才云科技

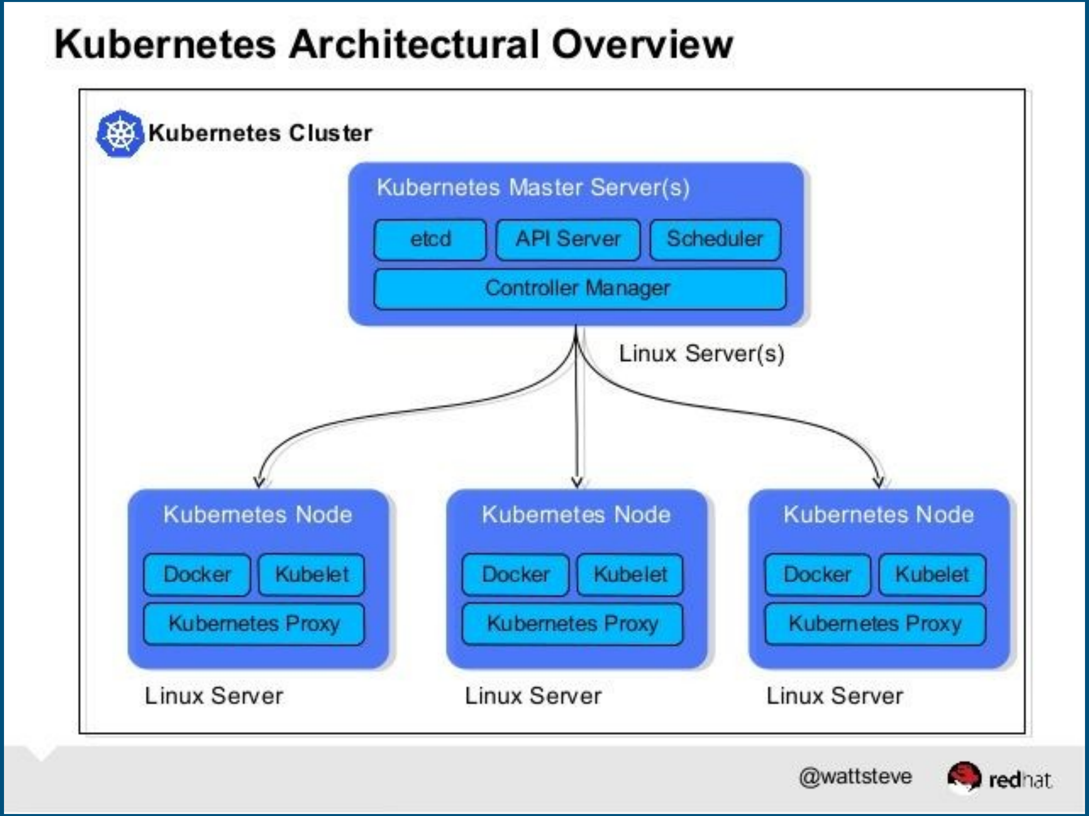
Kubernetes 1.3 +

Caicloud 邓德源

Kubernetes overview

- Basic Unit: Pod, Node, Volume, Labels, Endpoint, Binding, etc
- Aggregation: ReplicaSet, DaemonSet, PetSet, Deployment, etc
- Control loop: kube-proxy, scheduler, replica controller, etc

Kubernetes overview



Kubernetes 1.3+

- Infrastructure support for diverse application workloads
  - E.g. Legacy application, Stateful application, etc
- Enhanced cluster management policies and toolchains
  - E.g. Federation, Network policy, etc
- Performance and Performance
  - E.g. Protocol buffer serialization, etcd 3, etc

Diverse workloads

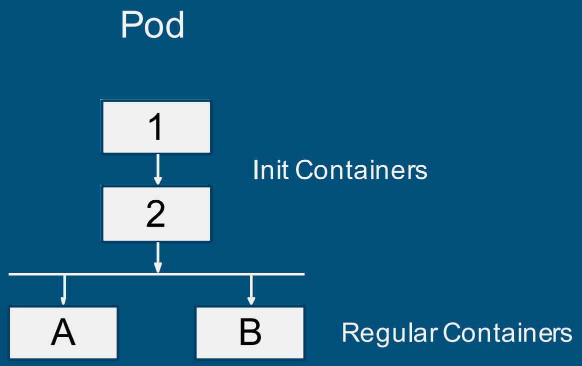
- Init Container
- PetSet
- Scheduled Job
- Disruption budget

Init container - alpha

Goal: Perform tasks before normal containers

Behavior:

- Init containers run in sequence
- One failed container fails entire pod
- Regular containers wait until all init containers complete



Use Cases:

- Dependency, Self-registration, Volume initialization, Decouple from application images, etc

Init container

stable

```
  Spec:
    initContainers:
      - name: install
        Image: busybox
        command: ["wget", "-O", "/work-dir/index.html", "http:
//kubernetes.io/index.html"]
        volumeMounts:
        - name: workdir
        - mountPath: "/work-dir"
    containers:
    - name:  nginx
      image: nginx
      ports:
      - containerPort: 80
```

alpha

```
apiVersion: v1
kind: Pod
metadata:
name: nginx
  annotations:
    pod.alpha.kubernetes.io/init-containers: '[
        {
            "name": "install",
            "image": "busybox",
            "command": ["wget", "-O", "/work-dir/index.html", "http:
//kubernetes.io/index.html"],
            "volumeMounts": [
              {
                "name": "workdir",
                "mountPath": "/work-dir"
              }
            ]
        }
    ]'
spec:
  containers:
  - name:  nginx
    image: nginx
    ports:
    - containerPort: 80
```

Init container

- Discussion Points
  - Pod status?
  - Health check?
  - Resources and QoS?
  - Update to init container?

PetSet - alpha

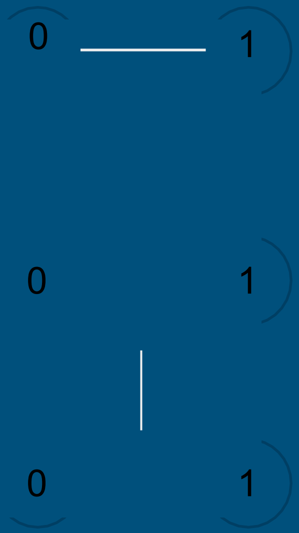
- Goal: Support stateful/clustered application which requires stronger identity
- Three identities:
  - Name (index)
  - Network
  - Storage
- Use cases:
  - Quorums with leader election: zookeeper, etcd
  - Decentralized Quorums: Cassandra
  - Databases like MySQL

PetSet

- Name (index)

```
$ kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
web-m63f0      1/1    Running      0       1d
Web-a29s4      1/1    Running      0       1d
```

```
$ kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
web-0          1/1    Running      0       1d
web-1          1/1    Running      0       1d
```

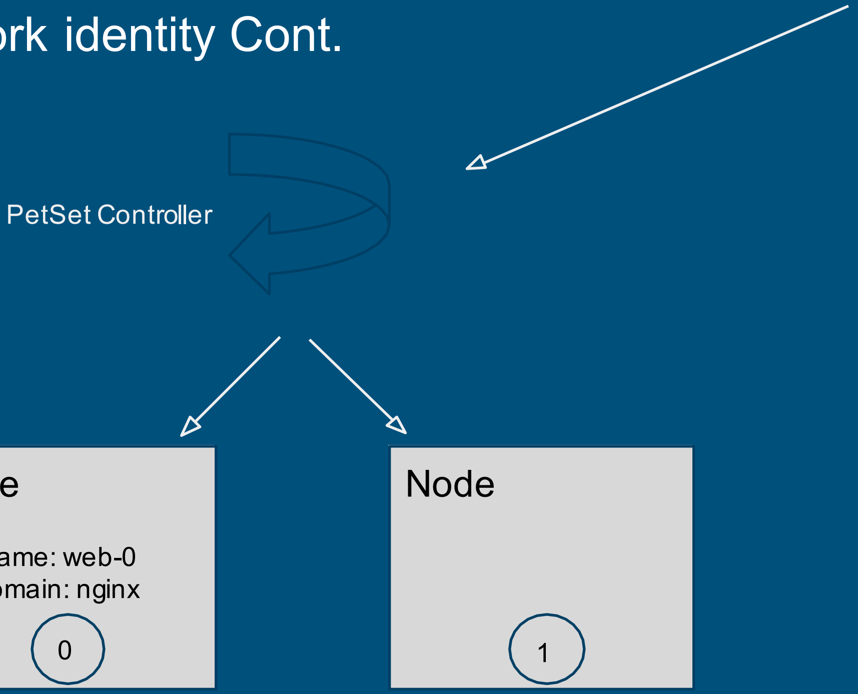


- Network identity
  - Stable hostname across cluster, across pod restart
  - Stable domain name using headless service

web-1.nginx.default.svc.cluster.local

PetSet

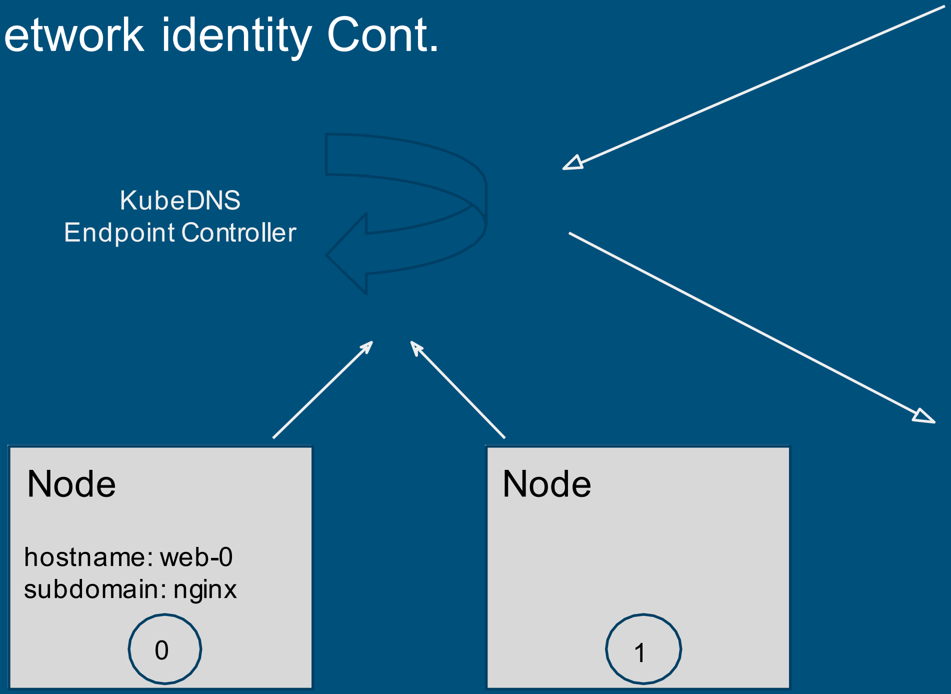
- Network identity Cont.



```
apiVersion: apps/v1alpha1
kind: PetSet
metadata:
name: web
spec:
 serviceName: "nginx"
 replicas: 2
 template:
  metadata:
  labels:
    app: nginx
   annotations:
    pod.alpha.kubernetes.io/initialized: "true"
  spec:
   containers:
   - name: nginx
    Image: nginx-slim:0.7
```

PetSet

- Network identity Cont.



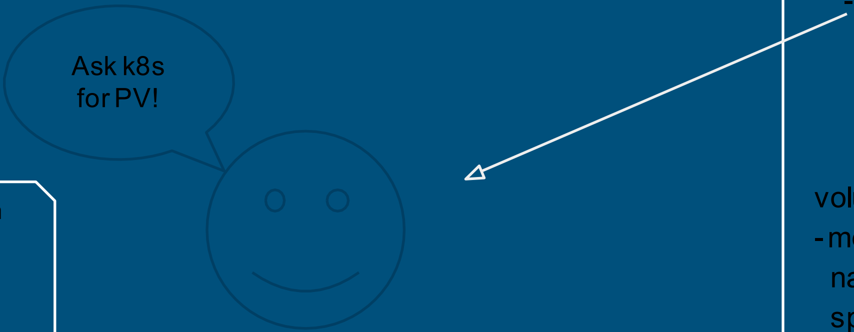
```
apiVersion: v1
kind: Service
metadata:
name: nginx
labels:
  app: nginx
spec:
 ports:
 - port: 80
   name: web
 # *.nginx.default.svc.cluster.local
 clusterIP: None
 selector:
   app: nginx
```

```
Name: web-0.nginx.default.svc.cluster.local
Address: 10.244.2.5

Name: web-1.nginx.default.svc.cluster.local
Address: 10.244.3.4

Name: nginx.default.svc.cluster.local
Address: 10.244.3.4
Name: nginx.default.svc.cluster.local
Address: 10.244.2.5
```

PetSet

- Storage identity
    - Each pet has its own persistent volumes

```
apiVersion: apps/v1alpha1
kind: PetSet
metadata:
name: web
spec:
template:
  spec:
   containers:
   - name: nginx
    image: nginx-slim:0.7
    volumeMounts:
    - name: www
     mountPath: /usr/share/nginx/html
 volumeClaimTemplates:
 - metadata:
  name: www
  spec:
   accessModes: [ "ReadWriteOnce" ]
   resources:
    requests:
     storage: 1Gi
```

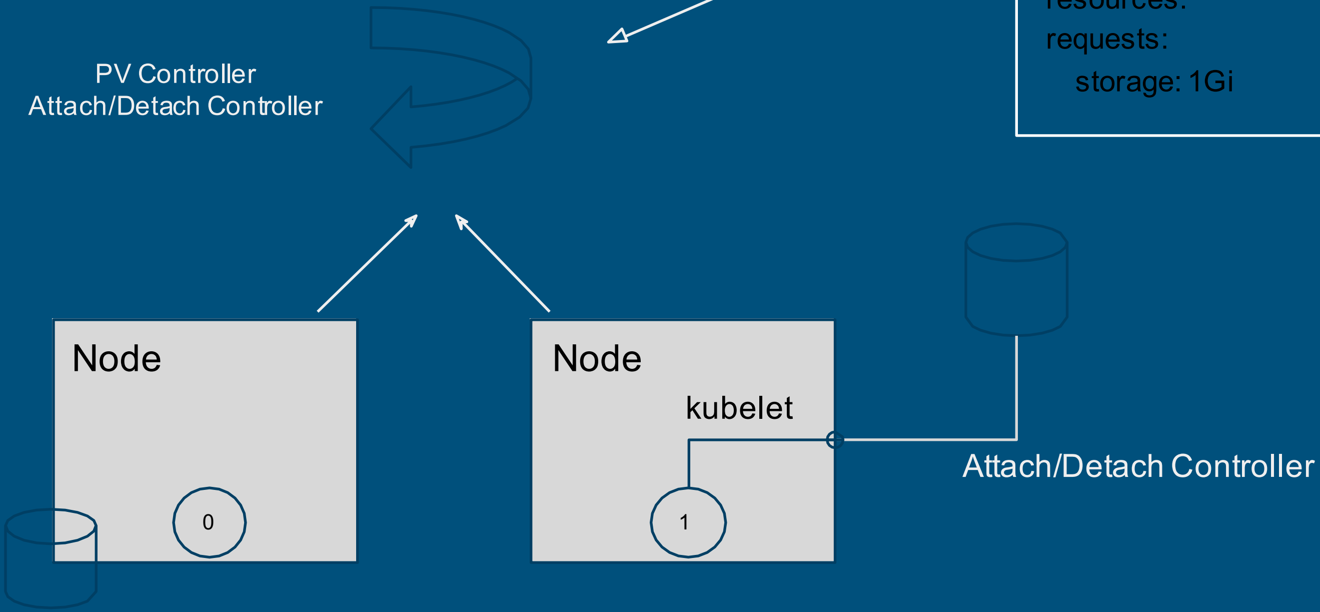


PetSet Controller

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
name: myclaim
spec:
accessModes:
   - ReadWriteOnce
 resources:
 requests:
   storage: 1Gi
```

PetSet

- Storage identity Cont.



```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
name: myclaim
spec:
accessModes:
  - ReadWriteOnce
 resources:
 requests:
   storage: 1Gi
```

PetSet

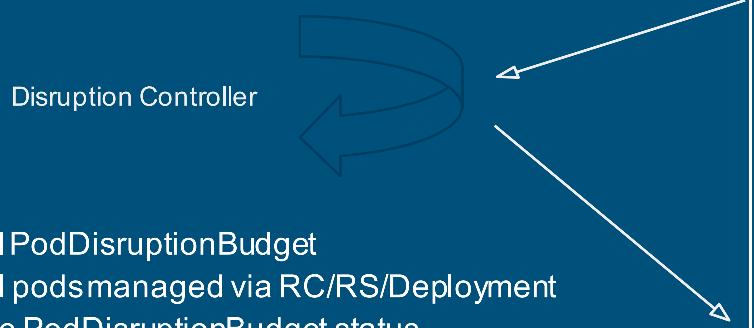
- Peer discovery
  - Query kubernetes api-server
  - Query DNS SRV records
- Important issues
  - Local storage
  - Public network identities
  - Pet upgrade
  - and more

Scheduled Job - 1.3+

- Goal:
  - Run Jobs at a given time
  - Run Jobs periodically at given time points
- Cron for the cluster
  - Use standard cron syntax
- Example:
  - kubectl run cleanup -image=cleanup --runAt="0 1 0 0 *" -- /scripts/cleanup.sh

Disruption Budget - 1.3+

- Guard against infrastructure initiated disruptions
  - Not unplanned, not self-inflicting problems
- Declarative policy around disruptions app will tolerate



- List all PodDisruptionBudget
- List all pods managed via RC/RS/Deployment
- Update PodDisruptionBudget.status

```
apiVersion: policy/v1alpha1
kind: PodDisruptionBudget
metadata:
 name: web
spec:
minAvailable:  3
selector:
   app: nginx
status:
 disruptionAllowed:  true
 currentHealthy: 4
 desiredHealthy: 3
 expectedPods: 5
```

Disruption Budget - 1.3+

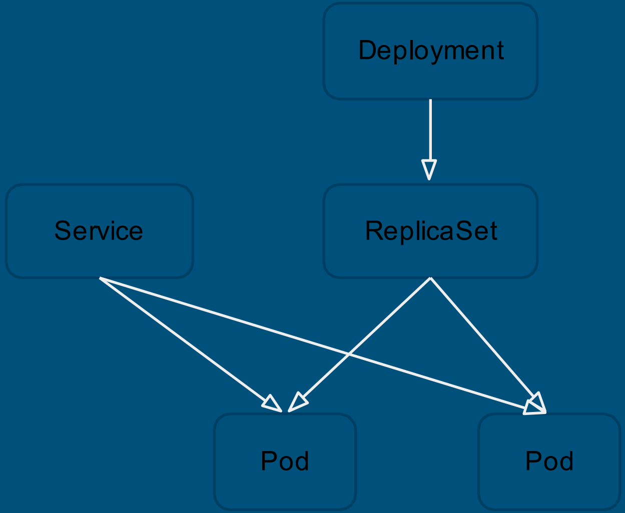
- Related topics:
  - Rescheduling: move pods around
    - How pod specify its tolerance for disruptions
    - Where and how is the decision made
  - Node eviction
    - Evict pods from overloaded nodes to preserve stability
    - More on later section
  - QoS and Priority
    - Low QoS app but strict tolerance? Quota and Admission Control !

Enhanced Cluster Management

- Cascading Deletion
- Kubelet/Node Eviction
- Network Policy

Cascading Deletion - alpha

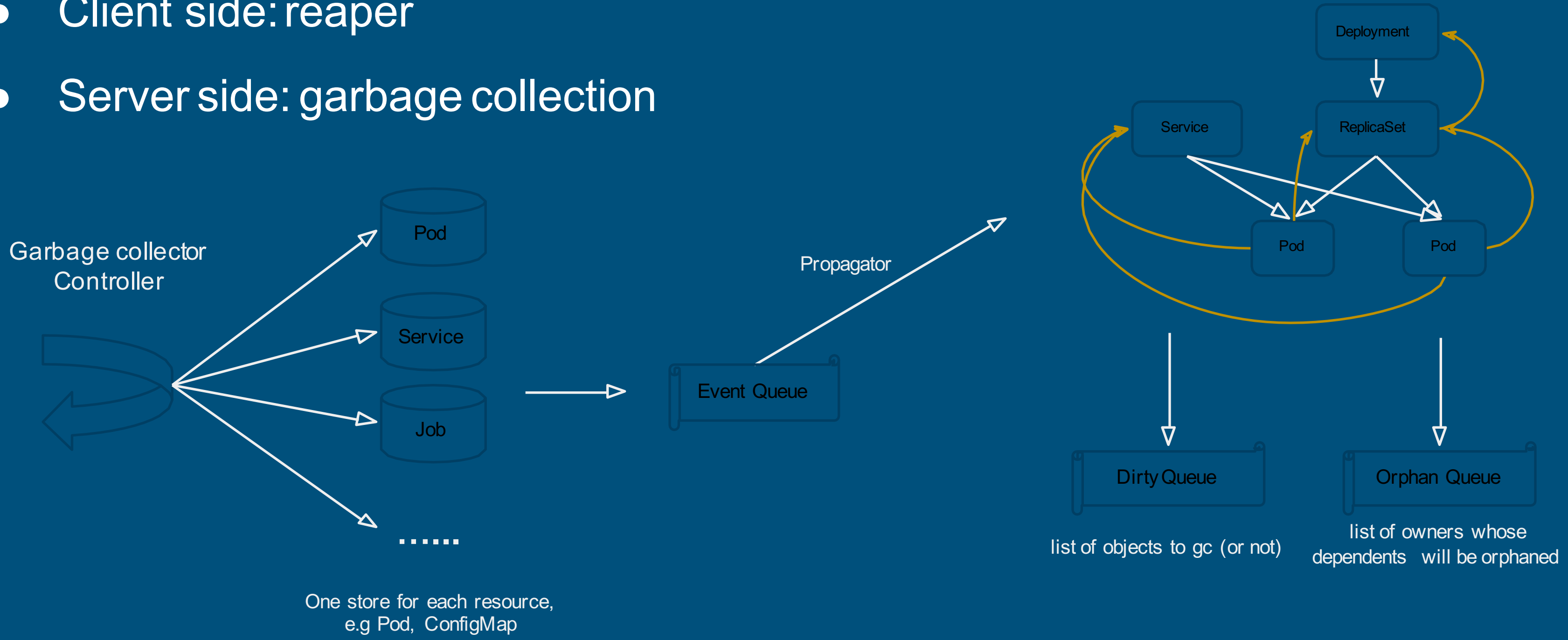
- Server side cleanup of all resources
- Example:



- Delete Deployment results in orphaned ReplicaSet
- Delete Deployment and ReplicaSet results in orphaned Pods
- Delete Service won’t touch Pods

Cascading Deletion

- Client side: reaper
- Server side: garbage collection



Garbage collector Controller
Pod
Service
Job
……
One store for each resource, e.g Pod, ConfigMap
Event Queue
Propagator
Deployment
Service
ReplicaSet
Pod
Pod
Dirty Queue
list of objects to gc (or not)
Orphan Queue
list of owners whose dependents will be orphaned

Kubelet/Node Eviction - opt-in

- **Proactively** evict pods from overloaded nodes to preserve stability
- Current
  - Memory: OOM killer
  - Disk: Image/container GC

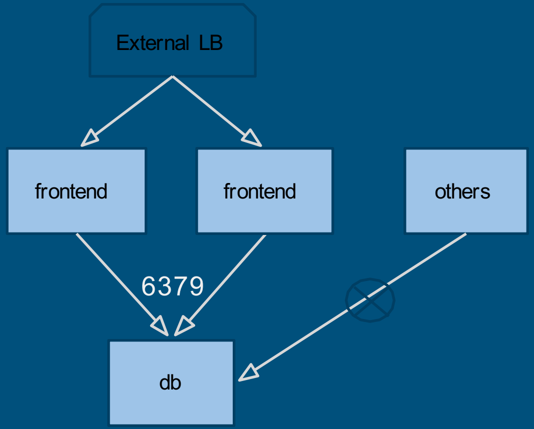
- Desired
  - Memory
    - memory.available
    - soft vs hard
  - Disk
    - nodefs.available, nodefs.inodesFree, imagefs.available, imagefs.inodesFree
    - soft vs hard

Kubelet/Node Eviction

- Policy
  - Low QoS pod first
  - Pod use most of its requested resources
- Problems?
  - Repeatedly schedule back
  - Oscillation
  - DaemonSet, Host Pin
  - Repeatedly reclaim for small resources

Network Policy - beta

- Define rules controlling pod traffic
- Expose only certain pods, and ports
- Implementation leaves to network vendors

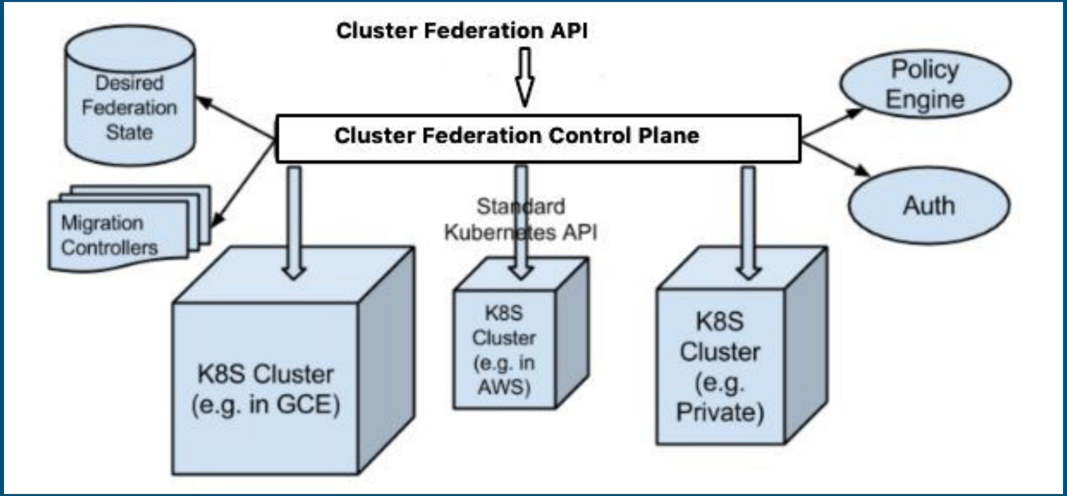


```
apiVersion: extensions/v1beta1
kind: NetworkPolicy
metadata:
 name: test-network-policy
spec:
 podSelector:
 matchLabels:
 role: db
 ingress:
  - from:
   podSelector:
   matchLabels:
   role: frontend
   ports:
    - protocol: tcp
     port: 6379
```

Caicloud 才云科技

Federation - beta

- Motivation
  - High-availability
  - CloudBursting
  - Avoid vendor lockin
  - Sensitive workflow
- Requirement
  - Location affinity
  - Cross-cluster scheduling
  - Cross-cluster service discovery
  - Cross-cluster monitoring and auditing
  - Cross-cluster load balancing
  - Application migration


Cluster Federation API
Desired Federation State
Cluster Federation Control Plane
Policy Engine
Auth
Migration Controllers
Standard Kubernetes API
K8S Cluster (e.g. in GCE)
K8S Cluster (e.g. in AWS)
K8S Cluster (e.g. Private)

Federation

- Federation-lite
  - kubernetes cluster nodes can span different zones
  - scheduler: take zone into consideration
  - kube-proxy: make sure packets do not bounce back and forth between different zone
  - volumes: add zone info label to volume
  - nodes: add zone info label to nodes
- Federation
  - A central control panel
    - scheduler, cluster controller, etc
  - Stock, dum kubernetes cluster

Performance

- etcd v3
- Protobuf serialization
- Controller shared caches
- Watch throughput optimization
- More

Performance

- Increased number of nodes
  100 nodes > 250 nodes > 1000 nodes -> 2000 nodes
- Increased number of pods
  30 pods per node > 40 pods per node > 100 pods per node -> 60000 pods

Performance

- etcd3 (sock testing)
  - https://coreos.com/blog/etcd3-a-new-etcd.html
- Protobuf serialization
  - The binary serialization for most API objects
  - For inter-component communication
  - 5 - 10x performance boost (compared to JSON)
- A lot others
  - Cluster shared cache
  - Watch throughput optimization
  - etc

Caicloud 才云科技

Thank you!