Building Container-based Cluster Management Systems

邓德源

About Me

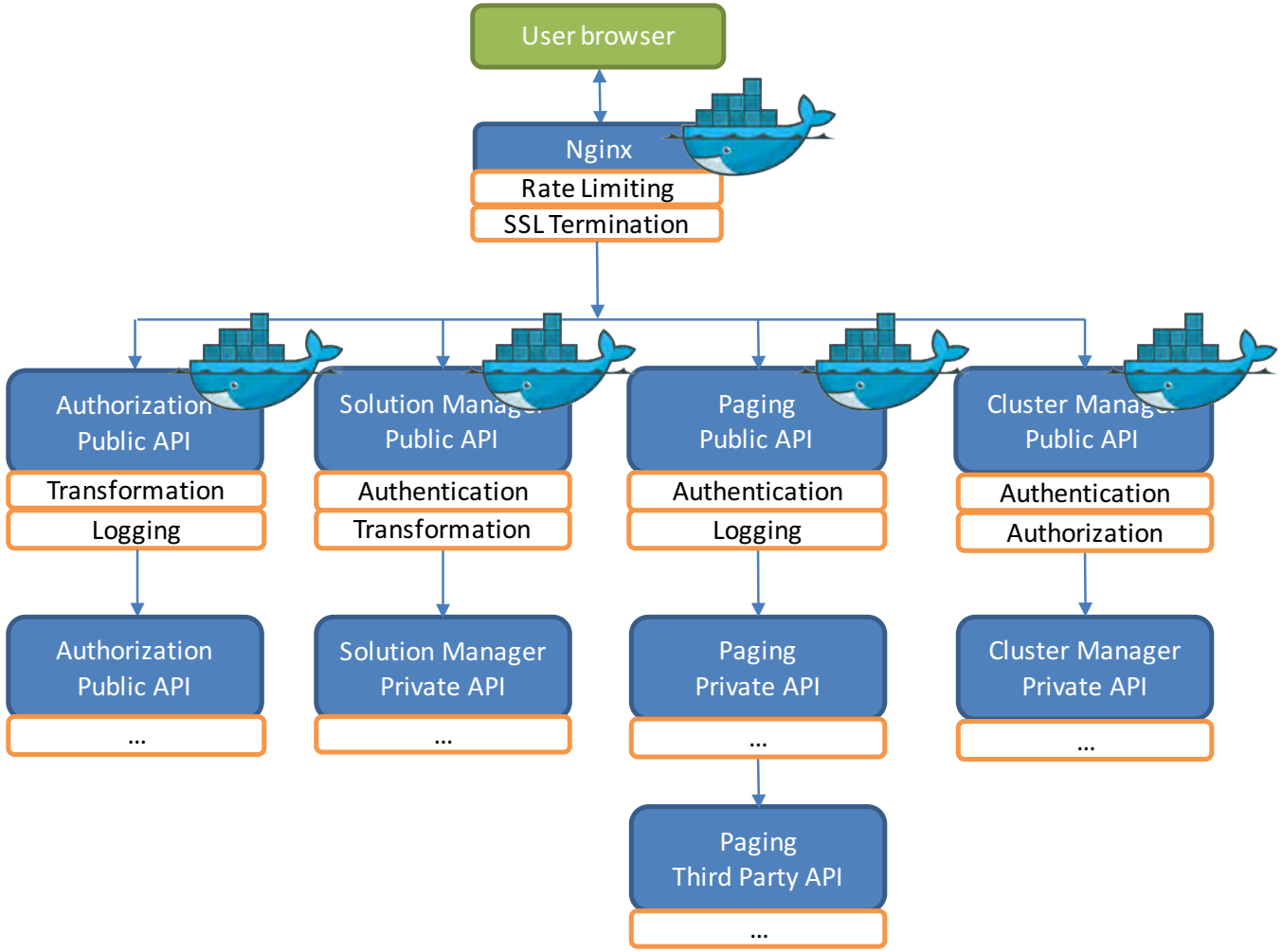
- 电子科大：计算机、算法、控制理论、机器人
- 卡内基梅陇大学：计算机、操作系统、分布式系统、存储系统
- FusionIO：存储系统
- 谷歌：集群管理系统及生态

Architecture

Caicloud
才云科技

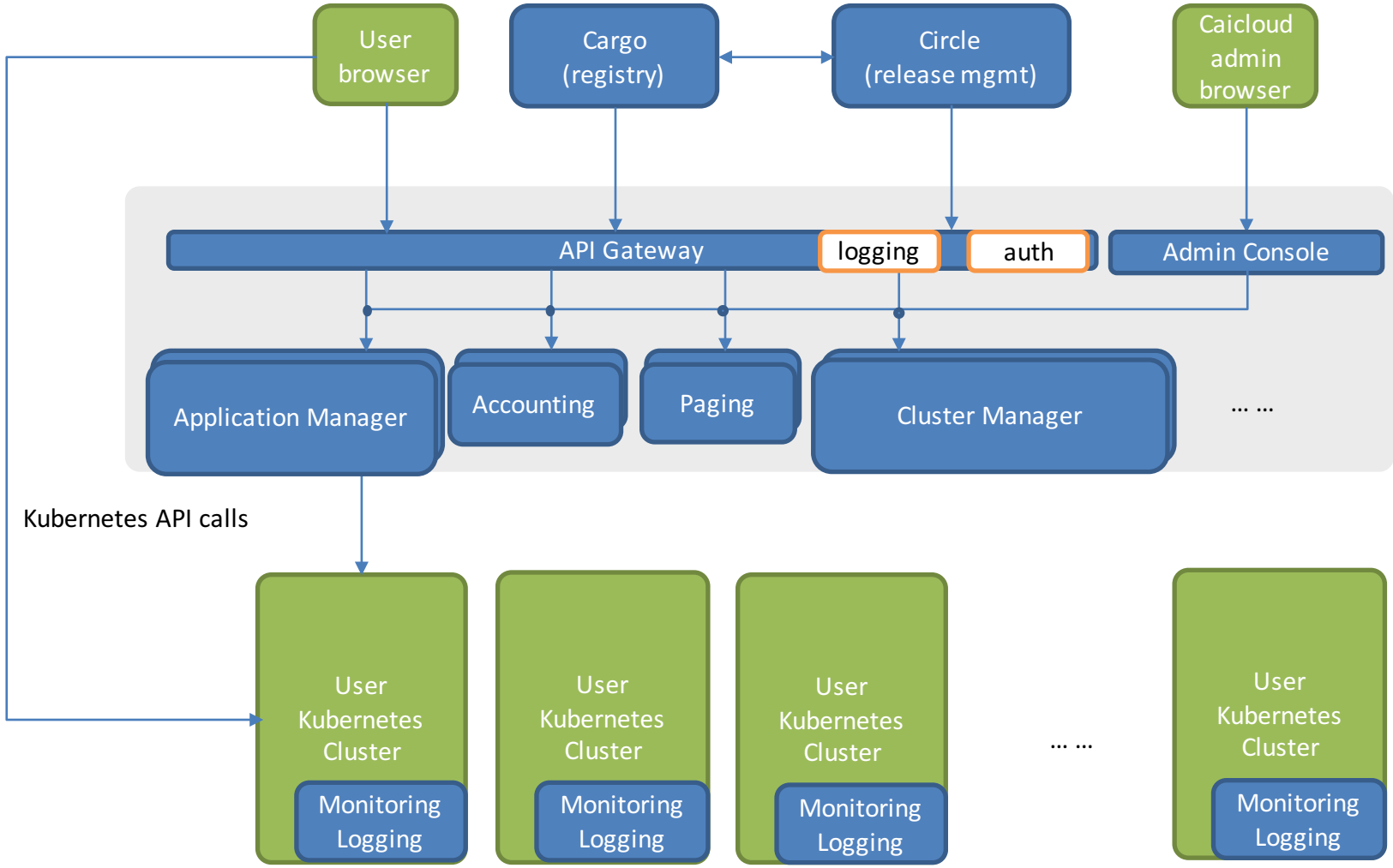
Problem:

- Duplicate Functionalities
- System tends to be monolithic
- Complex frontend logic due to varied API



Architecture Revamp

Caicloud
才云科技



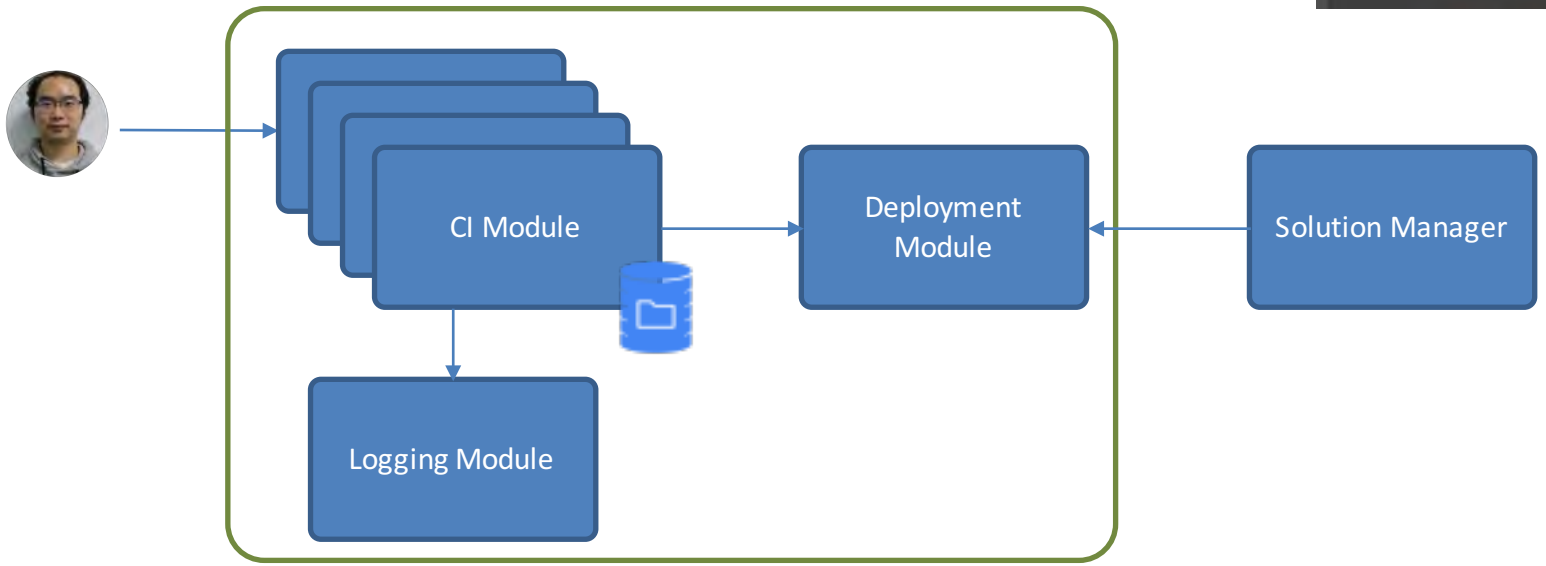
Circle: Goal

- 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?

Circle: Goal

- 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

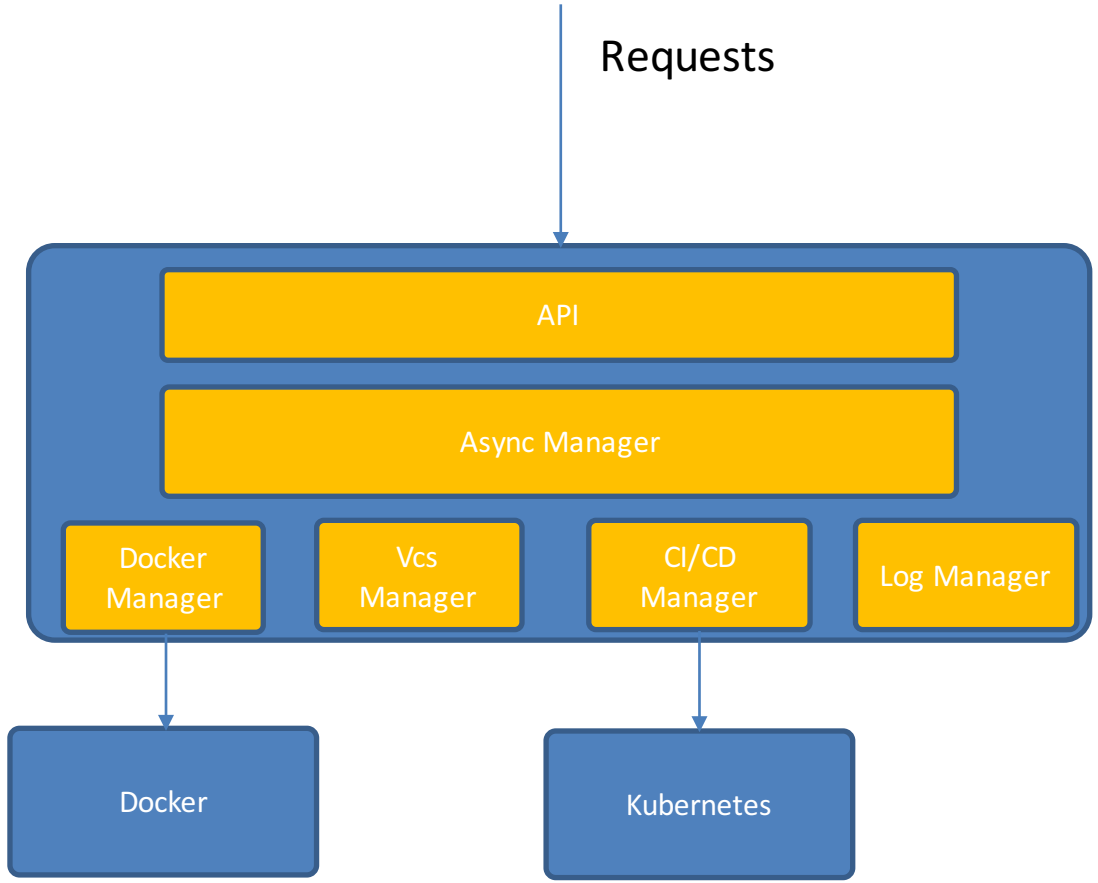
```
integration:
  image: golang
  commands:
    - go test ./..

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



Circle: POC

Caicloud
才云科技

- API:
  - Handle user requests, validation, etc
- Async Manager:
  - Asynchronously executing operations
  - go channel -> message queue
- Vcs Manager:
  - Handles version control tools
  - Stateless: run and go
- Log Manager
  - Build log streaming
  - local file -> kafka, web socket
- Docker Manager
  - Manage docker build/push
  - ?



Multiple Docker

Caicloud
才云科技

- Single Docker Daemon
  - Limited concurrent build/push: 2cores, 4G => 15 concurrent pushes
  - Isolation for multiple build/run
  - Race condition for images
  - Risk of leaking private images

- DIND
  - Not that confident

- Virtual Machine
  - Complete isolation
  - Slow, and lack of virtualization support on cloud environment
  - Provisioning overhead
  - VM based container is faster, but doesn't solve image problem

- Multiple Docker Daemon
  - Complete isolation
  - Lightweight: 2cores, 4G => 50 daemons + concurrent pushes

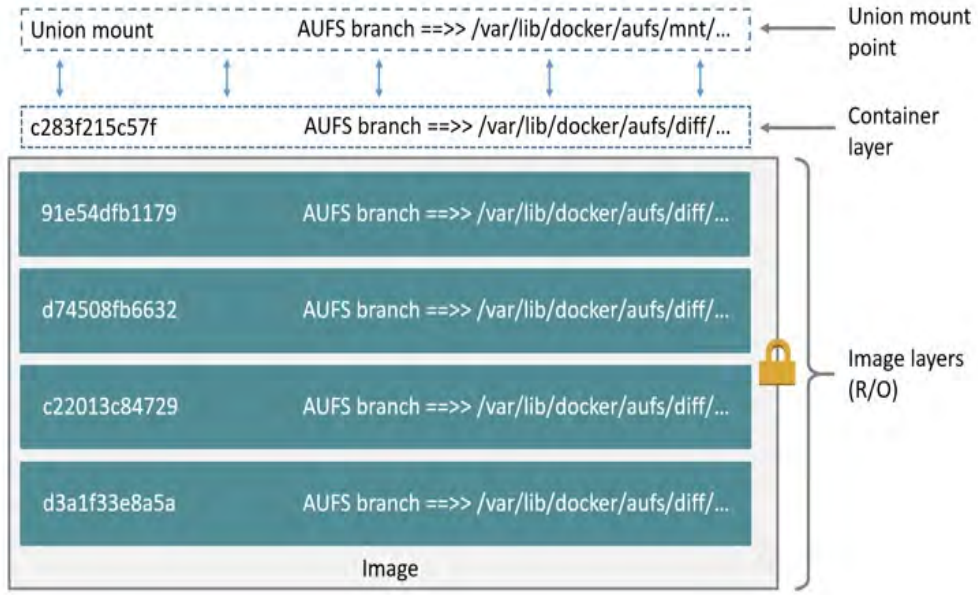
Multiple Docker

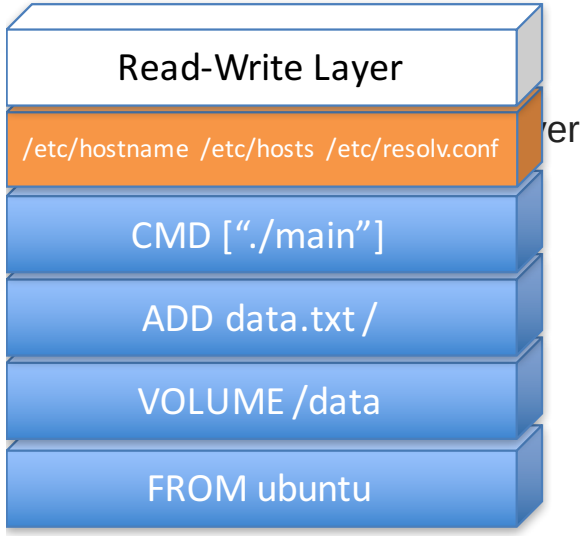
Caicloud
才云科技

- Use different working directory for docker daemon

```
docker@kube-dev:~$ sudo ls /var/lib/docker
aufs        containers  image       network     tmp         trust       volumes
```

```
docker@kube-dev:~$ sudo ls /var/run/docker
execdriver  netns
```

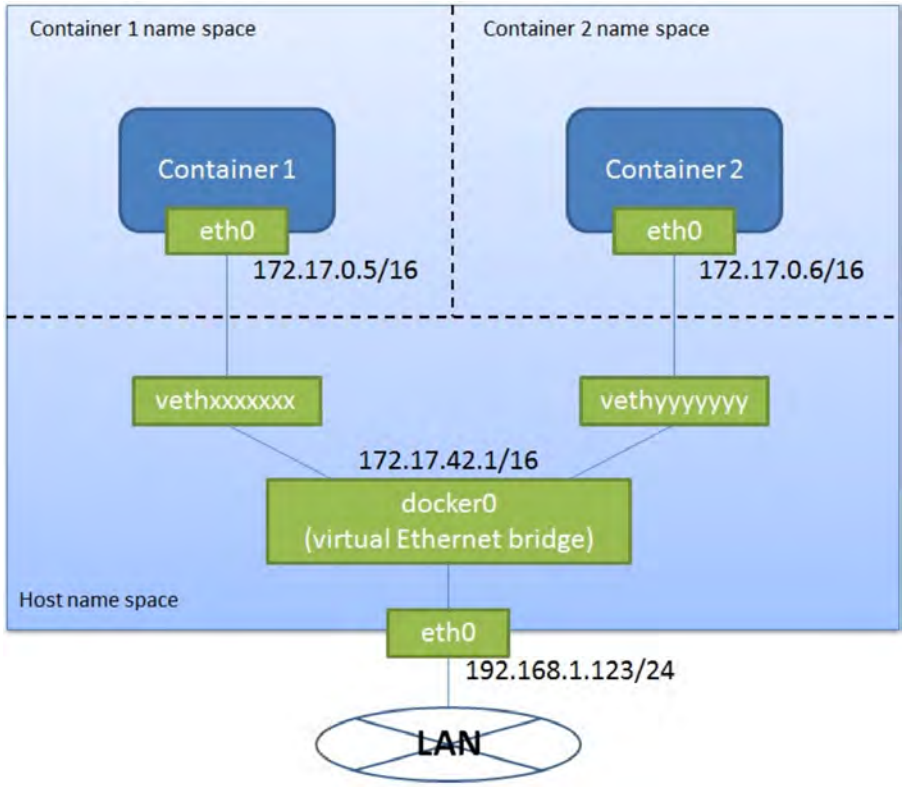


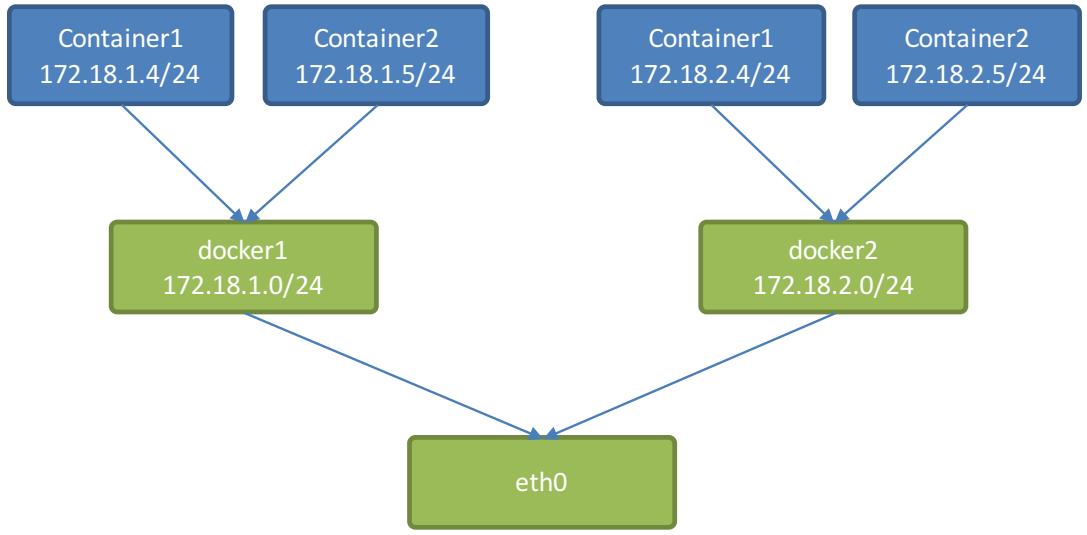


Multiple Docker

Caicloud
才云科技

- Different network setting for docker daemon






```
-A POSTROUTING -s 172.18.2.0/24 ! -o docker2 -j MASQUERADE
-A POSTROUTING -s 172.18.1.0/24 ! -o docker1 -j MASQUERADE
-A FORWARD -o docker2 -j DOCKER
-A FORWARD -o docker1 -j DOCKER
-A FORWARD -o docker2 -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
-A FORWARD -o docker1 -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
-A FORWARD -i docker2 ! -o docker2 -j ACCEPT
-A FORWARD -i docker2 -o docker2 -j ACCEPT
-A FORWARD -i docker1 ! -o docker1 -j ACCEPT
-A FORWARD -i docker1 -o docker1 -j ACCEPT
```

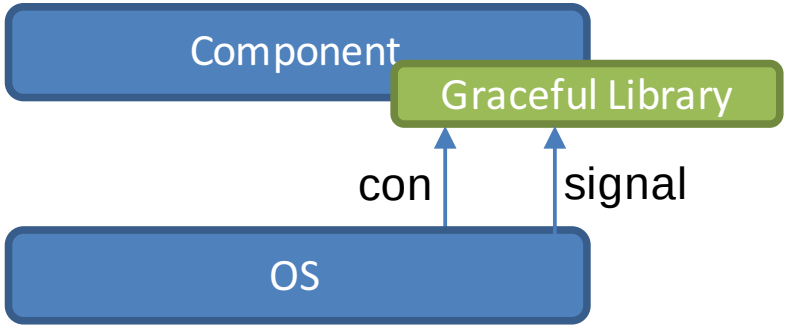
Graceful Termination

Caicloud
才云科技

Rolling update is great, when:

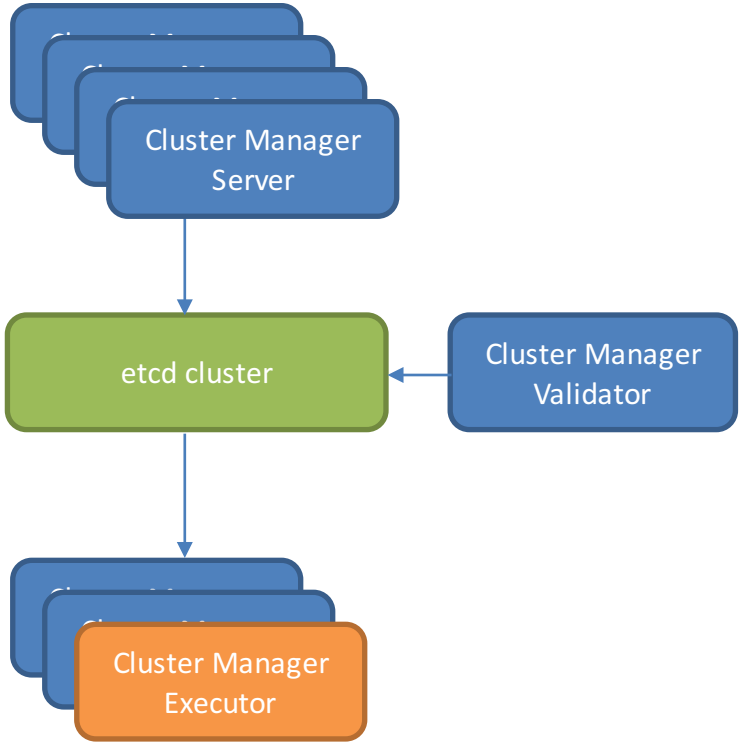
- You want to test multiple versions of code or configuration
- You want to update application without service interruption

However, you want to make sure there is really no “service interruption”




```
spec:
  containers:
  - image: index.caicloud.io/caicloud/go-echoserver
    imagePullPolicy: Always
    name: echoserver
    resources:
      limits:
        cpu: "1"
        memory: 256Mi
      requests:
        cpu: 100m
        memory: 120Mi
    securityContext: {}
    terminationMessagePath: /dev/termination-log
    volumeMounts:
    - mountPath: /var/run/secrets/kubernetes.io/serviceaccount
      name: default-token-gzcqo
      readOnly: true
  restartPolicy: Always
  terminationGracePeriodSeconds: 60
  volumes:
  - name: default-token-gzcqo
    secret:
      secretName: default-token-gzcqo
```

Task Offloading



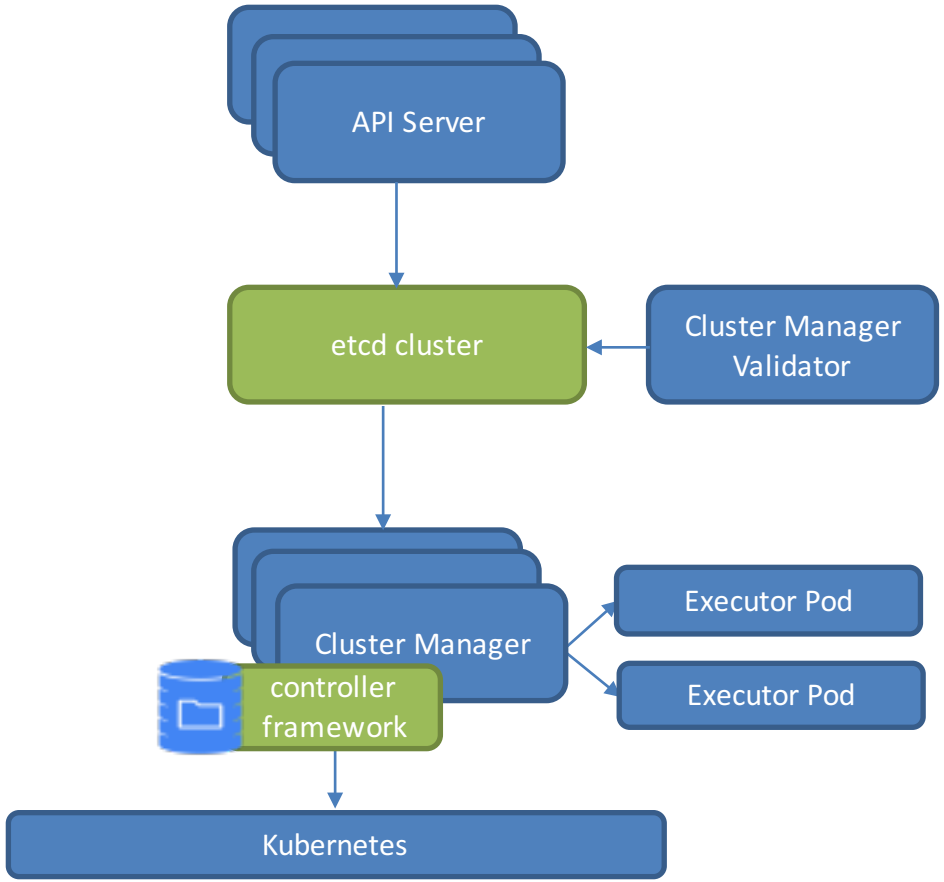
A service with a lot of states:

- Operation status
- Execution status
- Cluster status
- State transformation

Solution

- Distributed transaction?
- A state machine?
- State reconstruction?
- Graceful termination?
- Or just use k8s!

Task Offloading



Controller framework:

- List operation to reconcile cache
- Watch operation to reflect change ASAP
- Event system and hooks
- Cache reconstruction during restart

Kubernetes:

- Manage pod status
- Resource isolation
- Concurrency control

Caicloud
才云科技

Thank you!

http://caicloud.io info@caicloud.io