Kubernetes

标签（空格分隔）： Kubernetes k8s

- Kubernetes
  - 标签（空格分隔）： Kubernetes k8s
  - 环境
  - Topology
  - SYS Init
  - Hostname And Hosts
  - Master
    - Docker CE
    - kubelet
    - Flannel
  - Node
    - Node Join Clusters
  - Admin Guide
    - K8s 系统架构
    - node
    - API Server
    - Scheduler
    - pod
      - Pod 生命周期
      - liveness
      - readness
      - postStart preStop
    - control
      - Deployment
      - Daemonset
    - service
      - headless
      - sessionAffinity
    - Ingress Controller

        - 提供 SSL 服务
    - Traefik
    - secret configmap
        - emptyDir
        - gitRepo
        - HostPath
        - NFS
        - volume pv pvc StorageClass
    - StatefulSet
    - rbac
        - role
        - serviceaccount
    - ResourceQuota
    - kubeconfig
    - CNI
        -
            - Flannel
            - Flannel Directrouting
            - Flannel host-gw
        - Calico
    - Dashboard
    - Monitor
        - Metrics-Server
        - customresourcedefinitions
        - Prometheus
    - HPA
    - helm
        - tiller
        - Gitlab Install By Helm
- Creating Highly Available Clusters with kubeadm
    - Init
    - Create load balancer for kube-apiserver
    - Etcd cluster
        - 先在 master01 生成 etcd 证书
        - 在所有的 master 都执行
        - 检查集群状态
    - 创建 k8s 集群

        - Install CNI
            - Flannel And Enable Directrouting
            - Installing Calico for policy and flannel for networking
        - Add Node
        - Dashboard
            - 部署Web UI
            - 创建一个管理员用户
            - kubeconfig Login
        - Error
    - K8s With Paas
- Ingress
    - 启用 Ingress
    - gitlab 实例
        - 新加 node 并设置污点
        - gitlab安装
        - Traefik 设置
        - Ingress_nginx 设置
        - https

环境

```
CentOS Linux release 7.4.1708 (Core)
3.10.0-693.el7.x86_64
```

Topology

```
172.28.28.70 qa-k8s-master01.mljr.com
172.28.28.71 qa-k8s-master02.mljr.com
172.28.28.76 qa-k8s-master02.mljr.com
172.28.28.72 qa-k8s-node01.mljr.com
172.28.28.73 qa-k8s-node02.mljr.com
172.28.28.74 qa-k8s-node03.mljr.com
172.28.28.75 qa-k8s-node04.mljr.com
```

SYS Init

```
cat>> /etc/rc.loacl <<EOF
modprobe ip_vs
EOF

cat >> /etc/bashrc <<EOF
export HISTTIMEFORMAT="%Y-%m-%d-%H:%M:%S "
EOF

cat >> /etc/security/limits.conf <<EOF
* soft nofile 40960
* hard nofile 40960
EOF

cat > /usr/bin/safecron <<-'EOF'
#!/bin/bash
# remove -r function from crontab
# PATH

if [[ "$*" =~ "-r" ]] ; then
        echo "SB! Dangerous..."
        echo "Exit..."
        exit 2
else
        /usr/bin/crontab $*
        exit 0
fi
EOF

chmod +x /usr/bin/safecron
grep "crontab='/usr/bin/safecron'" /etc/bashrc || echo "alias crontab='/usr/bin
/safecron'" >> /etc/bashrc

setenforce 0
sed -i 's/SELINUX=enforcing/SELINUX=disabled/' /etc/selinux/config

systemctl stop firewalld.service
systemctl disable firewalld.service

systemctl stop  iptables.service
systemctl disable iptables.service
```

```
swapoff -a
sed -i 's/.*swap.*/#&/' /etc/fstab


cat <<EOF > /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
baseurl=http://mirrors.aliyun.com/kubernetes/yum/repos/kubernetes-el7-x86_64
enabled=1
gpgcheck=0
repo_gpgcheck=0
gpgkey=http://mirrors.aliyun.com/kubernetes/yum/doc/yum-key.gpg
        http://mirrors.aliyun.com/kubernetes/yum/doc/rpm-package-key.gpg
EOF

cd /usr/local/src
wget http://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm
rpm -ivh epel-release-latest-7.noarch.rpm

crontab  -e <<EOF
:1,1s#^#0 */2 * * * /usr/sbin/ntpdate 1.cn.pool.ntp.org \&#
:wq
EOF
```

Hostname And Hosts

```
# 每台服务服务器 各自执行

hostnamectl --pretty set-hostname qa-k8s-master03.mljr.com
hostnamectl --transient set-hostname qa-k8s-master03.mljr.com
hostnamectl --static set-hostname qa-k8s-master03.mljr.com
```

```
# hosts 文件添加
172.28.28.70 qa-k8s-master01.mljr.com master01
172.28.28.71 qa-k8s-master02.mljr.com master02
172.28.28.76 qa-k8s-master03.mljr.com master03
172.28.28.72 qa-k8s-node01.mljr.com node01
172.28.28.73 qa-k8s-node02.mljr.com node02
172.28.28.74 qa-k8s-node03.mljr.com node03
```

```
172.28.28.75 qa-k8s-node04.mljr.com node04
```

Master

Docker CE

```
yum remove docker docker-common docker-selinux docker-engine -y

yum install -y yum-utils device-mapper-persistent-data lvm2

yum-config-manager --add-repo  https://download.docker.com/linux/centos/docker-ce.repo

yum -y install docker-ce

systemctl enable docker && systemctl start docker
```

kubelet

```
yum install -y kubelet kubeadm kubectl
systemctl enable kubelet && systemctl start kubelet
```

> 禁止iptables对bridge数据进行处理

```
# 所有节点都要执行

cat <<EOF >  /etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
EOF
sysctl --system
```

> master init

```
kubeadm init --kubernetes-version=v1.11.0 --pod-network-cidr=10.244.0.0/16 --apiserver-advertise-address=172.28.28.70 --ignore-preflight-errors=all
```

```
//正常执行结果如下
Your Kubernetes master has initialized successfully!

To start using your cluster, you need to run the following as a regular user:

  mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config

You should now deploy a pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
  https://kubernetes.io/docs/concepts/cluster-administration/addons/

You can now join any number of machines by running the following on each node
as root:

  kubeadm join 172.28.28.70:6443 --token en8yv2.njpxt99hcv6hctcf --discovery-to
ken-ca-cert-hash sha256:3b800b8dc8c6bc23565fd80ffafe143ef7e4659324f4c10706fad83
d33b7bc0a
```

```
useradd k8s
mkdir -p /home/k8s/.kube
cp -i /etc/kubernetes/admin.conf /home/k8s/.kube/config
chown  k8s. /home/k8s/.kube/config
su - k8s
```

```
在 k8s 用户下执行 kubectl 命令 如:
$ kubectl get nodes
```

Flannel

```
// 也可以用 Calico, 基于 BGP 协议, 比 Flannel更 强大
```

```
$ kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Docu
mentation/kube-flannel.yml
```

Node

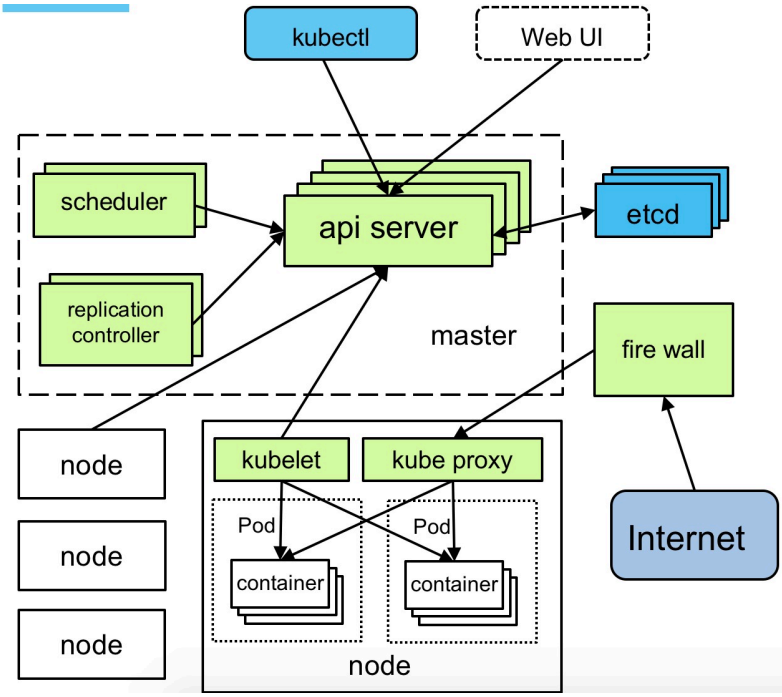
```
yum install docker-ce kubelet kubeadm kubectl -y

systemctl enable docker && systemctl start docker
systemctl enable kubelet.service
```

Node Join Clusters

```
kubeadm join 172.28.28.70:6443 --token en8yv2.njpxt99hcv6hctcf --discovery-token-ca-cert-hash sha256:3b800b8dc8c6bc23565fd80ffafe143ef7e4659324f4c10706fad83d33b7bc0a
```

Admin Guide

K8s 系统架构



- 用户通过kubectl提交需要运行的docker container(pod)
- api server把请求存储在etcd里面
- scheduler扫描，分配机器
- kubelet找到自己需要跑的container，在本机上运行

- 用户提交RC描述，replication controller监视集群中的容器并保持数量
- 用户提交service描述文件，由kube proxy负责具体的工作流量转发

```
1) etcd
Etcd是一个分布式的高性能key-value存储系统, 负责Kubernetes 中所有 REST API 对象的持久化保存.
```

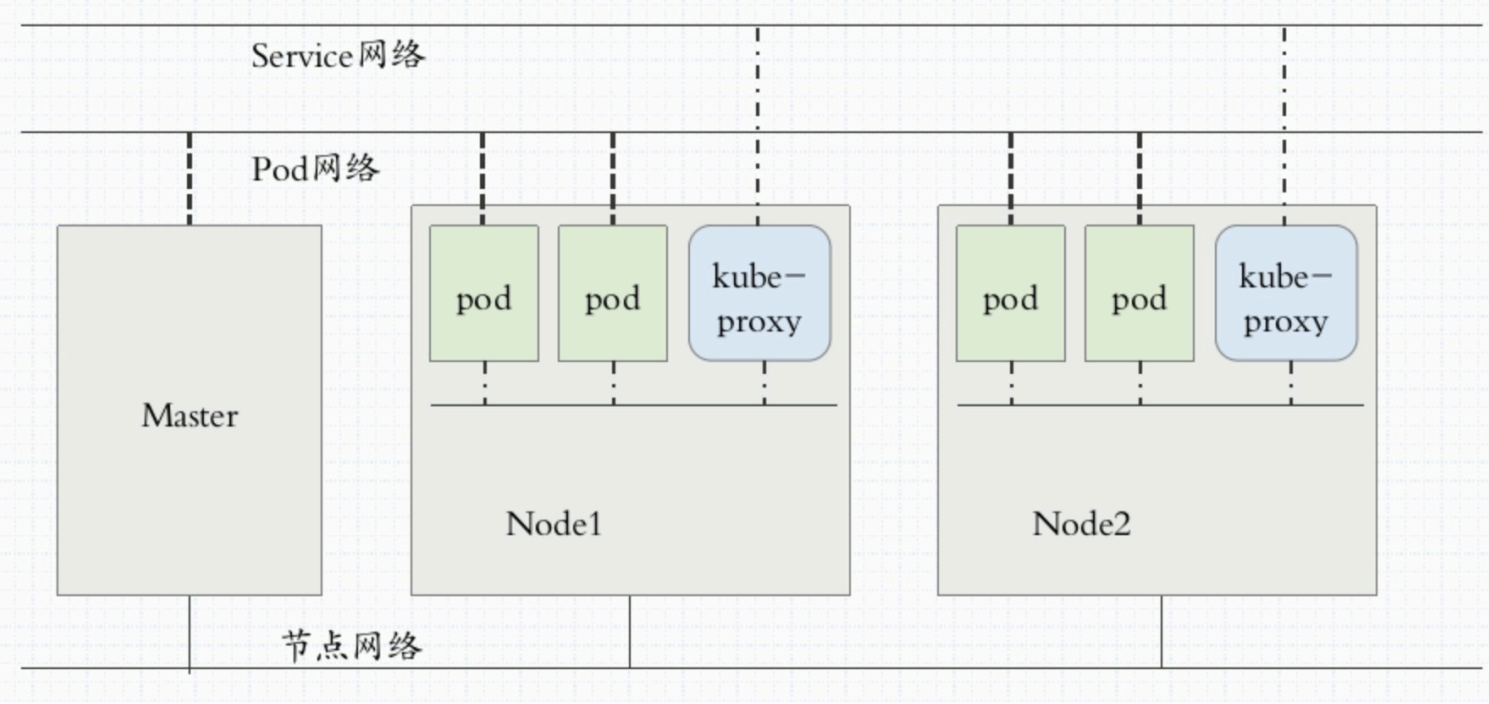
2) kube-apiserver
Kubernetes 对外提供统一的入口，通过 REST API 的方式供客户端调用，例如 kubectl.

3) kube-controller-manager
控制集群中程序任务的后台线程. 逻辑上，每个控制任务对应一个线程，包括**Node Controller**，Replication Controller, Service Controller, Endpoints Controller等.

4) kube-scheduler
维护可用的 **node 节点及资源列表，根据输入的待调度pod**请求，通过调度算法选择最优**node节点，输出绑定了pod**的**node节点.**

5) kubelet
Kubelet 是 Kubernetes 集群中每个**node和Api-server**的连接点，负责容器和pod的实际管理工作.

6) kube-proxy
为pod提供代理服务，用来实现 kubernetes 的 service 机制.

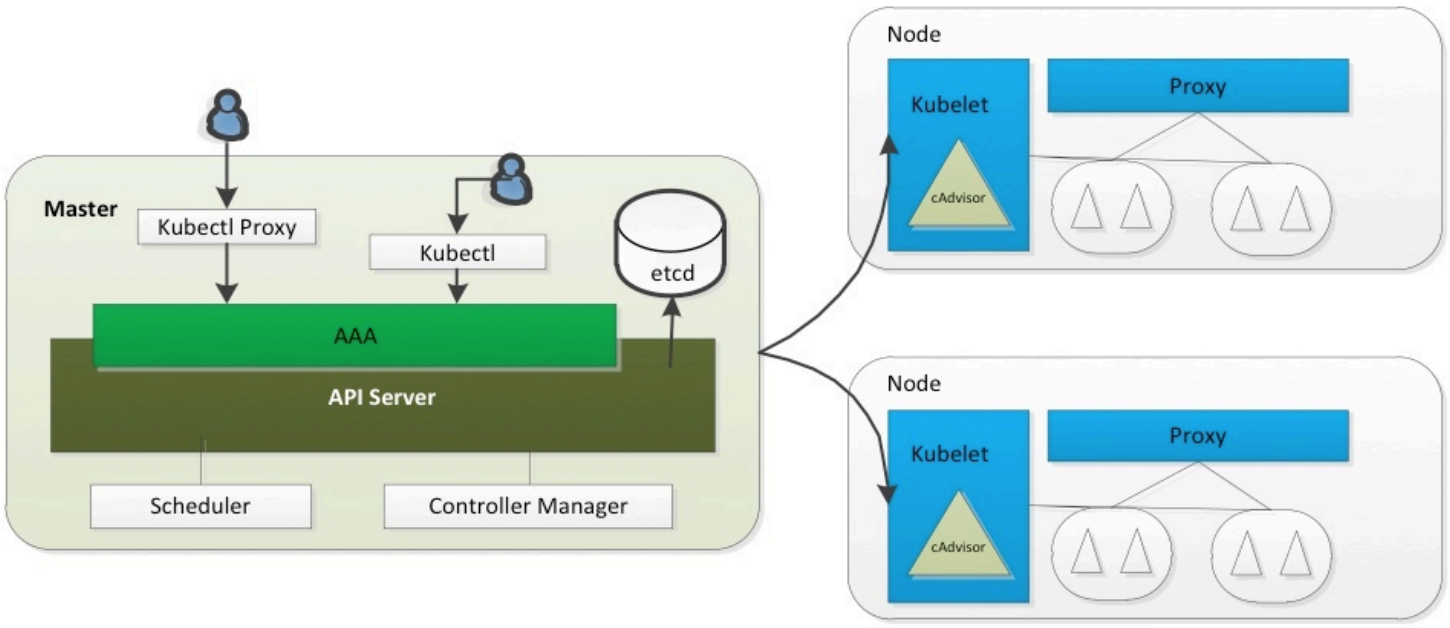


node

给 node 打标签，用于服务指定的服务.
kubectl label nodes qa-k8s-node01.mljr.com nodetype=vmware

```
[root@qa-k8s-master01 configmap]# cat pod-demo.yaml
apiVersion: v1
kind: Pod
metadata:
  name: pod-nginx
  namespace: default
  labels:
    app: yyfq
    type: nginx
spec:
  containers:
  - name: nginx
    image: nginx:1.15-alpine
    imagePullPolicy: IfNotPresent
    ports:
    - name: http
      containerPort: 80
  nodeSelector:
    nodetype: vmware
```

API Server

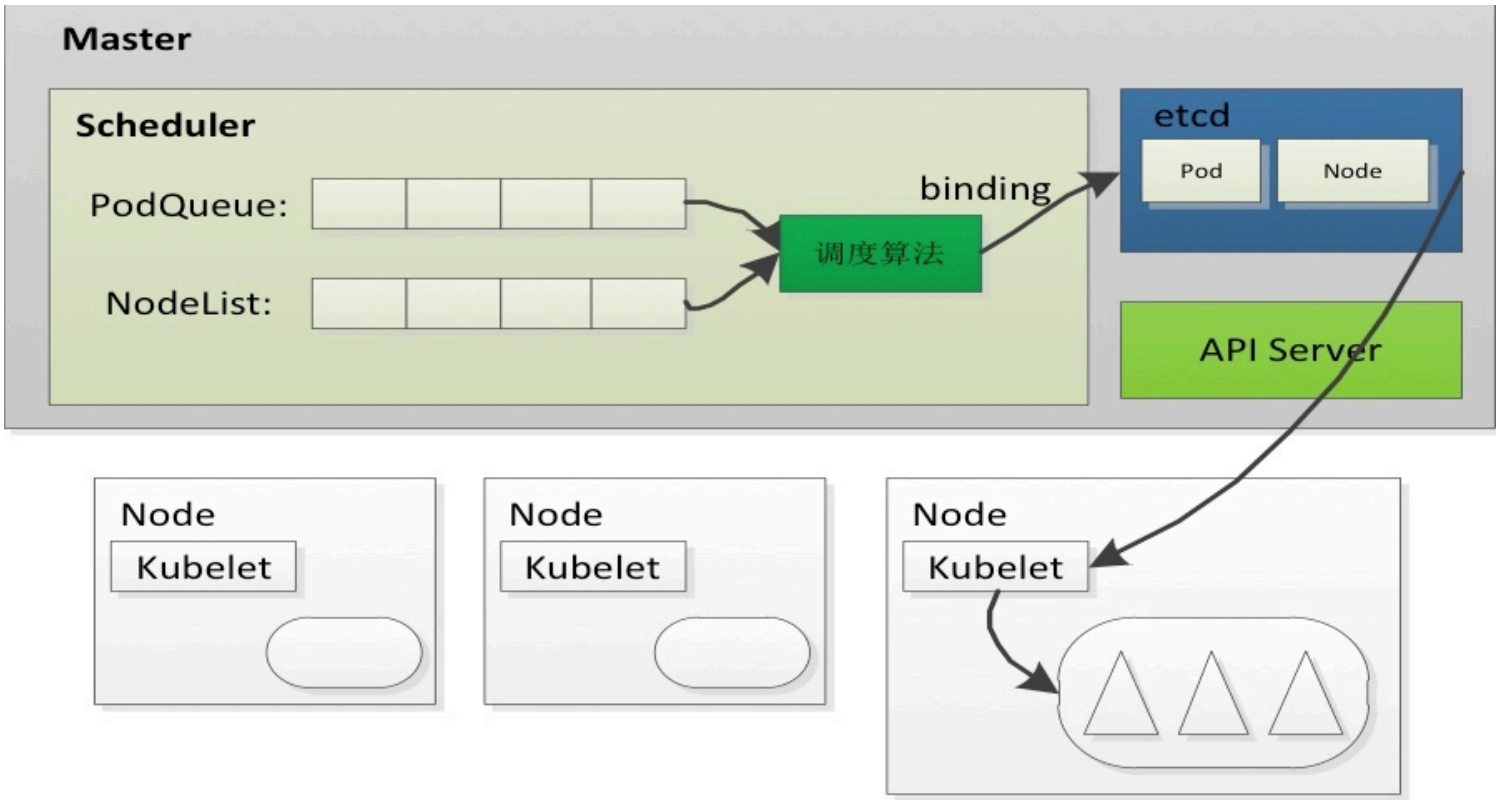


1）提供集群管理的API接口；

- api-server进程提供**http**(8080)/**https**(6443)两个端口供访问;

2) 成为集群内各个功能模块之间数据交互和通信的中心枢纽

Scheduler



- 负责Pod调度
- 通过调度算法为待调度Pod列表的每个Pod从Node列表中选择一个最合适的Node

pod

Pod 是一组紧密关联的容器集合, 它们共享 IPC, Network 和 UTS namespace, 是 Kubernetes 调度的基本单位.
Pod 的设计理念是支持多个容器在一个 Pod 中共享网络和文件系统, 可以通过进程间通信和文件共享这种简单高效的方式组合完成服务.

Pod 的特征:
    包含多个共享 IPC, Network 和 UTC namespace 的容器, 可直接通过 localhost 通信

    所有 Pod 内容器都可以访问共享的 Volume, 可以访问共享数据.

    无容错性, 直接创建的 Pod 一旦被调度后就跟 **Node 绑定, 即使 Node** 挂掉也不会被重新调度 (

而是被自动删除），因此推荐使用 Deployment, Daemonset 等控制器来容错

    优雅终止，Pod 删除的时候先给其内的进程发送 SIGTERM，等待一段时间（grace period）后才强制停止依然还在运行的进程

    特权容器（通过 SecurityContext 配置）具有改变系统配置的权限（在网络插件中大量应用）

```
// Kubernetes v1.8+ 还支持容器间共享 PID namespace，需要 docker >= 1.13.1，并配置 kubelet --docker-disable-shared-pid=false.
```

```
[root@qa-k8s-master01 configmap]# cat pod-demo.yaml
apiVersion: v1
kind: Pod
metadata:
  name: pod-nginx
  namespace: default
  labels:
    app: yyfq
    type: nginx
spec:
  containers:
  - name: nginx
    image: nginx:1.15-alpine
    imagePullPolicy: IfNotPresent
    ports:
    - name: http
      containerPort: 80

[root@qa-k8s-master01 configmap]# kubectl apply  -f pod-demo.yaml
pod/pod-nginx created
[root@qa-k8s-master01 configmap]# kubectl get pods -o wide --show-labels
NAME                     READY     STATUS    RESTARTS   AGE       IP
  NODE                     LABELS
pod-nginx                1/1       Running   0          23s       10.244.2.9
  qa-k8s-node02.mljr.com   app=yyfq,type=nginx
```

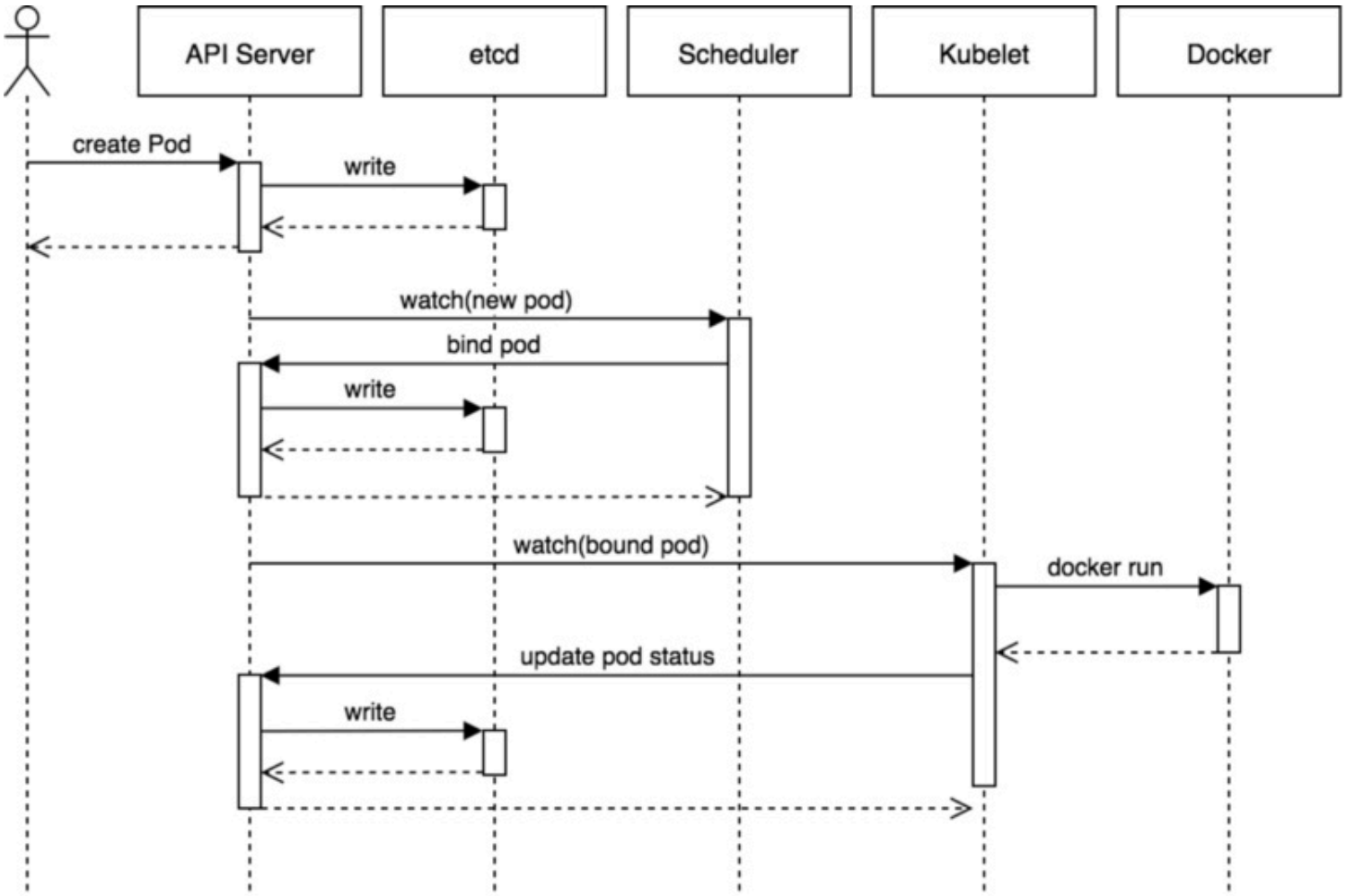
```
// -l 做标签过滤，显示有 app 标签(可以指定多个逗号分开)的 pod
[root@qa-k8s-master01 configmap]# kubectl get pods -o wide -l app
NAME                     READY     STATUS    RESTARTS   AGE       IP
  NODE
pod-nginx                1/1       Running   0          5m        10.244.2.9
  qa-k8s-node02.mljr.com
```

```
// -l 也可以做 等值/不等值 的标签选择器, 多个标签为与关系
[root@qa-k8s-master01 configmap]# kubectl get pods -l type=nginx,release=stable
NAME        READY     STATUS    RESTARTS   AGE
pod-nginx   1/1       Running   0          16m
[root@qa-k8s-master01 configmap]# kubectl get pods -l type=nginx,release!=stabl
e
No resources found.
[root@qa-k8s-master01 configmap]# kubectl get pods -l type=nginx,release
NAME        READY     STATUS    RESTARTS   AGE
pod-nginx   1/1       Running   0          16m

// -L 显示知道标签的值
[root@qa-k8s-master01 configmap]# kubectl get pods -L app,type
NAME                    READY     STATUS    RESTARTS   AGE       APP
TYPE
pod-nginx               1/1       Running   0          7m        yyfq
nginx
[root@qa-k8s-master01 configmap]# kubectl get pods -l "release in (canary,beta,
stable)"
NAME        READY     STATUS    RESTARTS   AGE
pod-nginx   1/1       Running   0          19m
[root@qa-k8s-master01 configmap]# kubectl get pods -l "release notin (canary,be
ta,stable)"
No resources found.
[root@qa-k8s-master01 configmap]# kubectl get pods -l "release in (canary,beta,
alpha)"
No resources found.

// 给 pod 打标签
[root@qa-k8s-master01 configmap]# kubectl  label  --overwrite pods pod-nginx re
lease=stable
pod/pod-nginx labeled
[root@qa-k8s-master01 configmap]# kubectl get pods -o wide --show-labels
NAME                    READY     STATUS    RESTARTS   AGE       IP
  NODE                   LABELS
pod-nginx               1/1       Running   0          11m       10.244.2.9
  qa-k8s-node02.mljr.com   app=yyfq,release=stable,type=nginx
```

```
// 给 node 打标签
[root@qa-k8s-master01 configmap]# kubectl get node
NAME                     STATUS    ROLES     AGE       VERSION
qa-k8s-master01.mljr.com   Ready     master    8d        v1.11.2
qa-k8s-node01.mljr.com     Ready     <none>    7d        v1.11.2
```

```
qa-k8s-node02.mljr.com     Ready     <none>    7d        v1.11.2
qa-k8s-node03.mljr.com     Ready     <none>    7d        v1.11.2
qa-k8s-node04.mljr.com     Ready     <none>    7d        v1.11.2
[root@qa-k8s-master01 configmap]# kubectl label nodes qa-k8s-node01.mljr.com no
detype=vmware
node/qa-k8s-node01.mljr.com labeled
[root@qa-k8s-master01 configmap]# kubectl get node --show-labels -l nodetype
NAME                     STATUS    ROLES     AGE       VERSION   LABELS
qa-k8s-node01.mljr.com   Ready     <none>    7d        v1.11.2   beta.kubernete
s.io/arch=amd64,beta.kubernetes.io/fluentd-ds-ready=true,beta.kubernetes.io/os=
linux,kubernetes.io/hostname=qa-k8s-node01.mljr.com,nodetype=vmware
[root@qa-k8s-master01 configmap]# cat pod-demo.yaml
apiVersion: v1
kind: Pod
metadata:
  name: pod-nginx
  namespace: default
  labels:
    app: yyfq
    type: nginx
spec:
  containers:
  - name: nginx
    image: nginx:1.15-alpine
    imagePullPolicy: IfNotPresent
    ports:
    - name: http
      containerPort: 80
  nodeSelector:
    nodetype: vmware
[root@qa-k8s-master01 configmap]# kubectl get pods -o wide
NAME                     READY     STATUS    RESTARTS   AGE       IP
  NODE
pod-nginx                1/1       Running   0          20s       10.244.1.13
  qa-k8s-node01.mljr.com
```

Pod 生命周期




```
状态: Pending, Running, Failed, Succeeded, Unknown
Pod生命周期中的重要行为:
    初始化容器
    容器探测:
        liveness: 存活行探测
        readiness: 就绪性探测

pod 容器策略:
restartPolicy:
        Always, OnFailure, Never. Default to Always.
```

liveness

```
[root@qa-k8s-master01 configmap]# cat liveness-exec-demo.yaml
apiVersion: v1
kind: Pod
metadata:
  name: pod-liveness-exec
```

```
  namespace: dev
  labels:
    app: yyfq
    type: nginx
spec:
  containers:
  - name: nginx
    image: nginx:1.15-alpine
    imagePullPolicy: IfNotPresent
    ports:
    - name: http
      containerPort: 80
    command: ["/bin/sh","-c","touch /tmp/healthy;sleep 30;rm -f /tmp/healthy;sl
eep 1200"]
    livenessProbe:
      exec:
        command: ["test","-c","/tmp/healthy"]
      initialDelaySeconds: 1
      periodSeconds: 3
      failureThreshold: 2
      timeoutSeconds: 1
  restartPolicy: Always
[root@qa-k8s-master01 configmap]# kubectl apply -f liveness-exec-demo.yaml
pod/pod-liveness-exec created
[root@qa-k8s-master01 configmap]# kubectl get pods -n dev
NAME                READY     STATUS    RESTARTS   AGE
pod-liveness-exec   1/1       Running   0          17s
[root@qa-k8s-master01 configmap]# kubectl get pods -n dev -w
NAME                READY     STATUS             RESTARTS   AGE
pod-liveness-exec   0/1       CrashLoopBackOff   5          5m
pod-liveness-exec   1/1       Running   6         5m


// 可以观察得到，一直检查容器，是否正常，发现失败，一直重启.
```

```
// httpGet 探针
[root@qa-k8s-master01 configmap]# cat liveness-httpget-demo.yaml
apiVersion: v1
kind: Pod
metadata:
  name: pod-liveness-httpget
  namespace: dev
  labels:
    app: yyfq
```

```
    type: nginx
spec:
  containers:
  - name: nginx
    image: nginx:1.15-alpine
    imagePullPolicy: IfNotPresent
    ports:
    - name: http
      containerPort: 80
    livenessProbe:
      httpGet:
        path: /index.html
        port: http
      initialDelaySeconds: 1
      periodSeconds: 3
      failureThreshold: 2
      timeoutSeconds: 1
  restartPolicy: Always

// 注意此刻用的  dev 命名空间
// 可以 连到 容器 , 删除 index.html  进行测试
```

readness

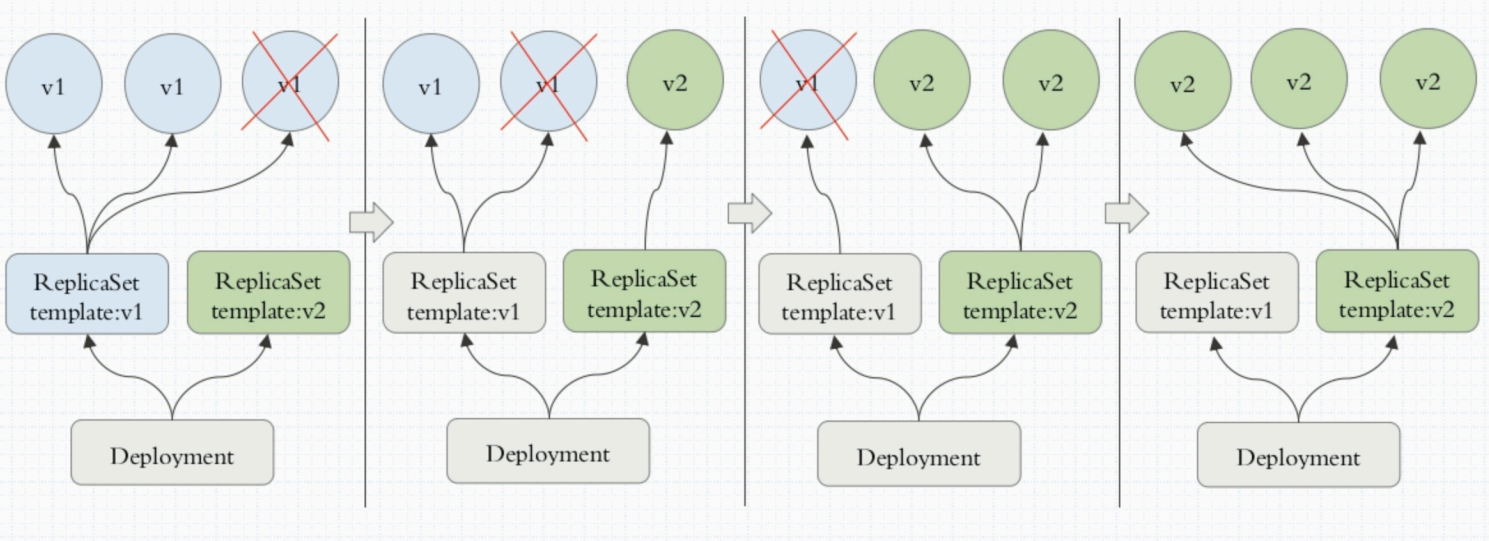
```
[root@qa-k8s-master01 configmap]# cat  readness-httpget-demo.yaml
apiVersion: v1
kind: Pod
metadata:
  name: pod-readness-httpget
  namespace: dev
  labels:
    app: yyfq
    type: nginx
spec:
  containers:
  - name: nginx
    image: nginx:1.15-alpine
    imagePullPolicy: IfNotPresent
    ports:
    - name: http
      containerPort: 80
    readinessProbe:
      httpGet:
```

```
        path: /index.html
        port: http
      initialDelaySeconds: 1
      periodSeconds: 3
  restartPolicy: Always
[root@qa-k8s-master01 configmap]# kubectl apply -f readness-httpget-demo.yaml
pod/pod-readness-httpget created
[root@qa-k8s-master01 configmap]# kubectl get pods -n dev -o wide
NAME                   READY     STATUS    RESTARTS   AGE       IP            N
ODE
pod-liveness-httpget   1/1       Running   7          27m       10.244.2.10   q
a-k8s-node02.mljr.com
pod-readness-httpget   0/1       Running   0          3s        10.244.2.11   q
a-k8s-node02.mljr.com
[root@qa-k8s-master01 configmap]# kubectl get pods -n dev -o wide
NAME                   READY     STATUS    RESTARTS   AGE       IP            N
ODE
pod-liveness-httpget   1/1       Running   7          27m       10.244.2.10   q
a-k8s-node02.mljr.com
pod-readness-httpget   1/1       Running   0          7s        10.244.2.11   q
a-k8s-node02.mljr.com

// 此时可以连上 pod-readness-httpget 容器, 删除 index.html 文件  , 可以看出 READY 变
成了 0/1.
```

postStart preStop

control

Deployment




```
[root@qa-k8s-master01 configmap]# cat deployment-demo.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: myapp-delpoy
  namespace: dev
spec:
  replicas: 2
  selector:
    matchLabels:
      app: myapp
      release: canary
  template:
    metadata:
      labels:
        app: myapp
        release: canary
    spec:
      containers:
      - name: myapp
        image: nginx:1.15-alpine
        imagePullPolicy: IfNotPresent
        ports:
        - name: nginx
          containerPort: 80
```

```
[root@qa-k8s-master01 configmap]# kubectl apply -f deployment-demo.yaml
deployment.apps/myapp-delpoy created
[root@qa-k8s-master01 configmap]# kubectl get deploy -n dev
```

```
NAME           DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
myapp-delpoy   2         2         2            2           5m
[root@qa-k8s-master01 configmap]# kubectl get rs -n dev -o wide
NAME                     DESIRED   CURRENT   READY     AGE       CONTAINERS
IMAGES              SELECTOR
myapp-delpoy-6b786474dc   2         2         2         2m        myapp
nginx:1.15-alpine   app=myapp,pod-template-hash=2634203087,release=canary
[root@qa-k8s-master01 configmap]# kubectl get pod -n dev
NAME                           READY     STATUS    RESTARTS   AGE
myapp-delpoy-6b786474dc-jkj2w   1/1       Running   0          3m
myapp-delpoy-6b786474dc-npnfr   1/1       Running   0          3m
```

Daemonset

```
[root@qa-k8s-master01 configmap]# cat daemonset-demo.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: redis
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: redis
      role: logstor
  template:
    metadata:
      labels:
        app: redis
        role: logstor
    spec:
      containers:
      - name: redis
        image: redis:4.0-alpine
        imagePullPolicy: IfNotPresent
        ports:
        - name: redis
          containerPort: 6379
---
apiVersion: apps/v1
kind: DaemonSet
metadata:
```

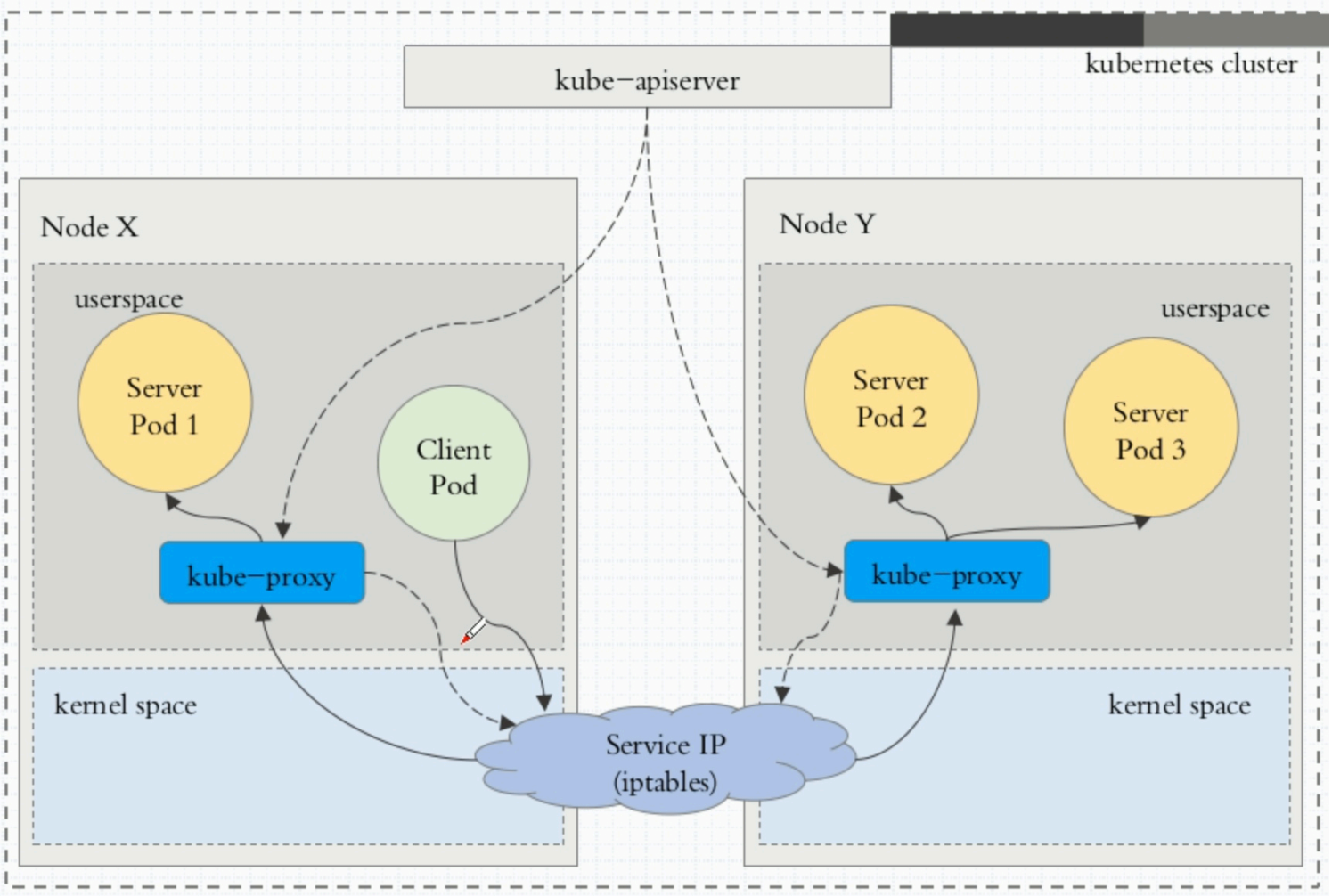
```
  name: filebeat-ds
  namespace: default
spec:
  selector:
    matchLabels:
      app: filebeat
      release: stable
  template:
    metadata:
      labels:
        app: filebeat
        release: stable
    spec:
      containers:
      - name: filebeat
        image: ikubernetes/filebeat:5.6.5-alpine
        imagePullPolicy: IfNotPresent
        env:
        - name: REDIS_HOST
          value: redis.default.svc.cluster.local
        - name: REDIS_LOG_LEVEL
          value: info
[root@qa-k8s-master01 configmap]# kubectl apply -f daemonset-demo.yaml
deployment.apps/redis created
daemonset.apps/filebeat-ds created
// 暴露 redis 端口
// kubectl expose deployment redis --port=6379
[root@qa-k8s-master01 configmap]# kubectl expose deployment redis --port=6379
service/redis exposed

// 4个 node 节点
[root@qa-k8s-master01 configmap]# kubectl get pods
NAME                        READY     STATUS    RESTARTS   AGE
filebeat-ds-92wcq           1/1       Running   1          15m
filebeat-ds-nzlg9           1/1       Running   0          15m
filebeat-ds-qrn5x           1/1       Running   0          15m
filebeat-ds-xbtnp           1/1       Running   0          15m
redis-5b5d6fbbbd-4ljfq      1/1       Running   0          16m
```

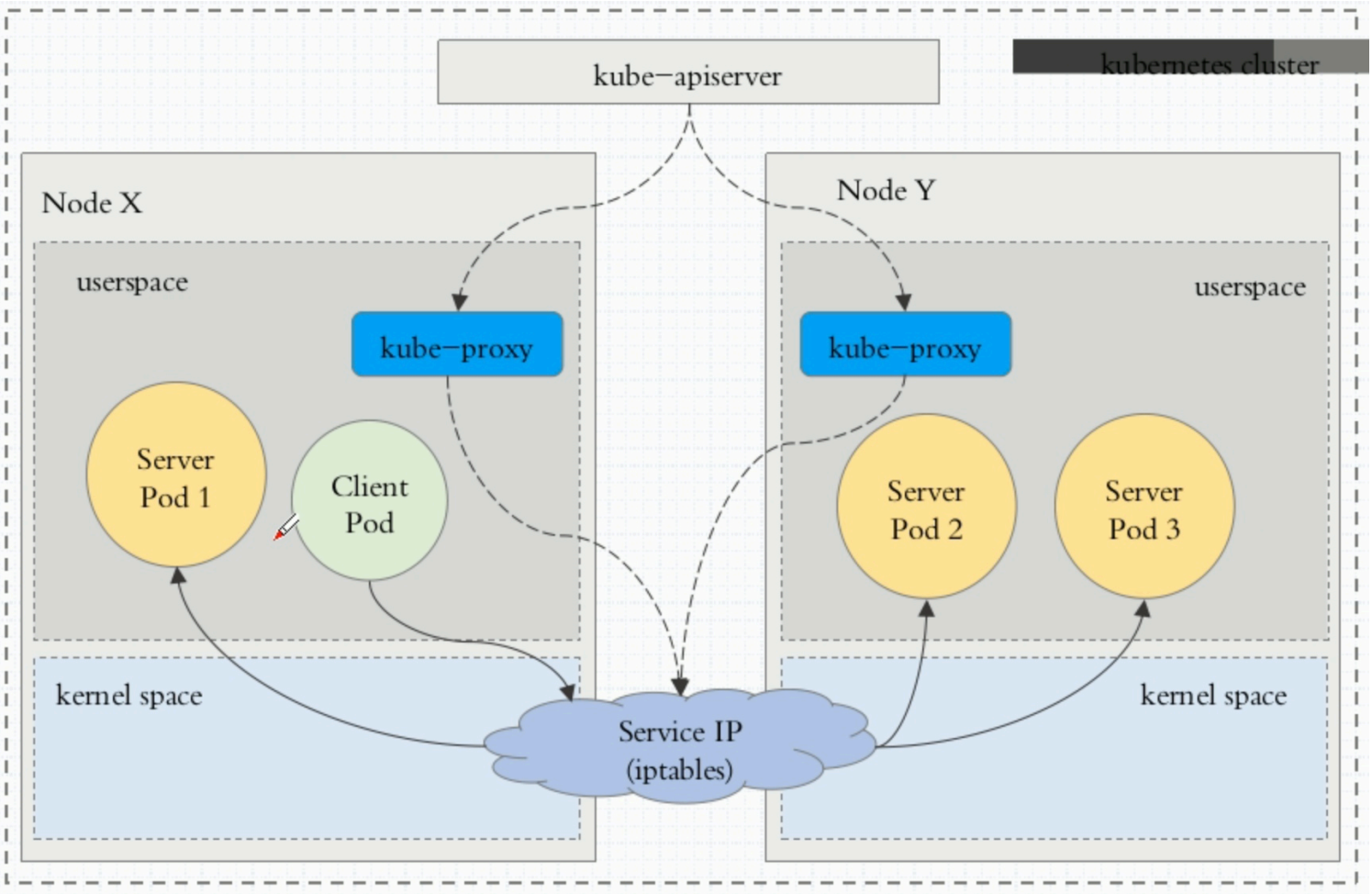
service

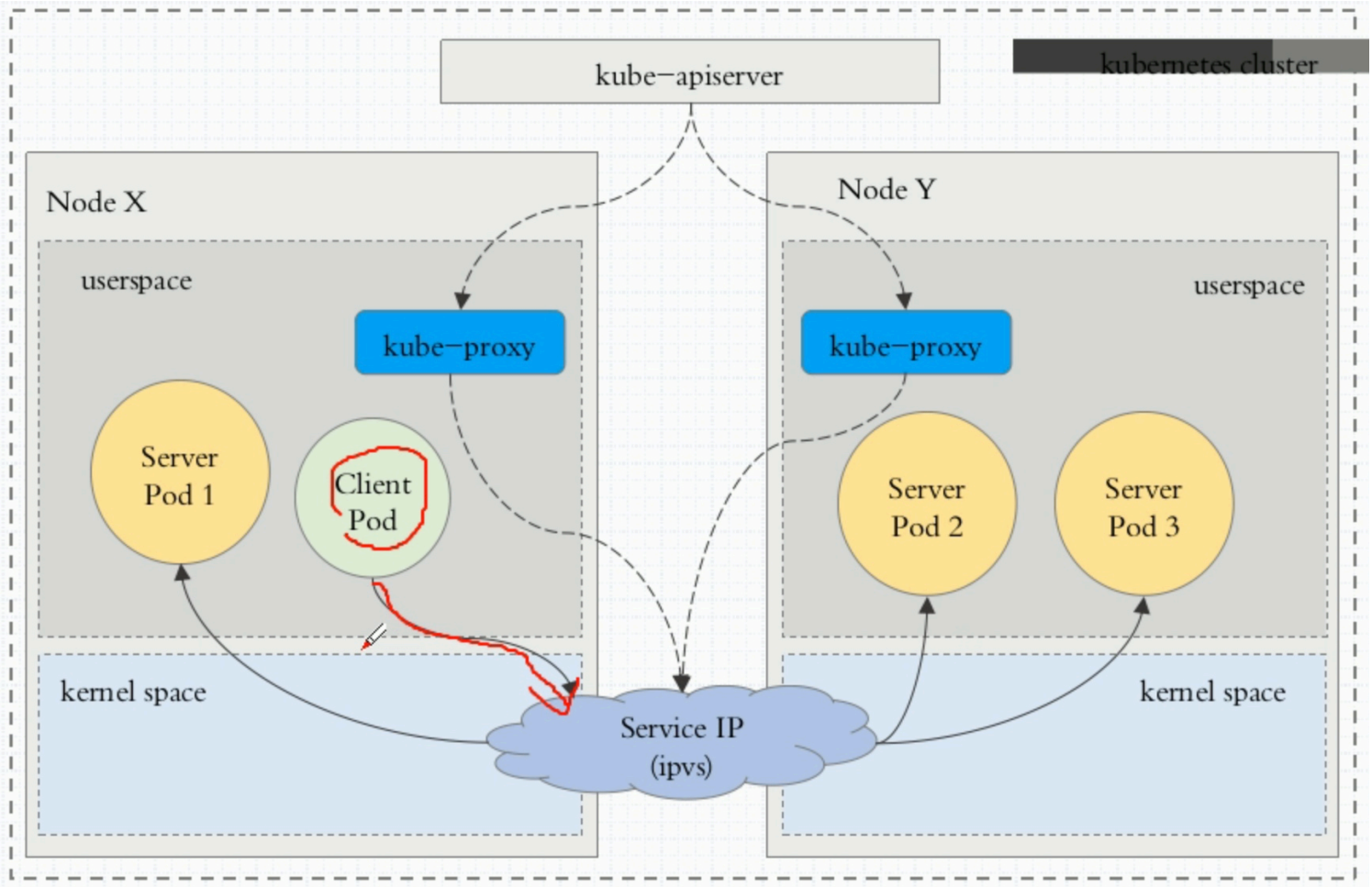
一个 Kubernetes 的 Service 是一种抽象，它定义了一个Pod的逻辑集合和一个用于访问它们的策略 – 有的时候被称之为微服务.

一个Service的目标Pod集合通常是由Label Selector 来决定的.

kube-proxy 中实现 service 的方案: userspace, iptables 和 ipvs.




kube-apiserver
kubernetes cluster
Node X
Node Y
userspace
userspace
kube-proxy
kube-proxy
Server Pod 1
Client Pod
Server Pod 2
Server Pod 3
kernel space
kernel space
Service IP (iptables)



> clusterip 例子

```
[root@qa-k8s-master01 configmap]# kubectl apply -f svc-clusterip-demo.yaml
service/redis created
deployment.apps/redis created
[root@qa-k8s-master01 configmap]# kubectl get pods
NAME                       READY     STATUS    RESTARTS   AGE
pod-nginx                  1/1       Running   0          1d
pod-nginx-hostpath         1/1       Running   0          2d
pod-nginx-pvc              1/1       Running   0          1d
pod-nginx-statefulset-0    1/1       Running   0          1d
pod-nginx-statefulset-1    1/1       Running   0          1d
pod-nginx-statefulset-2    1/1       Running   0          1d
redis-5b5d6fbbbd-kt6dk     1/1       Running   0          26s
[root@qa-k8s-master01 configmap]# kubectl get svc
NAME         TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)    AGE
kubernetes   ClusterIP   10.96.0.1      <none>        443/TCP    9d
myapp        ClusterIP   None           <none>        80/TCP     1d
redis        ClusterIP   10.97.97.98    <none>        6379/TCP   35s
[root@qa-k8s-master01 configmap]# kubectl get deploy
NAME      DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
```

```
redis     1         1         1           1           43s
```

> NodePort 例子

```
[root@qa-k8s-master01 configmap]# kubectl apply -f svc-nodeport-demo.yaml
service/nginx created
deployment.apps/myapp-delpoy created
[root@qa-k8s-master01 configmap]# kubectl get pods -n dev
NAME                           READY     STATUS    RESTARTS   AGE
myapp-delpoy-b6bc574fb-kmvmz   1/1       Running   0          9s
myapp-delpoy-b6bc574fb-rl8xs   1/1       Running   0          9s
[root@qa-k8s-master01 configmap]# kubectl -n dev get svc
NAME      TYPE       CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
nginx     NodePort   10.100.91.108   <none>        80:31659/TCP   34s
[root@qa-k8s-master01 configmap]# kubectl -n dev get deploy
NAME           DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
myapp-delpoy   2         2         2            2           39s
```

> externalName

headless

```
有时你不需要一个单独的服务IP地址，也不需要做负载均衡.
在这种情况下，你可以创建一个"headless"的Service，只需要把集群IP(spec.clusterIP)指定为"
None"即可.

对于这种类型的Service，没有集群IP地址的分配. 也不会有DNS的配置来为一个Service名称返回多个A
记录，这些记录会直接指向支撑这个Service的Pod.
另外，kube-proxy不会处理这些service，并且平台不会为它们做负载均衡或者代理.
endpoints controller仍然会在API中创建Endpoints的记录.
```

```
[root@qa-k8s-master01 configmap]# kubectl apply -f svc-headless-demo.yaml
service/nginx-svc created
deployment.apps/nginx-delpoy created
[root@qa-k8s-master01 configmap]# kubectl get svc
NAME         TYPE        CLUSTER-IP    EXTERNAL-IP   PORT(S)    AGE
kubernetes   ClusterIP   10.96.0.1     <none>        443/TCP    9d
nginx-svc    ClusterIP   None          <none>        80/TCP     18s

[root@qa-k8s-master01 configmap]# dig -t A nginx-svc.default.svc.cluster.local.
 @10.96.0.10
```

```
; <<>> DiG 9.9.4-RedHat-9.9.4-50.el7 <<>> -t A nginx-svc.default.svc.cluster.lo
cal. @10.96.0.10
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 65232
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;nginx-svc.default.svc.cluster.local. IN    A

;; ANSWER SECTION:
nginx-svc.default.svc.cluster.local. 5 IN A 10.244.1.29
nginx-svc.default.svc.cluster.local. 5 IN A 10.244.2.22

;; Query time: 0 msec
;; SERVER: 10.96.0.10#53(10.96.0.10)
;; WHEN: 四 8月 23 00:49:01 CST 2018
;; MSG SIZE  rcvd: 166
```

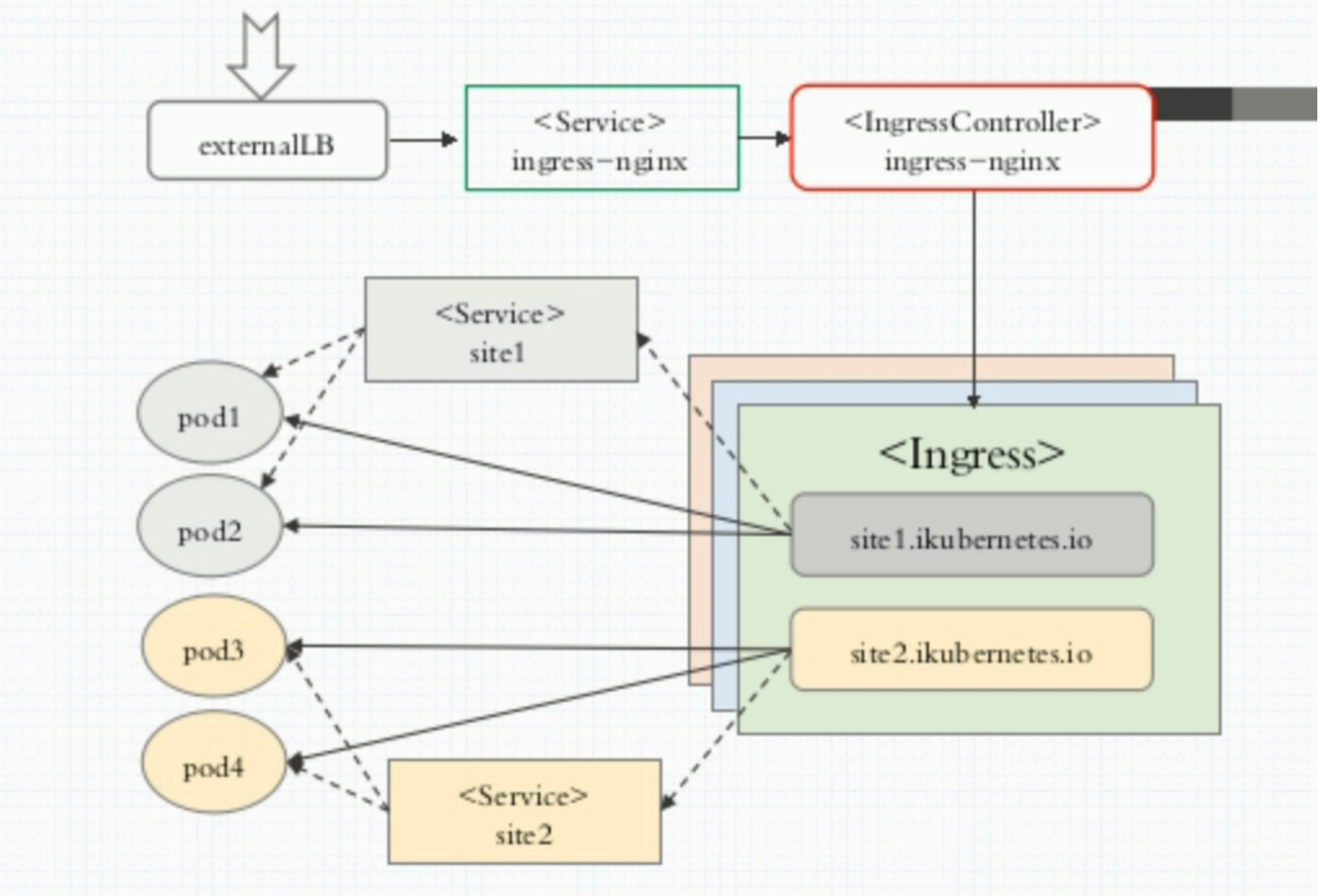
sessionAffinity

```
[root@qa-k8s-master01 configmap]# kubectl patch svc nginx-svc -p '{"spec":{"ses
sionAffinity":"ClientIP"}}'
service/nginx-svc patched
```

Ingress Controller

Ingress Controller 是一个或者一组提供7层代理的 pod.

Ingress 资源, 可以基于后端 **server**(仅仅做后端资源收集, 不做调用) 监控收集到后端 pod, 后端 pod 资源改变后, Ingress可以直接编辑 Ingress Controller,




```
// 安装 Ingress Nginx Controller https://kubernetes.github.io/ingress-nginx/
kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/master/deploy/mandatory.yaml
```

> 准备一个 NodePort Service

```
[root@qa-k8s-master01 ingress]# cat service-nodeport.yaml
apiVersion: v1
kind: Service
metadata:
  name: ingress-nginx
  namespace: ingress-nginx
  labels:
    app.kubernetes.io/name: ingress-nginx
    app.kubernetes.io/part-of: ingress-nginx

spec:
  type: NodePort
```

```
  ports:
  - name: http
    port: 80
    targetPort: 80
    nodePort: 30080
    protocol: TCP
  - name: https
    port: 443
    targetPort: 443
    nodePort: 30443
    protocol: TCP
  selector:
    app.kubernetes.io/name: ingress-nginx

[root@qa-k8s-master01 ingress]# kubectl apply -f service-nodeport.yaml
service/ingress-nginx created
[root@qa-k8s-master01 ingress]# kubectl get svc -n ingress-nginx
NAME                   TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)
                AGE
default-http-backend   ClusterIP   10.99.176.38     <none>        80/TCP
                4m
ingress-nginx          NodePort    10.106.101.154   <none>        80:30080/TCP,
443:30443/TCP   5s
```

> 准备 Ingress Server

```
[root@qa-k8s-master01 ingress]# cat ingress-tomcat-demo.yaml
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: ingress-tomcat
  namespace: default
  annotations:
    kubernetes.io/ingress.class: "nginx"
spec:
  rules:
  - host: tomcat.mljr.com
    http:
      paths:
      - path:
        backend:
          serviceName: tomcat
          servicePort: 8080
```

```
[root@qa-k8s-master01 ingress]# kubectl apply -f ingress-tomcat-demo.yaml
ingress.extensions/ingress-tomcat created
[root@qa-k8s-master01 ingress]# kubectl get  ing
NAME             HOSTS             ADDRESS   PORTS     AGE
ingress-tomcat   tomcat.mljr.com             80        25s
```

> 准备后端 tomcat service

```
[root@qa-k8s-master01 ingress]# cat deploy-tomcat-demo.yaml
apiVersion: v1
kind: Service
metadata:
  name: tomcat
  namespace: default
spec:
  selector:
    app: tomcat
    release: canary
  ports:
  - name: http
    targetPort: 8080
    port: 8080
  - name: ajp
    targetPort: 8009
    port: 8009
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: tomcat-deploy
  namespace: default
spec:
  replicas: 3
  selector:
    matchLabels:
      app: tomcat
      release: canary
  template:
    metadata:
      labels:
        app: tomcat
        release: canary
    spec:
      containers:
```

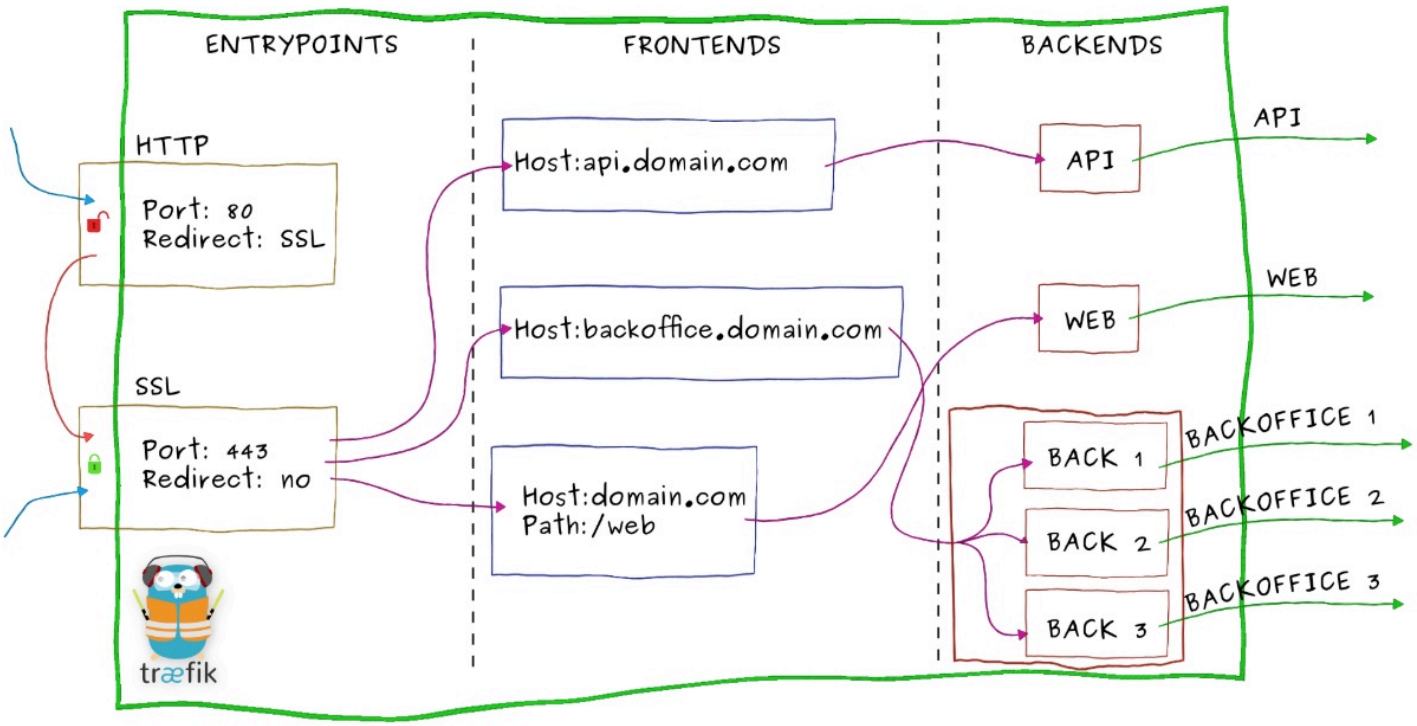
```
      - name: tomcat
        image: tomcat:8.5.32-jre8-alpine
        ports:
        - name: http
          containerPort: 8080
        - name: ajp
          containerPort: 8009

[root@qa-k8s-master01 ingress]# kubectl apply -f deploy-tomcat-demo.yaml
service/tomcat created
deployment.apps/tomcat-deploy created
[root@qa-k8s-master01 ingress]# kubectl get pods -o wide
NAME                             READY     STATUS    RESTARTS   AGE       IP
         NODE
tomcat-deploy-588c79d48d-5qqbq   1/1       Running   0          58s       10.24
4.4.11   qa-k8s-node04.mljr.com
tomcat-deploy-588c79d48d-d8nhc   1/1       Running   0          58s       10.24
4.1.31   qa-k8s-node01.mljr.com
tomcat-deploy-588c79d48d-q4dst   1/1       Running   0          58s       10.24
4.2.23   qa-k8s-node02.mljr.com
```

提供 SSL 服务

```
待补全
http://docs.traefik.cn/
```

Traefik

```
Ingress Controller 的存在就是为了能跟 kubernetes 交互, 又能写 nginx 配置, 还能 reload
 它, 这是一种折中方案.
而最近开始出现的 traefik 天生就是提供了对 kubernetes 的支持, 也就是说 traefik 本身就能跟
 kubernetes API 交互, 感知后端变化, 因此可以得知, 在使用 traefik 时, 整体架构如下.
```



secret configmap

配置容器化应用的方式：

1 自定义命令行参数

```
args: []
```

2 把配置文件直接焙进镜像

3 环境变量

(1) Cloud Native 的应用程序一般可直接通过环境变量加载配置

(2) 通过entrypoint脚本来预处理变量为配置文件中的配置信息

4 存储卷

```
// 配置文件从镜像解耦
configmap
secret
```

```
kubectl create configmap nginx-config --from-literal=nginx_port=8080
```

```
[root@qa-k8s-master01 ~]# kubectl get cm
NAME           DATA      AGE
nginx-config   1         25s
[root@qa-k8s-master01 ~]# kubectl describe cm nginx-config
Name:         nginx-config
Namespace:    default
Labels:       <none>
Annotations:  <none>

Data
====
nginx_port:
----
8080
Events:  <none>
```

```
// 从文件
kubectl create cm nginx-www --from-file=./www.config

[root@qa-k8s-master01 configmap]# cat www.config
server {
    server_name myapp.mljr.com;
    listen 80;
    root /data/web/html;
}
[root@qa-k8s-master01 configmap]# kubectl get cm nginx-www  -o yaml
apiVersion: v1
data:
  www.config: |
    server {
        server_name myapp.mljr.com;
        listen 80;
        root /data/web/html;
    }
kind: ConfigMap
metadata:
  creationTimestamp: 2018-08-20T07:46:43Z
  name: nginx-www
  namespace: default
  resourceVersion: "981539"
  selfLink: /api/v1/namespaces/default/configmaps/nginx-www
  uid: 2e555bd8-a44d-11e8-8cfc-0050569d5ecc
[root@qa-k8s-master01 configmap]#
```

```
[root@qa-k8s-master01 configmap]# cat pod1-configmap-demo.yaml
apiVersion: v1
kind: Pod
metadata:
  name: pod-nginx-configmap
  namespace: default
spec:
  containers:
  - name: mynginx
    image: nginx:1.15-alpine
    imagePullPolicy: IfNotPresent
    ports:
    - name: mynginx
      containerPort: 80
    env:
    - name: NGINX_SERVER_PORT
      valueFrom:
        configMapKeyRef:
          name: nginx-config
          key: nginx_port
    volumeMounts:
    - mountPath: /var/run/secrets/kubernetes.io/serviceaccount
      name: default-token-6chr7
      readOnly: true
  tolerations:
  - effect: NoExecute
    key: node.kubernetes.io/not-ready
    operator: Exists
    tolerationSeconds: 300
  - effect: NoExecute
    key: node.kubernetes.io/unreachable
    operator: Exists
    tolerationSeconds: 300
  volumes:
  - name: default-token-6chr7
    secret:
      defaultMode: 420
      secretName: default-token-6chr7

[root@qa-k8s-master01 configmap]# kubectl apply -f pod1-configmap-demo.yaml
pod/pod-nginx-configmap created
[root@qa-k8s-master01 configmap]# kubectl exec -it pod-nginx-configmap -- /bin/
sh
/ #
```

```
/ # env
KUBERNETES_PORT=tcp://10.96.0.1:443
KUBERNETES_SERVICE_PORT=443
HOSTNAME=pod-nginx-configmap
SHLVL=1
HOME=/root
NGINX_SERVER_PORT=8080
TERM=xterm
NGINX_VERSION=1.15.2
KUBERNETES_PORT_443_TCP_ADDR=10.96.0.1
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
KUBERNETES_PORT_443_TCP_PORT=443
KUBERNETES_PORT_443_TCP_PROTO=tcp
KUBERNETES_PORT_443_TCP=tcp://10.96.0.1:443
KUBERNETES_SERVICE_PORT_HTTPS=443
PWD=/
KUBERNETES_SERVICE_HOST=10.96.0.1
/ # exit
[root@qa-k8s-master01 configmap]#
```

emptyDir

```
// 定义  volume

// 在容器上 挂载 存储器 volumMounts

// 创建 pv

// 定义好 存储类

//
```

gitRepo

```
gitRepo 建立在  emptyDir 基础

git 仓库发生改变, pod 不会自己改变.
```

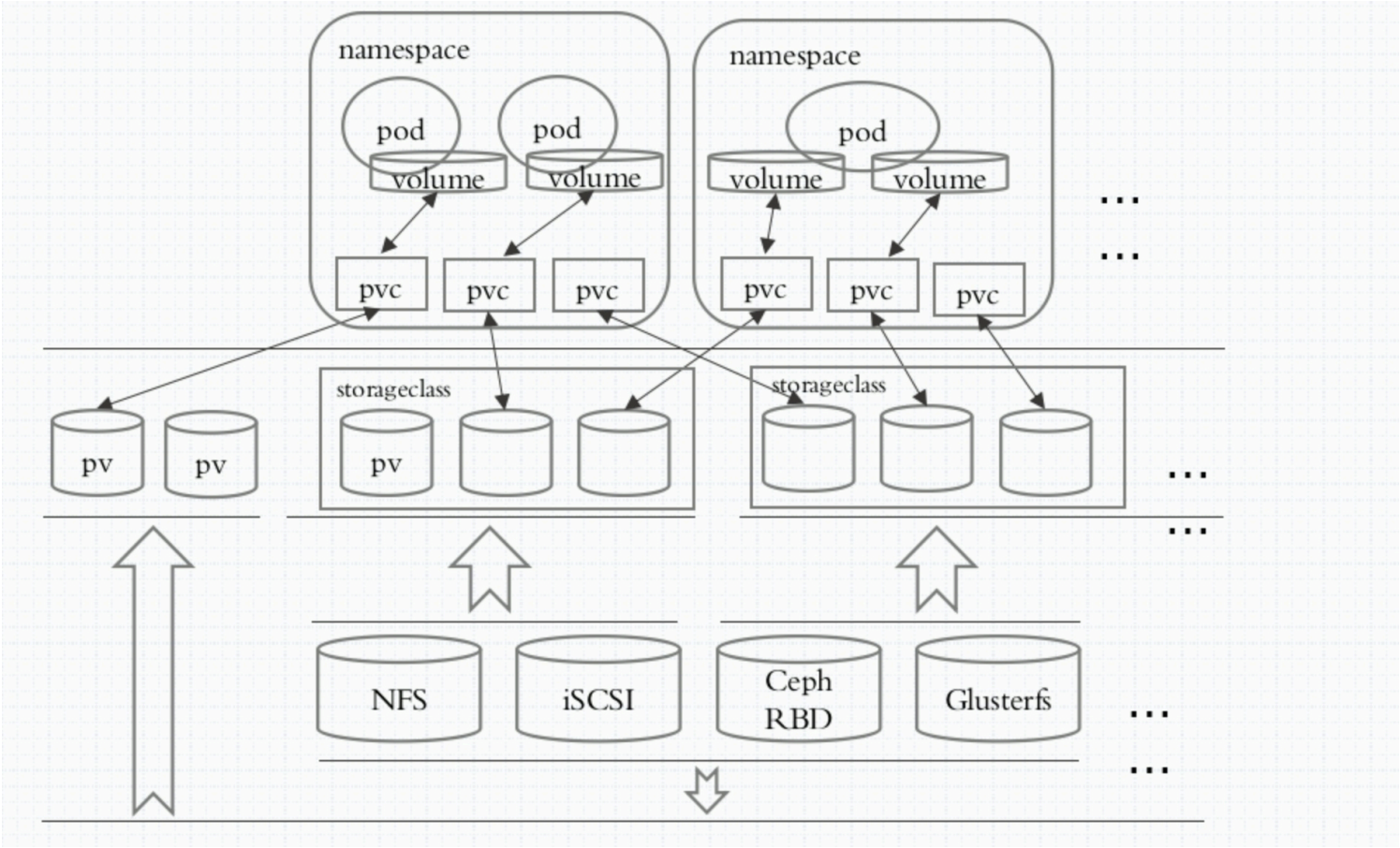
HostPath

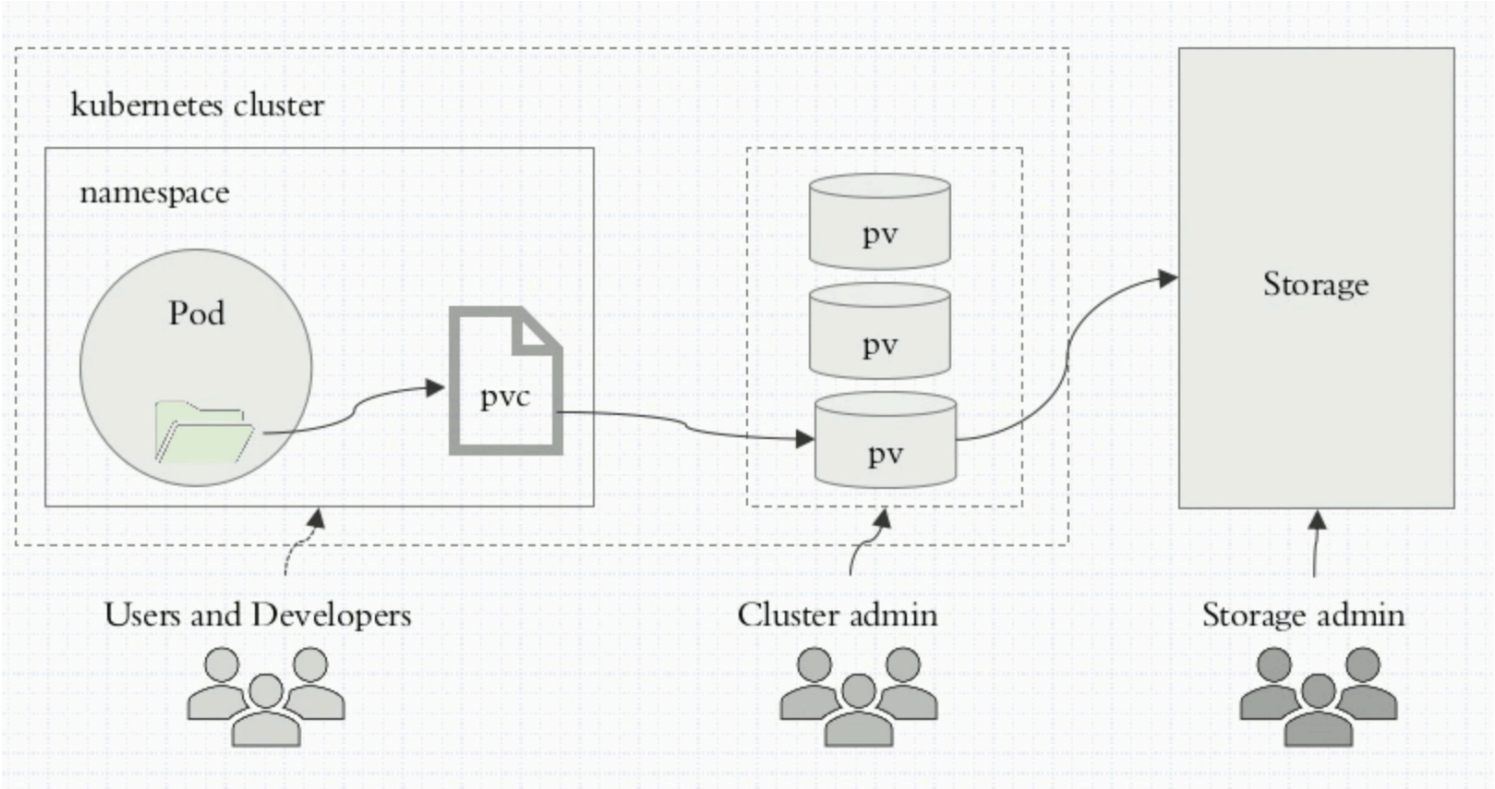
```
// 节点级别的 持久存储

// 所有节点
mkdir -p /data/pod/volume1
echo "$(hostname)" /data/pod/volume1/index.html
```

NFS

```
// 所有节点安装
yum install nfs-utils -y

//  nfs server
mkdir -p /pv/volumes
[root@qa-k8s-master03 ~]# cat /etc/exports
/pv/volumes 172.0.0.0/8(rw,no_root_squash)
[root@qa-k8s-master03 ~]# systemctl  start nfs

// master 节点
[root@qa-k8s-master01 configmap]# cat pod-nfs-demo.yaml
apiVersion: v1
kind: Pod
metadata:
  name: pod-nginx-nfs
  namespace: default
spec:
  containers:
  - name: nginx
    image: nginx:1.15-alpine
    imagePullPolicy: IfNotPresent
    volumeMounts:
    - name: html
      mountPath: /usr/share/nginx/
  volumes:
  - name: html
    nfs:
      path: /pv/volumes
      server: 172.28.28.76
```

volume pv pvc StorageClass






```
// k8s  工程师 创建 pvc, pv

mkdir -p  /pv/volumes/v{1,2,3,4,5}
```

```
[root@qa-k8s-master03 volumes]# cat /etc/exports
/pv/volumes/v1 172.0.0.0/8(rw,no_root_squash)
/pv/volumes/v2 172.0.0.0/8(rw,no_root_squash)
/pv/volumes/v3 172.0.0.0/8(rw,no_root_squash)
/pv/volumes/v4 172.0.0.0/8(rw,no_root_squash)
/pv/volumes/v5 172.0.0.0/8(rw,no_root_squash)
[root@qa-k8s-master03 volumes]# systemctl  daemon-reload
[root@qa-k8s-master03 volumes]# systemctl restart nfs
[root@qa-k8s-master03 volumes]# exportfs -arv
exporting 172.0.0.0/8:/pv/volumes/v5
exporting 172.0.0.0/8:/pv/volumes/v4
exporting 172.0.0.0/8:/pv/volumes/v3
exporting 172.0.0.0/8:/pv/volumes/v2
exporting 172.0.0.0/8:/pv/volumes/v1
[root@qa-k8s-master03 volumes]# showmount -e
Export list for qa-k8s-master03.mljr.com:
/pv/volumes/v5 172.0.0.0/8
/pv/volumes/v4 172.0.0.0/8
/pv/volumes/v3 172.0.0.0/8
/pv/volumes/v2 172.0.0.0/8
/pv/volumes/v1 172.0.0.0/8


[root@qa-k8s-master01 configmap]# cat pod-vol-pvc.yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mypvc
  namespace: default
spec:
  accessModes: ["ReadWriteMany","ReadWriteOnce"]
  resources:
    requests:
      storage: 6Gi
---
apiVersion: v1
kind: Pod
metadata:
  name: pod-nginx-pvc
  namespace: default
spec:
  containers:
  - name: nginx
    image: nginx:1.15-alpine
```

```
    imagePullPolicy: IfNotPresent
    volumeMounts:
    - name: html
      mountPath: /usr/share/nginx/
  volumes:
  - name: html
    persistentVolumeClaim:
      claimName: mypvc
[root@qa-k8s-master01 configmap]# kubectl get pv
NAME      CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM
  STORAGECLASS   REASON    AGE
pv001     5Gi        RWO,RWX        Retain           Available
                           10m
pv002     5Gi        RWO,RWX        Retain           Available
                           10m
pv003     10Gi       RWO,RWX        Retain           Bound       default/mypvc
                           10m
pv004     10Gi       RWO,RWX        Retain           Available
                           10m
pv005     20Gi       RWO,RWX        Retain           Available
                           12m
```

StatefulSet

volumeClaimTemplates 能自动完成两个功能:
    为 pod 定义vloume
    在 pod 名称空间中 自动创建 pvc

```
[root@qa-k8s-master01 configmap]# kubectl get pv
NAME      CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS     CLAIM
                         STORAGECLASS   REASON    AGE
pv001     5Gi        RWO,RWX        Retain           Bound      default/html-po
d-nginx-statefulset-0                              15h
pv002     5Gi        RWO,RWX        Retain           Bound      default/html-po
d-nginx-statefulset-1                              15h
pv003     10Gi       RWO,RWX        Retain           Released   default/mypvc
                                                   15h
pv004     10Gi       RWO,RWX        Retain           Bound      default/mypvc
                                                   15h
pv005     20Gi       RWO,RWX        Retain           Bound      default/varlog
                                                   15h
[root@qa-k8s-master01 configmap]# kubectl get pvc
```

```
NAME                            STATUS    VOLUME    CAPACITY   ACCESS MODES   ST
ORAGECLASS   AGE
html-pod-nginx-statefulset-0   Bound     pv001     5Gi        RWO,RWX
             10m
html-pod-nginx-statefulset-1   Bound     pv002     5Gi        RWO,RWX
             10m
mypvc                           Bound     pv004     10Gi       RWO,RWX
             15h
varlog                          Bound     pv005     20Gi       RWO,RWX
             38m
[root@qa-k8s-master01 configmap]# kubectl get pods
NAME                       READY     STATUS    RESTARTS   AGE
pod-nginx-hostpath         1/1       Running   0          16h
pod-nginx-pvc              1/1       Running   0          15h
pod-nginx-statefulset-0    1/1       Running   0          10m
pod-nginx-statefulset-1    1/1       Running   0          10m

[root@qa-k8s-master01 configmap]# kubectl delete -f  pod-statefulset-demo.yaml
service "myapp" deleted
statefulset.apps "pod-nginx-statefulset" deleted
[root@qa-k8s-master01 configmap]# kubectl get pods
NAME                       READY     STATUS        RESTARTS   AGE
pod-nginx-hostpath         1/1       Running       0          16h
pod-nginx-pvc              1/1       Running       0          15h
pod-nginx-statefulset-1    0/1       Terminating   0          12m
[root@qa-k8s-master01 configmap]# kubectl get pvc
NAME                            STATUS    VOLUME    CAPACITY   ACCESS MODES   ST
ORAGECLASS   AGE
html-pod-nginx-statefulset-0   Bound     pv001     5Gi        RWO,RWX
             12m
html-pod-nginx-statefulset-1   Bound     pv002     5Gi        RWO,RWX
             12m
mypvc                           Bound     pv004     10Gi       RWO,RWX
             15h
varlog                          Bound     pv005     20Gi       RWO,RWX
             41m
[root@qa-k8s-master01 configmap]# kubectl get pv
NAME      CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS     CLAIM
                      STORAGECLASS   REASON    AGE
pv001     5Gi        RWO,RWX        Retain           Bound      default/html-po
d-nginx-statefulset-0                             15h
pv002     5Gi        RWO,RWX        Retain           Bound      default/html-po
d-nginx-statefulset-1                             15h
pv003     10Gi       RWO,RWX        Retain           Released   default/mypvc
                                                  15h
```

```
pv004     10Gi       RWO,RWX        Retain           Bound      default/mypvc
                                                    15h
pv005     20Gi       RWO,RWX        Retain           Bound      default/varlog
                                                    15h
[root@qa-k8s-master01 configmap]# kubectl apply -f pod-statefulset-demo.yaml
service/myapp created
statefulset.apps/pod-nginx-statefulset created
```

```
nslookup  pod-nginx-statefulset-1.myapp.default.svc.cluster.local

// 字段如下
pod_name.service_name.namespce_name.svc.cluster.local

[root@qa-k8s-master01 configmap]# dig -t A pod-nginx-statefulset-1.myapp.defaul
t.svc.cluster.local @10.96.0.10

; <<>> DiG 9.9.4-RedHat-9.9.4-50.el7 <<>> -t A pod-nginx-statefulset-1.myapp.de
fault.svc.cluster.local @10.96.0.10
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 1809
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;pod-nginx-statefulset-1.myapp.default.svc.cluster.local. IN A

;; ANSWER SECTION:
pod-nginx-statefulset-1.myapp.default.svc.cluster.local. 5 IN A 10.244.2.5

;; Query time: 0 msec
;; SERVER: 10.96.0.10#53(10.96.0.10)
;; WHEN: 二 8月 21 11:26:53 CST 2018
;; MSG SIZE  rcvd: 155
```

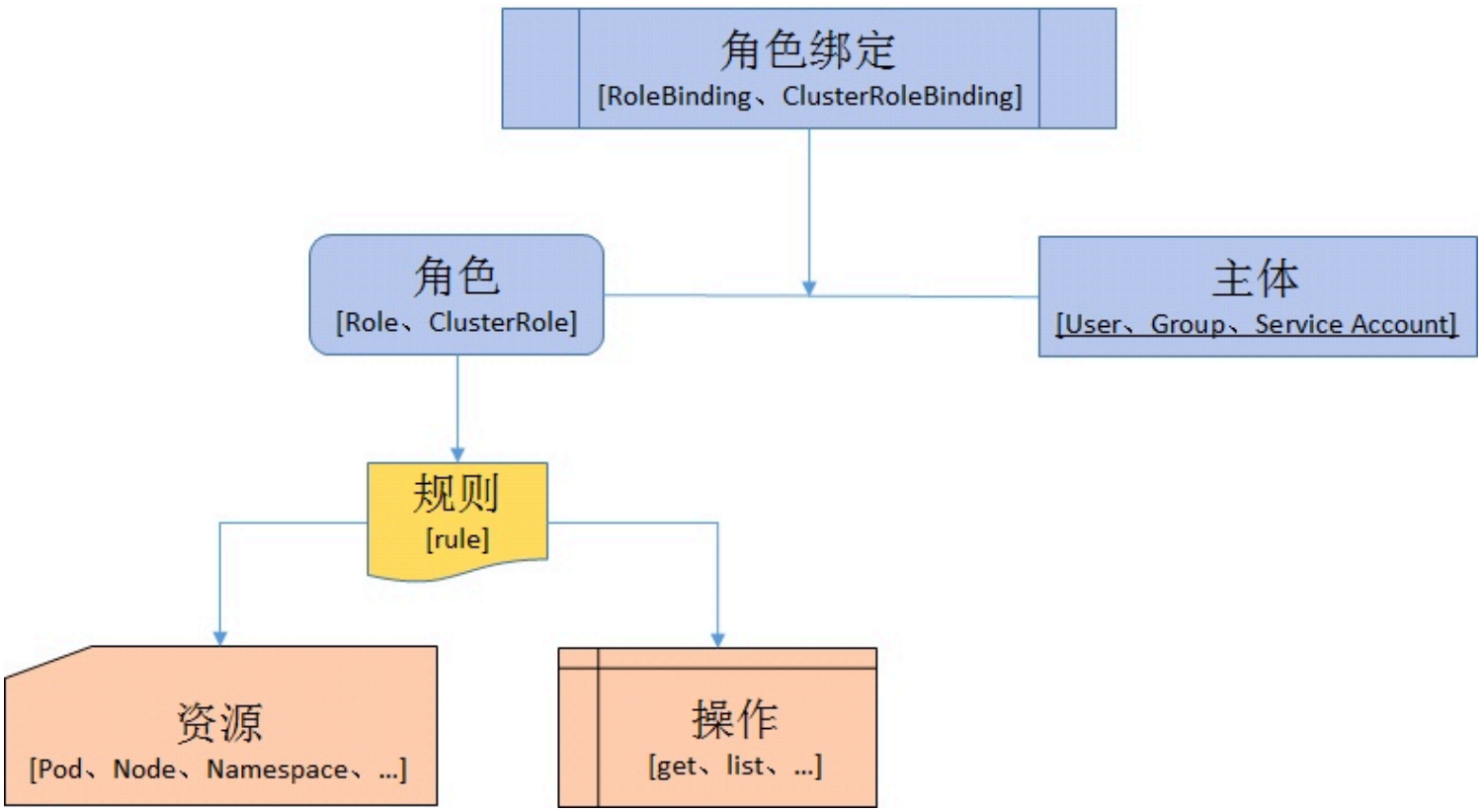
```
// 扩容
kubectl scale statefulset pod-nginx-statefulset --replices=3

[root@qa-k8s-master01 configmap]# kubectl scale statefulset pod-nginx-statefuls
et --replicas=3
statefulset.apps/pod-nginx-statefulset scaled
```

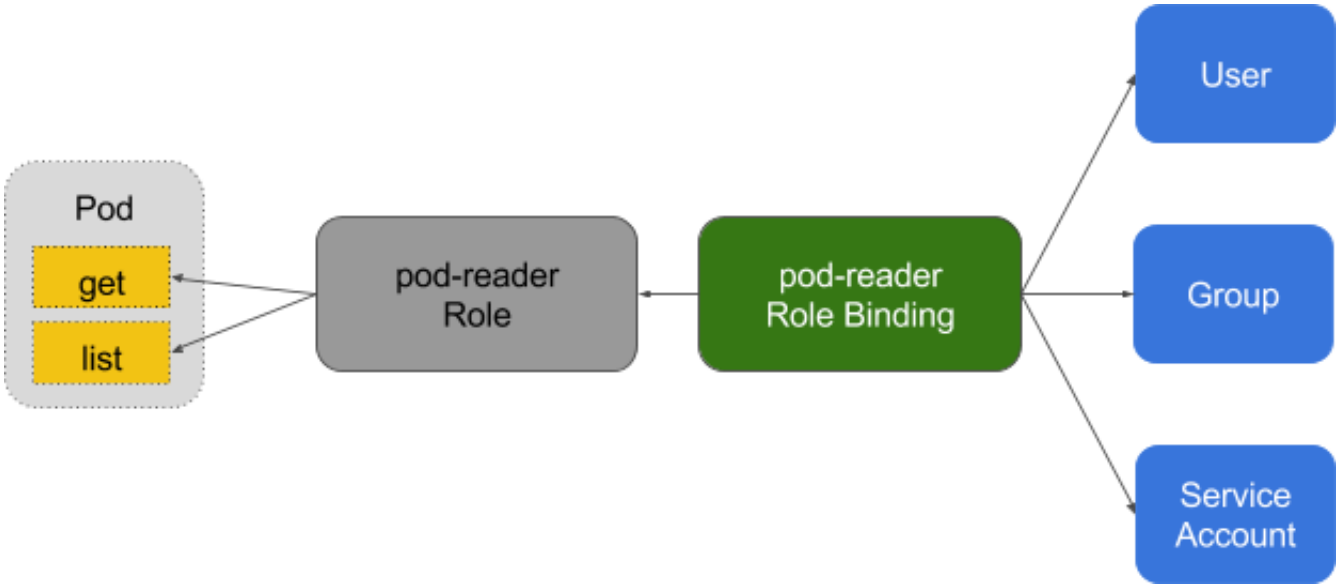
```
[root@qa-k8s-master01 configmap]# kubectl get pods
NAME                       READY     STATUS    RESTARTS   AGE
pod-nginx-hostpath         1/1       Running   0          17h
pod-nginx-pvc              1/1       Running   0          15h
pod-nginx-statefulset-0    1/1       Running   0          23m
pod-nginx-statefulset-1    1/1       Running   0          23m
pod-nginx-statefulset-2    1/1       Running   0          29s
```

```
// 更新策略
[root@qa-k8s-master01 configmap]# kubectl explain sts.spec.updateStrategy.rolli
ngUpdate
KIND:     StatefulSet
VERSION:  apps/v1

RESOURCE: rollingUpdate <Object>

DESCRIPTION:
     RollingUpdate is used to communicate parameters when Type is
     RollingUpdateStatefulSetStrategyType.

     RollingUpdateStatefulSetStrategy is used to communicate parameter for
     RollingUpdateStatefulSetStrategyType.

FIELDS:
   partition    <integer>
     Partition indicates the ordinal at which the StatefulSet should be
     partitioned. Default value is 0.

partition : N 用于模拟金丝雀更新
    表示 大于等于 N 的, 才会更新. 0 表示都更新.

// 查看 sts 更新策略
[root@qa-k8s-master01 configmap]# kubectl describe sts  pod-nginx-statefulset |
 grep Update
Update Strategy:    RollingUpdate

// 实现模拟金丝雀更新
[root@qa-k8s-master01 configmap]# kubectl patch sts pod-nginx-statefulset -p '{
"spec":{"updateStrategy":{"rollingUpdate":{"partition":2}}}}'
// 然后更新测试
[root@qa-k8s-master01 configmap]# kubectl set image sts/pod-nginx-statefulset m
yapp=nginx:1.14-alpine
statefulset.apps/pod-nginx-statefulset image updated
[root@qa-k8s-master01 configmap]# kubectl get sts -o wide
```

```
NAME                    DESIRED   CURRENT   AGE       CONTAINERS   IMAGES
pod-nginx-statefulset   3         3         1h        myapp        nginx:1.14-a
lpine
[root@qa-k8s-master01 configmap]# kubectl describe pod pod-nginx-statefulset-2
| grep 'Image:'
    Image:          nginx:1.14-alpine
[root@qa-k8s-master01 configmap]# kubectl describe pod pod-nginx-statefulset-1
| grep 'Image:'
    Image:          nginx:1.15-alpine
[root@qa-k8s-master01 configmap]#  kubectl patch sts pod-nginx-statefulset -p '
{"spec":{"updateStrategy":{"rollingUpdate":{"partition":0}}}}'
statefulset.apps/pod-nginx-statefulset patched
[root@qa-k8s-master01 configmap]# kubectl describe pod pod-nginx-statefulset-1
| grep 'Image:'
    Image:          nginx:1.14-alpine
[root@qa-k8s-master01 configmap]# kubectl describe pod pod-nginx-statefulset-0
| grep 'Image:'
    Image:          nginx:1.14-alpine
```

rbac





role

```
kubectl create role pods-reader --verb=get,list,watch --resource=pods --dry-run
 -o yaml > role-demo.yaml

kubectl apply -f role-demo.yaml


[root@qa-k8s-master01 ~]# kubectl create rolebinding dashboard-read-pods-rolebi
nding --role=pods-reader --user=dashboard  --dry-run -o yaml  > rolebinding-dem
o.yaml
[root@qa-k8s-master01 ~]# cat rolebinding-demo.yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  creationTimestamp: null
  name: dashboard-read-pods-rolebinding
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: pods-reader
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: User
  name: dashboard
```

serviceaccount

```
[root@qa-k8s-master01 networkpolicy]# kubectl create  serviceaccount mysa -o ya
ml --dry-run
apiVersion: v1
kind: ServiceAccount
metadata:
  creationTimestamp: null
  name: mysa
```

```
[root@qa-k8s-master01 networkpolicy]# kubectl create  serviceaccount myadmin
serviceaccount/myadmin created
[root@qa-k8s-master01 networkpolicy]# kubectl get sa
NAME             SECRETS   AGE
def-ns-default   1         9h
default          1         6d
myadmin          1         12s
[root@qa-k8s-master01 networkpolicy]# kubectl get secret
NAME                         TYPE                                  DATA      AG
E
def-ns-default-token-x4w2d   kubernetes.io/service-account-token   3         9h
default-token-6chr7          kubernetes.io/service-account-token   3         6d
myadmin-token-qpmvj          kubernetes.io/service-account-token   3         45
s
[root@qa-k8s-master01 networkpolicy]#
```

```
kubectl get pods client-7c9999bd74-cqz6h -o yaml
```

ResourceQuota

kubeconfig

```
[root@qa-k8s-master01 kubernetes]# kubectl config view
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data: REDACTED
    server: https://172.28.28.70:6443
```

```
  name: kubernetes
contexts:
- context:
    cluster: kubernetes
    user: kubernetes-admin
  name: kubernetes-admin@kubernetes
current-context: kubernetes-admin@kubernetes
kind: Config
preferences: {}
users:
- name: kubernetes-admin
  user:
    client-certificate-data: REDACTED
    client-key-data: REDACTED
```

```
kubectl config  set-credentials dashboard-admin --client-certificate=./dashboard.crt --client-key=./dashboard.key

kubectl config set-context dashboard-admin@kubernetes --cluster=kubernetes --user=dashboard-admin

kubectl config use-context dashboard-admin@kubernetes
```

CNI

```
Kubernetes网络通信:
    (1) 容器间通信: 同一个Pod内的多个容器间的通信, lo
    (2) Pod通信: Pod IP <--> Pod IP
    (3) Pod与Service通信: PodIP <--> ClusterIP
    (4) Service与集群外部客户端的通信;
```

Flannel

Flannel Directrouting

```
使用Flannel, 添加一台新机器到集群时, Flannel 做了三件事:
    使用 etcd 为新增机器分配一个子网
    在宿主机上创建一个虚拟桥接接口(名为 docker0 的桥接器)
    设置一个网络转发后端
```

```
flannel 支持多种后端:
    VxLAN: 创建一个虚拟的 VXLAN 接口
        (1) vxlan
        (2) Directrouting (同一二层网络使用 host-gw, 跨三层使用 vxlan)
    host-gw: 通过远程机器 IP, 创建到子网的IP路由.这要求运行 flannel 的不同主机在二层直接互通.
    UDP:
```

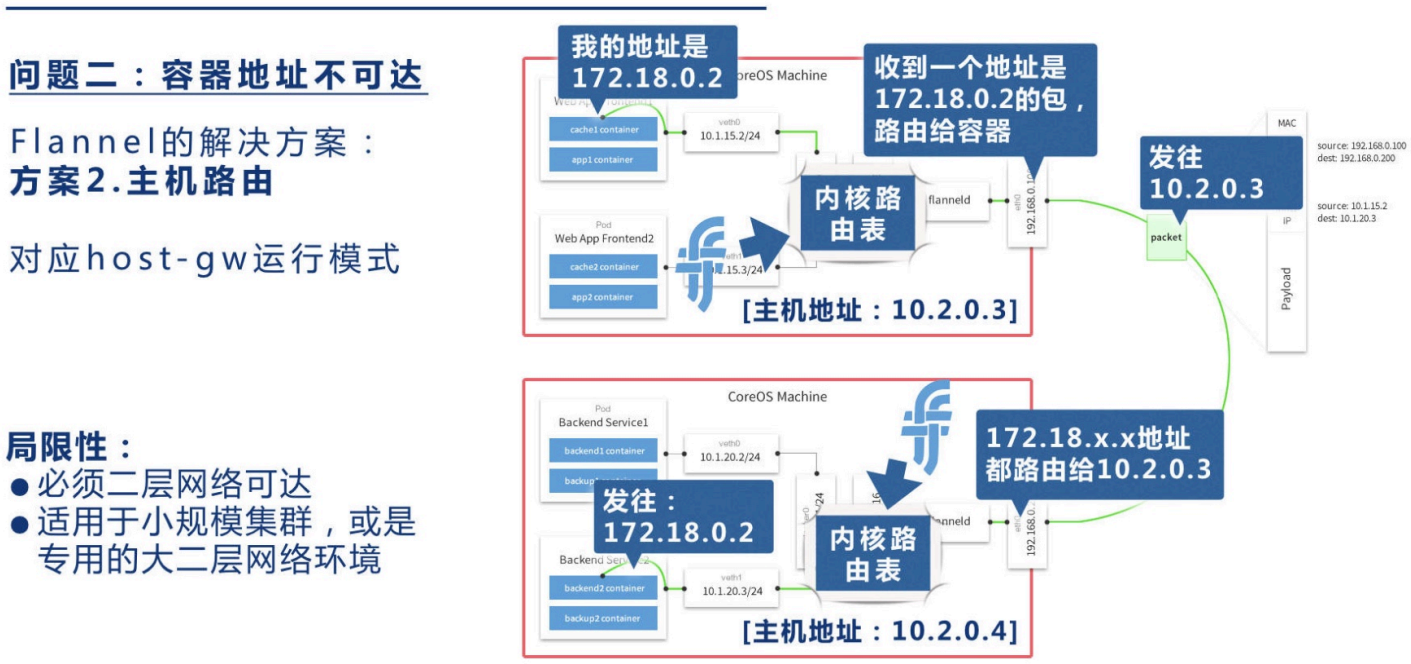
```
[root@qa-k8s-master01 configmap]# kubectl get pods -o wide
NAME                           READY     STATUS    RESTARTS   AGE       IP
       NODE                     NOMINATED NODE
nginx-delpoy-869b48bf9-glqf8   1/1       Running   0          2m        10.244.
4.94   qa-k8s-node03.mljr.com   <none>
nginx-delpoy-869b48bf9-ldvzm   1/1       Running   0          2m        10.244.
3.5    qa-k8s-node04.mljr.com   <none>

[root@qa-k8s-node04 ~]# ip route show
default via 172.28.3.254 dev ens160 proto static metric 100
10.244.0.0/24 via 10.244.0.0 dev flannel.1 onlink
10.244.1.0/24 via 10.244.1.0 dev flannel.1 onlink
10.244.2.0/24 via 10.244.2.0 dev flannel.1 onlink
10.244.3.0/24 dev cni0 proto kernel scope link src 10.244.3.1
10.244.4.0/24 via 10.244.4.0 dev flannel.1 onlink
172.17.0.0/16 dev docker0 proto kernel scope link src 172.17.0.1
172.28.3.0/24 dev ens160 proto kernel scope link src 172.28.3.55 metric 100

[root@qa-k8s-node03 ~]# ip route show
default via 172.28.3.254 dev ens160 proto static metric 100
10.244.0.0/24 via 10.244.0.0 dev flannel.1 onlink
10.244.1.0/24 via 10.244.1.0 dev flannel.1 onlink
10.244.2.0/24 via 10.244.2.0 dev flannel.1 onlink
10.244.3.0/24 via 10.244.3.0 dev flannel.1 onlink
10.244.4.0/24 dev cni0 proto kernel scope link src 10.244.4.1
172.17.0.0/16 dev docker0 proto kernel scope link src 172.17.0.1
172.28.3.254 dev ens160 proto static scope link metric 100
172.28.28.0/24 dev ens160 proto kernel scope link src 172.28.28.74 metric 100
```

```
// 下载 kube-flannel.yml , 修改如下字段, 然后重新应用.
net-conf.json: |
    {
      "Network": "10.244.0.0/16",
      "Backend": {
```

```
        "Type": "vxlan",
        "Directrouting": true
      }
    }

kubectl delete -f kube-flannel.yml
kubectl apply -f kube-flannel.yml
```

```
// 重新生成pod节点查看路由
[root@qa-k8s-node03 ~]# ip route show
default via 172.28.3.254 dev ens160 proto static metric 100
10.244.0.0/24 via 172.28.28.75 dev ens160
10.244.1.0/24 via 172.28.28.71 dev ens160
10.244.2.0/24 via 172.28.28.76 dev ens160
10.244.3.0/24 via 10.244.3.0 dev flannel.1 onlink
10.244.4.0/24 dev cni0 proto kernel scope link src 10.244.4.1
172.17.0.0/16 dev docker0 proto kernel scope link src 172.17.0.1
172.28.3.254 dev ens160 proto static scope link metric 100
172.28.28.0/24 dev ens160 proto kernel scope link src 172.28.28.74 metric 100
```

Flannel host-gw



Calico

> Installing Calico for policy and flannel for networking

```
kubectl apply -f \
https://docs.projectcalico.org/v3.1/getting-started/kubernetes/installation/hosted/canal/rbac.yaml

kubectl apply -f \
https://docs.projectcalico.org/v3.1/getting-started/kubernetes/installation/hosted/canal/canal.yaml
```

```
mkdir /etc/kubernetes/networkpolicy
cd /etc/kubernetes/networkpolicy

// 建立 2个 命名空间  dev  prod
kubectl create  ns  dev
kubectl create  ns  prod

// 创建 ingress 默认 deny 规则
[root@qa-k8s-master01 networkpolicy]# cat ingress-def.ymal
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
    name: deny-all-ingress
spec:
    podSelector: {}
    policyTypes:
    - Ingress

// 应用到 dev 命名空间上
kubectl apply -f ingress-def.ymal  -n dev

// 创建 ingress 默认放行规则
[root@qa-k8s-master01 networkpolicy]# cat ingress-def.ymal
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
    name: allow-all-ingress
spec:
    podSelector: {}
    ingress:
    - {}
    policyTypes:
    - Ingres
```

```
[root@qa-k8s-master01 networkpolicy]# cat allow-netpol-demo.yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-myapp-ingress
spec:
  podSelector:
    matchLabels:
      app: myapp
  ingress:
  - from:
    - ipBlock:
        cidr: 10.244.0.0/16
        except:
        - 10.244.1.2/32
    ports:
    - protocol: TCP
      port: 80
```

calico的缺点:

路由的数目与容器数目相同, 非常容易超过路由器, 三层交换, 甚至**node的处理能力, 从而限制了整个网络的扩张.**

calico的每个**node上会设置大量(海量)的iptables**规则, 路由, 运维/排障难度大.

calico的原理决定了它不可能支持VPC, 容器只能从calico设置的网段中获取ip.

calico目前的实现没有流量控制的功能, 会出现少数容器抢占**node多数带宽的情况.**

calico的网络规模受到BGP网络规模的限制.

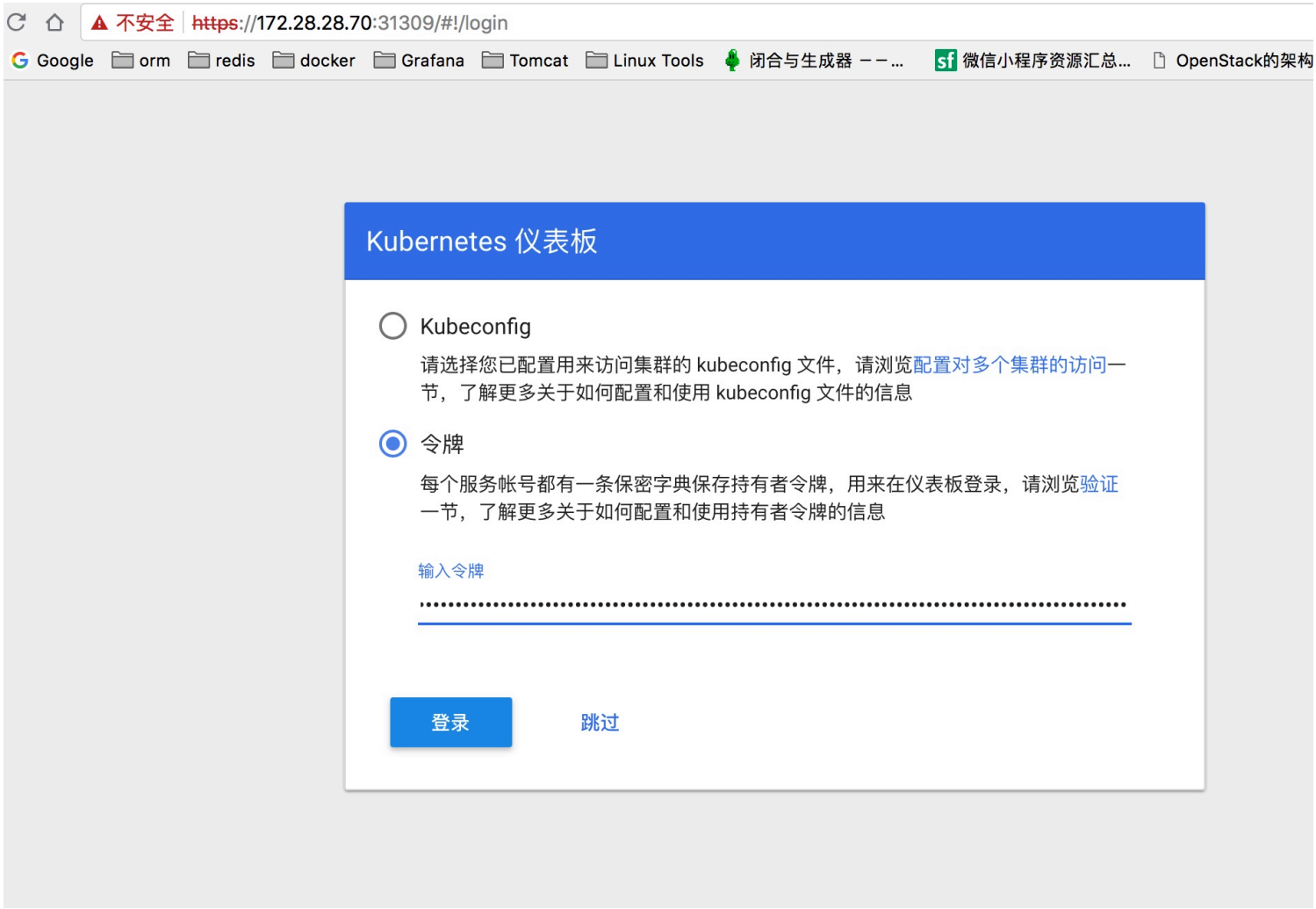
Dashboard

> token

```
cd /etc/kubernetes/pki/

// 生成 dashboard 私钥
(umask 077; openssl genrsa -out dashboard.key 2048)
```

```
// 建立证书签署请求
openssl req -new -key dashboard.key  -out dashboard.csr -subj "/O=mljr/CN=dashboard.mljr.com/"

// 签署证书
[root@qa-k8s-master01 pki]# openssl  x509 -req -in  dashboard.csr -CA ca.crt  -CAkey ca.key  -CAcreateserial -out dashboard.crt -days 3650
Signature ok
subject=/O=mljr/CN=dashboard.mljr.com
Getting CA Private Key

// 查看证书信息
[root@qa-k8s-master01 pki]# openssl x509 -in dashboard.crt -text -noout
Certificate:
    Data:
        Version: 1 (0x0)
        Serial Number:
            c2:12:50:e1:21:65:5c:88
    Signature Algorithm: sha256WithRSAEncryption
        Issuer: CN=kubernetes
        Validity
            Not Before: Aug 19 04:01:52 2018 GMT
            Not After : Aug 16 04:01:52 2028 GMT
        Subject: O=mljr, CN=dashboard.mljr.com

// 创建 secret
kubectl create secret generic dashboard-cert -n kube-system --from-file=dashboard.crt=./dashboard.crt --from-file=dashboard.key=./dashboard.key

// 创建 serviceaccount
kubectl  create  clusterrolebinding dashboard-cluster-admin --clusterrole=cluster-admin  --serviceaccount=kube-system:dashboard-admin

// 查看 token
kubectl get secret  -n kube-system | grep dashboard-admin
kubectl describe secret dashboard-admin-token-vqdnr -n kube-system

// 部署 Dashboard
kubectl apply -f https://raw.githubusercontent.com/kubernetes/dashboard/master/src/deploy/recommended/kubernetes-dashboard.yaml

// 设置  service NodePort
kubectl patch  svc kubernetes-dashboard -p '{"spec":{"type":"NodePort"}}' -n kube-system
```

```
[root@qa-k8s-master01 pki]# kubectl get svc -n kube-system
NAME                   TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)
  AGE
kube-dns               ClusterIP   10.96.0.10      <none>        53/UDP,53/TCP
  5d
kubernetes-dashboard   NodePort    10.97.156.106   <none>        443:31309/TCP
  4m

// token 登陆 dashboard
https://172.28.28.70:31309/#!/login
```



> kubeconfig

```
# 实现 对 default 完全管理

// 创建 serviceaccount
kubectl create serviceaccount def-ns-default -n default
```

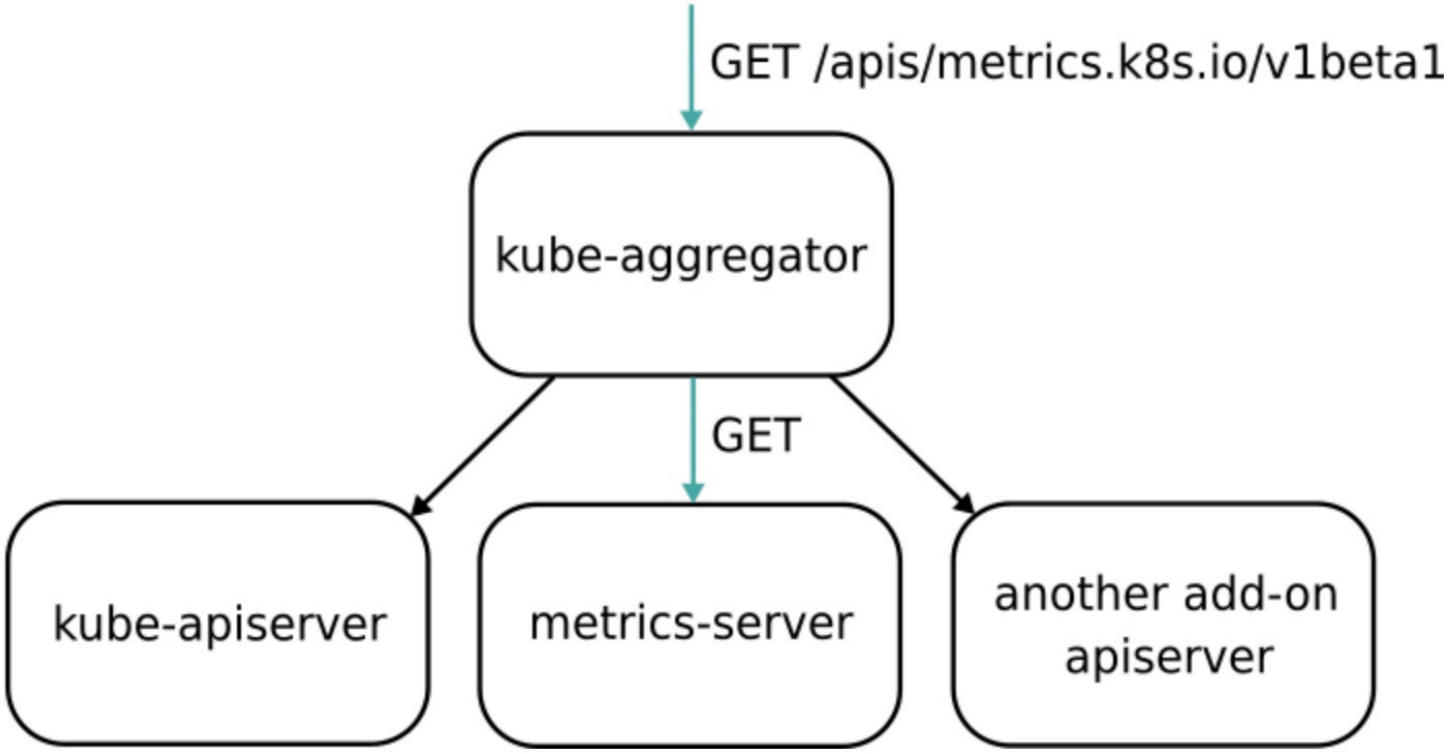
```
// 绑定
kubectl create  rolebinding def-ns-admin --clusterrole=admin --serviceaccount=default:def-ns-default

// 此时 kubectl get secret  | grep def-ns-default 找出的账号 的 token 对 default 有完全管理权限
kubectl get secret def-ns-default-token-x4w2d  -o jsonpath={.data.token}

kubectl config  set-credentials dashboard-admin --
```

Monitor

Metrics-Server

```
Kubernetes Metrics Server 是一个集群范围的资源使用数据聚合器, 是 Heapster 的继承者.

The metrics server collects CPU and memory usage for nodes and pods by pooling data from the kubernetes.summary_api.
```

```
核心指标流水线: 由kubelet, metrics-server以及由API server提供的api组成;
CPU累积使用率, 内存实时使用率,
Pod的资源占用率及容器的磁盘占用率;

监控流水线: 用于从系统收集各种指标数据并提供终端用户, 存储系统以及HPA, 它们包含核心指标及许多非核心指
标.
非核心指标本身不能被k8s所解析.
```




```
// 部署 Metrics Server 服务后,  kubectl top 命令就可以访问了
```

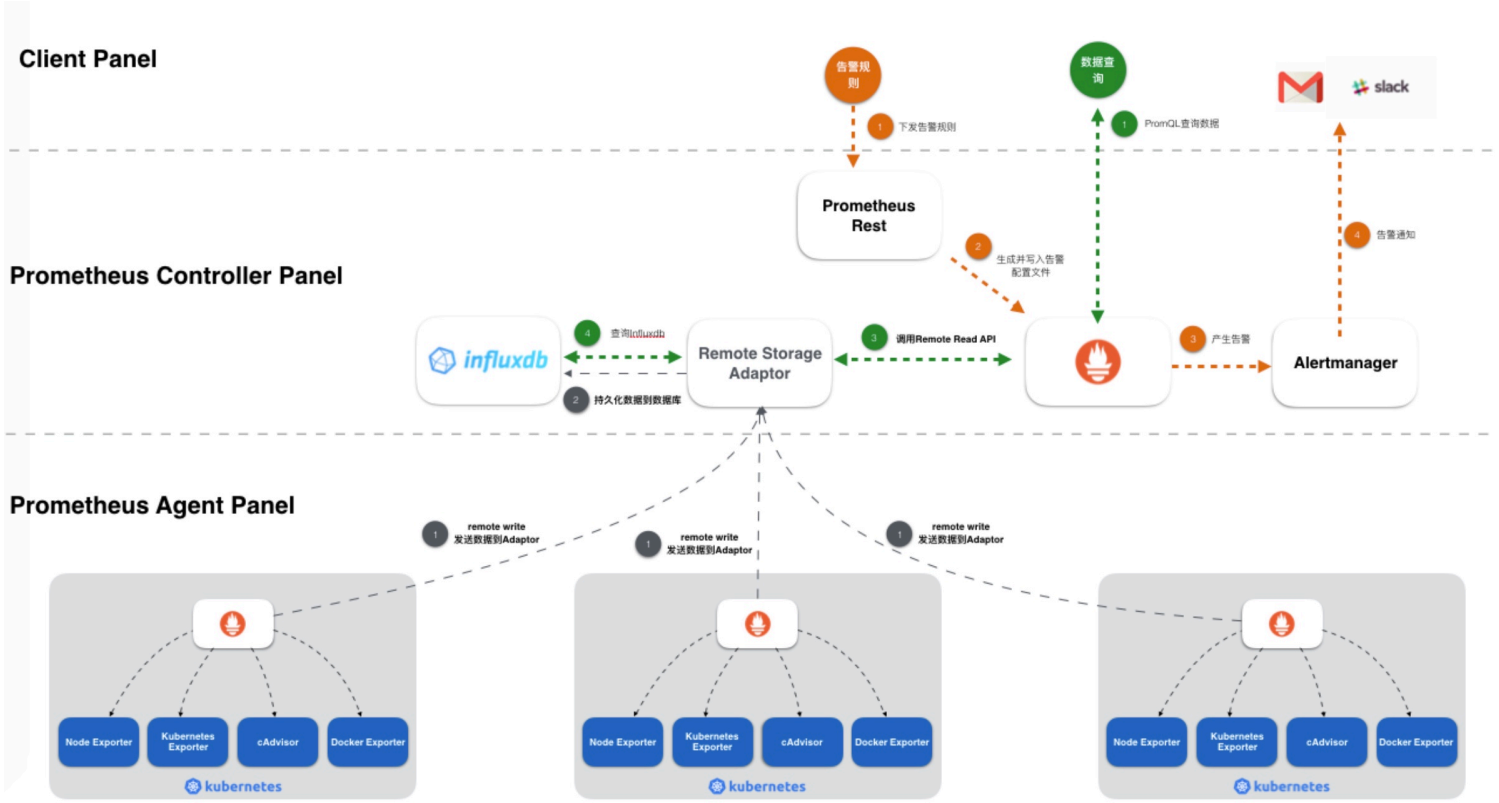
```
nohup kubectl proxy --port 8080 &
curl http://localhost:8080/apis/metrics.k8s.io/v1beta1/nodes
```

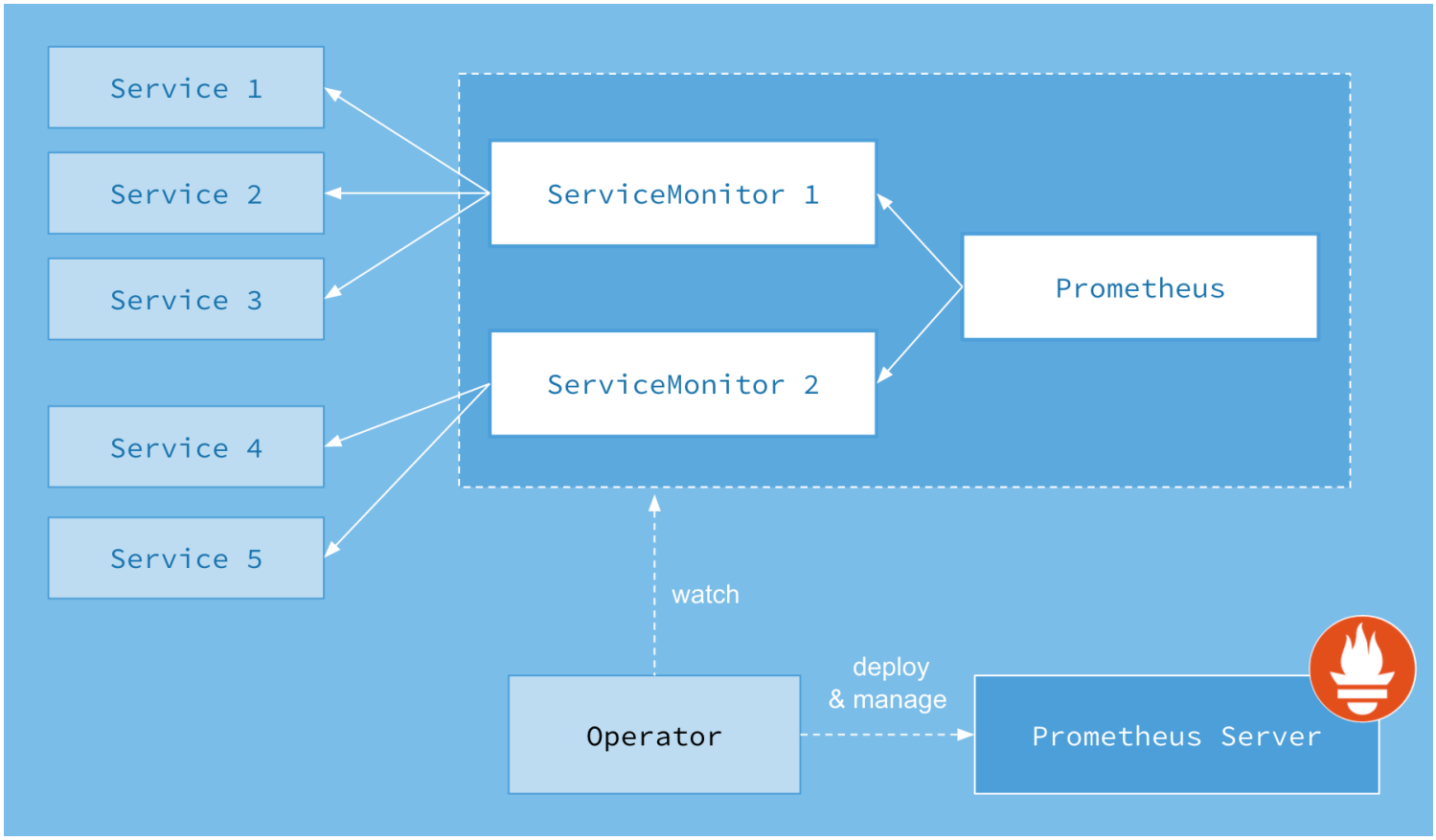
customresourcedefinitions

```
在 Kubernetes 中一切都可视为资源, Kubernetes 1.7 之后增加了对 CRD 自定义资源二次开发能力来扩展 Kubernetes API.
通过 CRD 我们可以向 Kubernetes API 中增加新资源类型, 而不需要修改 Kubernetes 源码或创建自定义的 API server, 该功能大大提高了 Kubernetes 的扩展能力.
```

Prometheus

```
https://github.com/kubernetes/kubernetes/tree/master/cluster/addons/prometheus
```





1) Operator: Operator 是整个系统的主要控制器.
Operator 以 Deployment 方式执行于 Kubernetes 集群上, 并根据自定义资源(Custom Resource Definition, CRDs)来负责管理与部署 Prometheus Server.

同时监控这些自定义资源事件的变化来做相应的处理.

2) Prometheus Server: Operator 根据自定义资源 Prometheus类型中定义的内容而部署的 Prometheus Server 集群, 这些自定义资源可以看作是用来管理 Prometheus Server 集群的 StatefulSets 资源.

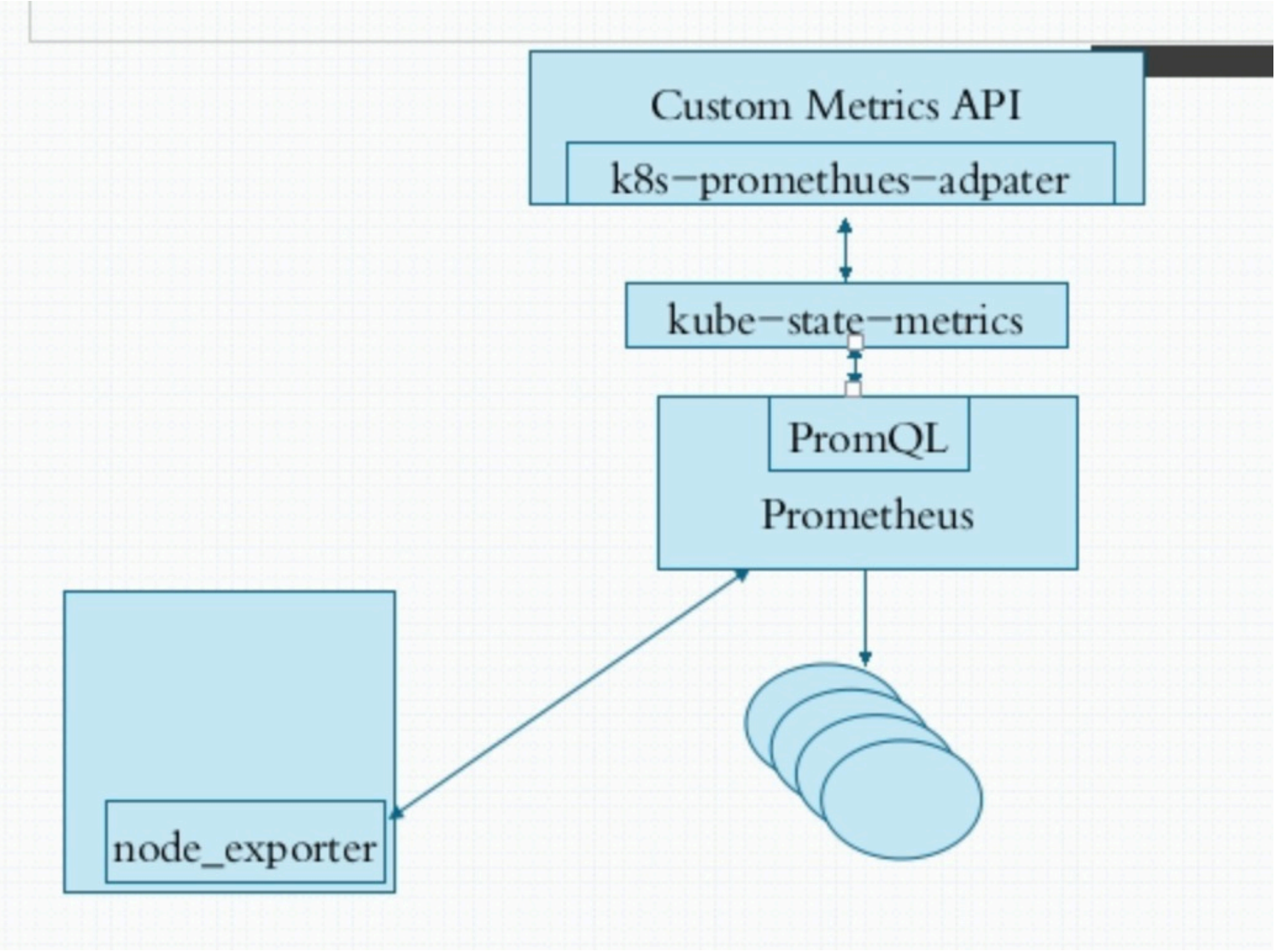
3) Prometheus: Prometheus资源是声明性地描述 Prometheus部署的期望状态.

4) ServiceMonitor: ServiceMonitor 也是一个自定义资源, 它描述了一组被 Prometheus 监控的 targets 列表.
该资源通过 Labels 来选取对应的 Service Endpoint, 让
Prometheus Server 通过选取的 Service 来获取 Metrics 信息.

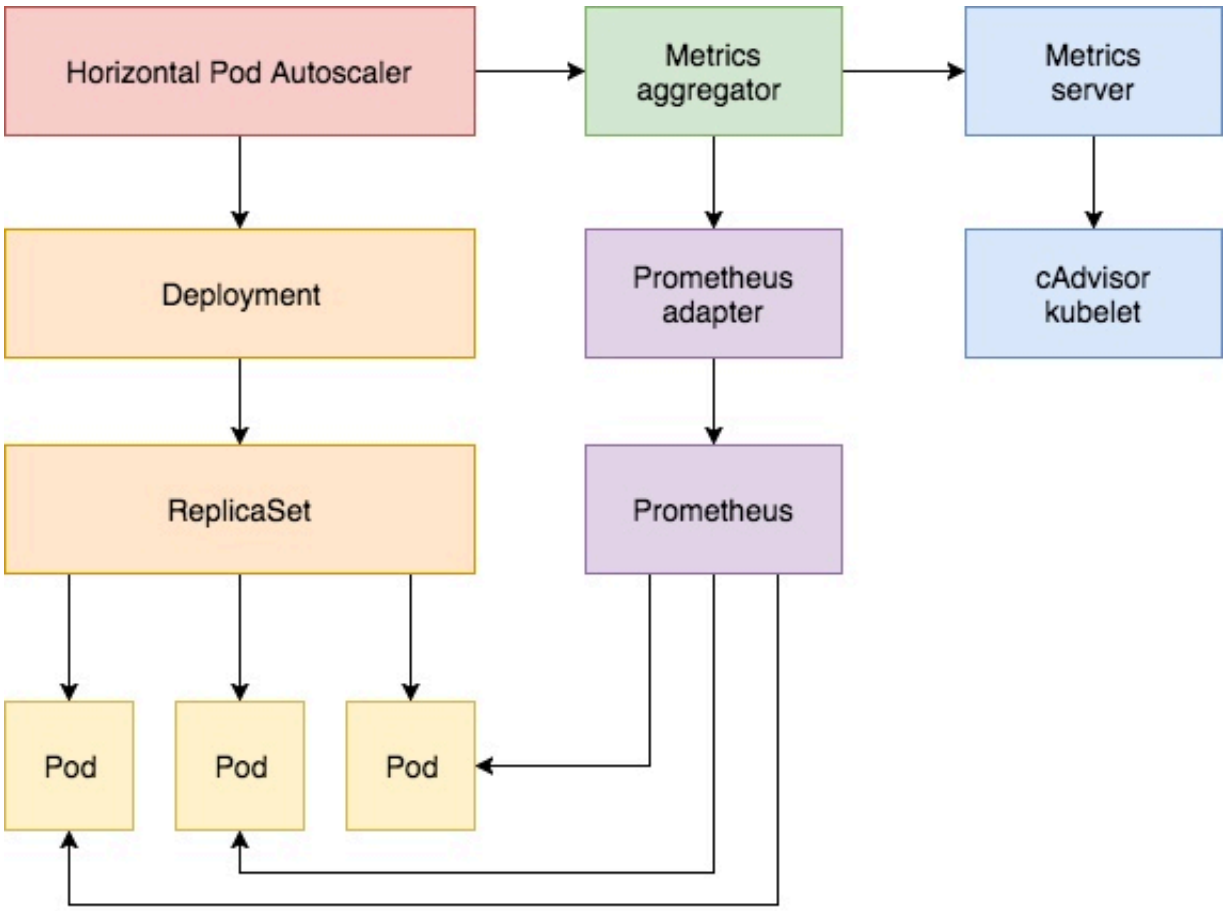
5) Service: Service 资源主要用来对应 Kubernetes 集群中的 Metrics Server Pod, 来提供给 ServiceMonitor 选取让 Prometheus Server 来获取信息.
简单的说就是 Prometheus 监控的对象, 例如之前了解的 Node Exporter Service, Mysql Exporter Service 等等.

6) Alertmanager: Alertmanager 也是一个自定义资源类型, 由 Operator 根据资源描述内容来部署 Alertmanager 集群.

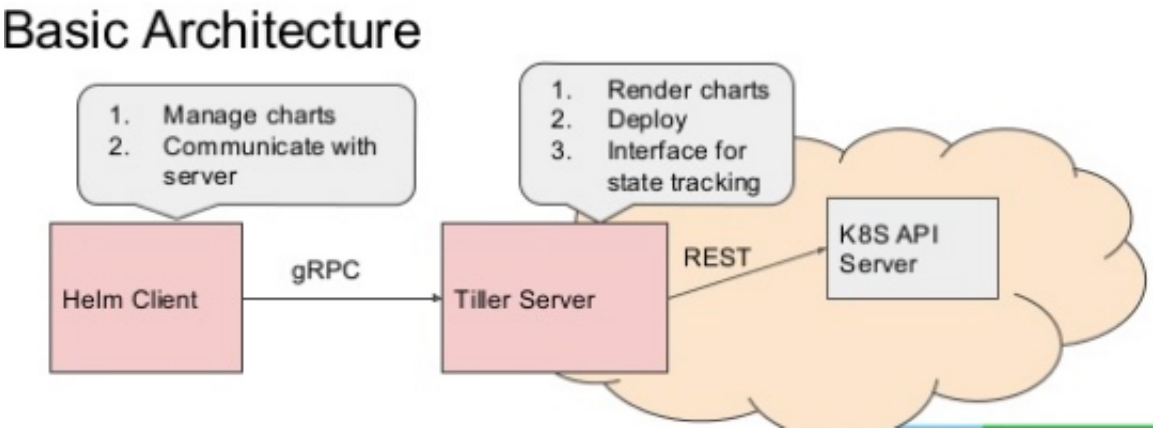
https://github.com/camilb/prometheus-kubernetes


Custom Metrics API
k8s-promethues-adpater
kube-state-metrics
PromQL
Prometheus
node_exporter

HPA



helm




```
Charts: A bundle of kubernetes resource data, it operates the System Package

Repository: A collection of release charts, something like NPM, or Ubuntu Repos
itory.

Release: A chart instance that was loaded into Kubernetes. If the same chart is
 installed several times inside the same cluster, each time the cart will have
its own release.
```

```
wget https://storage.googleapis.com/kubernetes-helm/helm-v2.9.1-linux-amd64.tar.gz
tar xf helm-v2.9.1-linux-amd64.tar.gz
cp linux-amd64/helm  /usr/local/bin/
```

```
核心术语:
    Chart:一个helm程序包;
    Repository:Charts仓库，https/http服务器;
    Release:特定的Chart部署于目标集群上的一个实例;
    Chart -> Config -> Release

程序架构:
    helm:客户端，管理本地的Chart仓库，管理Chart，与Tiller服务器交互，发送Chart，实例安装、查询、卸载等
    Tiller:服务端，接收helm发来的Charts与Config，合并生成relase;

RBAC配置文件示例: https://github.com/helm/helm/blob/master/docs/rbac.md

官方可用的Chart列表: https://hub.kubeapps.com/
```

tiller

```
// tiller rbac

kubectl  apply -f tiller-rbac-config.yaml

//  tiller init
helm init --service-account tiller
```

```
[root@qa-k8s-master01 helm]# helm version
Client: &version.Version{SemVer:"v2.9.1", GitCommit:"20adb27c7c5868466912eebdf6664e7390ebe710", GitTreeState:"clean"}
Server: &version.Version{SemVer:"v2.9.1", GitCommit:"20adb27c7c5868466912eebdf6664e7390ebe710", GitTreeState:"clean"}
[root@qa-k8s-master01 helm]# helm repo list
NAME    URL
stable  https://kubernetes-charts.storage.googleapis.com
local   http://127.0.0.1:8879/charts
```

Gitlab Install By Helm

```
// 安装
helm install --name gitlab -f values.yaml stable/gitlab-ce

// 更新
helm upgrade -f values.yaml gitlab stable/gitlab-ce
```

Creating Highly Available Clusters with kubeadm

```
172.28.28.75 qa-k8s-master01.mljr.com master01
172.28.28.71 qa-k8s-master02.mljr.com master02
172.28.28.76 qa-k8s-master03.mljr.com master03
172.28.28.72 qa-k8s-node01.mljr.com node01
172.28.28.73 qa-k8s-node02.mljr.com node02
172.28.28.74 qa-k8s-node03.mljr.com node03
172.28.3.55 qa-k8s-node04.mljr.com node04
```

Init

```
// 在所有的 master 节点上 执行
yum install -y kubelet kubeadm kubectl docker-ce
systemctl enable docker && systemctl start docker
systemctl enable kubelet

// 更多 init 需要看 单 master 集群内容


curl -o /usr/local/bin/cfssl https://pkg.cfssl.org/R1.2/cfssl_linux-amd64
curl -o /usr/local/bin/cfssljson https://pkg.cfssl.org/R1.2/cfssljson_linux-amd
64
chmod +x /usr/local/bin/cfssl*
```

Create load balancer for kube-apiserver

> master01

```
yum install haproxy  -y

cd /etc/haproxy/
mv haproxy.cfg haproxy.cfg.orgin

cat >/etc/haproxy/haproxy.cfg<<EOF
global
  log 127.0.0.1 local0
  log 127.0.0.1 local1 notice
  tune.ssl.default-dh-param 2048

defaults
  log global
  mode http
  option dontlognull
  timeout connect 5000ms
  timeout client  600000ms
  timeout server  600000ms

listen stats
    bind :9090
    mode http
    balance
    stats uri /haproxy_stats
    stats auth admin:admin123
    stats admin if TRUE

frontend kube-apiserver-https
   mode tcp
   bind :8443
   default_backend kube-apiserver-backend

backend kube-apiserver-backend
    mode tcp
    balance roundrobin
    stick-table type ip size 200k expire 30m
    stick on src
    server apiserver1 172.28.28.75:6443 check
    server apiserver2 172.28.28.71:6443 check
    server apiserver3 172.28.28.76:6443 check
EOF

systemctl start haproxy
systemctl enable haproxy
```

```
yum install keepalived  -y
/etc/keepalived

mv keepalived.conf keepalived.conf.origin

cat > keepalived.conf <<EOF
! Configuration File for keepalived
global_defs {
  router_id LVS_DEVEL
}

vrrp_instance VI_1 {
    state MASTER
    interface ens160
    virtual_router_id 51
    priority 90
    authentication {
        auth_type PASS
        auth_pass 35b4c9t019x4d1600c483e10ad1d
    }
    virtual_ipaddress {
        172.28.28.200
    }
}
EOF

 systemctl start keepalived
 systemctl enable keepalived
```

> master02 / master03

```
// haproxy 配置文件同 master01

// keepalived  配置文件
state => BCKUP
priority => < 90
```

Etcd cluster

先在 master01 生成 etcd 证书

```
mkdir -p /etc/kubernetes/pki/etcd
cd /etc/kubernetes/pki/etcd

cat >ca-config.json <<EOL
{
    "signing": {
        "default": {
            "expiry": "87600h"
        },
        "profiles": {
            "server": {
                "expiry": "87600h",
                "usages": [
                    "signing",
                    "key encipherment",
                    "server auth",
                    "client auth"
                ]
            },
            "client": {
                "expiry": "87600h",
                "usages": [
                    "signing",
                    "key encipherment",
                    "client auth"
                ]
            },
            "peer": {
                "expiry": "87600h",
                "usages": [
                    "signing",
                    "key encipherment",
                    "server auth",
                    "client auth"
                ]
            }
        }
    }
}
EOL


cat >ca-csr.json <<EOL
{
    "CN": "etcd",
```

```
    "key": {
        "algo": "rsa",
        "size": 2048
    }
}
EOL

// 生成 CA 证书
cfssl gencert -initca ca-csr.json | cfssljson -bare ca -

cat >client.json <<EOL
{
    "CN": "client",
    "key": {
        "algo": "ecdsa",
        "size": 256
    }
}
EOL

cfssl gencert -ca=ca.pem -ca-key=ca-key.pem -config=ca-config.json -profile=cli
ent client.json | cfssljson -bare client


scp -r  -P 37100 /etc/kubernetes/pki/etcd master02:/etc/kubernetes/pki/
scp -r  -P 37100 /etc/kubernetes/pki/etcd master03:/etc/kubernetes/pki/
```

在所有的 master 都执行

```
mkdir -p /etc/kubernetes/pki/etcd
cd /etc/kubernetes/pki/etcd

export PEER_NAME=$(hostname)
export PRIVATE_IP=$(ip addr show ens160 | grep -Po 'inet \K[\d.]+')

cfssl print-defaults csr > config.json
sed -i '0,/CN/{s/example\.net/'"$PEER_NAME"'/}' config.json
sed -i 's/www\.example\.net/'"$PRIVATE_IP"'/' config.json
sed -i 's/example\.net/'"$PUBLIC_IP"'/' config.json

cfssl gencert -ca=ca.pem -ca-key=ca-key.pem -config=ca-config.json -profile=ser
ver config.json | cfssljson -bare server
cfssl gencert -ca=ca.pem -ca-key=ca-key.pem -config=ca-config.json -profile=pee
```

```
r config.json | cfssljson -bare peer

cd /usr/local/src
export ETCD_VERSION=v3.3.9
curl -sSL https://github.com/coreos/etcd/releases/download/${ETCD_VERSION}/etcd
-${ETCD_VERSION}-linux-amd64.tar.gz | tar -xzv --strip-components=1 -C /usr/loc
al/bin/
rm -rf etcd-$ETCD_VERSION-linux-amd64*

touch /etc/etcd.env
echo "PEER_NAME=$PEER_NAME" >> /etc/etcd.env
echo "PRIVATE_IP=$PRIVATE_IP" >> /etc/etcd.env

cat >/etc/systemd/system/etcd.service <<EOL
[Unit]
Description=etcd
Documentation=https://github.com/coreos/etcd
Conflicts=etcd.service
Conflicts=etcd2.service

[Service]
EnvironmentFile=/etc/etcd.env
Type=notify
Restart=always
RestartSec=5s
LimitNOFILE=40000
TimeoutStartSec=0

ExecStart=/usr/local/bin/etcd --name ${PEER_NAME} \
    --data-dir /var/lib/etcd \
    --listen-client-urls https://${PRIVATE_IP}:2379 \
    --advertise-client-urls https://${PRIVATE_IP}:2379 \
    --listen-peer-urls https://${PRIVATE_IP}:2380 \
    --initial-advertise-peer-urls https://${PRIVATE_IP}:2380 \
    --cert-file=/etc/kubernetes/pki/etcd/server.pem \
    --key-file=/etc/kubernetes/pki/etcd/server-key.pem \
    --client-cert-auth \
    --trusted-ca-file=/etc/kubernetes/pki/etcd/ca.pem \
    --peer-cert-file=/etc/kubernetes/pki/etcd/peer.pem \
    --peer-key-file=/etc/kubernetes/pki/etcd/peer-key.pem \
    --peer-client-cert-auth \
    --peer-trusted-ca-file=/etc/kubernetes/pki/etcd/ca.pem \
     --initial-cluster qa-k8s-master01.mljr.com=https://172.28.28.75:2380,qa-k8
s-master02.mljr.com=https://172.28.28.71:2380,qa-k8s-master03.mljr.com=https://
172.28.28.76:2380 \
```

```
    --initial-cluster-token mljr-etcd-token \
    --initial-cluster-state new

[Install]
WantedBy=multi-user.target
EOL

systemctl daemon-reload
systemctl start etcd
systemctl enable etcd
```

检查集群状态

```
[root@qa-k8s-master01 etcd]# etcdctl --cert-file /etc/kubernetes/pki/etcd/peer.pem --key-file /etc/kubernetes/pki/etcd/peer-key.pem --ca-file /etc/kubernetes/pki/etcd/ca.pem --endpoints https://172.28.28.71:2379,https://172.28.28.76:2379 cluster-health
member 555d61d38df58f48 is healthy: got healthy result from https://172.28.28.75:2379
member c494646ef59eff60 is healthy: got healthy result from https://172.28.28.71:2379
member eb5d91cdcad637f0 is healthy: got healthy result from https://172.28.28.76:2379
cluster is healthy


[root@qa-k8s-master01 etcd]# etcdctl --cert-file /etc/kubernetes/pki/etcd/peer.pem --key-file /etc/kubernetes/pki/etcd/peer-key.pem --ca-file /etc/kubernetes/pki/etcd/ca.pem --endpoints https://172.28.28.71:2379,https://172.28.28.76:2379 member list
555d61d38df58f48: name=qa-k8s-master01.mljr.com peerURLs=https://172.28.28.75:2380 clientURLs=https://172.28.28.75:2379 isLeader=false
c494646ef59eff60: name=qa-k8s-master02.mljr.com peerURLs=https://172.28.28.71:2380 clientURLs=https://172.28.28.71:2379 isLeader=false
eb5d91cdcad637f0: name=qa-k8s-master03.mljr.com peerURLs=https://172.28.28.76:2380 clientURLs=https://172.28.28.76:2379 isLeader=true
```

创建 k8s 集群

> 在所有的 master 节点生成kubeadm-config.yaml文件

```
cat > kubeadm-config.yaml <<EOL
apiVersion: kubeadm.k8s.io/v1alpha2
kind: MasterConfiguration
kubernetesVersion: v1.11.2
apiServerCertSANs:
- "172.28.28.200"
api:
    controlPlaneEndpoint: "172.28.28.200:8443"
etcd:
    external:
        endpoints:
        - https://172.28.28.75:2379
        - https://172.28.28.71:2379
        - https://172.28.28.76:2379
        caFile: /etc/kubernetes/pki/etcd/ca.pem
        certFile: /etc/kubernetes/pki/etcd/client.pem
        keyFile: /etc/kubernetes/pki/etcd/client-key.pem
networking:
    # This CIDR is a calico default. Substitute or remove for your CNI provider
.
    podSubnet: "10.244.0.0/16"
EOL
```

> 在 master01 上执行

```
kubeadm init --config kubeadm-config.yaml

// 执行成功后 保留添加 node 命令
kubeadm join 172.28.28.200:8443 --token 99fxwc.wbjgtzf10z2sn1d2 --discovery-tok
en-ca-cert-hash sha256:31fcbe3cc4da9bb872c2d0fab6ac4747c0634246796060dfa0d8265b
f9bfbc76

// 配置本地命令行管理
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config

// cp ca信息到 master02 master03
scp -P 37100 /etc/kubernetes/pki/* master02:/etc/kubernetes/pki/
scp -P 37100 /etc/kubernetes/pki/* master03:/etc/kubernetes/pki/
```

> 在 master02 和 master03 上执行 init

```
kubeadm init --config kubeadm-config.yaml
```

Install CNI

Flannel And Enable Directrouting

```
见 CNI
```

Installing Calico for policy and flannel for networking

```
见 CNI
```

Add Node

```
// 同单节点, 略
```

Dashboard

部署Web UI

```
wget https://raw.githubusercontent.com/kubernetes/dashboard/master/src/deploy/recommended/kubernetes-dashboard.yaml

下载并编辑kubernetes-dashboard.yaml, 并在此文件中的Service部分下添加type: NodePort和nodePort: 30443

修改部分如下.
```

```
kind: Service
apiVersion: v1
metadata:
  labels:
    k8s-app: kubernetes-dashboard
  name: kubernetes-dashboard
```

```
  namespace: kube-system
spec:
  ports:
    - port: 443
      targetPort: 8443
      nodePort: 30443
  type: "NodePort"
  selector:
    k8s-app: kubernetes-dashboard
```

```
kubectl apply -f kubernetes-dashboard.yaml

// 访问
使用集群的任何一个 ip 都能访问
https://172.28.28.71:30443
[root@qa-k8s-master01 dashboard]# kubectl get svc -n kube-system
NAME                   TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)
  AGE
kube-dns               ClusterIP   10.96.0.10      <none>        53/UDP,53/TCP
  4h
kubernetes-dashboard   NodePort    10.103.31.215   <none>        443:30443/TCP
  3m
```

创建一个管理员用户

```
cat >admin-user.yaml <<EOF
#----Create Service Account----#

apiVersion: v1
kind: ServiceAccount
metadata:
  name: admin-user
  namespace: kube-system


---
#-----Create ClusterRoleBinding----#

apiVersion: rbac.authorization.k8s.io/v1beta1
kind: ClusterRoleBinding
metadata:
  name: admin-user
```

```
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
- kind: ServiceAccount
  name: admin-user
  namespace: kube-system
EOF
```

> token

```
[root@qa-k8s-master01 dashboard]# kubectl apply -f amind-user.yaml
serviceaccount/admin-user created
clusterrolebinding.rbac.authorization.k8s.io/admin-user created

// 查看 token 相关信息
[root@qa-k8s-master01 dashboard]# kubectl -n kube-system describe secret $(kube
ctl -n kube-system get secret | grep admin-user | awk '{print $1}')
Name:         admin-user-token-fxls7
Namespace:    kube-system
Labels:       <none>
Annotations:  kubernetes.io/service-account.name=admin-user
              kubernetes.io/service-account.uid=93350806-ab7c-11e8-9192-0050569
d46cc

Type:  kubernetes.io/service-account-token

Data
====
ca.crt:     1025 bytes
namespace:  11 bytes
token:      eyJhbGciOiJSUzI1NiIsImtpZCI6IiJ9.eyJpc3MiOiJrdWJlcm5ldGVzL3NlcnZpY2
VhY2NvdW50Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9uYW1lc3BhY2UiOiJrdWJlLXN5c
3RlbSIsImt1YmVybmV0ZXMuaW8vc2VydmljZWFjY291bnQvc2VjcmV0Lm5hbWUiOiJhZG1pbi11c2Vy
LXRva2VuLWZ4bHM3Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9zZXJ2aWNlLWFjY291bnQ
ubmFtZSI6ImFkbWluLXVzZXIiLCJrdWJlcm5ldGVzLmlvL3NlcnZpY2VhY2NvdW50L3NlcnZpY2UtYW
Njb3VudC51aWQiOiI5MzM1MDgwNi1hYjdjLTExZTgtOTE5Mi0wMDUwNTY5ZDQ2Y2MiLCJzdWIiOiJze
XN0ZW06c2VydmljZWFjY291bnQ6a3ViZS1zeXN0ZW06YWRtaW4tdXNlciJ9.Q4T2iGAtTXg2iRNQyTD
wMqQDkjFKDR8R3TSIIcy_SKoEau-j1yFL2huE89w2y5UDVV8DBoUmCXrHEc8acps0krFAfSqeSK1kkX
WCxznqoSVdeoXNKV45TnR2jholVOvWFU91isbNTlFsH5AsPbgCjTZxKanmCSgwZza3tnHIorcPqPlKW
4iWGP07QyzpLh2vn7BAYvol7oOdJT4sDbKiF3AIxc5S2pKZ78s8nnF5ib1ZCfg5UPZ76PUERTet8S-E
0lfkQPOylBrOWnRa19T0zP2VgfLeHdCyH4TWLgh-0RGxrF8rBufD00cVNQ5tePOC_hl0Ok1g-urNuw7
zR0r4lQ
```

```
// 命令行获取 token
[root@qa-k8s-master01 dashboard]# kubectl -n kube-system get secret $(kubectl -
n kube-system get secret | grep admin-user | awk '{print $1}')    -o jsonpath={
.data.token} | awk '{print $0}' | base64 -d | awk '{print $0}'
```

kubeconfig Login

> 添加一个用户对 dev namespace 有管理权限

```
[root@qa-k8s-master01 dashboard]# cat >  dev-admin.yaml <<EOF
#----Create dev NS-----#
apiVersion: v1
kind: Namespace
metadata:
  creationTimestamp: null
  name: dev
spec: {}
status: {}

---
#----Create Rule----#

kind: Role
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  namespace: dev
  name: dev-admin
rules:
- apiGroups: ["*"]
  resources: ["*"]
  verbs: ["*"]

---
#----Create Service Account----#

apiVersion: v1
kind: ServiceAccount
metadata:
  name: dev-admin
  namespace: dev
```

```
---
#-----Create RoleBinding----#

apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: dev-admin
  namespace: dev
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: dev-admin
subjects:
- kind: ServiceAccount
  name: dev-admin
  namespace: dev
EOF
[root@qa-k8s-master01 dashboard]# kubectl apply -f default-admin.yaml
```

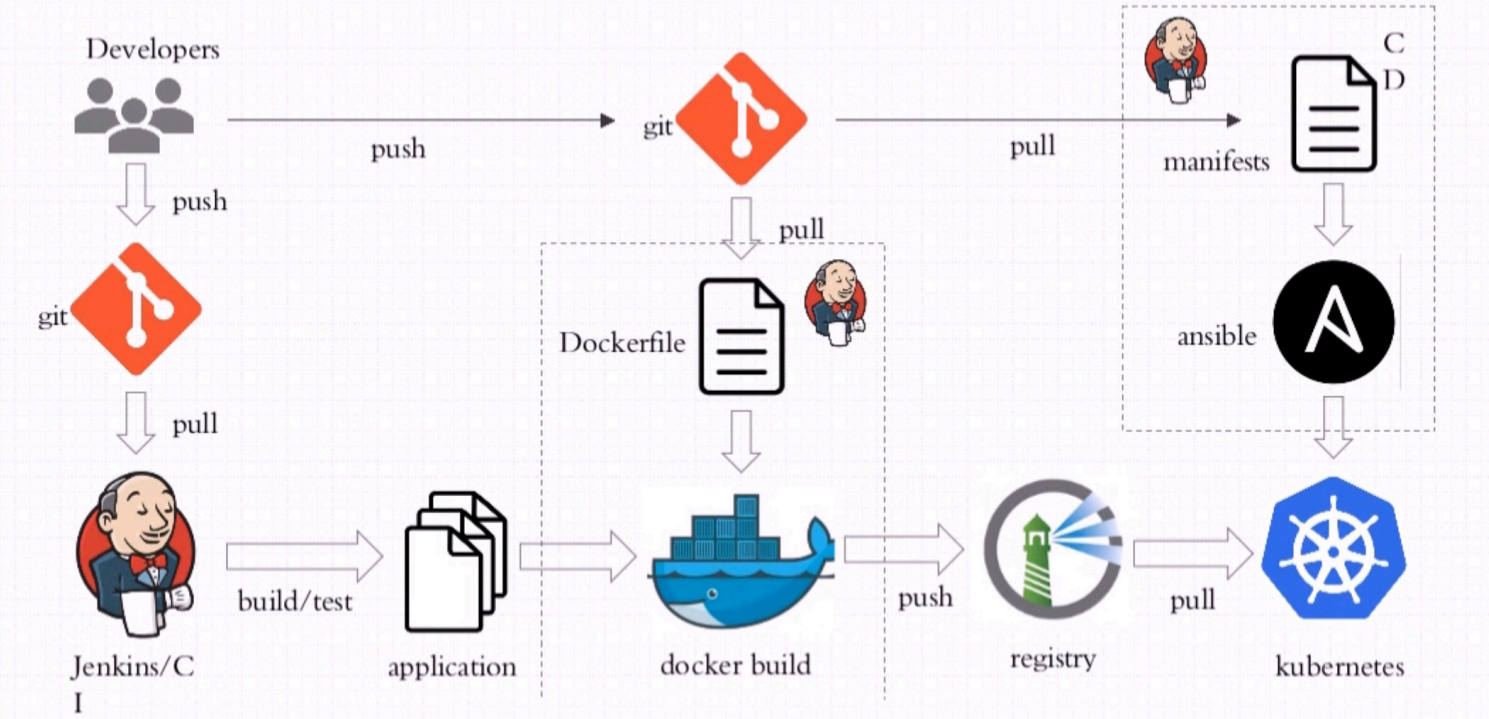
```
dev_token=`kubectl -n dev get secret $(kubectl -n dev get secret | grep dev-adm
in | awk '{print $1}')    -o jsonpath={.data.token} | awk '{print $0}' | base64
 -d | awk '{print $0}'`

kubectl config  set-credentials dev-admin --token=$dev_token  --kubeconfig=/roo
t/dev-ns-admin.conf

kubectl config  set-context dev-admin@kubernetes --cluster=kubernetes --user=de
v-admin --kubeconfig=/root/dev-ns-admin.conf

kubectl config use-context dev-admin@kubernetes --kubeconfig=/root/dev-ns-admin
.conf

[root@qa-k8s-master01 dashboard]# kubectl config view --kubeconfig=/root/dev-ns
-admin.conf
```

Error

```
r: code = Unknown desc = failed to set up sandbox container "ab14be8f0b48977fae
b57576a757997e9dc1efaedf394d2ec1162713118d49a0" network for pod "coredns-78fcdf
6894-6872b": NetworkPlugin cni failed to set up pod "coredns-78fcdf6894-6872b_k
ube-system" network: failed to set bridge addr: "cni0" already has an IP addres
```

```
s different from 10.244.0.1/24


==> 此时coredns 容器是运行不起来的 出现这种情况是因为之前有些机器已经做 node 了, 纯净服务器不会出现这样问题.
systemctl stop kubelet
ifconfig  cni0 down
brctl delbr cni0
ip link delete flannel.1
systemctl start kubelet
reboot
```

K8s With Paas



Ingress

```
Ingress 使用开源的反向代理负载均衡器来实现对外暴漏服务.
```

启用 Ingress

gitlab 实例

```
// 准备
在之前集群的基础上，新添加一个节点 node5(生成环境建议3个以上)，并设置污点，专用于暴露服务给外部.

使用 Traefik(80) 和 Ingress_nginx(8080) 两种方式提供服务.

基于https, Traefik(443) 和 Ingress_nginx(8443)

hostNetwork
```

新加 node 并设置污点

gitlab安装

```
// 需要创建4个 pv，具体大小看要求.

# cat pv-demo.yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv002
  labels:
     name: pv002
spec:
  nfs:
    path: /pv/volumes/v2
    server: master03
  accessModes: ["ReadWriteMany","ReadWriteOnce"]
  capacity:
    storage: 20Gi
```

```
[root@qa-k8s-master01 gitlab]# cat values.yaml
## GitLab CE image
## ref: https://hub.docker.com/r/gitlab/gitlab-ce/tags/
##
image: gitlab/gitlab-ce:9.4.1-ce.0
```

```
## Specify a imagePullPolicy
## 'Always' if imageTag is 'latest', else set to 'IfNotPresent'
## ref: http://kubernetes.io/docs/user-guide/images/#pre-pulling-images
##
# imagePullPolicy:
imagePullPolicy: IfNotPresent

## The URL (with protocol) that your users will use to reach the install.
## ref: https://docs.gitlab.com/omnibus/settings/configuration.html#configuring
-the-external-url-for-gitlab
##
#externalUrl: http://your-domain.com/
externalUrl: http://my.gitlab.com/

## Change the initial default admin password if set. If not set, you'll be
## able to set it when you first visit your install.
##
#gitlabRootPassword: ""
gitlabRootPassword: "ss1234560.0"

## For minikube, set this to NodePort, elsewhere use LoadBalancer
## ref: http://kubernetes.io/docs/user-guide/services/#publishing-services---se
rvice-types
##
#serviceType: LoadBalancer
serviceType: ClusterIP


## Ingress configuration options
##
ingress:
  annotations:
      # kubernetes.io/ingress.class: nginx
      # kubernetes.io/tls-acme: "true"
  enabled: false
  tls:
      # - secretName: gitlab.cluster.local
      #   hosts:
      #     - gitlab.cluster.local
  url: gitlab.cluster.local

## Configure external service ports
## ref: http://kubernetes.io/docs/user-guide/services/
sshPort: 22
httpPort: 80
```

```
httpsPort: 443
## livenessPort Port of liveness probe endpoint
livenessPort: http
## readinessPort Port of readiness probe endpoint
readinessPort: http

## Configure resource requests and limits
## ref: http://kubernetes.io/docs/user-guide/compute-resources/
##
resources:
  ## GitLab requires a good deal of resources. We have split out Postgres and
  ## redis, which helps some. Refer to the guidelines for larger installs.
  ## ref: https://docs.gitlab.com/ce/install/requirements.html#hardware-require
ments
  requests:
    memory: 1Gi
    cpu: 500m
  limits:
    memory: 2Gi
    cpu: 1

## Enable persistence using Persistent Volume Claims
## ref: http://kubernetes.io/docs/user-guide/persistent-volumes/
## ref: https://docs.gitlab.com/ce/install/requirements.html#storage
##
persistence:
  ## This volume persists generated configuration files, keys, and certs.
  ##
  gitlabEtc:
    enabled: true
    size: 1Gi
    ## If defined, volume.beta.kubernetes.io/storage-class: <storageClass>
    ## Default: volume.alpha.kubernetes.io/storage-class: default
    ##
    # storageClass:
    accessMode: ReadWriteOnce
  ## This volume is used to store git data and other project files.
  ## ref: https://docs.gitlab.com/omnibus/settings/configuration.html#storing-g
it-data-in-an-alternative-directory
  ##
  gitlabData:
    enabled: true
    size: 10Gi
    ## If defined, volume.beta.kubernetes.io/storage-class: <storageClass>
    ## Default: volume.alpha.kubernetes.io/storage-class: default
```

```
    ##
    # storageClass:
    accessMode: ReadWriteOnce

## Configuration values for the postgresql dependency.
## ref: https://github.com/kubernetes/charts/blob/master/stable/postgresql/README.md
##
postgresql:
  # 9.6 is the newest supported version for the GitLab container
  imageTag: "9.6"
  cpu: 1000m
  memory: 1Gi

  postgresUser: gitlab
  postgresPassword: gitlab
  postgresDatabase: gitlab

  persistence:
    size: 10Gi

## Configuration values for the redis dependency.
## ref: https://github.com/kubernetes/charts/blob/master/stable/redis/README.md
##
redis:
  redisPassword: "gitlab"

  resources:
    requests:
      memory: 1Gi

  persistence:
    size: 10Gi
```

```
helm install --name gitlab -f values.yaml stable/gitlab-ce
```

Traefik 设置

```
// Traefik 略
```

> 准备 Traefik Ingress 配置文件

```
[root@qa-k8s-master01 Ingress]# cat ingress-traefie.yaml
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: my-traefik
  annotations:
    kubernetes.io/ingress.class: traefik
    traefik.frontend.rule.type: PathPrefixStrip
spec:
  rules:
  - host: my.gitlab.com
    http:
      paths:
      - path: /
        backend:
          serviceName: gitlab-gitlab-ce
          servicePort: 80
```

> 设置traefik service

```
[root@qa-k8s-master01 gitlab]# cat my-gitlab-com-traefik-service.yaml
apiVersion: v1
kind: Service
metadata:
  name: ingress-traefik-gitlabe
  namespace: default

spec:
  type: NodePort
  ports:
  - name: http
    port: 80
    targetPort: 80
    nodePort: 30081
    protocol: TCP
  - name: https
    port: 443
    targetPort: 443
    protocol: TCP
  selector:
    app: gitlab-gitlab-ce
```

```
// 配置
http://my.gitlab.com:30081/
```

Ingress_nginx 设置

```
// 安装 Ingress Controller 见上
```

> 准备 Nginx Ingress

```
[root@qa-k8s-master01 gitlab]# cat my-gitlab-com-ingress-nginx.yaml
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: my-gitlab-com
  namespace: default
  annotations:
    kubernetes.io/ingress.class: "nginx"
spec:
  rules:
  - host: my.gitlab.com
    http:
      paths:
      - backend:
          serviceName: gitlab-gitlab-ce
          servicePort: 80
```

> 准备 Nginx Ingress Service

```
// cat my-gitlab-com-service.yaml
apiVersion: v1
kind: Service
metadata:
  name: ingress-nginx
  namespace: ingress-nginx
  labels:
    app.kubernetes.io/name: ingress-nginx
    app.kubernetes.io/part-of: ingress-nginx

spec:
```

```
  type: NodePort
  ports:
  - name: http
    port: 80
    targetPort: 80
    nodePort: 30080
    protocol: TCP
  - name: https
    port: 443
    targetPort: 443
    protocol: TCP
  selector:
    app.kubernetes.io/name: ingress-nginx
```

```
访问 http://my.gitlab.com:30080/
```

https

```
(umask 077; openssl genrsa -out gitlabe.key 2048)
openssl req -new  -x509 -key gitlabe.key -out gitlabe.crt -subj /C=CN/ST=Shenzh
eng/O=DevOps/CN=my.gitlab.com
kubectl create secret tls gitlabe-ingress-secret --cert=gitlabe.crt --key=gitla
be.key
```