KubeCon | CloudNativeCon

Europe 2019

Embracing Upstream Kubernetes in Web Scale Organization

Lei Zhang, Alibaba & Jun Chen, Ant Financial

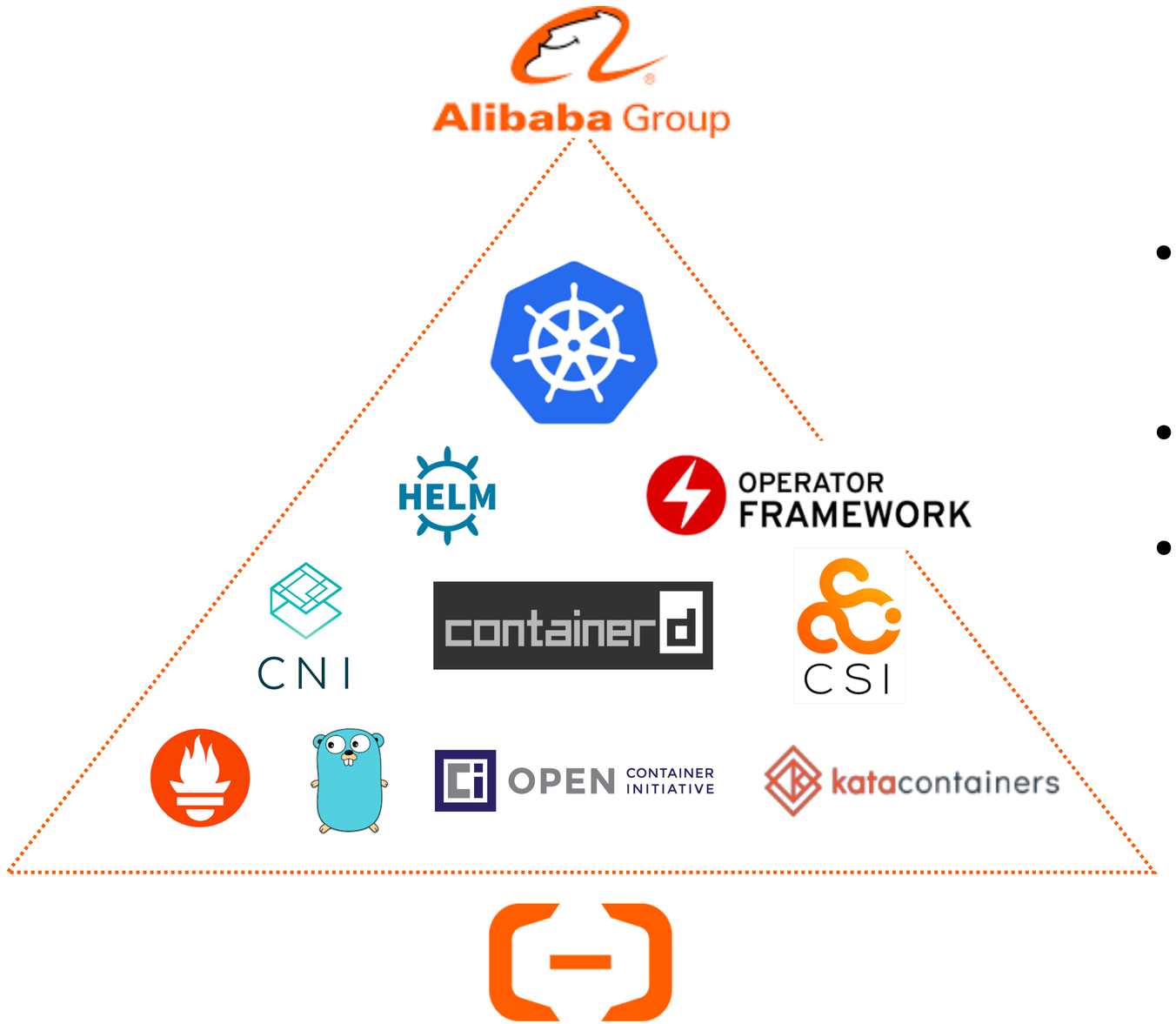
Alibaba Cloud | Worldwide Cloud Services Partner

CONTENT

- Background
- Hard Multi-tenancy Architecture
- The “Container Headache”?
- Workload Management
- Workload Predictability
- Scalability Verifying & Trouble Shooting
- Dance with Upstream

Background

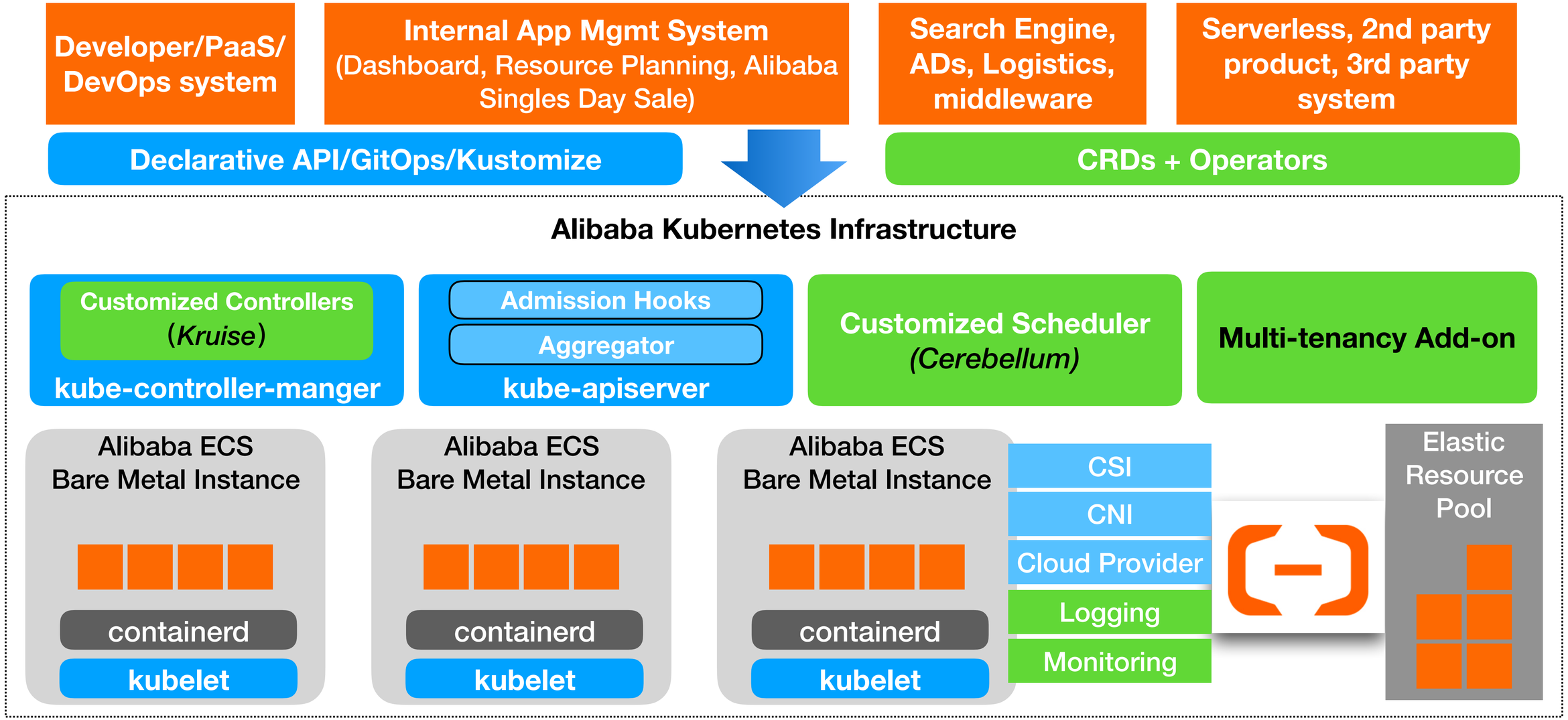
Alibaba Cloud | Worldwide Cloud Services Partner



- **Our team serves both Alibaba Cloud & Alibaba Group**
- Alibaba Cloud: managed K8s service
- Alibaba Group: several internal K8s cluster (5k~10k nodes each) to serve world's largest e-business platform
  - **The main topic today :-)**

Architecture - Large Cluster to Serve All Tenants

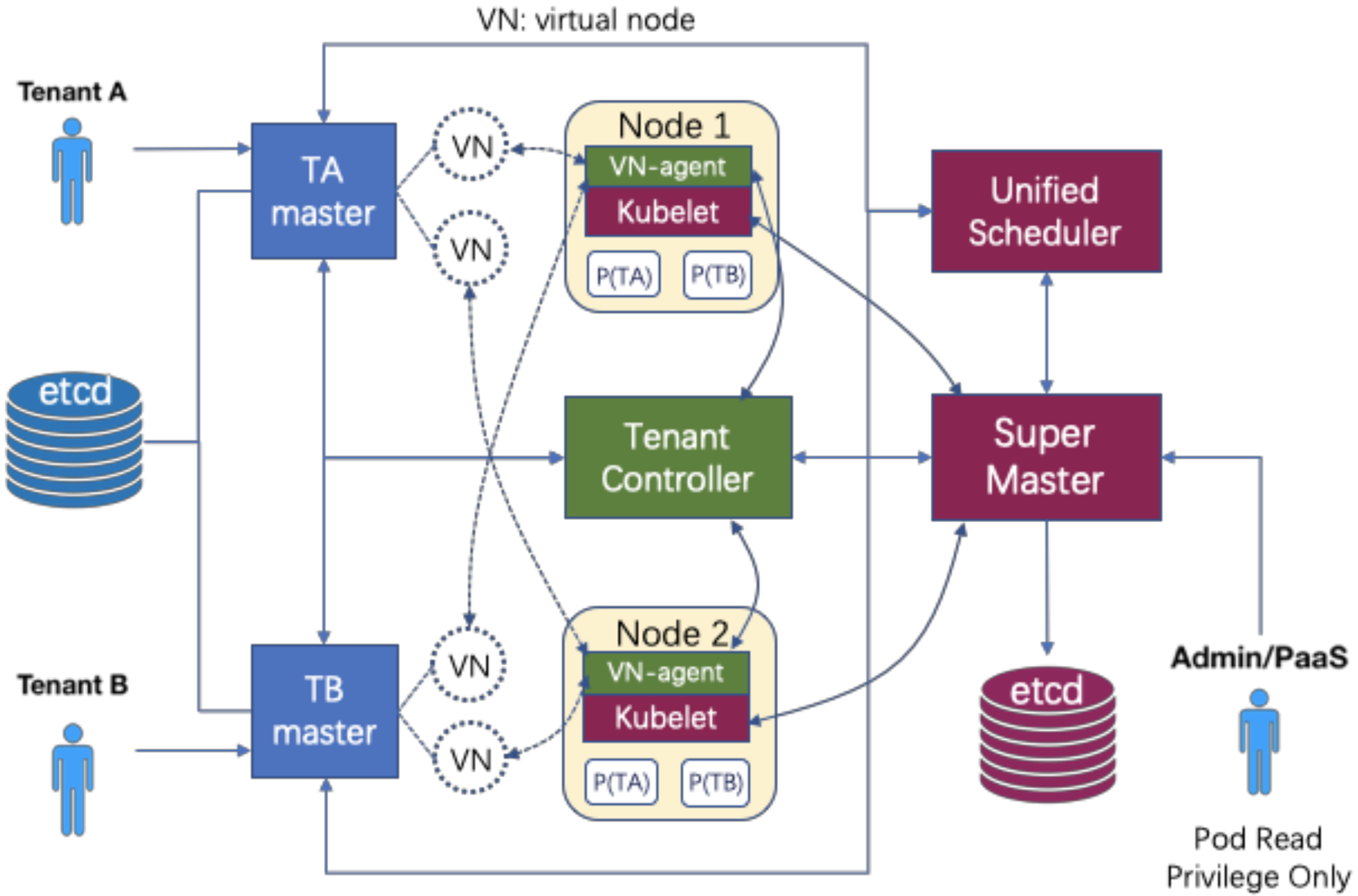
Alibaba Cloud | Worldwide Cloud Services Partner



“Virtual Cluster” Hard Multi-tenancy Model

Alibaba Cloud | Worldwide Cloud Services Partner

- **Per “tenant” per “tenant cluster”**
- Every “tenant cluster” is composed by a **dedicated K8s control plane** and **several “virtual nodes”**
  - virtual nodes are “virtualized” from real nodes by VN-agent (**a proxy of kubelet**)
    - So tenants workloads **could share** the same “super” K8s cluster
- Tenant control plane do not have kube-scheduler
  - **Unified Scheduler** to achieve high resource utilization by job co-location

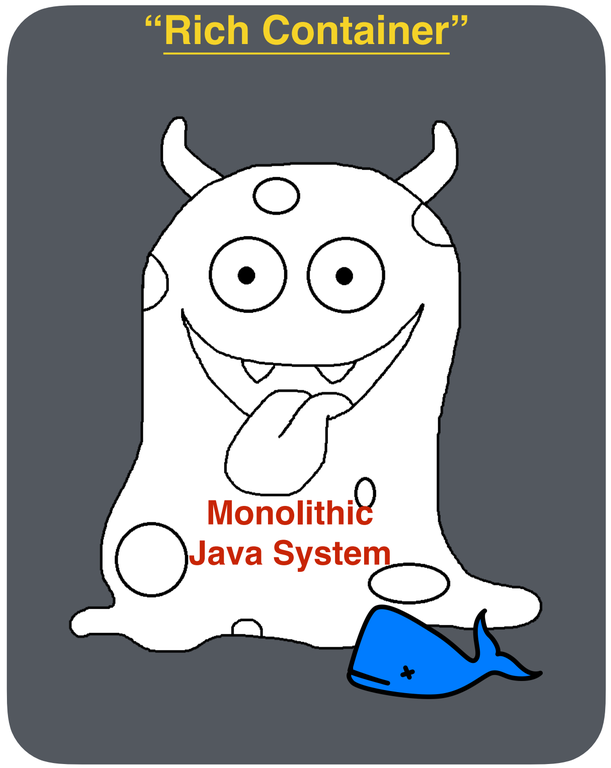


[Please check this upstream design doc for more details]

Open source at KubeCon NA 2019!

“Container Headache”

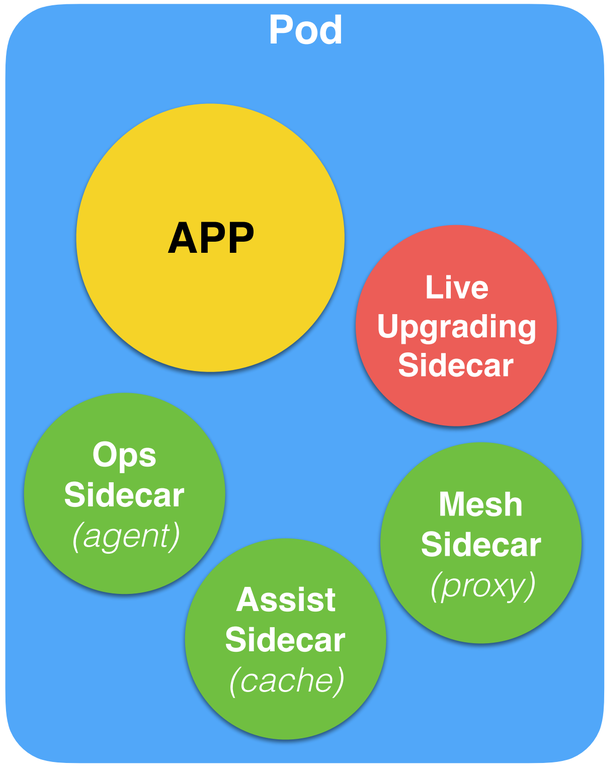
Alibaba Cloud | Worldwide Cloud Services Partner



- Before 2018
  - **Anti-container patterns**
    - *“Rich Container”*
      - You can find everything inside the container
        - app, start/stop scripts, sshd, log, monitoring, cache, VIP, DNS, proxy, service mesh agent …
        - *PID 1 process is Systemd*
      - **Aha, it’s basically a container but acts as a VM**
  - **Traditional operating workflow**
    - Start container -> SSH into container -> Start the app
    - Log files & user data are distributed everywhere in the container
  - **In-house orchestration & scheduling system**

Fix “Container Headache”

Alibaba Cloud | Worldwide Cloud Services Partner



- Shared volume
- Different resource QoS
- Fine-grained lifecycle control & health check

```
apiVersion: v1
kind: Pod
spec:
  containers:
  - env:
    - name: ali_start_app
      value: "no”
    name: main
    lifecycle:
      postStart:
        exec:
          command:
          - /bin/sh
          - -c
          - for i in $(seq 1 60); do [ -x /home/admin/.start ] && break ; sleep 5
            ; done; sudo -u admin /home/admin/.start>/var/log/kubeapp/start.log 2>&1
            && sudo -u admin /home/admin/health.sh>>/var/log/kubeapp/start.log 2>&1
      preStop:
        exec:
          command:
          - /bin/sh
          - -c
          - sudo -u admin /home/admin/stop.sh>/var/log/kubeapp/stop.log 2>&1
    livenessProbe:
        exec:
            command:
            - /bin/sh
            - -c
           - sudo -u admin /home/admin/health.sh>/var/log/kubeapp/health.log 2>&1
       initialDelaySeconds: 20
        periodSeconds: 60
        timeoutSeconds: 20
```

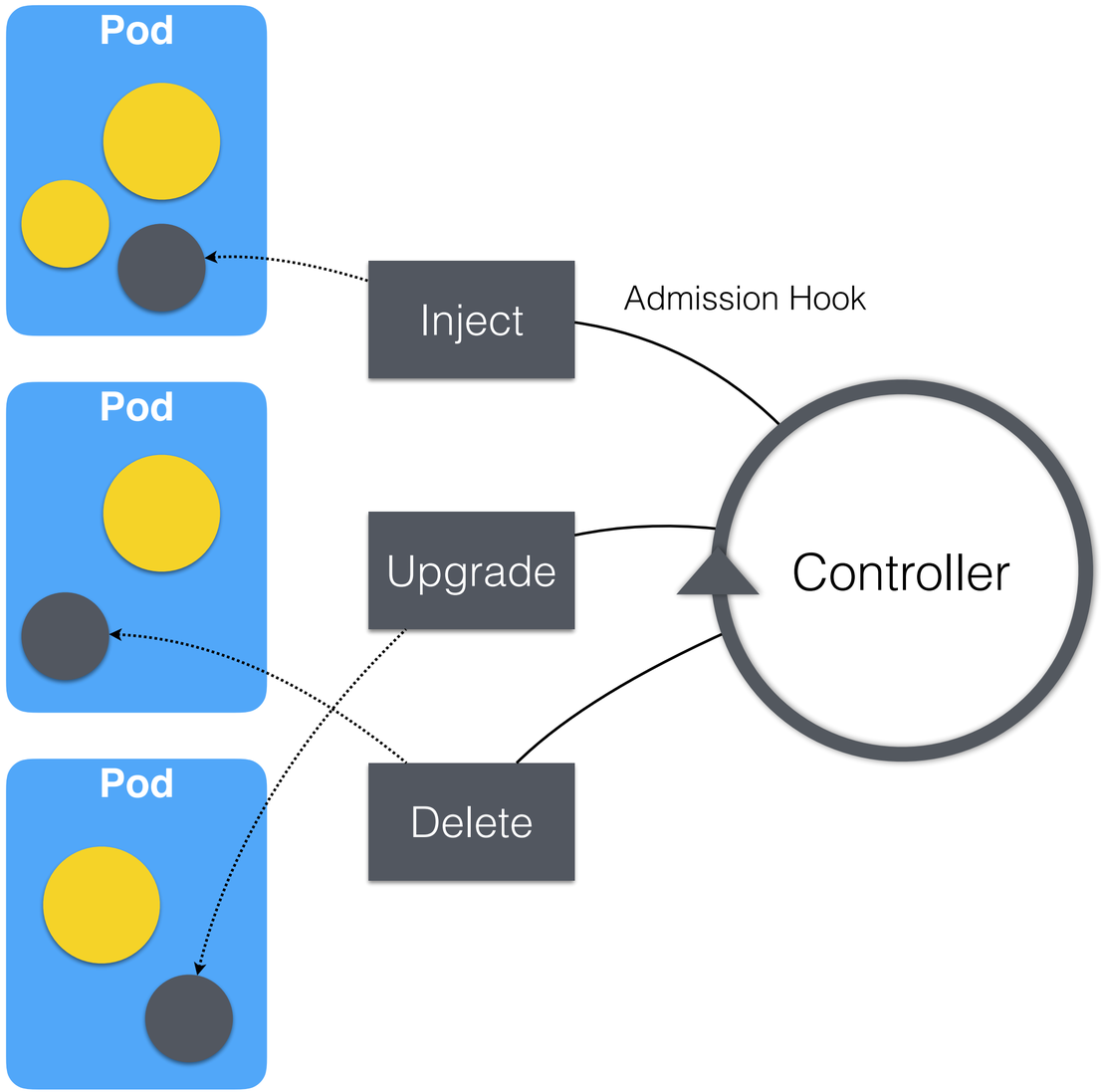
App start script

App stop script

Health check

Sidecar Operator

Alibaba Cloud | Worldwide Cloud Services Partner

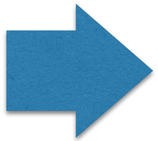


- When we have thousands sidecars, we'll need:
- A SidecarSet CRD:
  - Describe all sidecars need to be operated
- A SidecarOperator:
  1. Inject sidecar containers to selected Pods
  2. Upgrade sidecar containers following rollout policy when SidecarSet is updated
  3. Delete sidecar containers when SidecarSet is deleted

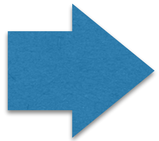
Workloads Management

Alibaba Cloud | Worldwide Cloud Services Partner

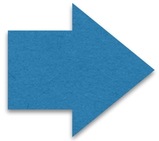
- **Kubernetes Application** = **YAML**
- **Kubernetes Workloads** = **Operating Model**
  - StatefulSet
  - Deployment
  - Job
  - CronJob
  - DaemonSet



- **Pre-defined models of**
  - Rollout Policy
  - Instance Recovery
  - Batch Deploy
  - Blue-Green Deploy
  - Canary Deploy



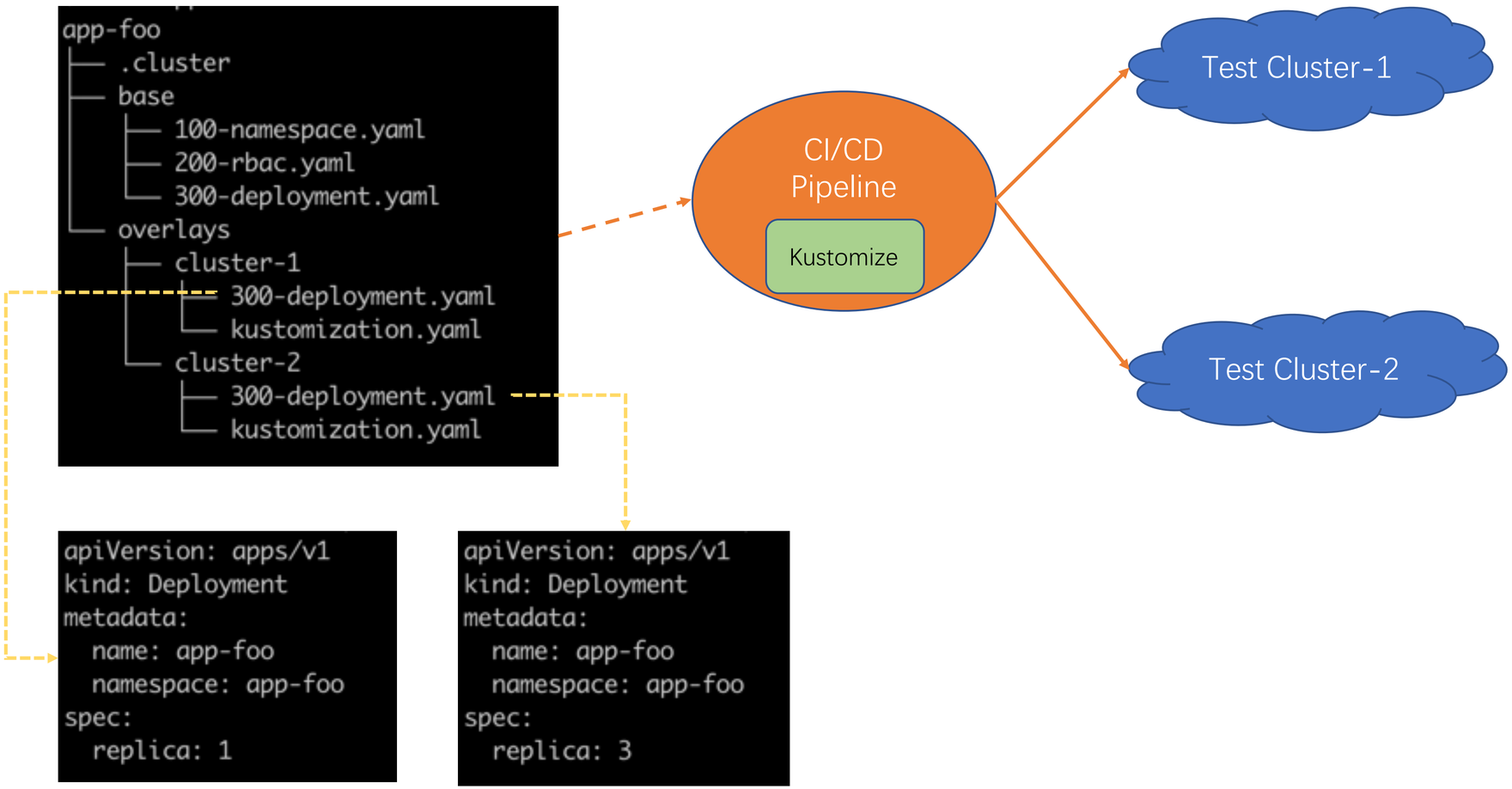
- **Lessons learned:**
  - They are well defined & convenient;
    - may not fit to all cases though …



- **Lessons learned:**
  - Managing YAML files in large cluster is a nightmare

GitOps + Kustomize = Awesome!

Alibaba Cloud | Worldwide Cloud Services Partner

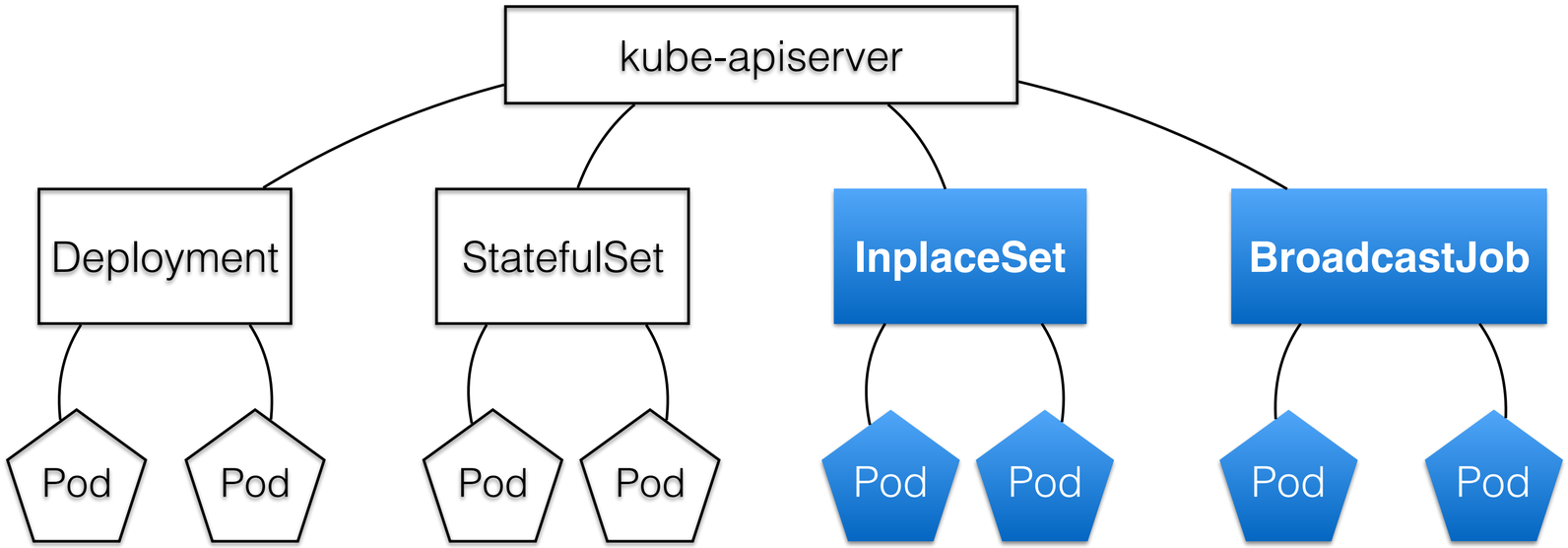


Git + Base YAML + patch YAML = **Easy YAML mgmt in large K8s cluster**

Kruise: Kubernetes Workloads Advanced

Alibaba Cloud | Worldwide Cloud Services Partner

- A fleet of customized **CRD + controllers** that operate applications at web scale.
- **Pluggable, repeatable and Kubernetes native** (Declarative API + Controller Pattern)
- **100 % Open Source** (very soon!)



Kruise - InplaceSet

- **InplaceSet:**
  - **Predictability** is critical in web-scale cluster
    - We prefer **In-Place-Upgrade**, because with thousands of pods reshuffled across cluster:
      - Topology changes, image re-warm, unexpected overhead, resource allocation churn …
  - Generally, we ❤️ *StatefulSet*, **but**:
    - *SS* will still **tear down** pods during rolling upgrade
    - Less rollout strategy than Deployment

	Deployment	StatefulSet	**InplaceSet**
Ordering	No	Yes	Yes
Naming	Random	Ordered	Ordered
PVC reserve	No	Yes	Yes
Retry on other nodes	No	No	Yes
Rollout policy	Rolling, Recreate	Rolling, On-delete	Rolling, On-delete, **In-place**
Pause/Resume	Yes	No	Yes
Partition	No	Yes	Yes
Max unavailable	Not yet	Yes	Yes
Pre/Post update hook	No	No	Yes

InplaceSet = A in-place "StatefulSet" with more rollout strategies

Scalability Matters

Alibaba Cloud | Worldwide Cloud Services Partner

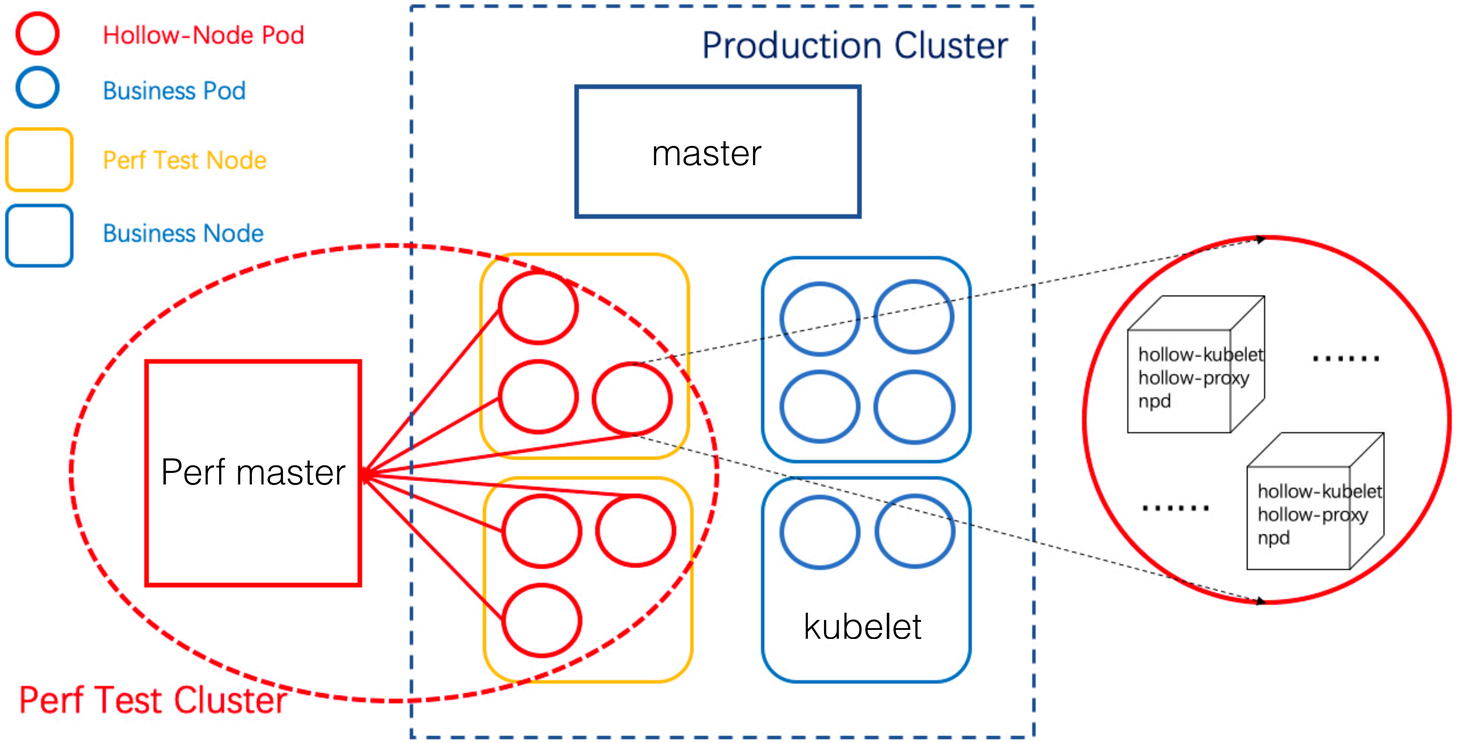
- Scalability boundary of upstream K8s (v1.14)
  - No more than 5k nodes
  - No more than 150k total pods
  - No more than 300k total containers
  - No more than 100 pods per node

- Scalability goal in our web-scale cluster
  - More than 10k nodes
  - More than 300k pods
- Non-goal:
  - Total containers & pods per node

- Question:
  - **How to discover scalability issue in 10k nodes cluster?**

Performance Benchmark Toolkit

Alibaba Cloud | Worldwide Cloud Services Partner



- **kubemark** with HTTP interface
  - **Hollow-Node** Pods
    - cmd/kubemark/hollow-node.go
- Taint and drain nodes for perf test, and run it
- Typical test cases in 10k nodes cluster:
  - Start up time during scaling pods
  - Time of creating and deleting pods
  - Pod listing RT
  - Failure counts

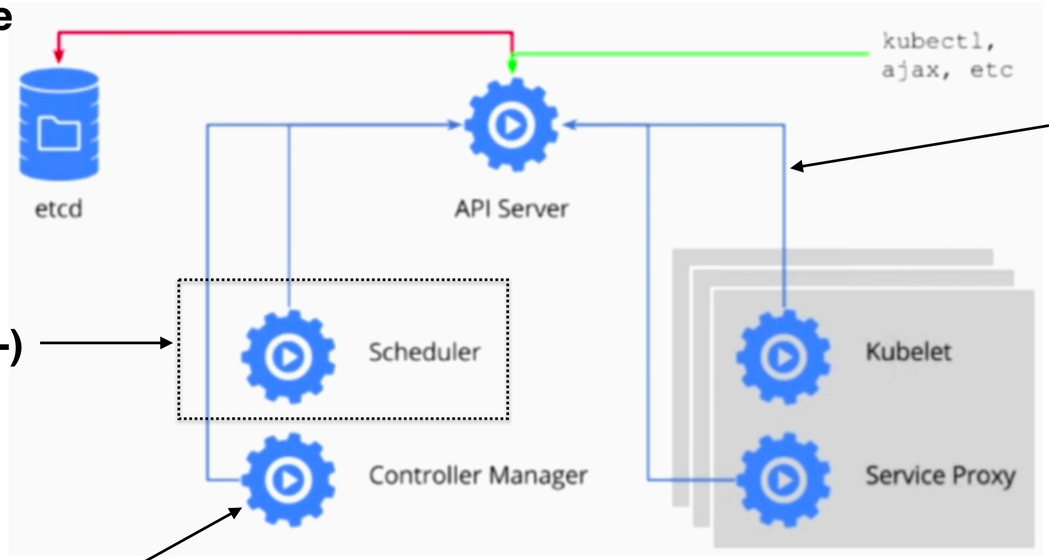
How to run?

```
curl -X POST -H "Content-Type: application/json"  \
"https://k8s-performance-toolkit.alibaba-inc.com/api/kubemark/test" \
-d '{"test_focus":"\\[Feature:Performance\\]","test_skip":"handle","node_count":10000,"pods_per_node":30}'
```

Discover Performance Bottlenecks

Alibaba Cloud | Worldwide Cloud Services Partner

Concurrency, locking, data store



High pressure caused by large amount of data

Our own implementation, no worries :-)

Large amount of lister & watcher

Fix Performance Issue - etcd

- **etcd**
  - Periodic commit operation **does not block concurrent read** transactions: etcd-io/etcd#9296
  - **Fully allow concurrent large read**: etcd-io/etcd#9384
  - Improve index compaction blocking by using a copy on write clone to **avoid holding the lock for the traversal** of the entire index: etcd-io/etcd#9511
  - Improve lease expire/revoke operation performance: etcd-io/etcd#9418
  - Use **segregated hash map** to boost the freelist allocate and release performance: etcd-io/bbolt#141
  - Add **backend batch limit/interval** fields: etcd-io/etcd#10283
- Benchmark:
  - 100 clients, 1 million random key value pairs, 5000 QPS
    - Completion time: **~200s**
    - Latency: 99.9% in **97.6ms**

Fix Performance Issue - kube-apiserver

Alibaba Cloud | Worldwide Cloud Services Partner

- **kube-apiserver:** indexing, caching & reduce data scale
  - **Pod List Indexing**: ~35x improvement *(will be upstream soon)*
  - **Watch Bookmark**: k8s.io/kubernetes#75474 ***(New!)***
  - Cherry pick: k8s.io/kubernetes#14733 (incremental heartbeat), k8s.io/kubernetes#63606
- Benchmark:
  - 10k nodes, 100K exiting pods, scale 2000 pod
    - QPS: 133.3 pods/s, **99 %ile 3.474s**
  - On going: metrics data will crash Prometheus

We Prefer “non fork”

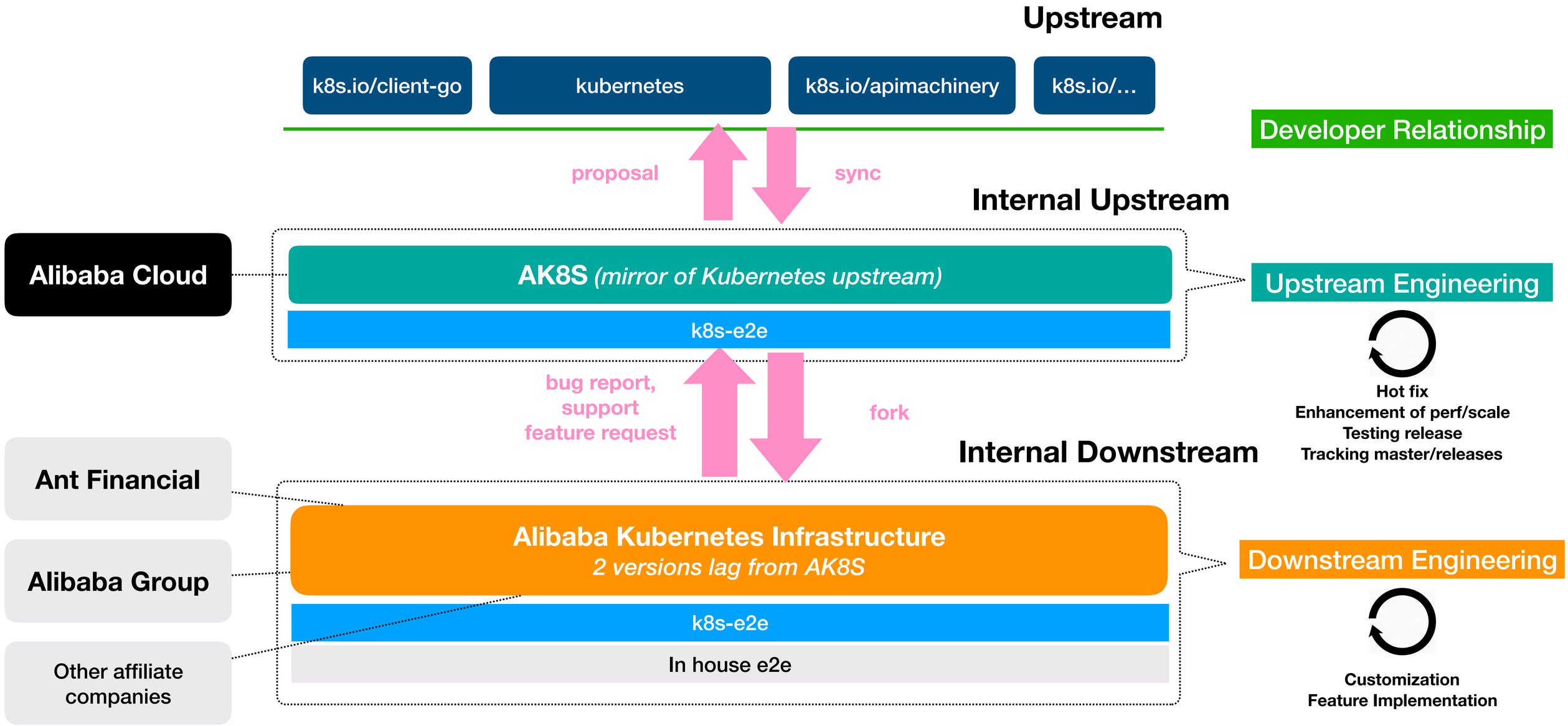
Alibaba Cloud | Worldwide Cloud Services Partner

- ***“fork”***
  - Lock down on specific K8s release, never upgrade
  - In-house/modified K8s API, hide/wrap K8s API
  - Bypass K8s core workflow
  - Bypass K8s interface (CSI, CNI, CRI)
  - Replace kubelet with some other agent
  - …

- ***“non fork”***
  - Keep upgrading with 2 releases lag with upstream
  - No API change
    - annotations, aggregator, CRD etc
  - Always respect K8s philosophy
    - Declarative API & Controller Pattern
  - Leverage K8s standard extensibility points
    - CNI, CSI, admission hook, initializer, extender etc
  - Respect kubelet & CRI

AK8S: A Tiger Team to Dance with Upstream

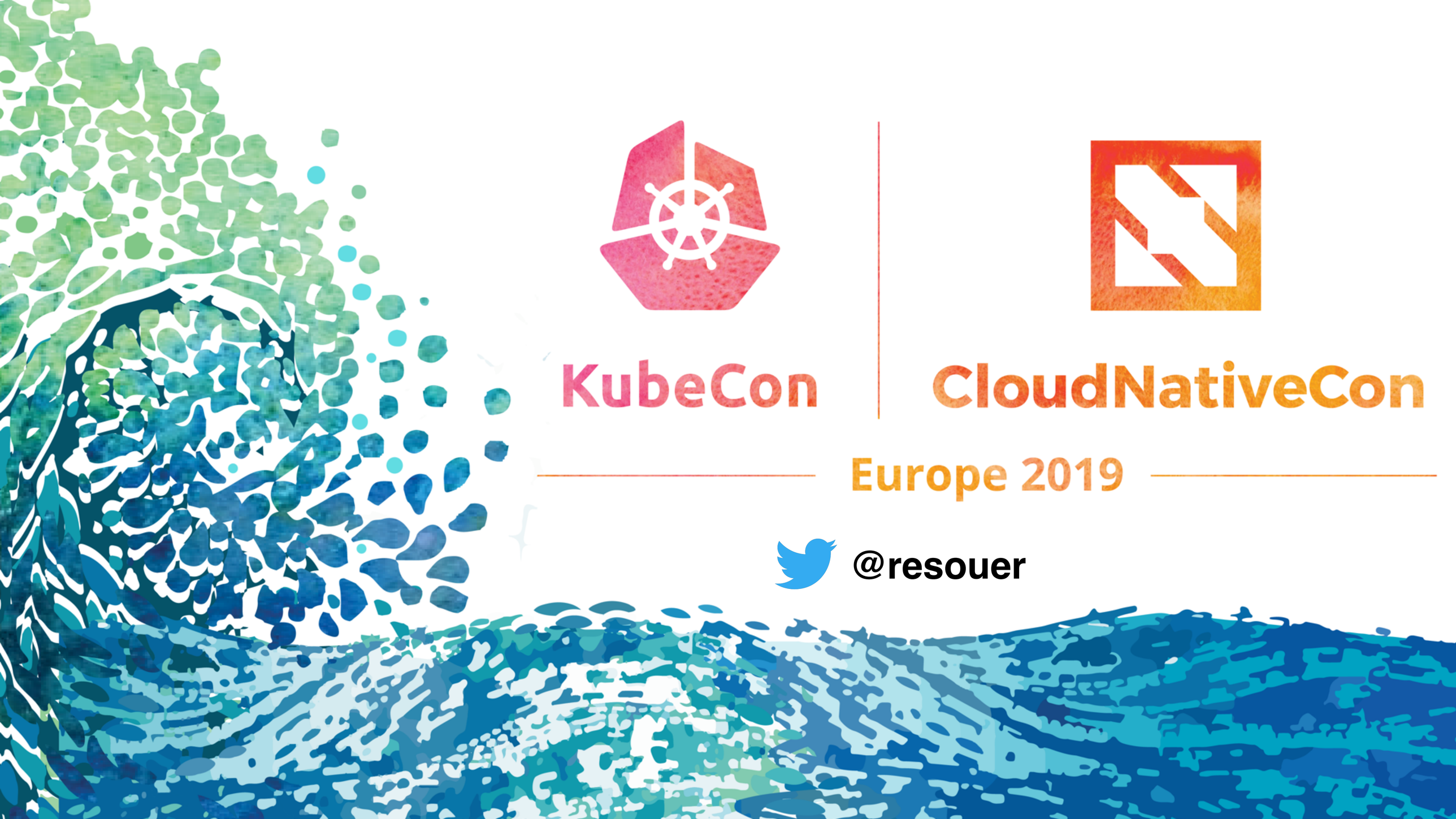
Alibaba Cloud | Worldwide Cloud Services Partner



Wrap Up

Alibaba Cloud | Worldwide Cloud Services Partner

- We use K8s **as both end user and public cloud vendor**
- **Container design pattern** is the key for web-scale users to migrate APPs to K8s
- **Customized workloads** is the key for web-scale users to run APPs
- Web-scale Kubernetes cluster is huge, **perf testing system is your #1 priority**
- **Virtual Cluster** based hard multi-tenancy model to serve customers
- **We fork upstream as “non fork”**, it’s also how we serve both internal & public cloud
- Build a small upstream team, it’s fun and rewarding!


KubeCon
CloudNativeCon
Europe 2019
@resouer