Geekbang 极客邦科技 InfoQ

QCon 全球软件开发大会【北京站】2016

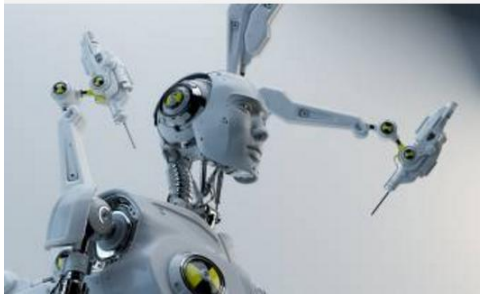
构建基于Kubernetes的容器云系统

才云科技CTO / 邓德源

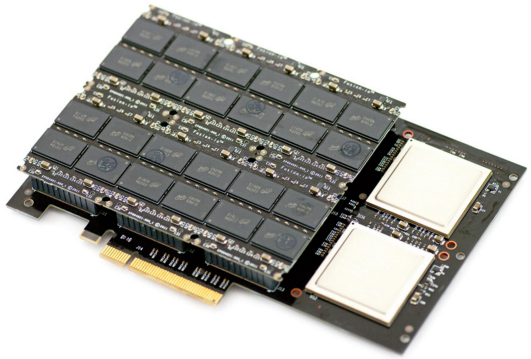
Intemational Software Development Conference

About me

And when I was young in the good old days…



Carnegie Mellon University



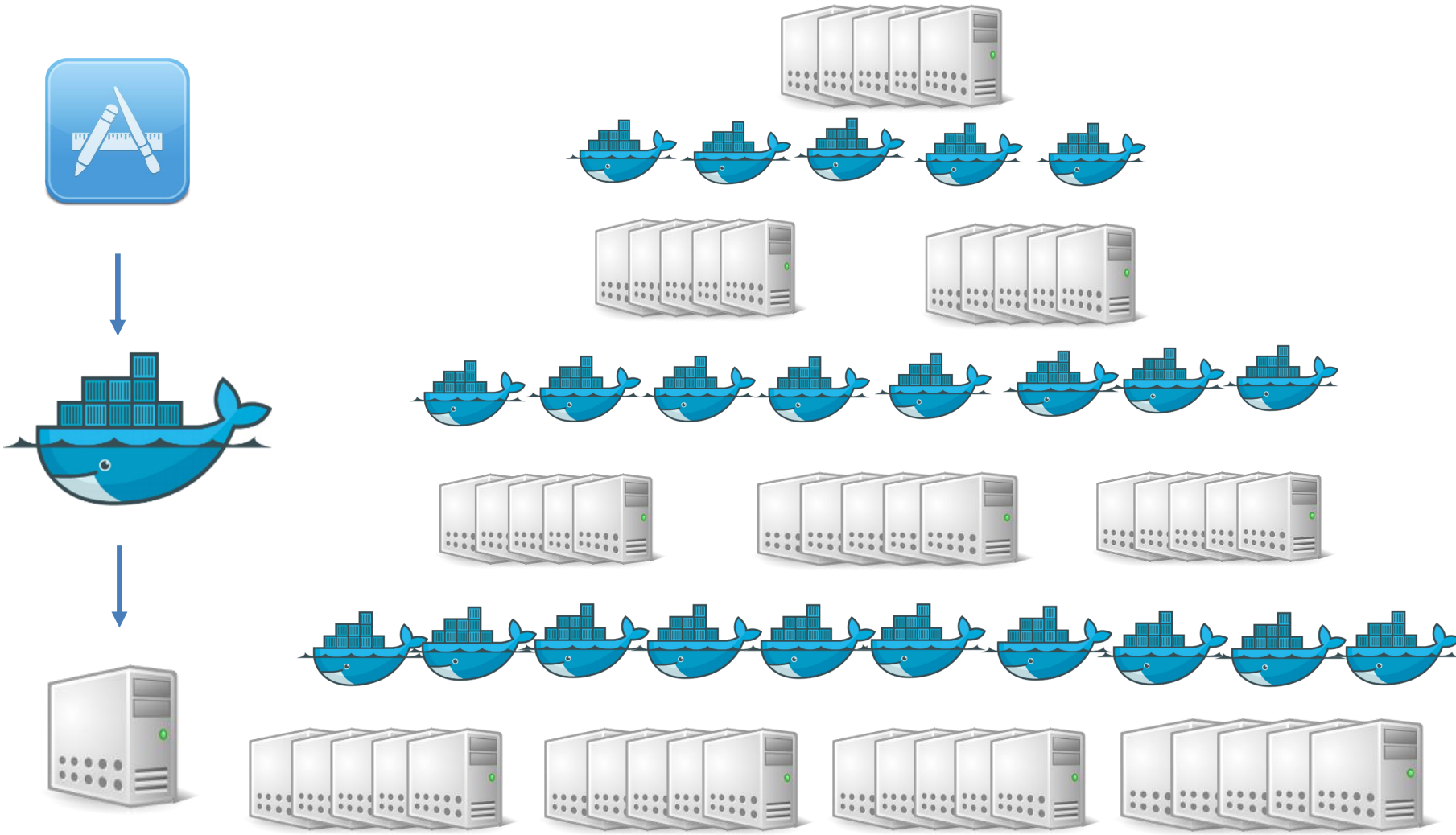
Google

kubernetes

Case Study: Containers in Google

- Using containers for a decade
- Running 2 billion containers a week
- Solves application migration nightmare
- Saves billion dollars a year
- Docker: 5x yearly growth rate

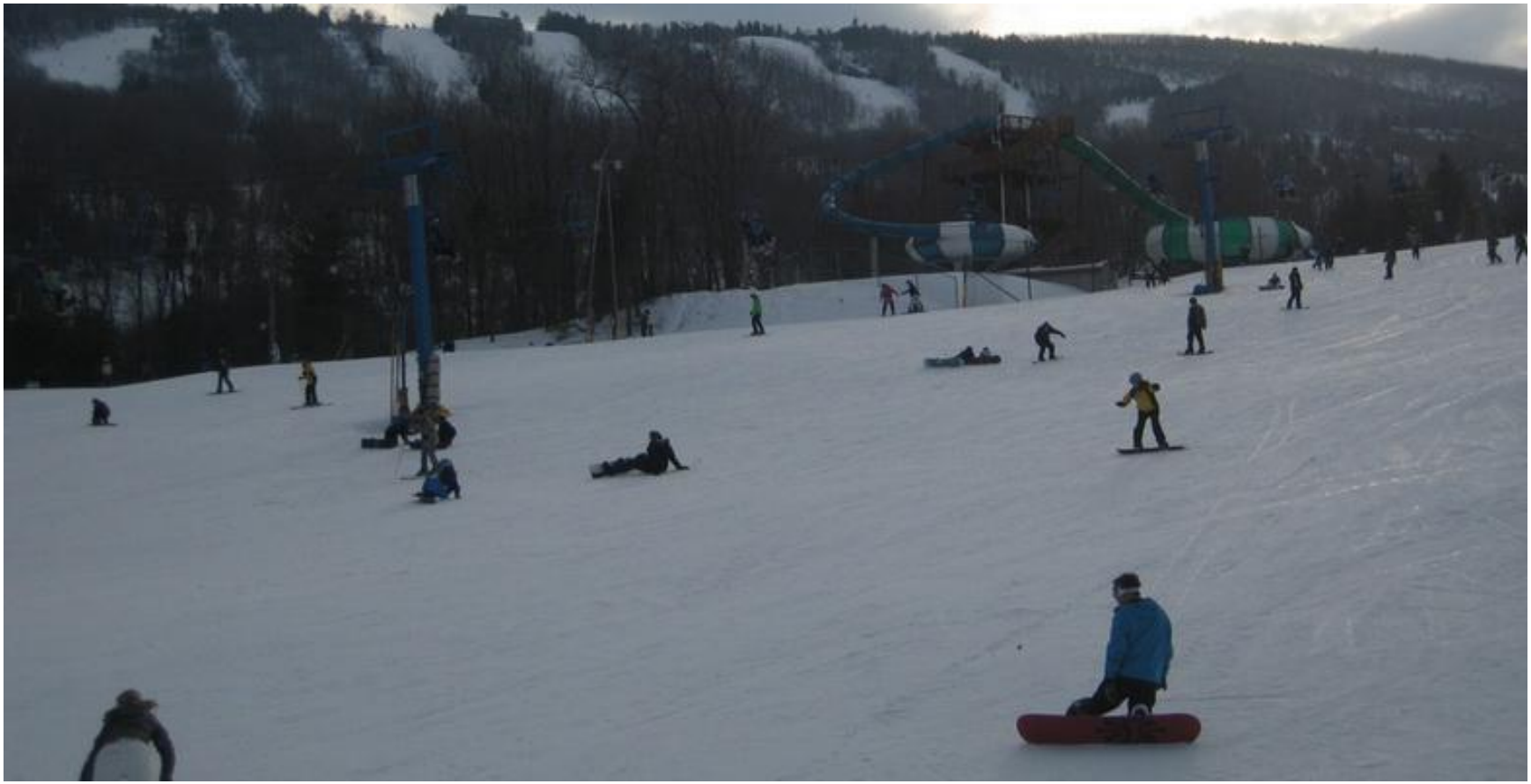
Solved Problems of the World?



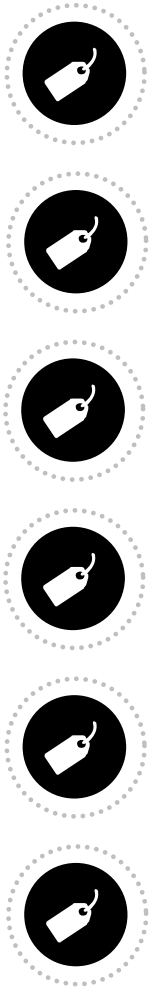
Case Study: Cluster Management in Google

- 1 SRE handles ~ 10,000 machines with 99.999% reliability
- Clustering is the hard part. In Google:
  - *NO* team dedicated for container study
  - *HUNDREDS* of engineers built THREE cluster manager systems
  - *HUNDREDS* of teams building ecosystems
- Clustering is the real value in building serious production container systems

All the Fun Stuff Began Here



Kubernetes Design Principles

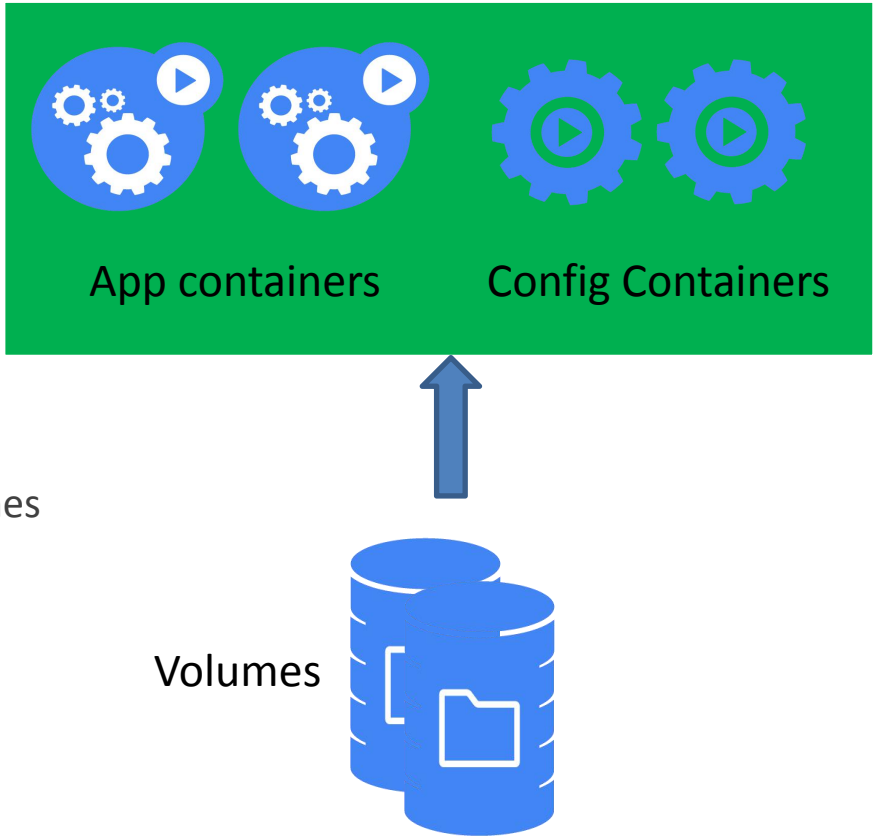


- declarative > imperative
- simple > complex
- labels > hierarchy
- legacy compatible
- extensible and pluggable
- application centric

Kubernetes Highlights

How to group resources?

- Pods



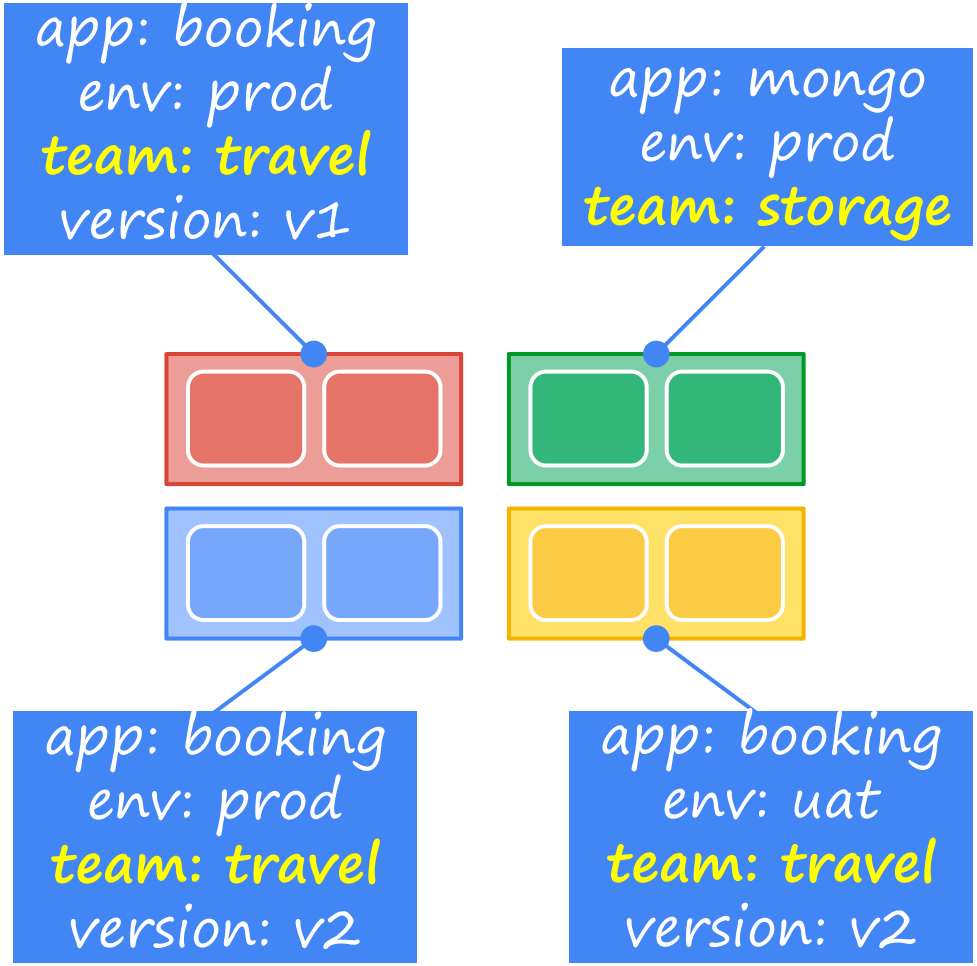
Lessen learned from Borg

- Jobs are usually grouped
  - e.g. log offloading
- Allow teams to develop distinct part of application
- improve robustness
  - e.g. log can be offloaded even if container crashes
- Atomic scheduling
  - C1: 1core, C2: 2cores
  - M1: 2cores, M2: 8 cores
  - C1 -> M1?

Kubernetes Highlights

How to manage *massive* resources in a *flexible* way?

- Labels and its query API
- Selectors



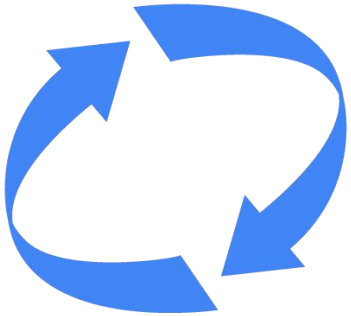
Kubernetes Highlights

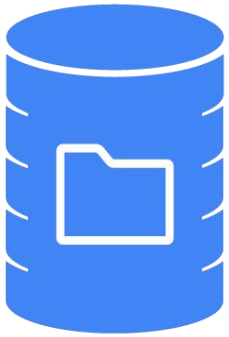
How to do service discovery for external services?

- Use services and endpoints together

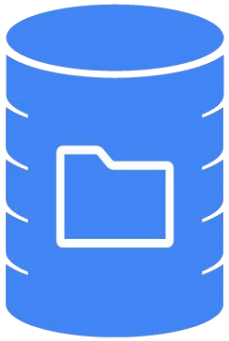
SERVICE

Name: “Oracle”
(NO IP)





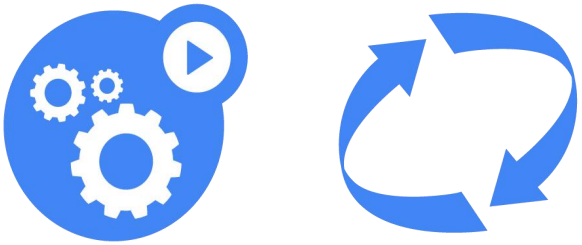
Prod Endpoint:
Name: “Oracle”
IP: 10.254.1.1



QA Endpoint:
Name: “Oracle”
IP: 192.168.1.1

Kubernetes Highlights

How to deal with configurations *varying* in different environments?



APP in Pod:
ref: ENV[ORACLE_PASSWD]

Single image, decoupled
from varying configurations

ConfigMap in Pod:
ORACLE_PASSWD: “7h6#f)”
JETTY_CONFIG_PATH: …

ConfigMap in UAT:
ORACLE_PASSWD: “123456”
JETTY_CONFIG_PATH: …

Kubernetes Highlights

How to NOT let docker persist my credentials?

- Secrets

```
apiVersion: v1
kind: Secret
metadata:
  name: aliyun-api-keys
data:
  api-client1: …
  api-client2: …
  aliyun-api-keys: …
```

```
apiVersion: v1
kind: ReplicationController
metadata:
  name: caicloud-cluster-manager
spec:
  …
   volumes:
        - name: aliyun-api-keys
          secret:
            secretName: aliyun-api-keys
        - name: caicloudapp-ssl-cert
          secret:
            secretName: caicloudapp-ssl-cert
```

Kubernetes Highlights

How to perform *fine-grained* resource control and access control?

- Namespaces and service accounts

How to handle services or applications that are *stateful*?

- L7 load balancer
- Ingress controller
- node affinity
- PetSet (coming)
- Pod
- Lifecyle interfaces

How to *automatically* create, delete, and allocate storage resources?

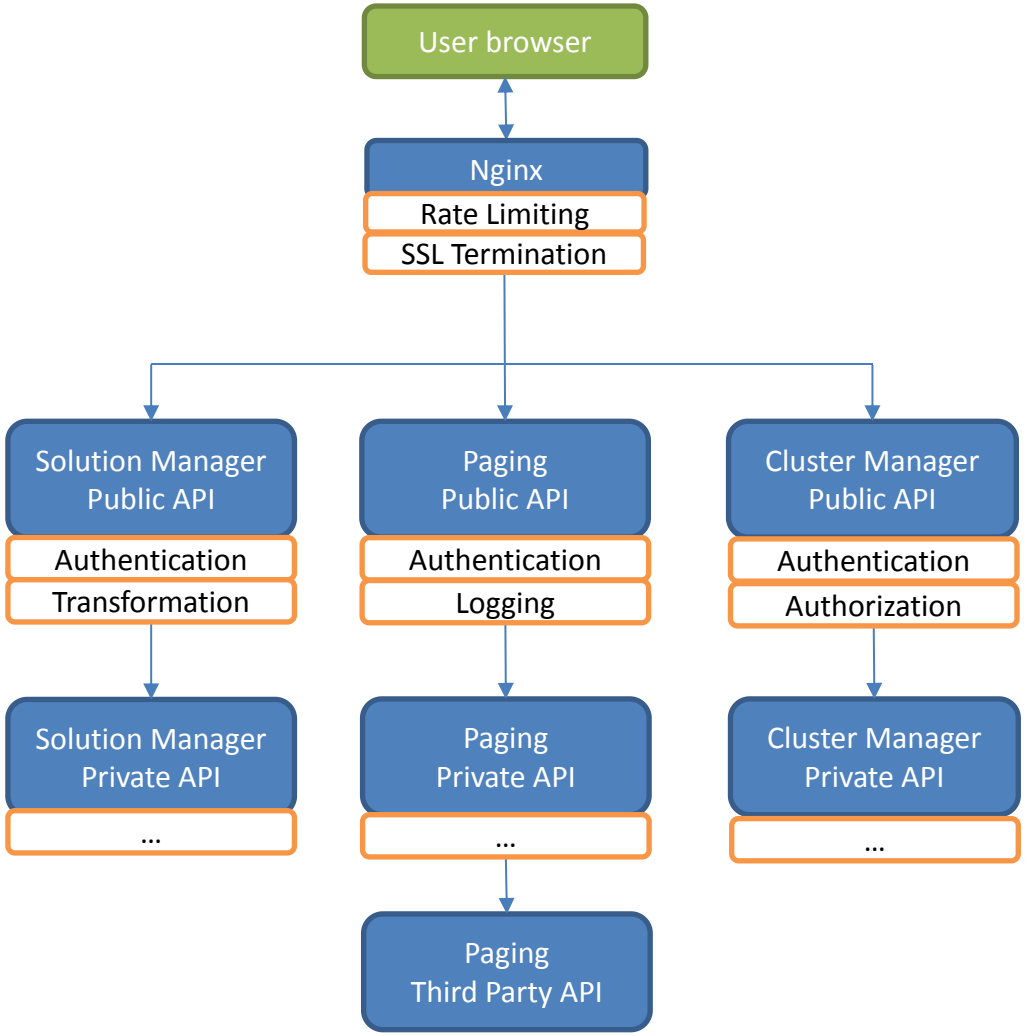
- Persistent volumes and claims

Our Practice on Using Kubernetes to Build Cluster Management System

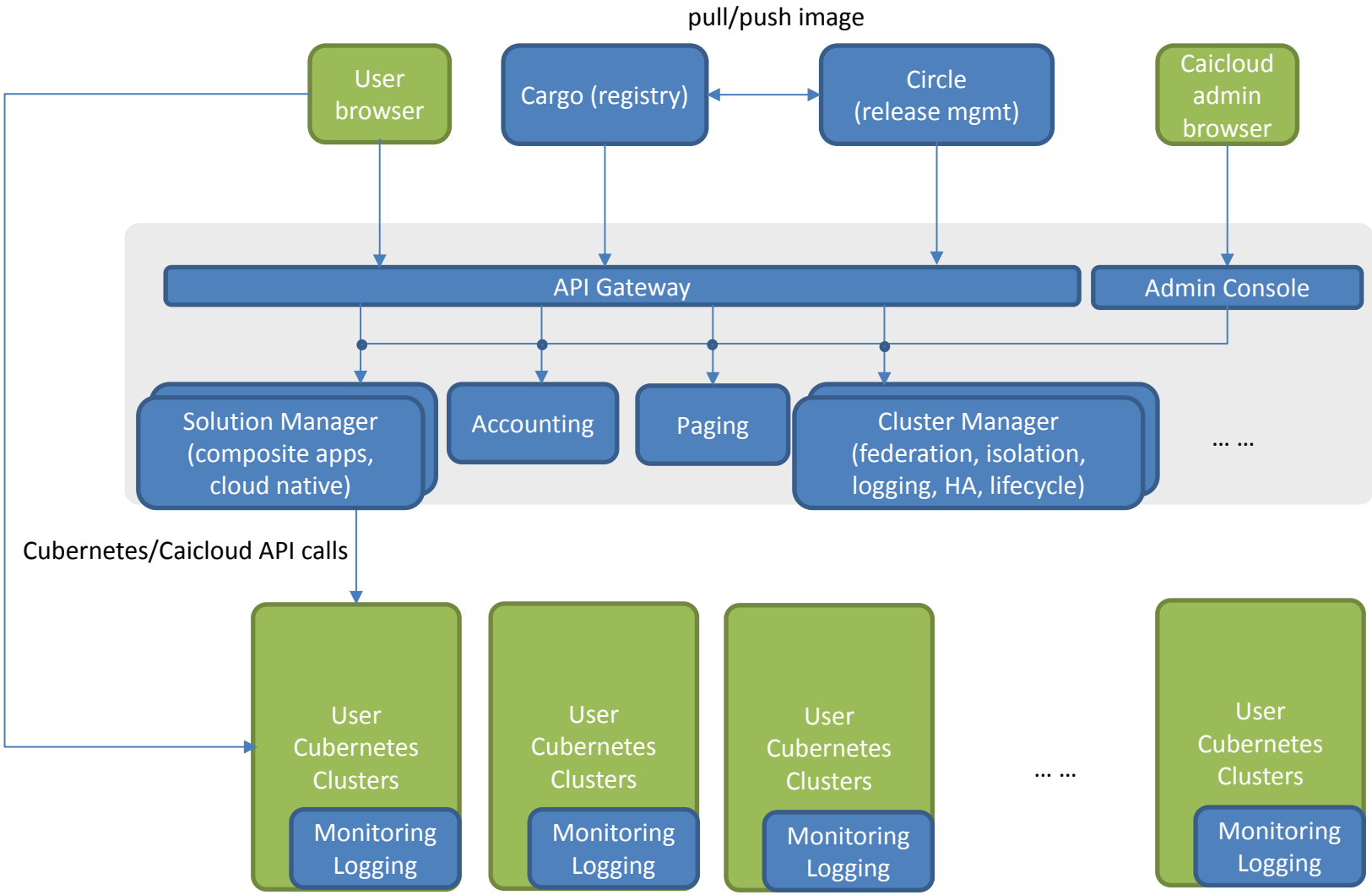
Architecture

Problems:

- Duplicate functionalities
- System tends to be monolithic
- Complex frontend logic due to varied APIs



Architecture



Design: CLaaS vs CaaS

Cluster vs Container as the operation units

- Additional higher-level management
- E.g., clone entire clusters and ensure holistic consistency (e.g., config) beyond just image consistency

Clustered applications vs container processes

- CLaaS: ES cluster with data, client, and master nodes + offload data processing
- CaaS: `docker run elasticsearch:1.7.4`

Host cluster exposure vs container black box

- CLaaS: requires dedicated clusters; additional host information and access points
- CaaS: hosts are abstracted away; obscures debugging, tooling, and customization

Example: The True Consistency and Portability

Scenario:

- Tomcat, Redis Cluster, Elastic Search, Mongo DB
- Want to setup development, testing, and production environments and be 1) fast and 2) consistent

CaaS: *image-level* consistency

- $Tomcat_{1...3}$, $Redis_{1...3}$ and $ES_{1...3}$ have consistent images
- How to handle different IPs, references, config files, dependencies? ***The MOST troublesome part unsolved!***

CLaaS: system-*level* consistency

- Tomcat1, tomcat2 and tomcat3 have consistent images
- IPs use consistent names (even for external services)
- Config uses consistent references, dependencies are respected

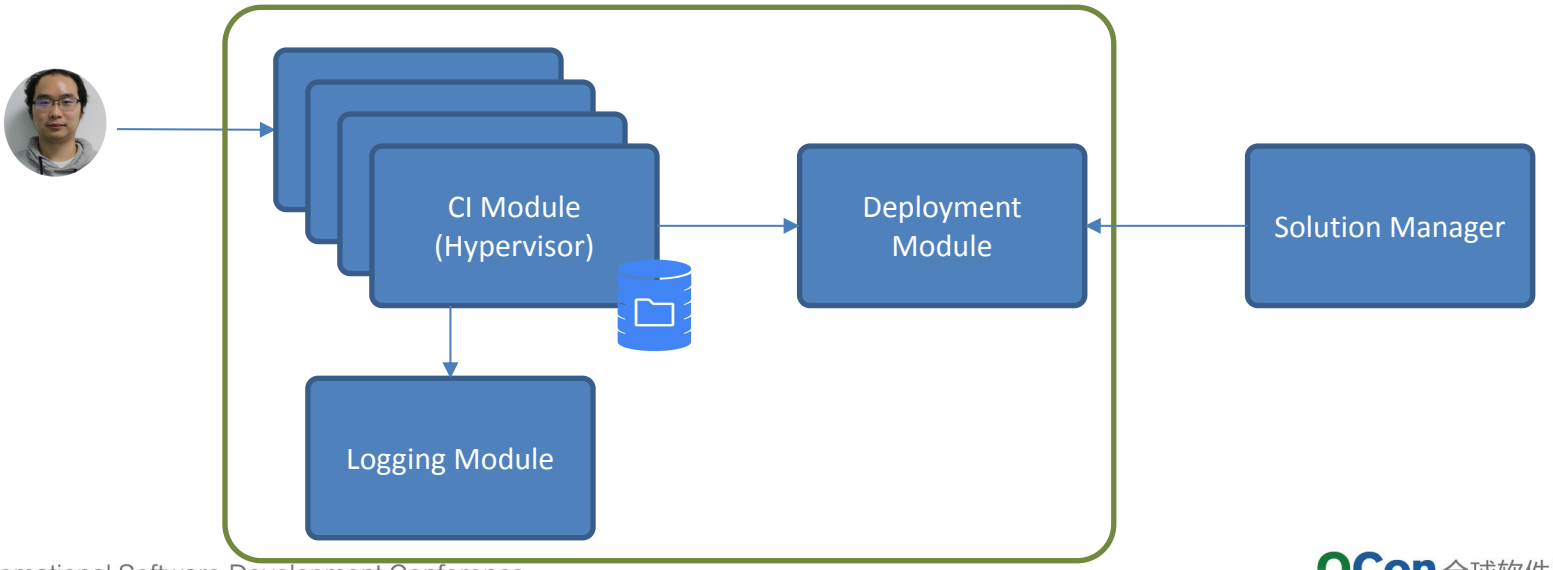
Design: Release Management

- Where is my-awesome-app running?
- What is the latest version of my-awesome-app?
- What is the live version of my-awesome-app?
- Is version Y running long enough to roll out (and upgrade version X)?
- Can I continuously deploy my-awesome-app to test cluster.
- How can I upgrade my-awesome-app with his-xxx-app now that I have to depend on it?

Design: Release Management

- Static Configuration
  - Easy but ‘static’, works well in most cases
- Dynamic tracking
  - Record status while deploying
  - Use kubernetes annotation for tracking
  - Dynamic dependency management remains unsolved

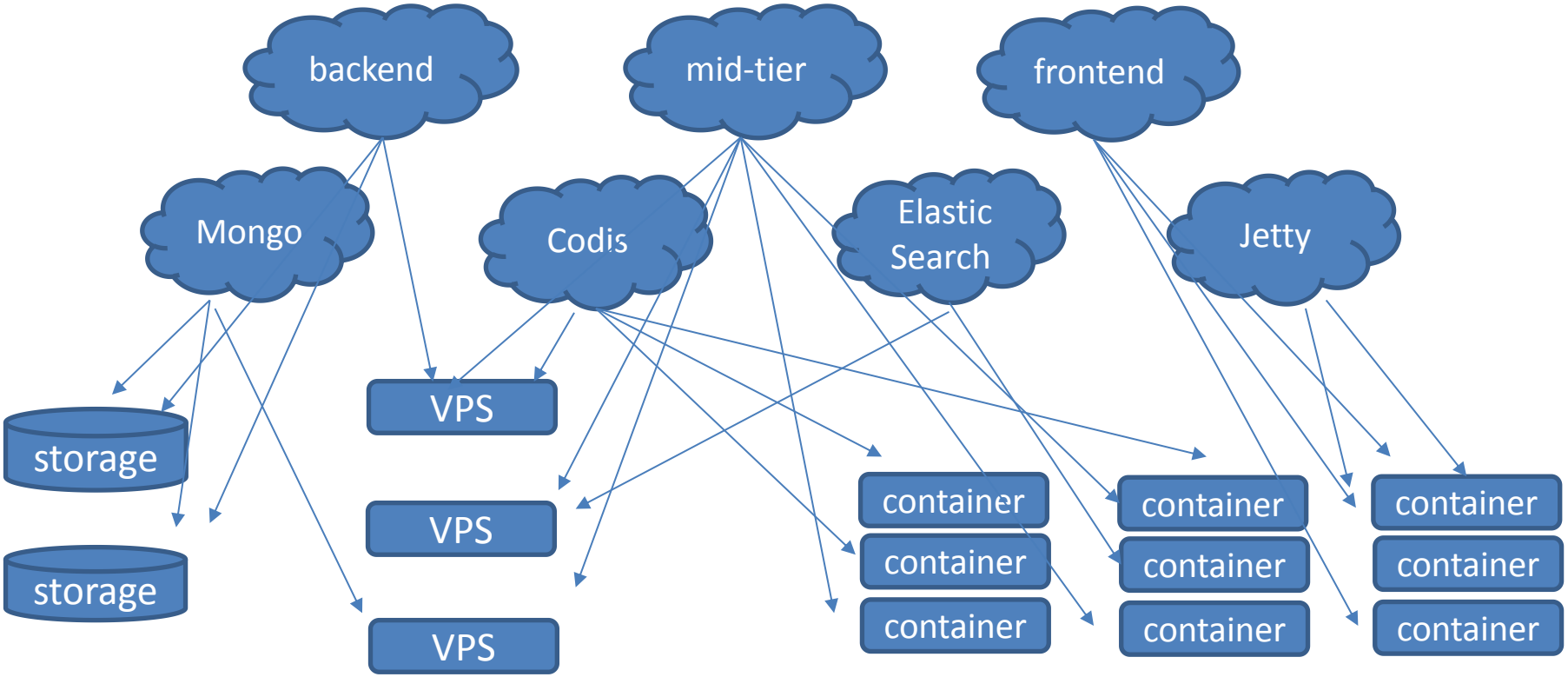
```
integration:
  image: golang
  commands:
    - go test ./..

deploy:
  - cluster: first-cluster
    namspace: test
    resources:
      cpu: "100m"
      mem: "512Mi"
    rollout:
      - cluster: first-cluster
        namespace: prod
```



Design: Managing Solutions Not Containers

How Cloud Treats Their Customers Today

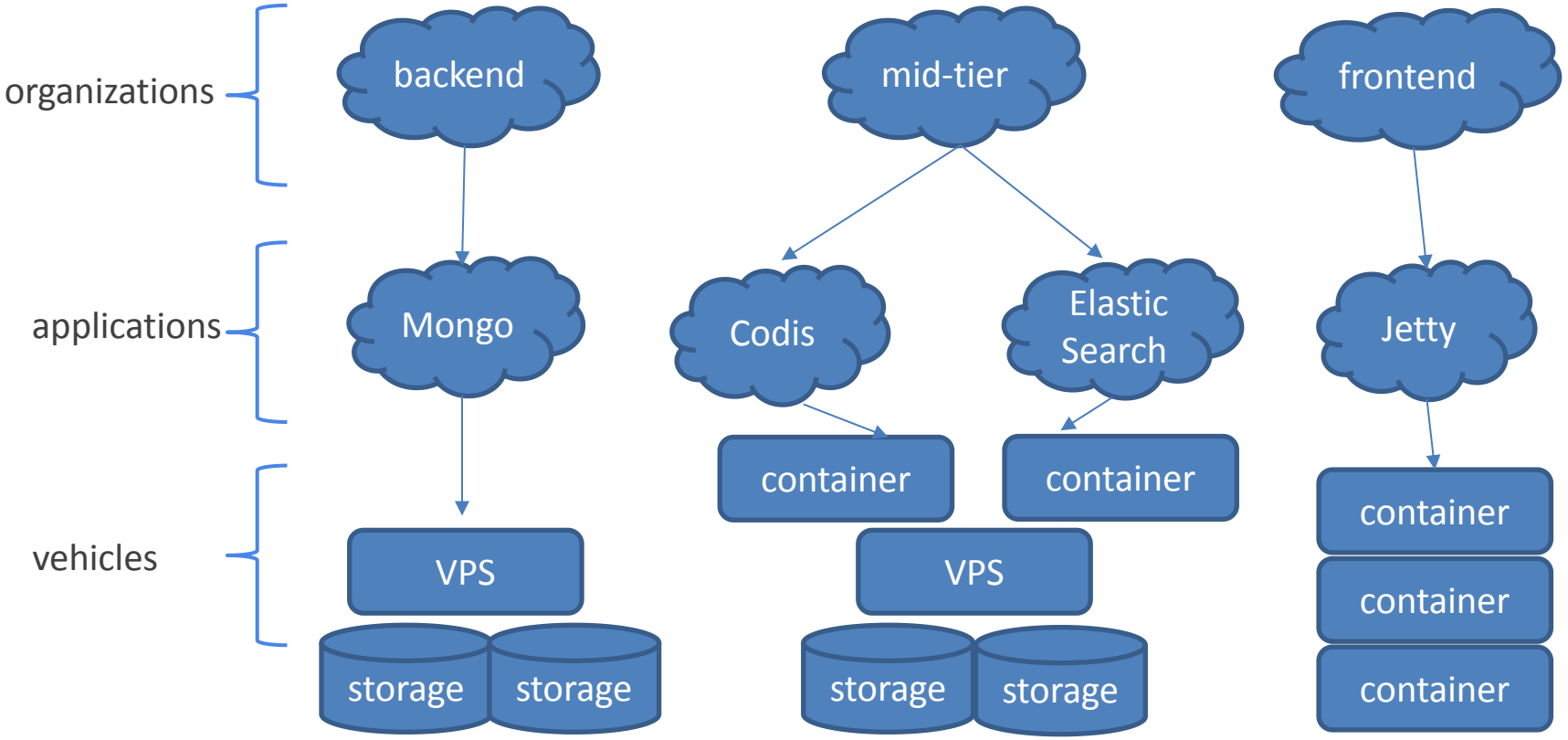
- Users have to map *logical solutions* to the list of *resources/containers*
- Containers DO NOT help!



Solution Manager: Solutions as 1st Class Citizen

Key benefits

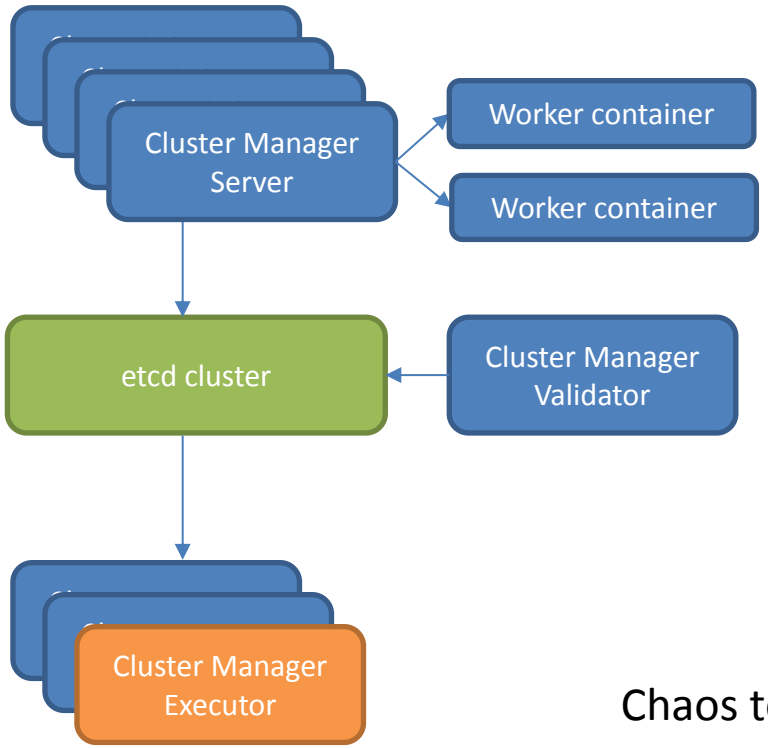
- Includes both workloads and infrastructure and the *topology*
- *Additional* higher level meta-data and management interfaces



Implementation: Stateful -> Stateless

Rolling update is great, when:

- you want to test multiple versions of code or configuration
- you want to update application without service interruption



- Restart creating cluster
  - Easy, but bad user experience

- Pick up where it left
  - Error prone

- Graceful termination
  - 1min is too small and too large

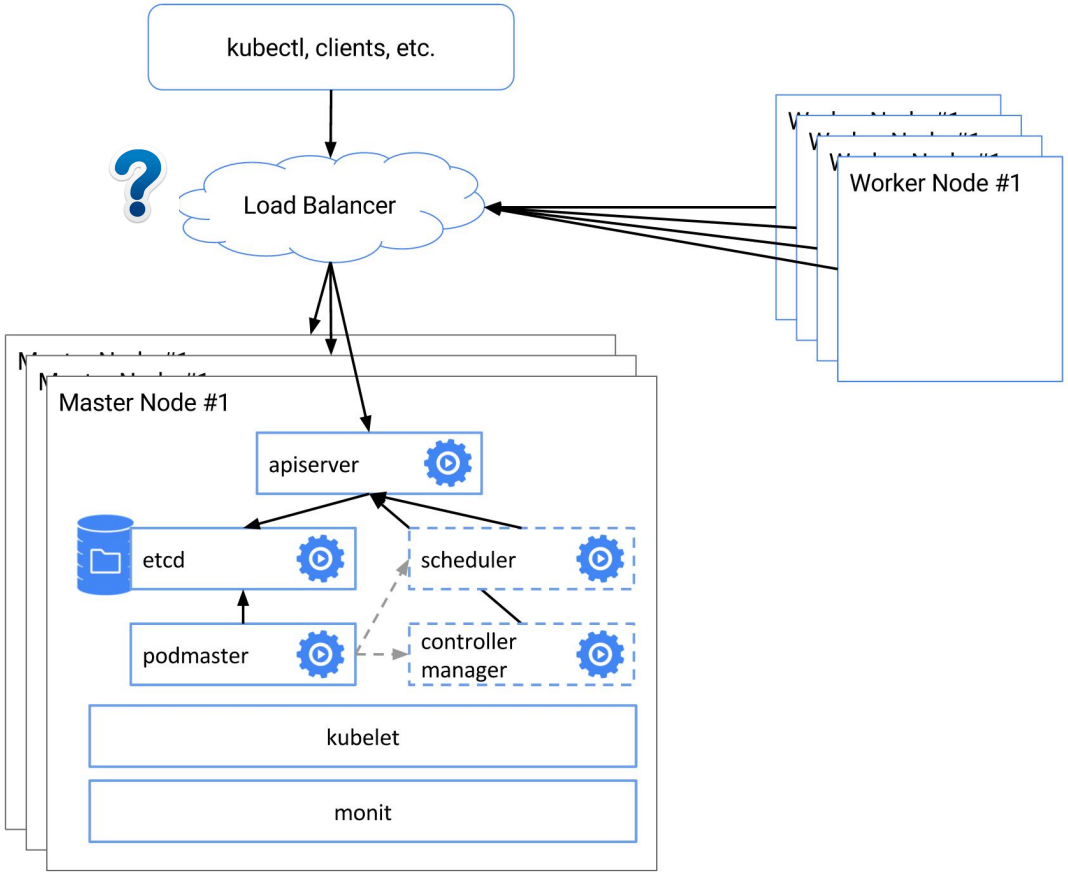
- A new docker container
  - Spawn a new docker container

Chaos testing: https://github.com/gaia-adm/pumba

Implementation: High Availability and Load balancing

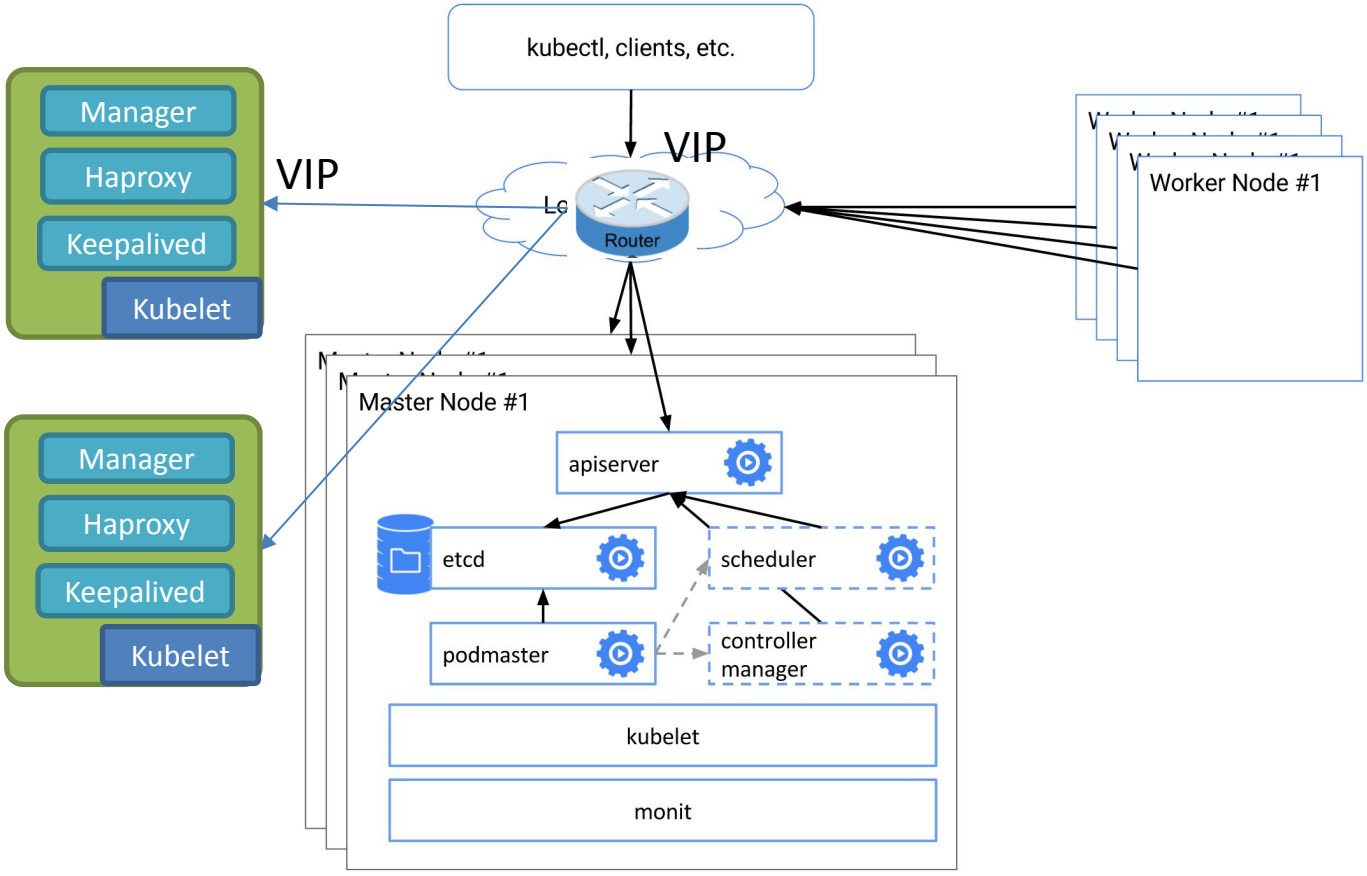
Kubernetes supports:

- etcd cluster for storage
- Multiple API servers
- Master elected scheduler and controller manager
- Kubelet babysitter



Implementation: High Availability and Load balancing

High Availability:



About Us: Cloud Team from Google + Amazon + CMU

- **CEO | Xin Zhang**
- Ex-***Googler*** specialized on Google private cloud, GAE and GCE, received multiple spot bonuses from several ***Google VPs***
- ***CS Ph.D from CMU*** specialized in distributed systems and security

- **COO | Jiayao Han**
- Experienced series ***Entrepreneur*** in the US
- ***Four degrees*** in Information Science, Law, Art, History from University of Pittsburgh

- **Chief Architect | Pengcheneg Tang**
- Ex-***Amazon*** engineer and expert in Docker and Kubernetes
- ***CMU*** ECE

- **CTO | Deyuan Deng**
- Ex-***Googler*** and ***top open-source*** Docker and container cluster contributor
- ***1st Prize*** in ***World*** Robotics Competition
- ***CMU*** ECE

- **Chief Data Scientist | Zeyu Zheng**
- Ex-***Googler*** specialized in Big Data
- ***ACM*** competition team lead
- ***CMU*** Computer Science


Caicloud
才云科技