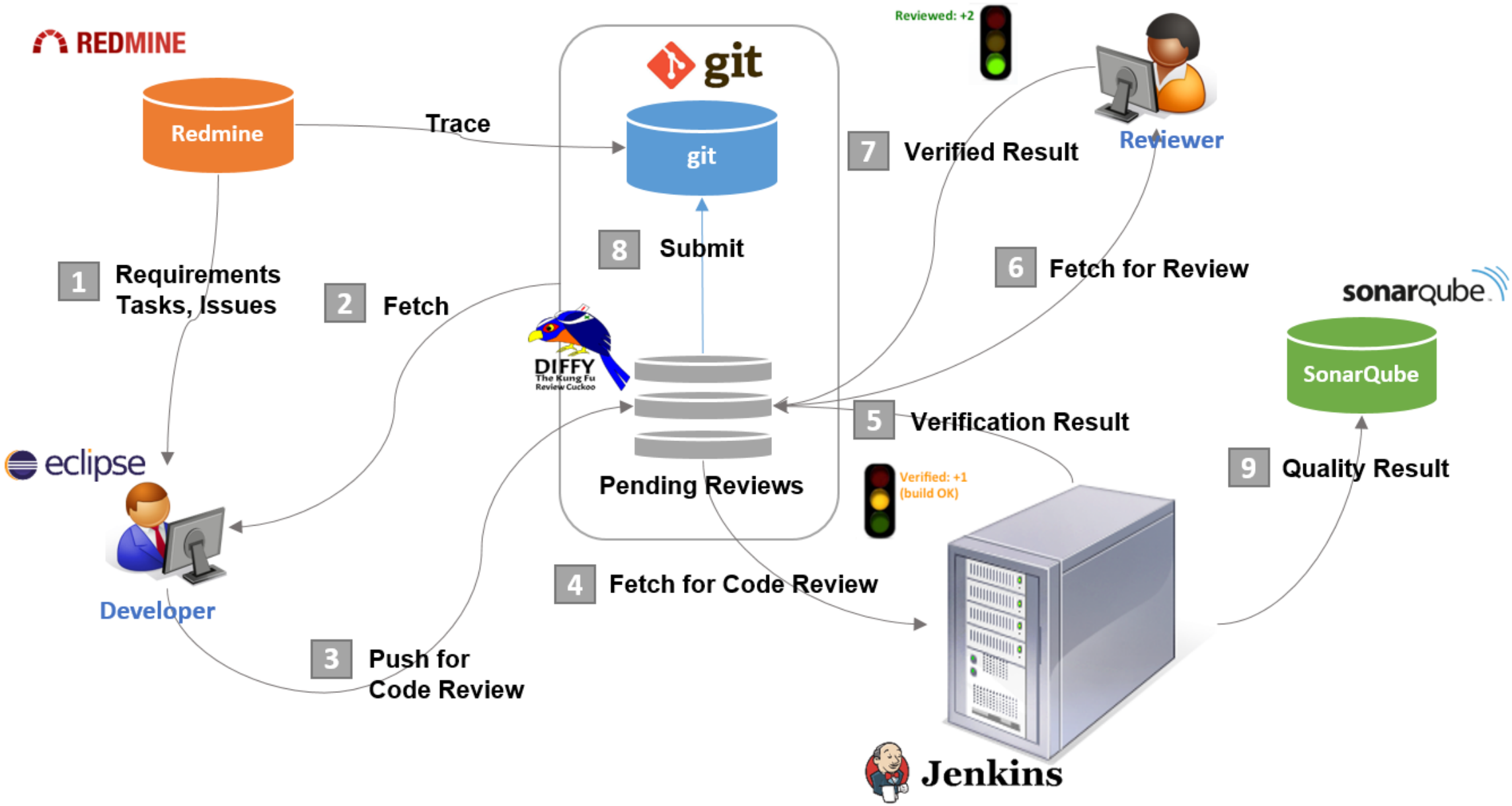
We are HIRING! baohyang@cn.ibm.com

Flexible



Repeatable
Confident

Stable
Standard
Automated

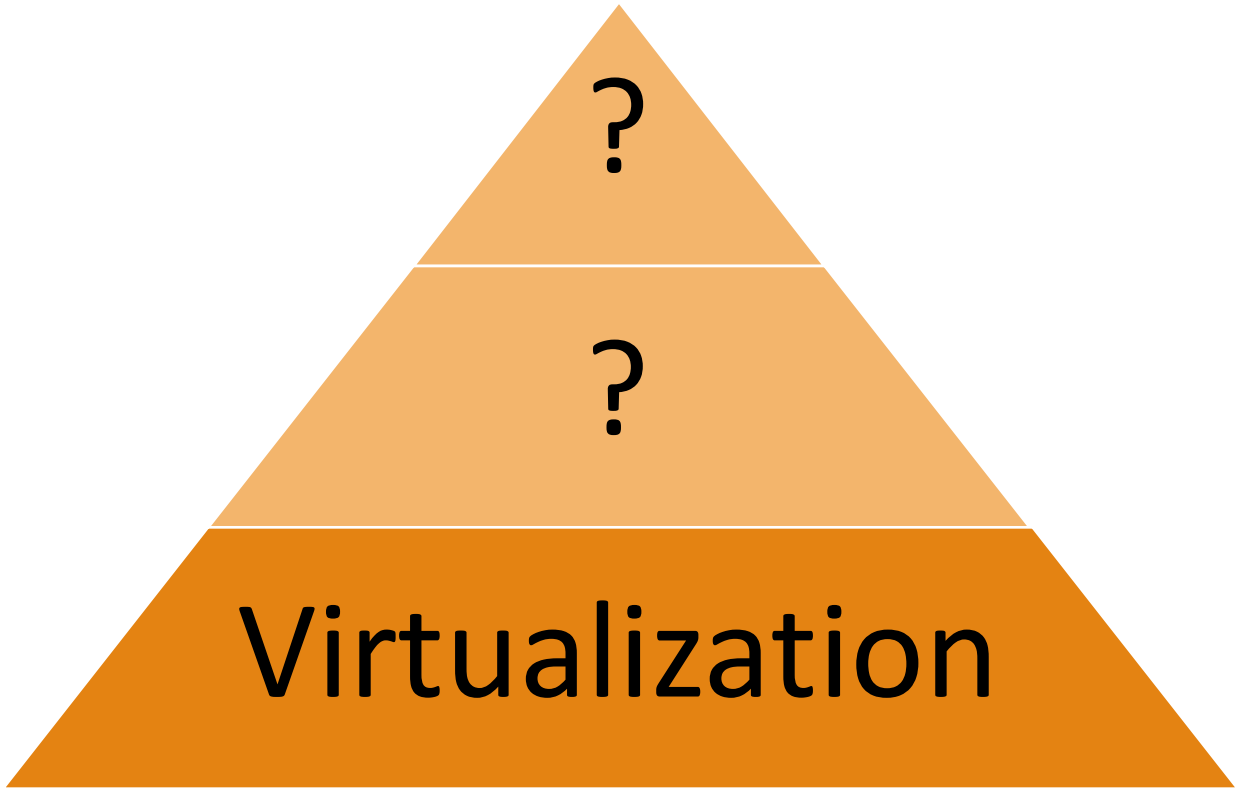

REDMINE
Redmine
Trace
git
git
Reviewed: +2
Reviewer
7 Verified Result
8 Submit
6 Fetch for Review
1 Requirements
Tasks, Issues
2 Fetch
DIFFY
The Kung Fu
Review Cuckoo
sonarqube
SonarQube
5 Verification Result
9 Quality Result
eclipse
Verified: +1
(build OK)
Pending Reviews
4 Fetch for Code Review
Developer
3 Push for
Code Review
Jenkins


sumologic
Jenkins
SUBVERSION
WebSphere
DEVELOPER'S JOURNAL
Solano Labs
REDMINE
flexible project management
Squid
eclipse
splunk>
DEV OPS
VAGRANT
Apache
SOFTWARE FOUNDATION
New Relic
IntelliJIDEA
SNORT
Ganglia
IBM. Rational. software
now
git
CFEngine
jazz
APACHE ANT
PAGERDUTY
logstash
Nagios
Maven
puppet labs
ZABBIX
docker
ANSIBLE
CHEF
Capistrano
SALTSTACK
mercurial
ScriptRock

The Matrix From Hell

	Development VM	QA Server	Single Prod Server	Onsite Cluster	Public Cloud	Contributor's laptop	Customer Servers
Static website	?	?	?	?	?	?	?
Web frontend	?	?	?	?	?	?	?
Background workers	?	?	?	?	?	?	?
User DB	?	?	?	?	?	?	?
Analytics DB	?	?	?	?	?	?	?
Queue	?	?	?	?	?	?	?

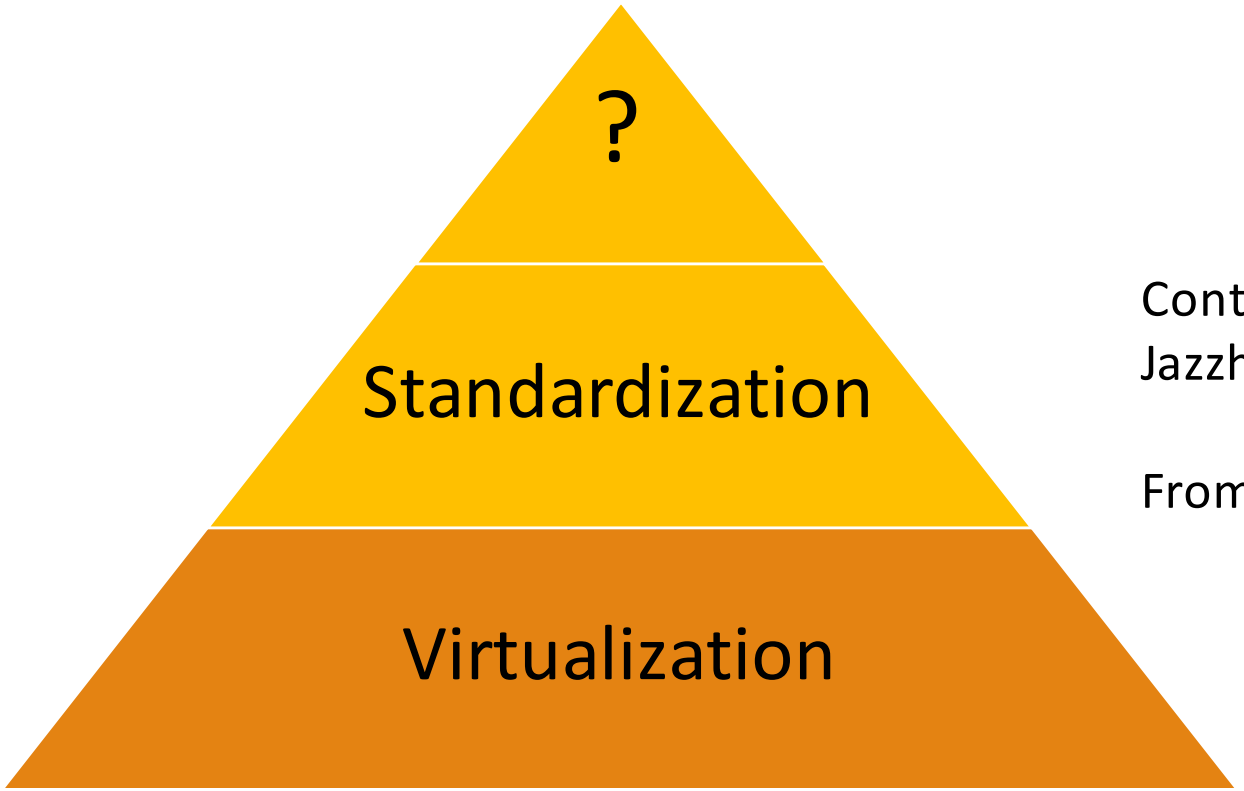
Devops Evolution



Efficient
Consistent
Repeatable
High-quality
Cost-saving
Portable
Agile
…

Next Generation of Devops?

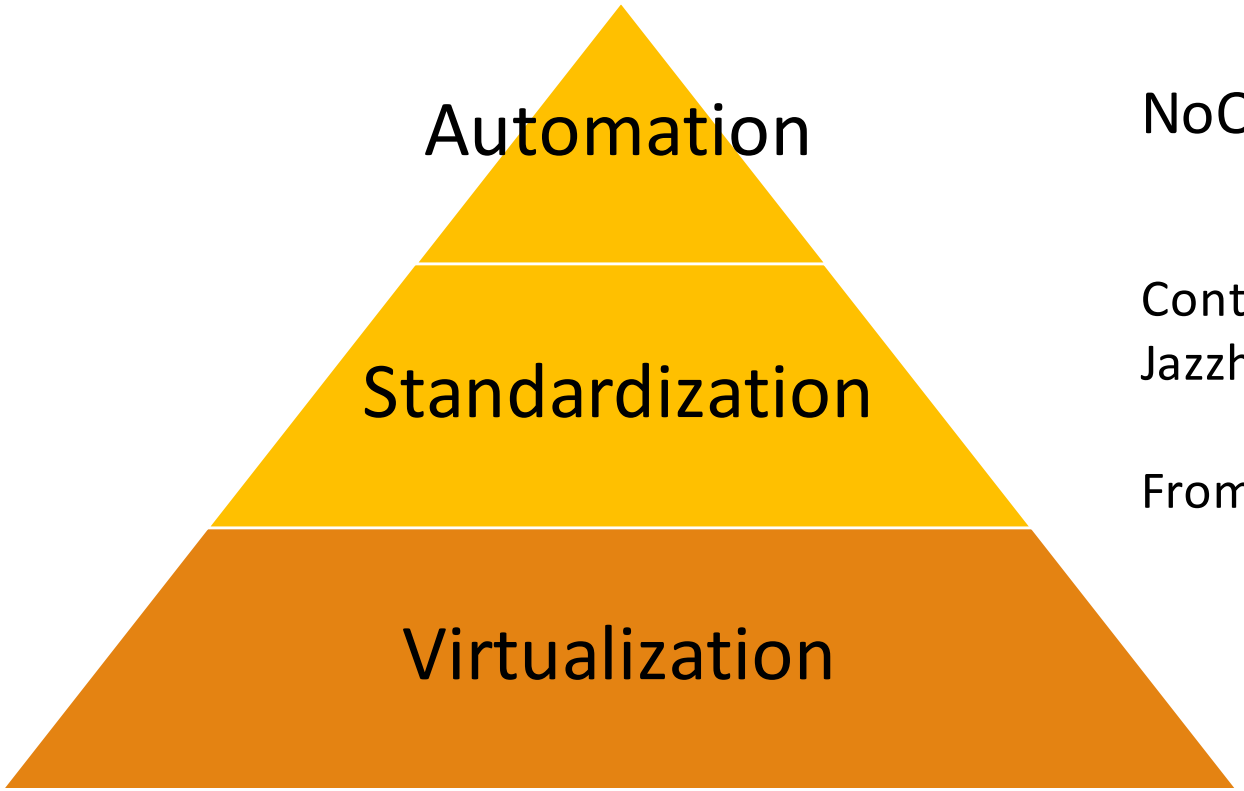
Devops evolution



ContainerOps, CloudOps, ITOps…
Jazzhub, Travis-ci, drone.io, gitlab…

From *moving code* to *moving service*

Devops evolution

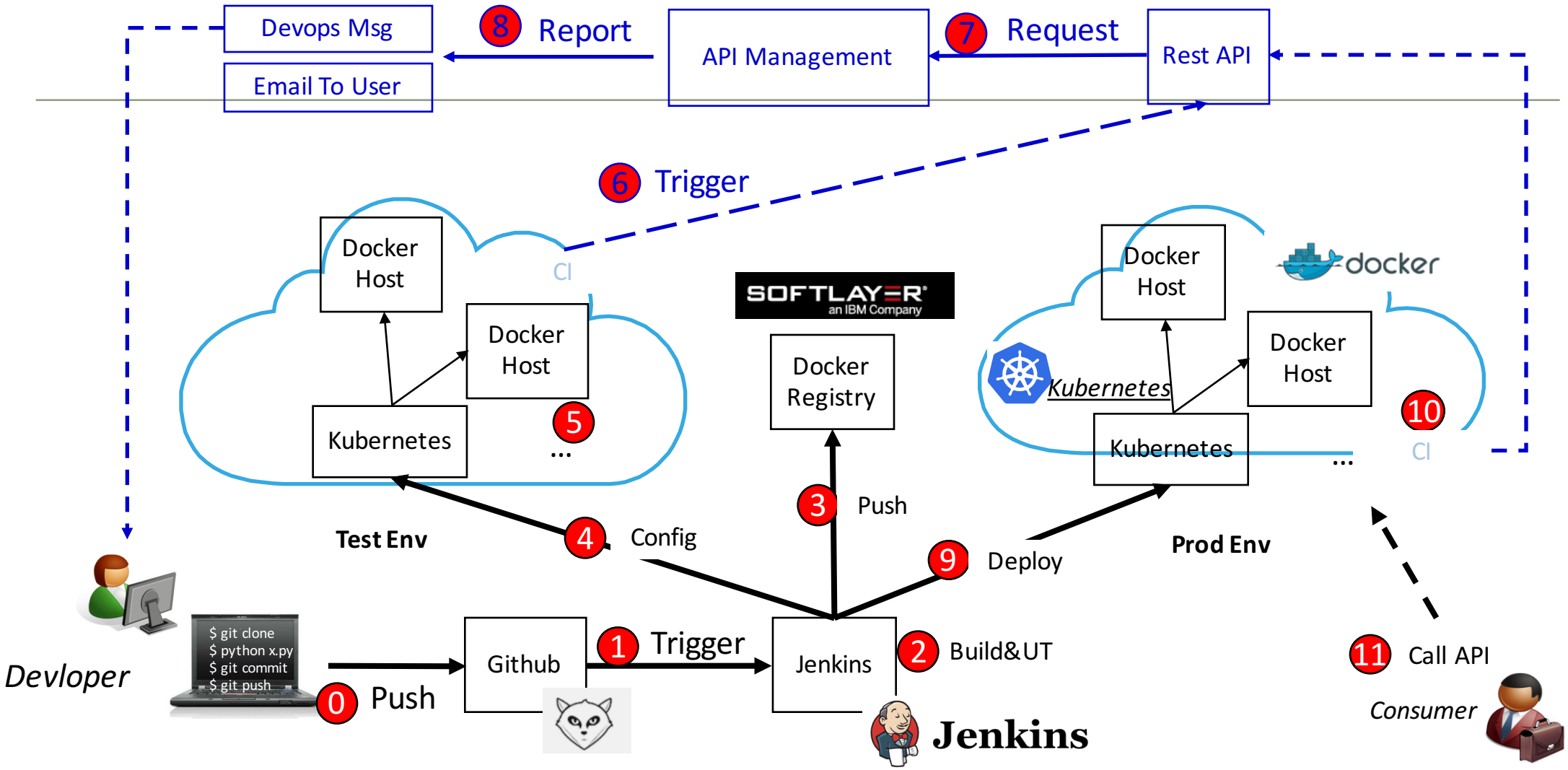


NoOps!

ContainerOps, CloudOps, ITOps…
Jazzhub, Travis-ci, drone.io, gitlab…

From *moving code* to *moving service*

Service Oriented


Devops Msg
Email To User
8 Report
API Management
7 Request
Rest API
6 Trigger
Docker Host
Docker Host
Kubernetes
5
...
CI
Test Env
SOFTLAYER
an IBM Company
Docker Registry
3 Push
4 Config
9 Deploy
Docker Host
Docker Host
Kubernetes
Kubernetes
...
10
CI
Prod Env
docker
Devloper
$ git clone
$ python x.py
$ git commit
$ git push
0 Push
Github
1 Trigger
Jenkins
2 Build&UT
Jenkins
11 Call API
Consumer

APITestingPlatform

ZheMing

ZheMing
Online

快速测试

Dashboard

用例执行

报告查询

Search...

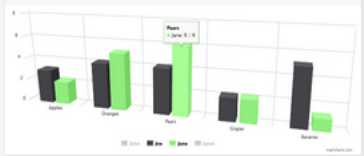
报告查询

Home / 报告查询

Cobalt
By PeiNi

项目创建时间：2015-1-1
项目版本:V1.0
创建者:LiuPeiNi
成功：105 失败：5

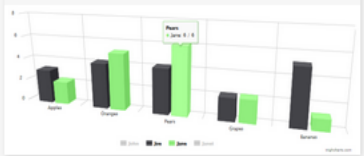
最近执行对比图：



CMIOT
By ZhongX

项目创建时间：2015-1-1
项目版本:V1.0
创建者:ZhongX
成功：89 失败：2

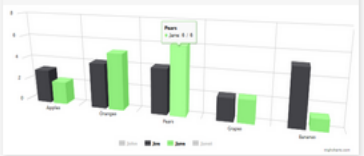
最近执行对比图：



Iot
By KeWei

项目创建时间：2014-7-15
项目版本:V1.0
创建者:SunKeWei
成功：62 失败：8

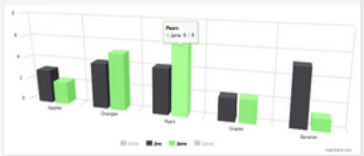
最近执行对比图：



TCWebPortal
By ZheMing

项目创建时间：2014-7-10
项目版本:V1.0
创建者:TuZheMing
成功：117 失败：3

最近执行对比图：



Execution Engine API
By PeiNi

项目创建时间：2014-11-1
项目版本:V1.0
创建者:LiuPeiNi
成功：45 失败：5

最近执行对比图：

JobManagement
By ZhongXiao

项目创建时间：2015-3-1
项目版本:V1.0
创建者:ZhongXiao
成功：37 失败：3

最近执行对比图：

API PLATFORM
By ZheMing

项目创建时间：2014-7-1
项目版本:V1.0
创建者:TuZheMing
成功：18 失败：2

最近执行对比图：

DebugTool
By ZheMing

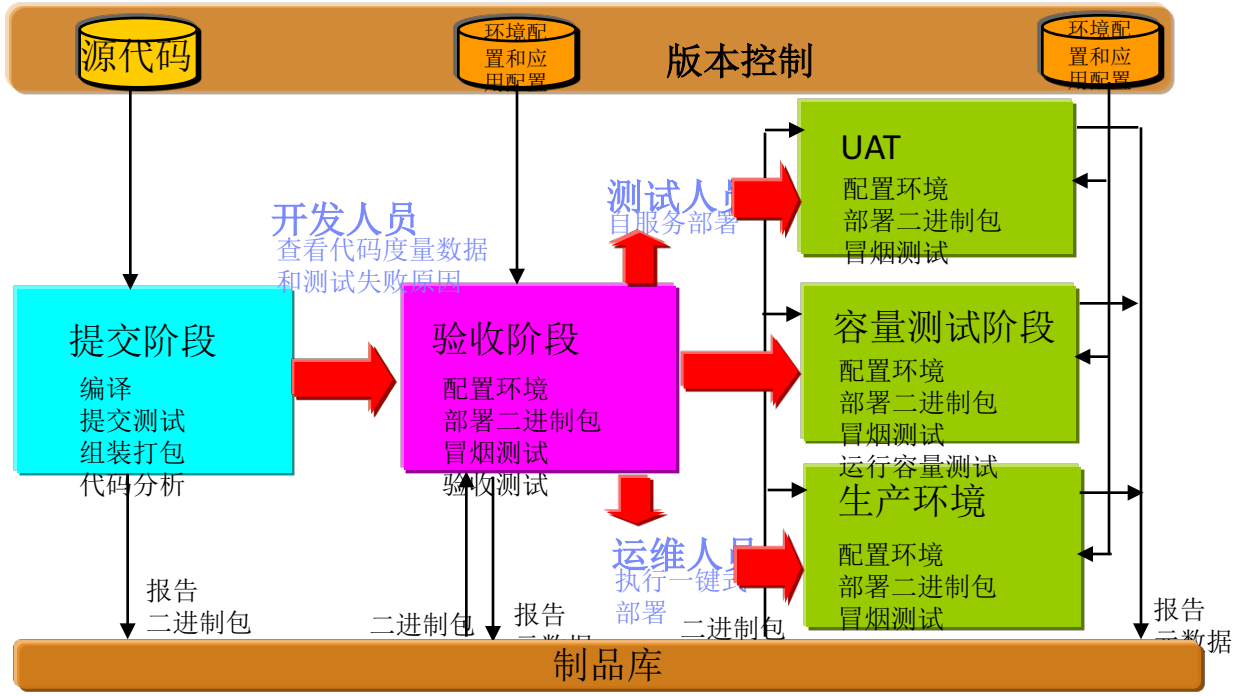
项目创建时间：2014-12-15
项目版本:V1.0
创建者:TuZheMing
成功：92 失败：8

最近执行对比图：

Cloud Based

DevOps最佳实践: 持续交付流水线

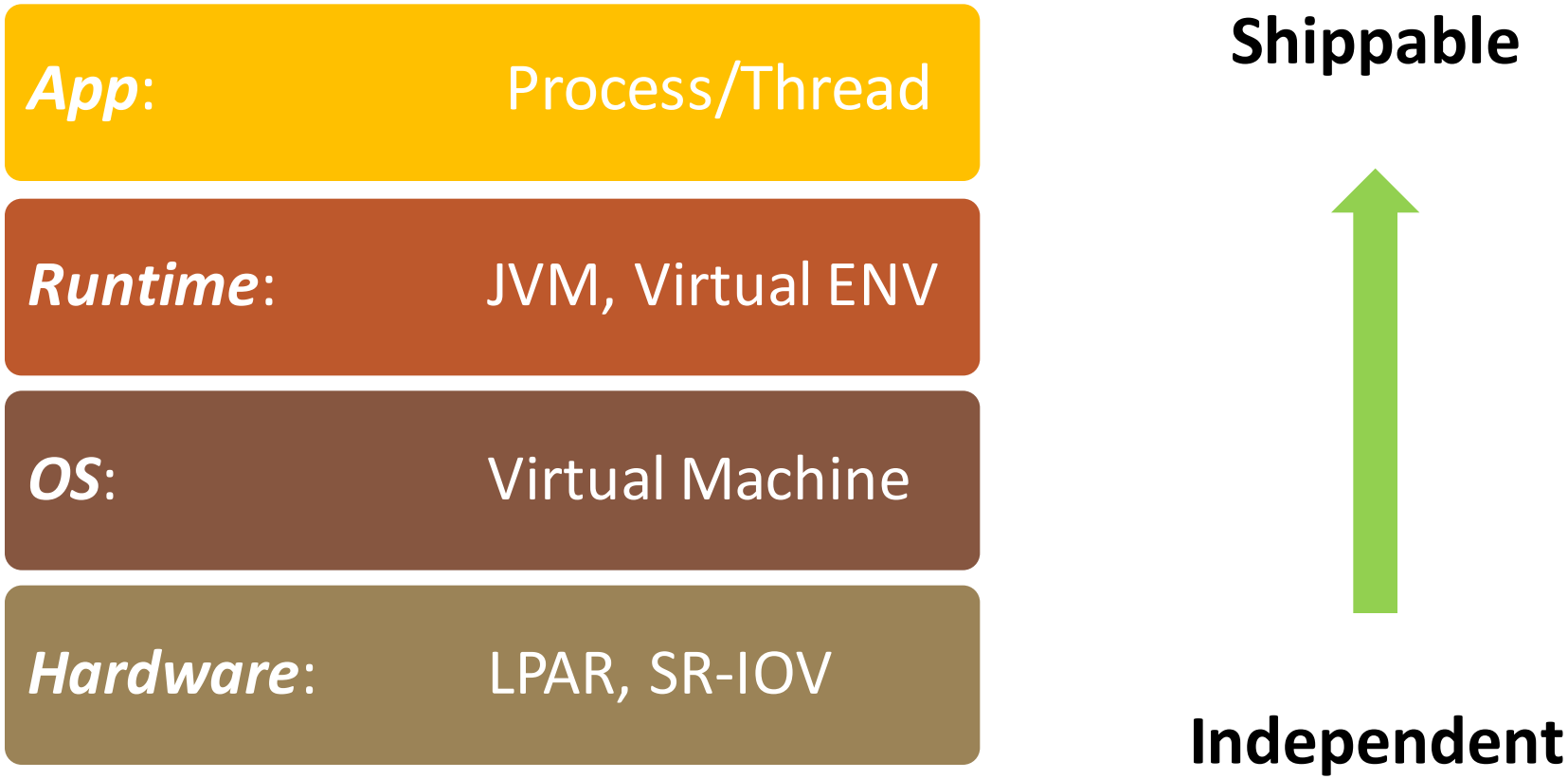
通过建立持续交付流水线整合和管理应用的构建, 测试, 部署和发布流程, 是企业实现快速和高质量应用交付的最佳DevOps实践。



Demo ready

The ultimate secret is to decouple!

Abstractions on different levels

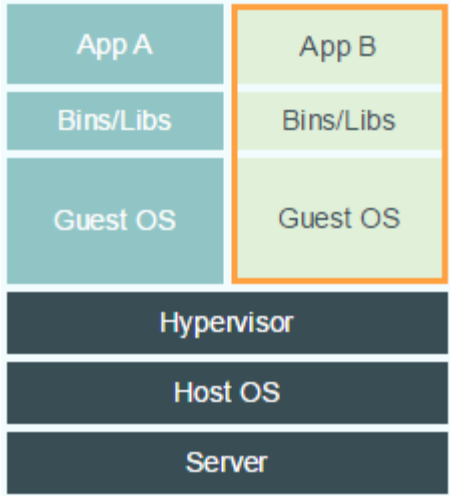


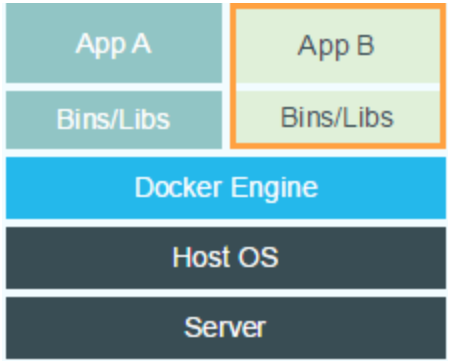
Docker: Build once, run anywhere!

$ sudo docker run busybox echo "hello world"

Decouple between app and infrastructure!

- App + Env
- LXC/Libcontainer + Wrapper (UFS, etc.) + DockerHub
  - Share kernel with host
  - Own process/network name space
  - Regardless of host distro (2.6.32+)





LXC, Garden, AppC/Rkt…

CoreOS, Atomic, LXD, Photon…

Docker issues in Devops

Data

Networking

Security

Configuration

Monitoring

And?

Data

/var/lib/docker

- *aufs*: UFS storage
- *containers*: Information of each container
- *execdriver/native*: Running container information
- *graph*: Images information
- *init*: docker init binary versions
- *linkgraph.db*: SQLite db file keeping the links between containers
- *repositories-aufs*: Info for images
- *trust*: signatures
- *volumes*: Randomly created volumes on host

Data/Backends

No Reliability guarantee!

AUFS

- Not upstreamed

Device Mapper

- Thin provisioning
- Loopback mounted sparse file

Btrfs

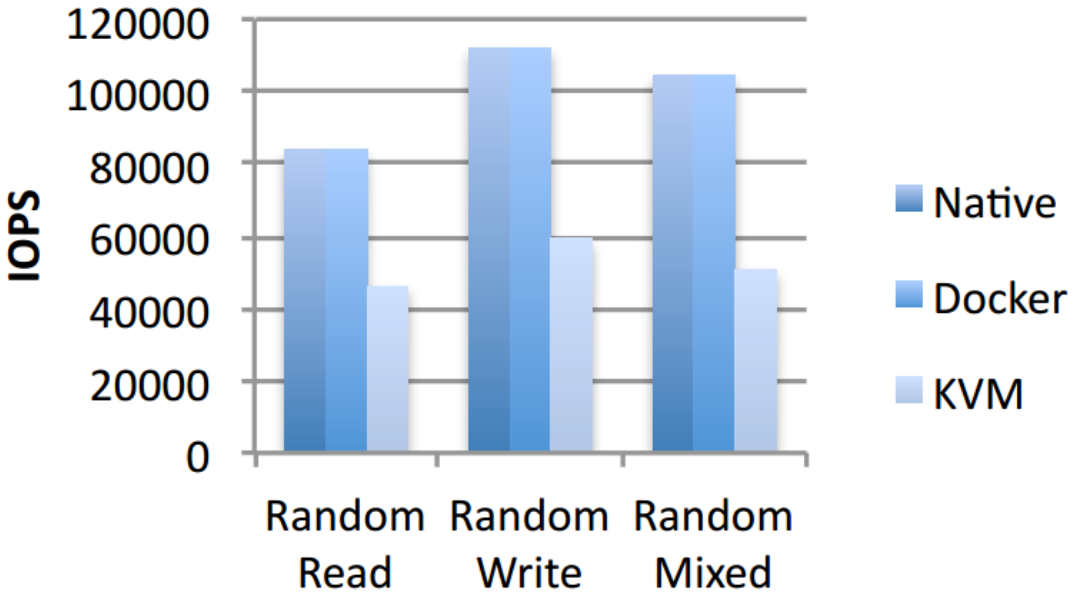
- Docker upstream
- No selinux
- No page cache sharing

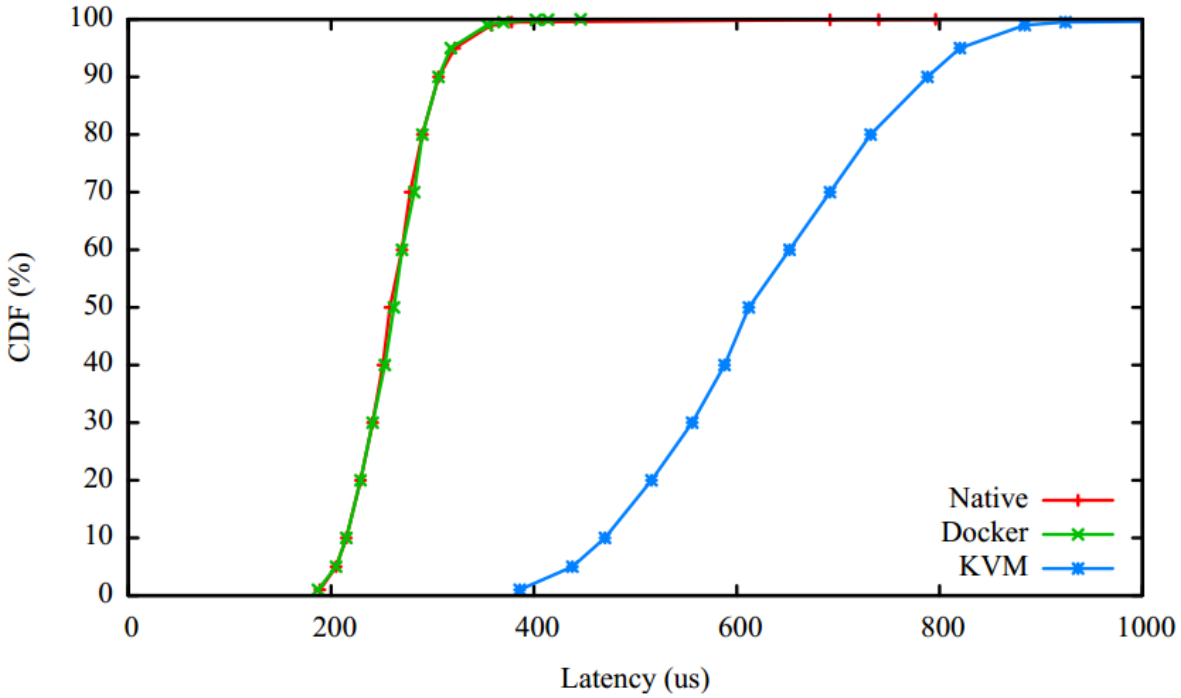
-s to select your favorite

```go
// Slice of drivers that should be used in an order
priority = []string{
    "aufs",
    "btrfs",
    "devicemapper",
    "overlay",
    "vfs",
}
```

daemon/graphdriver/driver.go

Data/Performance





An Updated Performance Comparison of Virtual Machines and Linux Containers, IBM Research, 2014

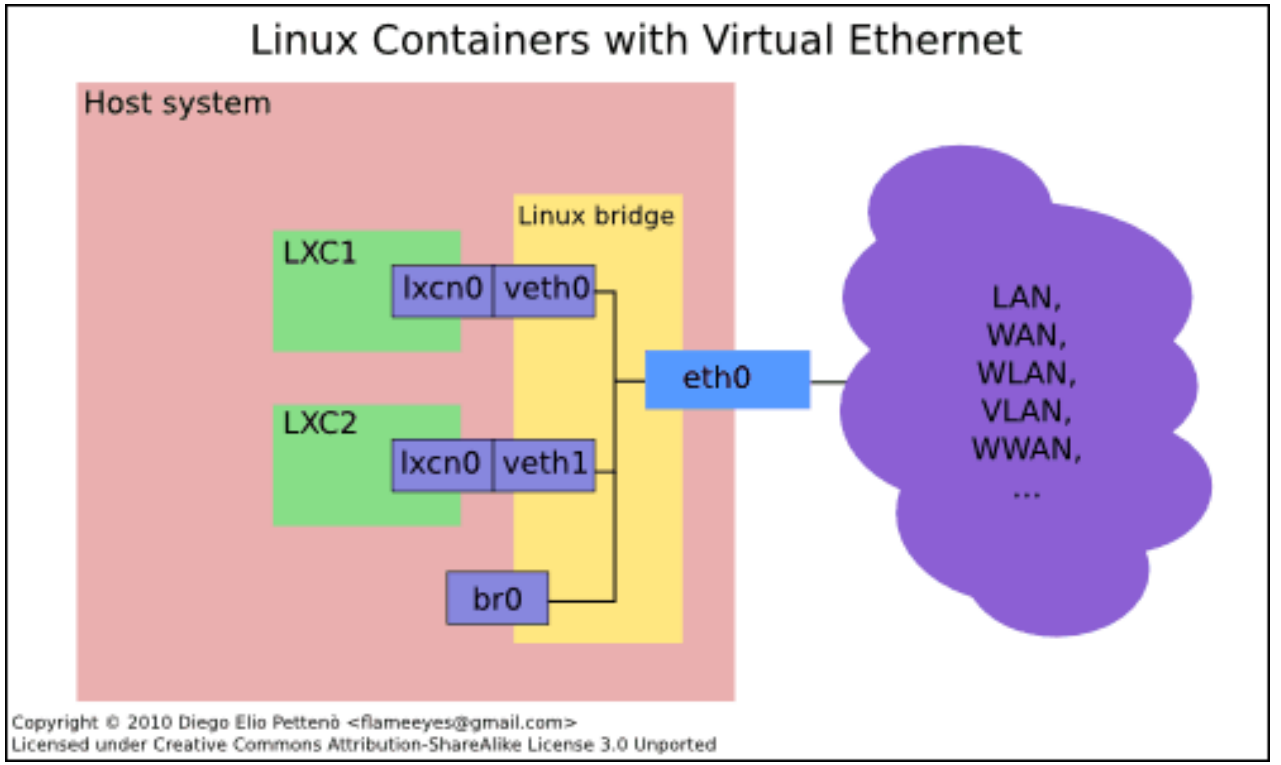
Networking

Think container as virtual machine! But

- Quicker booting/exiting
- More instances
- Shorter life (depends…)

Based on Linux Networking

- veth
- macvlan
- namespace
- iptables



Networking/DHCP

DHCP problems

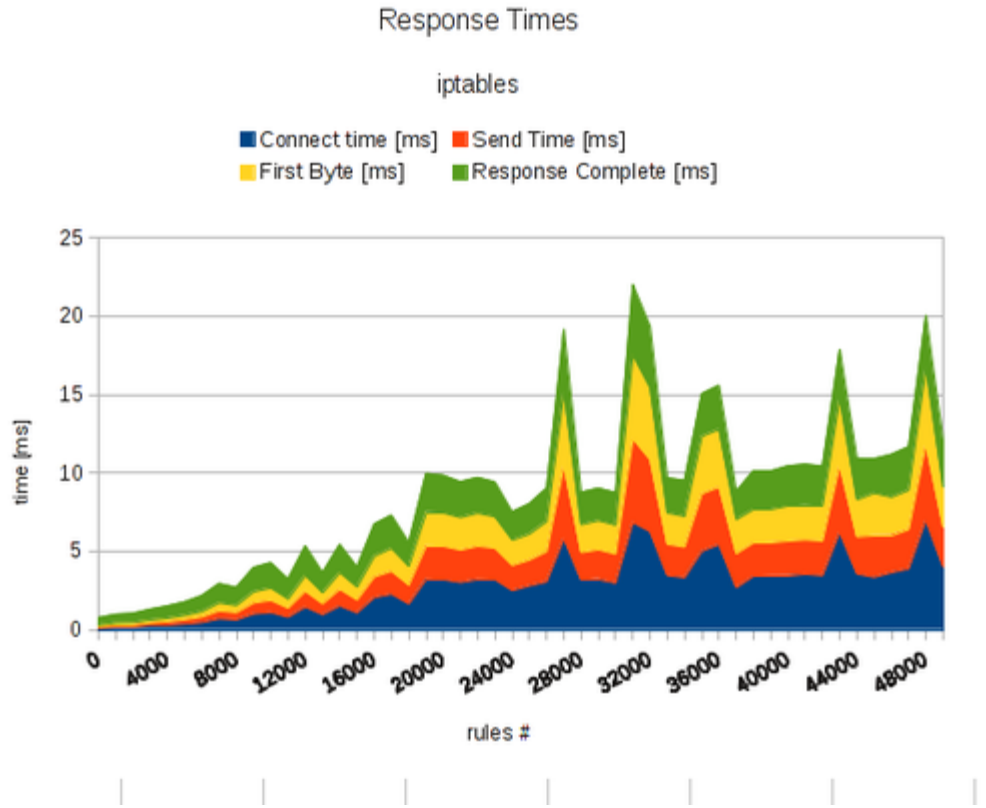
- Docker internally assigns IP for each container and change on restart even in 1.6 (#issue 2801).
- Check bridge, routing table and /etc/resolve.conf to obtain local network information
- DHCP service: fast!

```
addrs = []string{
    // Here we don't follow the convention of using the 1st IP of the range for the gateway.
    // This is to use the same gateway IPs as the /24 ranges, which predate the /16 ranges.
    // In theory this shouldn't matter - in practice there's bound to be a few scripts relying
    // on the internal addressing or other things like that.
    // They shouldn't, but hey, let's not break them unless we really have to.
    "172.17.42.1/16", // Don't use 172.16.0.0/16, it conflicts with EC2 DNS 172.16.0.23
    "10.0.42.1/16",   // Don't even try using the entire /8, that's too intrusive
    "10.1.42.1/16",
    "10.42.42.1/16",
    "172.16.42.1/24",
    "172.16.43.1/24",
    "172.16.44.1/24",
    "10.0.42.1/24",
    "10.0.43.1/24",
    "192.168.42.1/24",
    "192.168.43.1/24",
    "192.168.44.1/24",
}
```

daemon/networkdriver/bridge/driver.go

Networking/iptables

- Docker will generate static iptables nat rules to do port mapping
  - Existing rules confliction
  - Performance problems
    - Slow in update/config
    - Complex in grammar/ more rules
  - Dynamic rule generation?
- nftables since 3.13
  - Faster in update/config
  - Less kernel work



Networking/ideal solution?

Kubernetes
- Flannel

Weave

Socket

OpenStack Neutron

pipework

tenus

docknet

…

SDN!

Security

Think container as application!

Validate external images

Only put mutually-trusted containers on the same host

- --icc=false + manual link

Use AppArmor/SELinux

Minimize privileges enforced inside the container

- docker run -it –rm –u user1 --cap-drop SETUID --cap-drop SETGID …

Limit resource usage using cgroups

- docker run -it --rm --cpuset=0,1 -c 2 -m 128m …
- docker -d --storage-opt dm.basesize=5G

Mount volume with proper permission

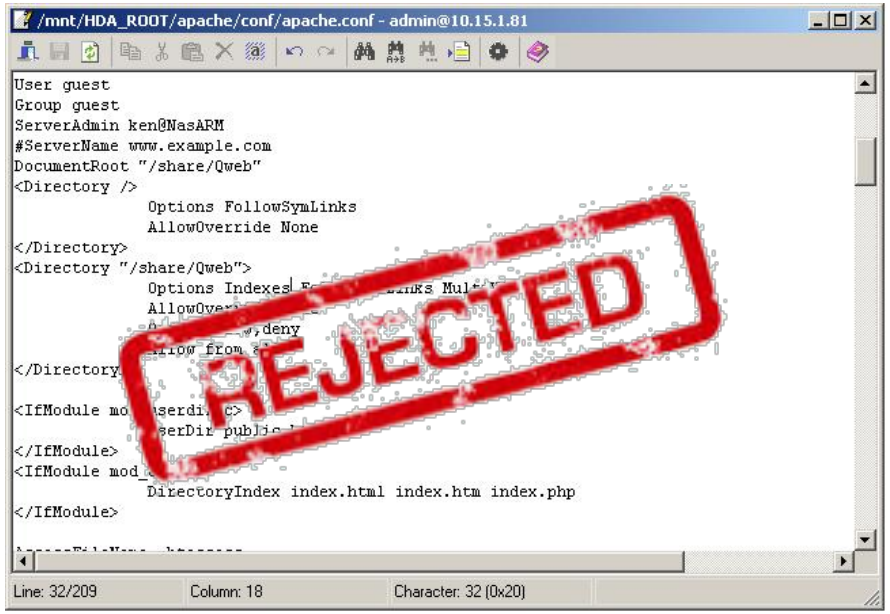
Check the Docker Secure Deployment Guidelines

Some one chooses running containers inside VM

Configuration

Config file is the legacy way

- Formats vary by xml, json, manual-defined, etc.
- Where to store for application in container?
- How to update config values flexibly?
- Management is too hard



Decouple from application itself

Configuration = Key+Value

- Store in centric db
- ENV variables/running options?

Monitoring

Easier to monitor container than virtual machine

- CPU
- Memory
- IO
- Network
- FD

Root Tracing

- docker logs
- Tools like ELK

Inject limit information into container proactively!

Miscellaneous

Supervisor

Discovery

Boot order

Fat container

Zombie-reaping, syslog-ng, ssh, cron, runit, setuser problems

- phusion/baseimage

No Use Docker Manually

Docker Inc
- Compose
- Machine
- Swarm

IBM
- Bluemix

Google
- Kubernetes
- Borg/Omega

Baidu
- Matrix

OpenStack

4 PRINCIPLES

4 Basic Principles

0. No silver bullet!

1. DONOT use container before understanding enough

2. Use container transient, stateless, and fault-tolerant

3. Do you care IO or security heavily?

Q&A