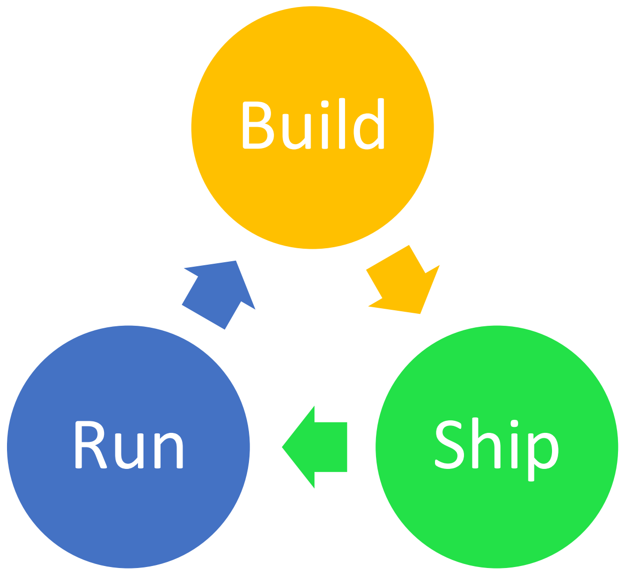
# Container Ecosystem

Baohua Yang

2015-11-05

# Overview

- Problem 1: runtime
- Problem 2: packaging & distribution
- Problem 3: service composition
- Problem 4: machine management
- Problem 5: clustering
- Problem 6: networking – Docker Networking
- Problem 7: extensibility – Docker Plugin

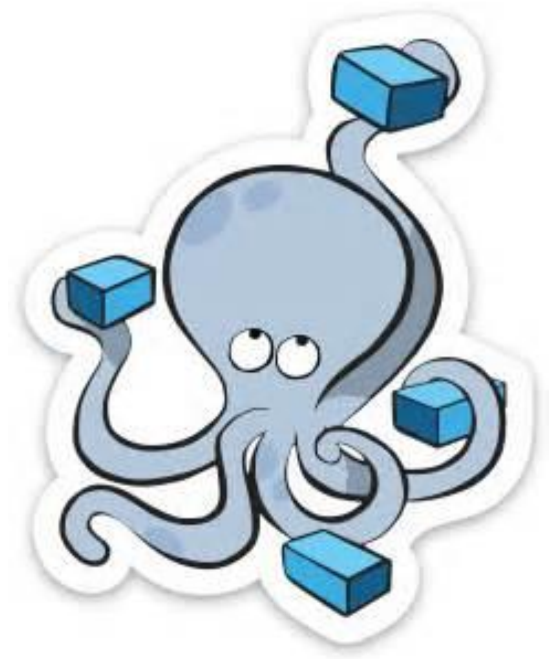


## Docker Machine

- Create Docker hosts on your computer, on cloud providers, and inside your own data center.
- Multiple drivers: vm, host, cloud platforms
- Operate on them through docker-machine

## Docker Compose

- Define and run multi-container applications with Docker
- Previously known as Fig
- Write a template to define your app clusters, manage it with compose



## Docker Swarm

- Docker Swarm is native clustering for Docker. It turns a pool of Docker hosts into a single, virtual host.
- Serves the standard Docker API
- Pluggable backends (can be integrated with Mesos)

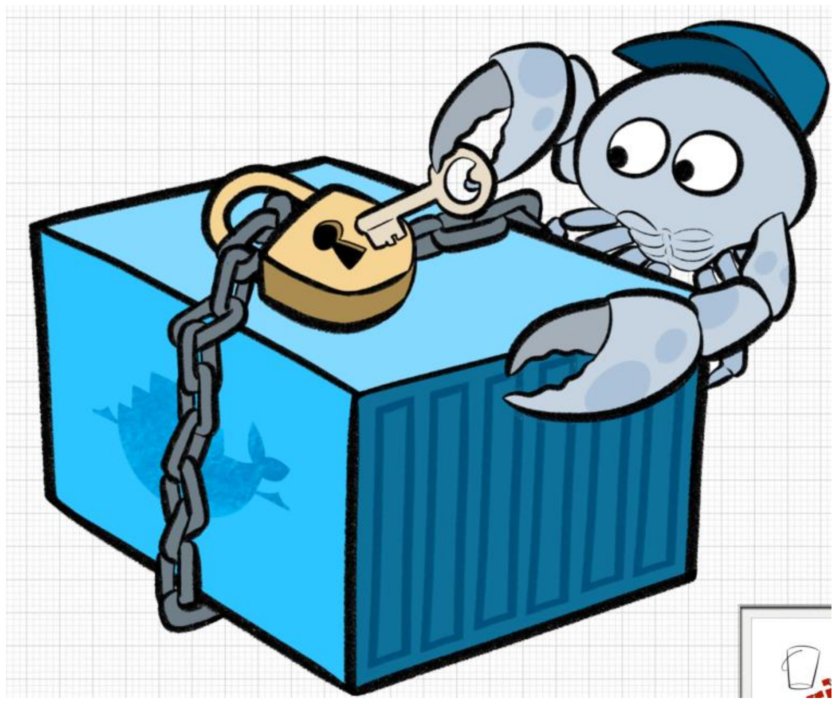
## Docker Distribution

- The Docker toolset to pack, ship, store, and deliver content (images).
- Main part is the Registry.

- tightly control where your images are being stored
- fully own your images distribution pipeline
- integrate images storage and distribution into your inhouse development workflow

## Docker Notary

- https://github.com/docker/notary
- A trusted publishing system for any content.



## Docker Trusted Registry

- Run and manage your own Docker image storage service locally

- An image registry to store, manage, and collaborate on Docker images
- Pluggable storage drivers
- Configuration options to let you run DTR in your particular enterprise environment.
- Easy, transparent upgrades
- Logging, usage and system health metrics

## Docker Plugins

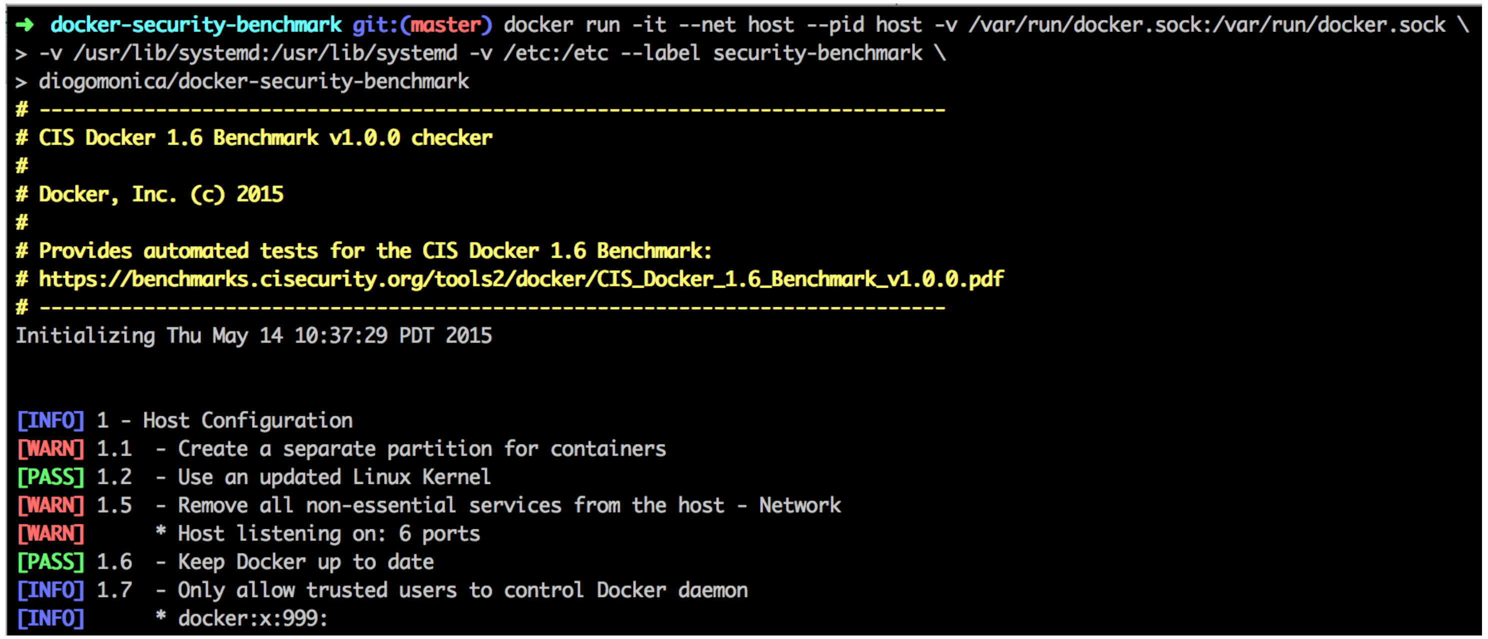
- Plugin system for Docker Engine, since 1.7
  - Networking
  - Volumes

  - Scheduler
  - Service discovery

  - And will be More!

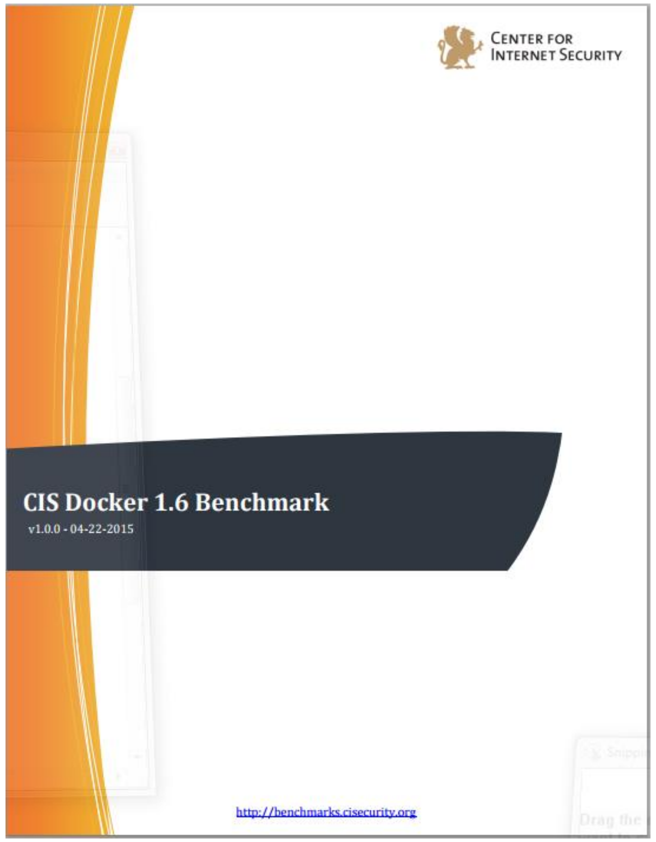
## Docker Security

- dockerbench.com



```
➜ docker-security-benchmark git:(master) docker run -it --net host --pid host -v /var/run/docker.sock:/var/run/docker.sock \
> -v /usr/lib/systemd:/usr/lib/systemd -v /etc:/etc --label security-benchmark \
> diogomonica/docker-security-benchmark
# ------------------------------------------------------------------------------
# CIS Docker 1.6 Benchmark v1.0.0 checker
#
# Docker, Inc. (c) 2015
#
# Provides automated tests for the CIS Docker 1.6 Benchmark:
# https://benchmarks.cisecurity.org/tools2/docker/CIS_Docker_1.6_Benchmark_v1.0.0.pdf
# ------------------------------------------------------------------------------
Initializing Thu May 14 10:37:29 PDT 2015


[INFO] 1 - Host Configuration
[WARN] 1.1  - Create a separate partition for containers
[PASS] 1.2  - Use an updated Linux Kernel
[WARN] 1.5  - Remove all non-essential services from the host - Network
[WARN]      * Host listening on: 6 ports
[PASS] 1.6  - Keep Docker up to date
[INFO] 1.7  - Only allow trusted users to control Docker daemon
[INFO]      * docker:x:999:
```

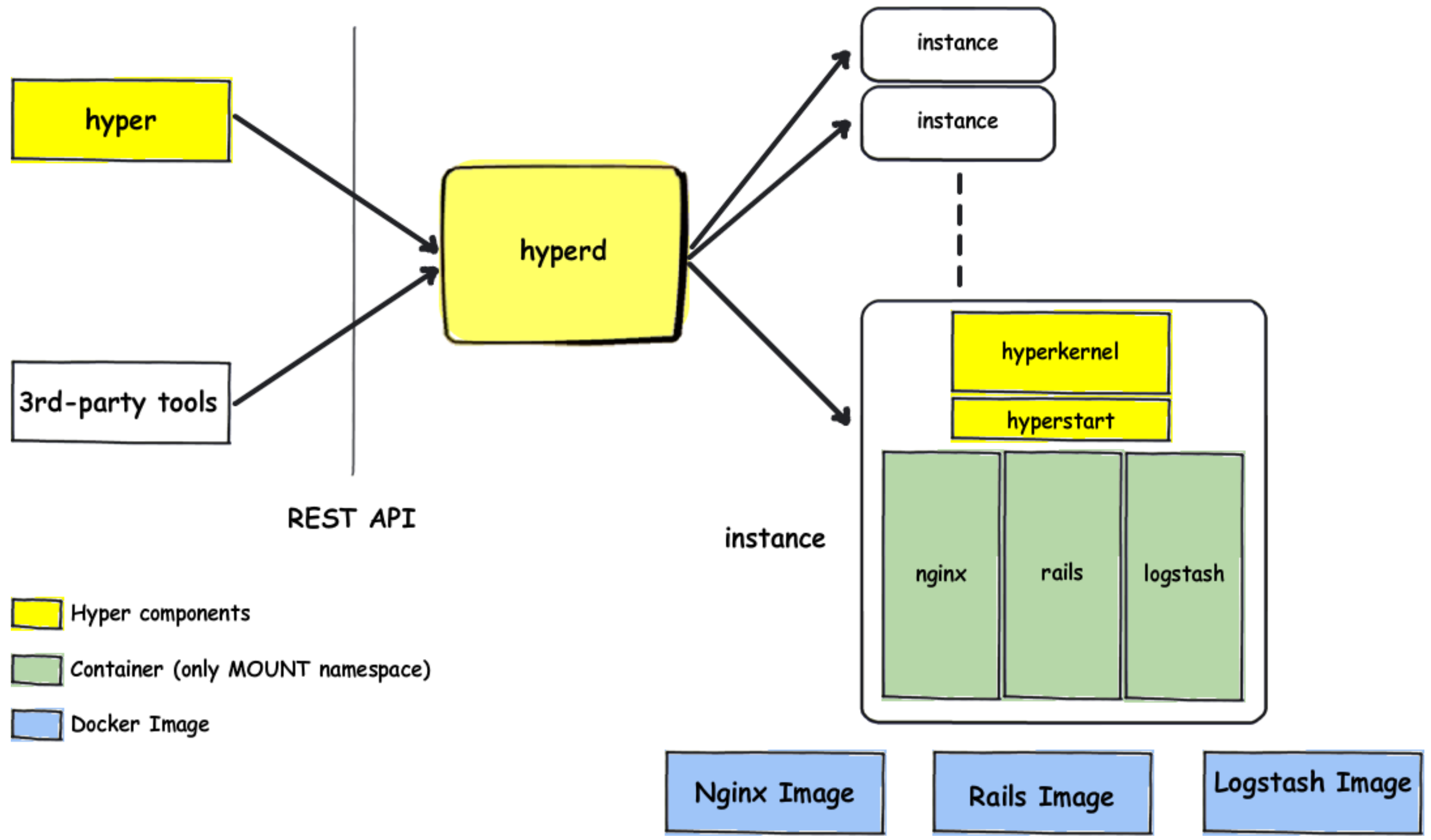

Center for Internet Security
CIS Docker 1.6 Benchmark
v1.0.0 - 04-22-2015
http://benchmarks.cisecurity.org

# Docker in OpenStack

- Nova-Docker
  - Treat Docker as VM
- Docker in Heat
  - Docker as a resource type
- Magnum
  - The main idea is to use Nova to boot (preferably large) VMs, and then schedule containers on top of VMs
  - OpenStack would keep track of the VMs in Nova’s database, and Container (inside VMs) on Magnum’s database
- Kuryr
  - Bridging libnetwork to Neutron

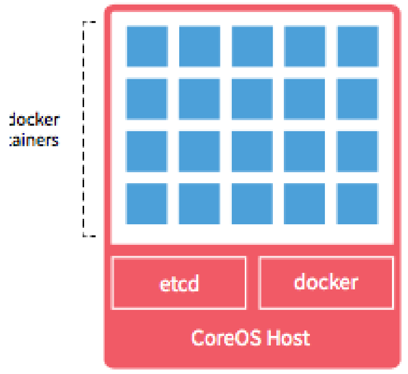
# Hyper_

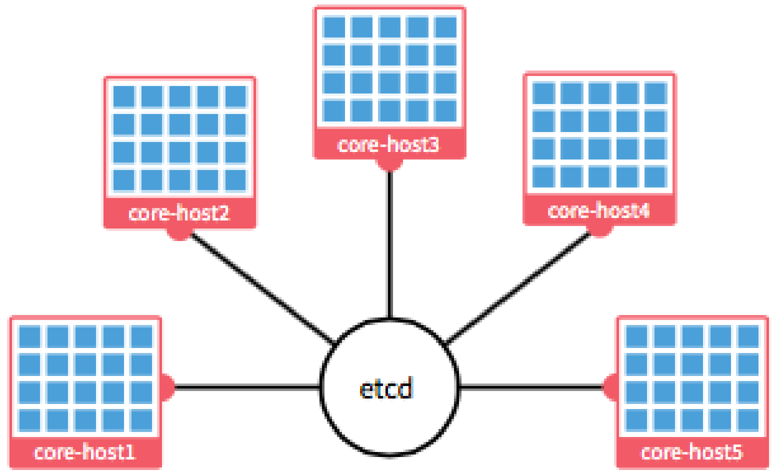
- Similar with Intel Clean Container



# CoreOS

- A Linux distribution for the Cloud age
- Based on Google’ s ChromeOS
- Relies on Docker to run workloads
- Cluster Management
- Service Discovery
- Fleet, Etcd





# Cloud Native Application

- **I. Codebase**
- One codebase tracked in revision control, many deploys
- **II. Dependencies**
- Explicitly declare and isolate dependencies
- **III. Config**
- Store config in the environment
- **IV. Backing Services**
- Treat backing services as attached resources
- **V. Build, release, run**
- Strictly separate build and run stages
- **VI. Processes**
- Execute the app as one or more stateless processes
- **VII. Port binding**
- Export services via port binding
- **VIII. Concurrency**
- Scale out via the process model
- **IX. Disposability**
- Maximize robustness with fast startup and graceful shutdown
- **X. Dev/prod parity**
- Keep development, staging, and production as similar as possible
- **XI. Logs**
- Treat logs as event streams
- **XII. Admin processes**
- Run admin/management tasks as one-off processes

# Open Container Initiative

- The OCI was launched on June 22nd 2015
- Target: formal, open, industry specification on container image format/runtime
- Libcontainer, appC, runC
- Cloud Native Computing Foundation

LINUX FOUNDATION COLLABORATIVE PROJECTS

## Others

- Management: Panamax, Shipyard, cadvisor,
- Paas: Flynn, Deis, solum.io
- CI: Drone.io, Travis CI

Q&A