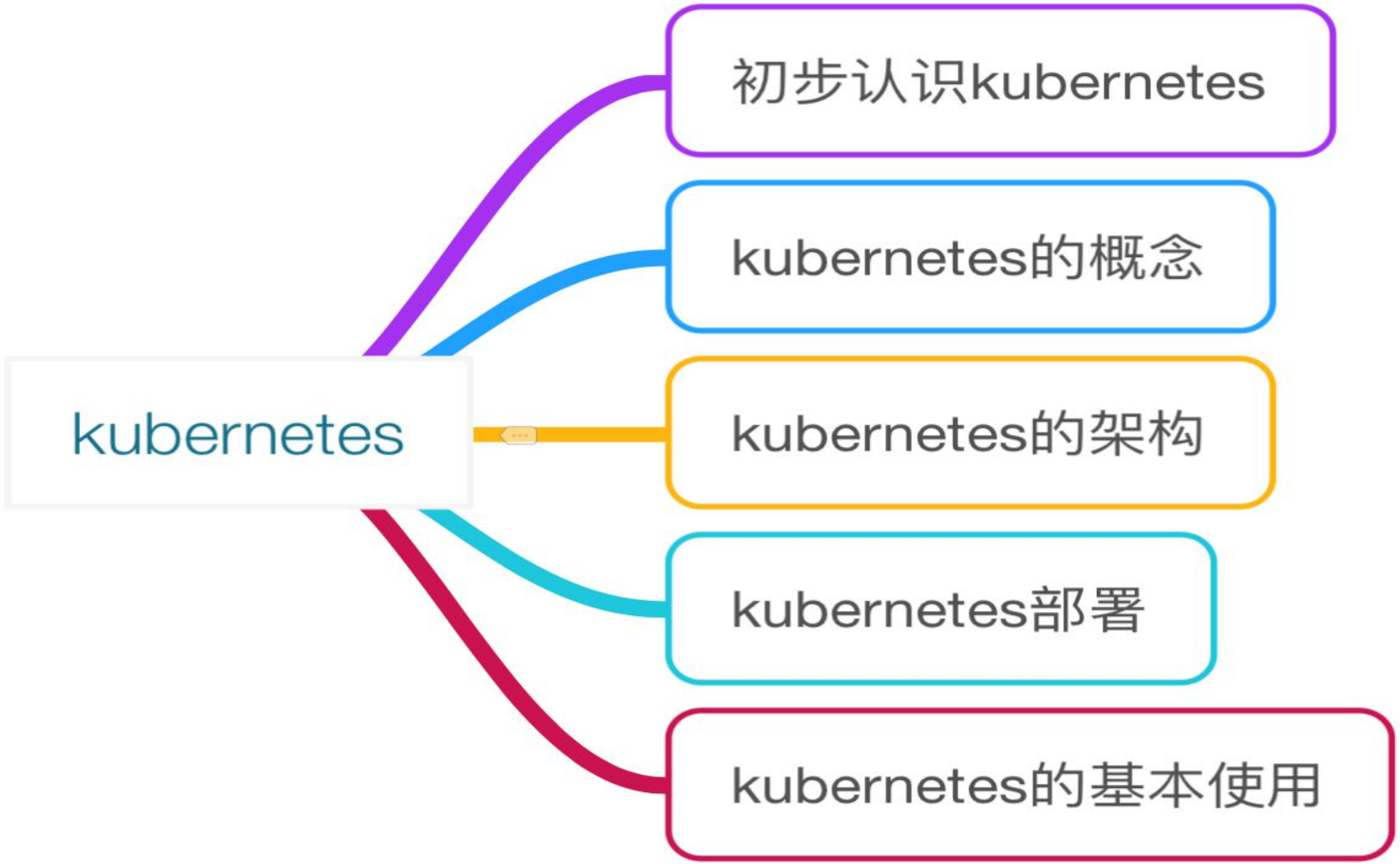
RANCHER 在线培训第3季第2期

Kubernetes部署与使用入门

内容大纲



RANCHER

1、初步认识kubernetes

What is kubernetes ?

RANCHER

1、初步认识kubernetes

Kubernetes is an open-source system for automating deployment, scaling, and management of containerized applications.

特点：

Automating deployment:自动部署

Scaling: 伸缩

Management of containerized applications: 管理容器应用

RANCHER

1、初步认识kubernetes

1、google容器内部容器编排系统borg目前又称omega的开源版，该项目由google 2014年开始启动，kubernetes 1.0版本在2015年7月21日正式发布。

2、2015年7月google联合 linux基金会(Linux Foundation)创办了CNCF基金会(Cloud Native Computing Foundation)并将kubernetes种子项目捐献给了CNCF基金会。

3、2017年kubernetes总共发布了四个版本，主要是在多用户、多负载、安全性和易用性方面做了改进。分别是：

Kubernetes 1.6：伸缩性SLO支持包含5000个节点（15万pod)的集群和动态存储。

Kubernetes 1.7：network policy API加强网络安全性

Kubernetes 1.8：角色访问控制功能变为稳定版

Kubernetes 1.9：支持window系统

4、2018年

Kubernetes.1.10 容器存储接口（CSI)进入beta测试阶段

RANCHER

1、初步认识kubernetes

kubernetes能做什么？

1、容器的自动化部署和升级

2、容器的自动伸缩

3、以集群的方式部署容器，并提供容器间的负载均衡

4、容器的自动修复和健康检查

RANCHER

1、初步认识kubernetes

为什么使用kubernetes？

背景维度

1、大量巨头加入CNCF基金会，人多力量大，出现问题也不怕。

2、2017年下半年kubernetes已经完全赢得了容器编排大战。

技术维度

1、Kubernetes具备超强的横向扩展能力，只要架构设计的好，甚至可以在线的线性扩展。

2、kubernetes采用传统的master-slave架构在container上层又封装了层pod，这样设计非常简单灵活，能更好的适应和管理微服务。

3、kubernetes 调度分发、服务发现、健康检查、平滑升级回滚.....完善。

RANCHER

1、初步认识kubernetes

总结

RANCHER

2、kubernetes的基本概念

Cluster

cluster是计算、存储、网络的资源的集合，kubernetes利用这些资源运行各种基于容器的应用。

POD

pod是kubernetes最小的调度单元，一个pod对应一个或多个container，通常会将紧密相连的一组应用放到一个pod中，同一个pod里的container关系一个网络栈和一块存储卷空间。

为什么使用pod?

1，在kubernetes中pod是container的载体，一个pod里面拥有一个或多个container，做为一个逻辑单元，**方便管理；**

2，同一个pod中的container共享一个网络栈和存储，相互之间可以直接通过localhost进行通信，同时也**共享同一块存储卷；**

3，kubernetes**直接管理对象是pod**，而不是底层的docker，对于docker的操作，被封装在pod中，不会直接操作，这也意味着，pod包含也可以是其他公司的容器产品，比如coreos的rkt或阿里巴巴的pouch。

RANCHER

2、kubernetes的基本概念

Controller

kubernetes使用controller来创建和管理pod，controller中定义了pod的部署特性，

比如部署几个副本，在什么样的node上运行。为满足不同的业务需求，

kubernetes提供了如下controller。

Deployment

是最常用的 Controller，是以前Replication Controller的升级版。Deployment 可以管理 Pod 的多个副本，并确保 Pod 按照期望的状态运行。集成上线部署、滚动升级、弹性伸缩等功能。

Replicaset

实现了 Pod 的多副本管理。使用 Deployment 时会自动创建 ReplicaSet，也就是说 Deployment 是通过 ReplicaSet 来管理 Pod 的多个副本，我们通常不需要直接使用 ReplicaSet。

RANCHER

2、kubernetes的基本概念

Daemonset

用于每个 Node 最多只运行一个 Pod 副本的场景。正如其名称所揭示的，DaemonSet 通常用于运行 daemon。

StatefuleSet

能够保证 Pod 的每个副本在整个生命周期中名称是不变的。而其他 Controller 不提供这个功能，当某个 Pod 发生故障需要删除并重新启动时，Pod 的名称会发生变化。同时 StatefuleSet 会保证副本按照固定的顺序启动、更新或者删除。

Job

用于运行结束就删除的应用。而其他 Controller 中的 Pod 通常是长期持续运行。

接下来主要介绍service

RANCHER

2、kubernetes的基本概念

service

直接接通过pod的ip加端口去访问应用是不靠谱的，因为pod的生命周期是不断变化的，每次重新生成的pod ip地址都是都是不一样的，并且当pod有多个副本时，这样的访问就更痛苦了，所以kubernetes通过service来解决这些问题，简单来说，你可以把service理解为一个负载均衡器，也可以说是service是为一组功能相同的pod提供统一入口。Service默认有自己的ip和端口的叫cluster-ip和port，内部可以直接通过这个endpoint(clusterip+port)去访问应用。

不过有一点需要注意，这个cluster-ip是个Virtual IP，它是ping不通的，底层转发是通过node节点上的kube-proxy调用iptables生成对应的转发规则，
新版本的kubernetes的kube-proxy可以直接使用ipvs，不过目前还是beat阶段。

RANCHER

2、kubernetes的基本概念

service

Service的转发后端的四种方式

1、clusterip
2、nodeport
3、ingress
4、loadblance

Service的三种端口类型

1、port
2、targetport
3、nodeport

不过有一点需要注意，这个cluster-ip是个Virtual IP，它是ping不通的，底层转发是通过node节点上的kube-proxy调用iptables生成对应的转发规则，
新版本的kubernetes的kube-proxy可以直接使用ipvs，不过目前还是beat阶段。

RANCHER

2、kubernetes的基本概念

namespace

这里根linux的namespace不一样，这个更像是一个分组。通过namespace将用户创建的controller和pod等资源分开。namespace可以将一个物理的cluster划成多个虚拟的cluster，每个cluster就是一个namespace，不同的namespace里面资源是完全隔离的。便于不同分组共享使用集群物理资源的同时，还能被管理。类似openstack的project，还可以对不同namespace设置quota。

kubernetes默认创建了三个namespace，default、kube-public、kube-system。

```
[root@master ~]# kubectl get namespace
NAME          STATUS    AGE
default       Active    2d
kube-public   Active    2d
kube-system   Active    2d
[root@master ~]#
```

default：默认创建的资源将放到这个namespace里面。

kube-system：kubernetes自己创建的系统资源将放这里。

kube-public：kubernetes自己创建的命名空间，用于确保整个集群中公开的看到某些资源。

创建pod时指定namespace

kubectl --namespace=<insert-namespace-name-here> run nginx --image=nginx

RANCHER

2、kubernetes的基本概念

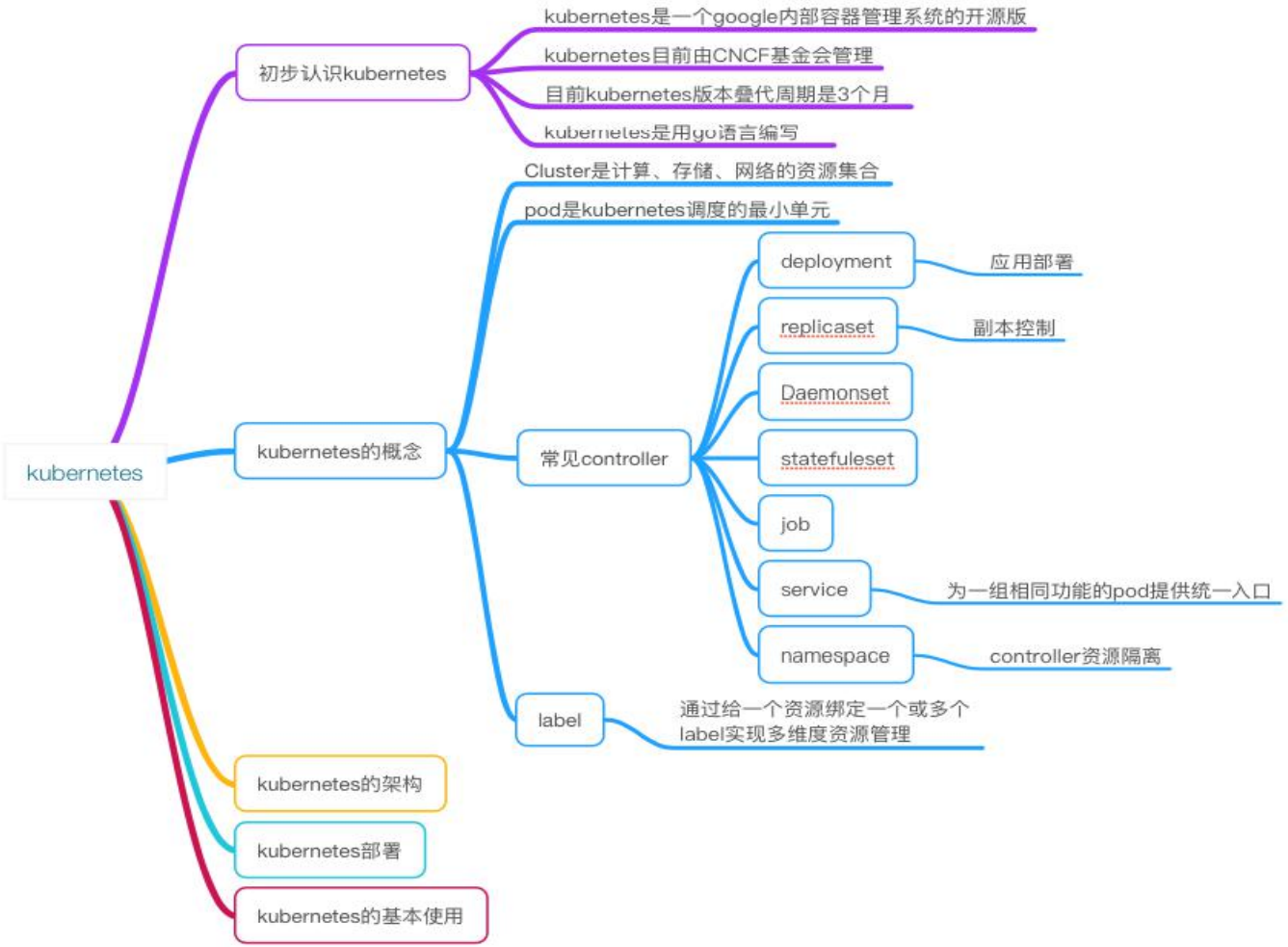
Label

Label以key/value键值对的形式附加到各种对象上，如Pod、Node等。Service通过selector label建立service和pod的关联，简单来说，资源和资源之间的关联都是通过label，通过给一个资源绑定一个或多个不同的label，实现多维度的资源管理，selector label类似于sql语句的where。

RANCHER

2、kubernetes的基本概念

总结



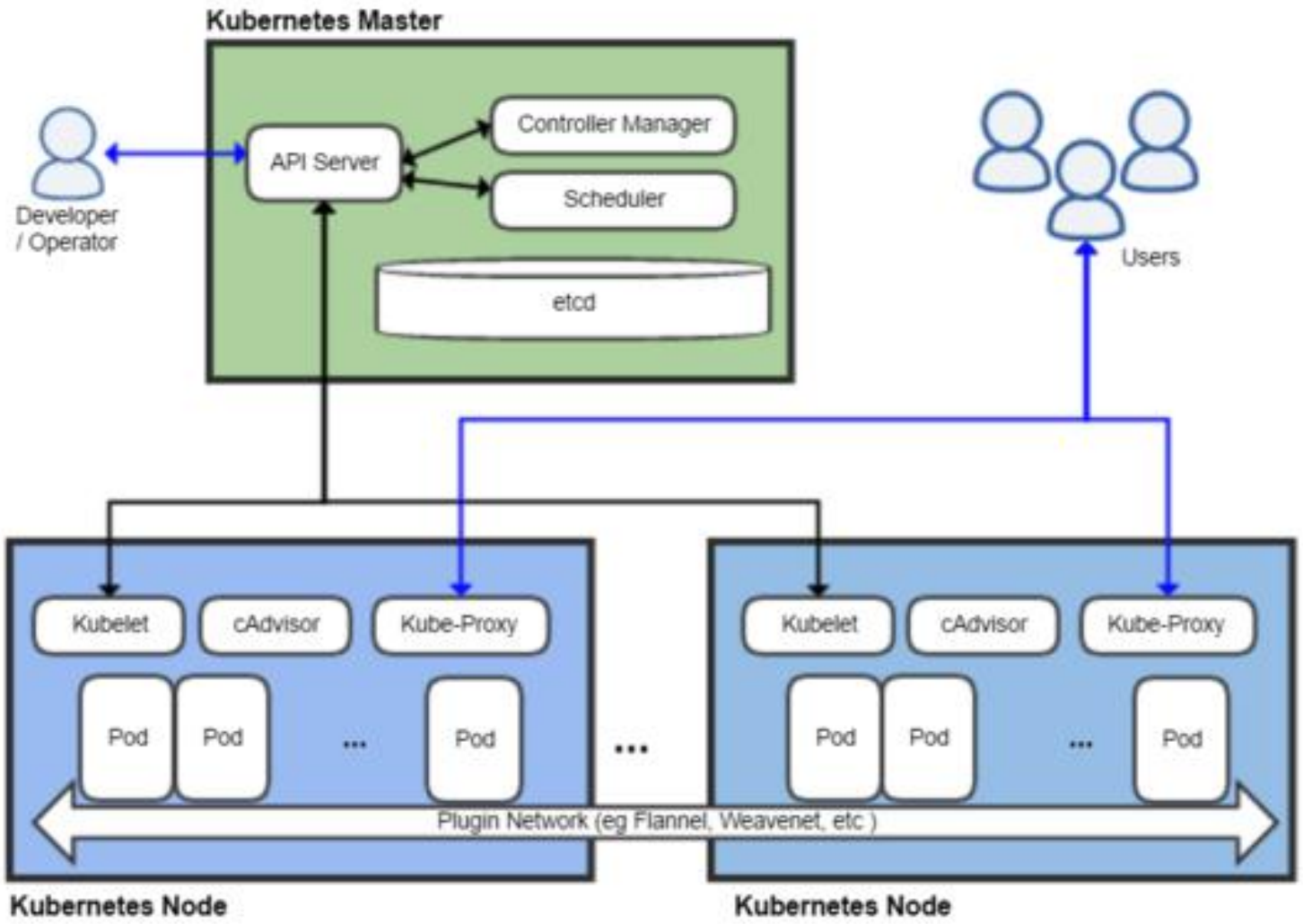
RANCHER

3、kubernetes的架构

角色上：
典型的master-slave架构

master节点上：
node节点或叫Minion节点

功能上：
master节点做为集群大脑
node节点用于承载部署的容器应用

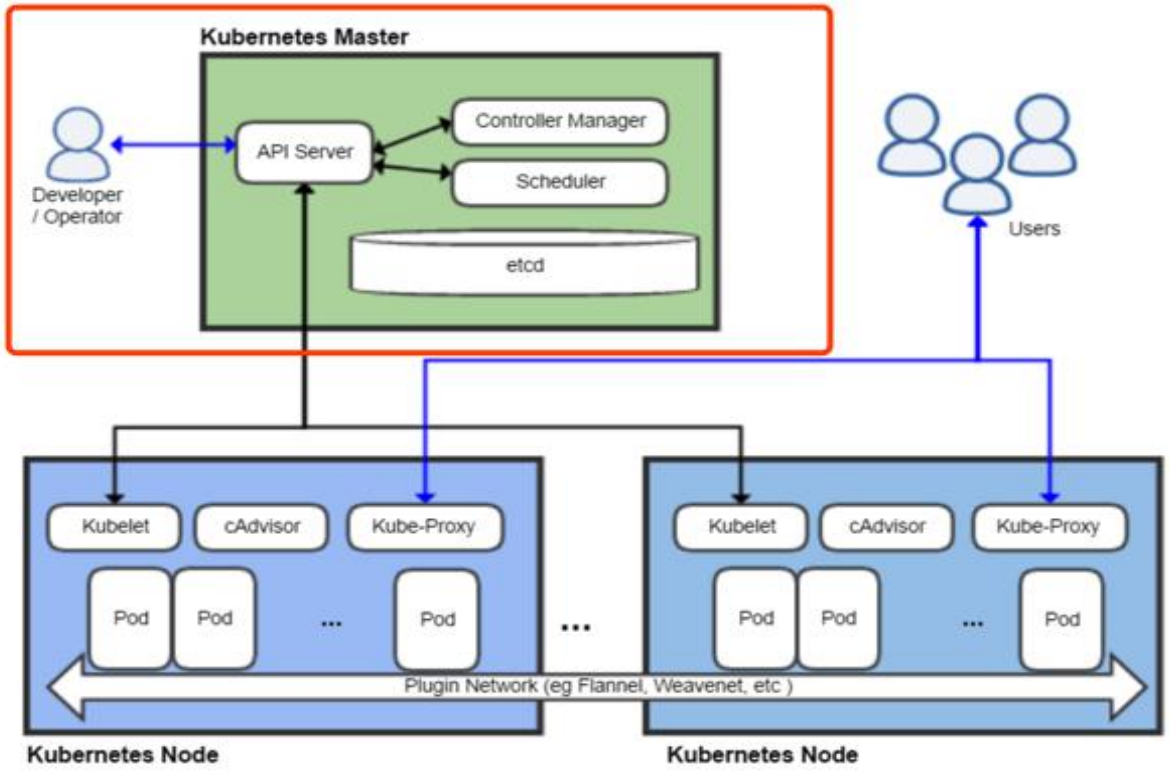


RANCHER

3、kubernetes的架构

master节点：

master是cluster的大脑，类似openstack的控制节点，
运行着kube-apiserver、kube-controller-manager、etcd
、kube-scheduler它的主要职责是对资源管理、调度，还有认证、弹性伸缩、安全认证。



RANCHER

3、kubernetes的架构

master节点组件介绍

etcd

kubernetes的后端数据存储系统，cluster中的所有资源对象数据都存储在这，用于配置共享和服务发现。

kube-apiserver

1、整个集群管理的api接口，所有对集群的操作和查询都需要经过api-server；

2、集群内各个模块通信的枢钮，集群内其他模块，互相之间并不能直接通信，都是通过api-server打交道，来完成自己那部分工作；

3、集群安全控制，apiserver提供了集群的安全验证，和角色权限分配；

4、连接etcd，集群内所有组件都不能直接连接etcd只能通过api-server去连接etcd。

RANCHER

3、kubernetes的架构

master节点组件介绍

kube-controller-manager

1、负责集群的故障检测和修复；

2、根据deployment的定义，维持正确的pod副本数；

3、根据Service与Pod的管理关系，完成服务的Endpoints对象的创建和更新；

4、为新的namespace创建帐户和api访问token。

kube-scheduler

创建pod时，选择合适的node进行调度。

flannel

flannel的网络组件；

kube-dashboard；

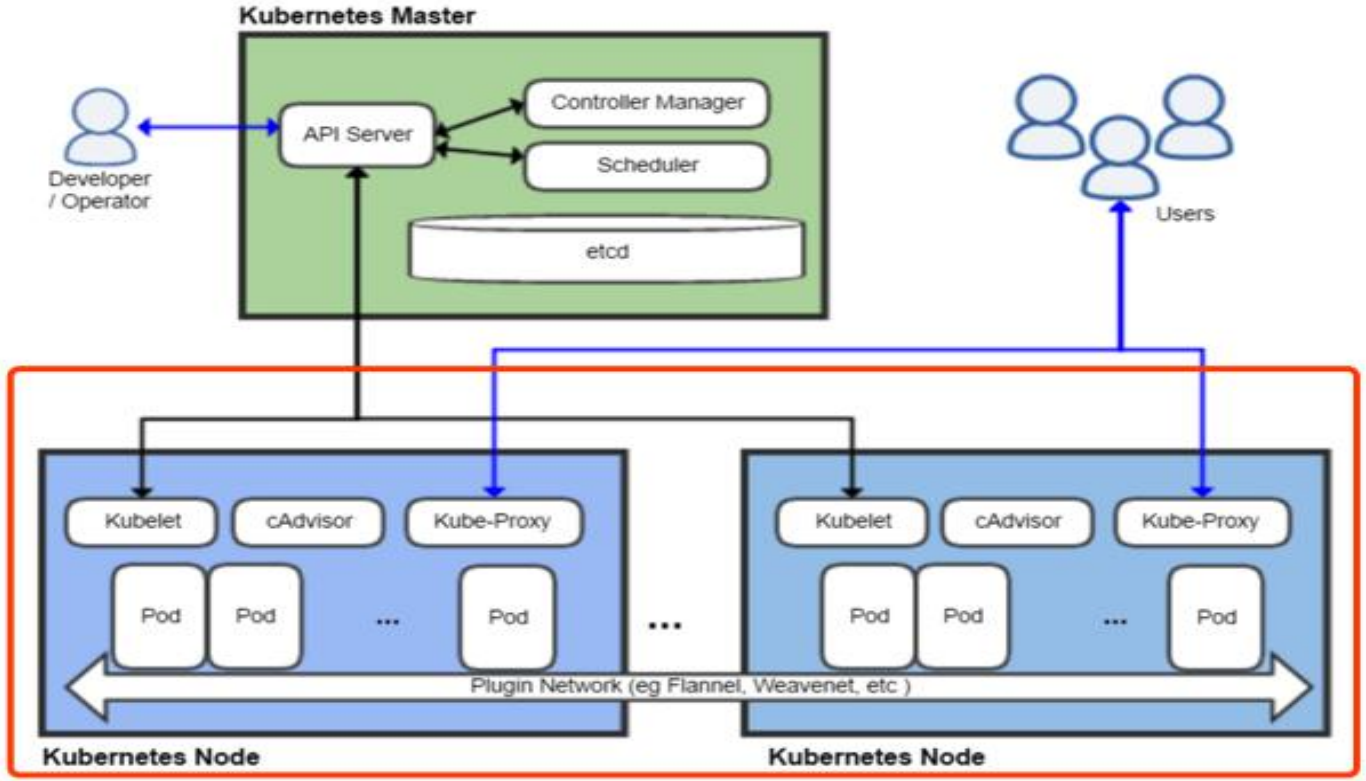
kubernetes的web控制端。

RANCHER

3、 kubernetes的架构

node节点

运行容器应用，由mater管理，接收master节点的各类请求，进行容器的创建和管理，并将运行在上面的容器应用上报到master节点，类似openstack的计算节点。



RANCHER

3、kubernetes的架构

node节点

Kubelet

负责node节点容器的创建、删除监控等生命周期管理，

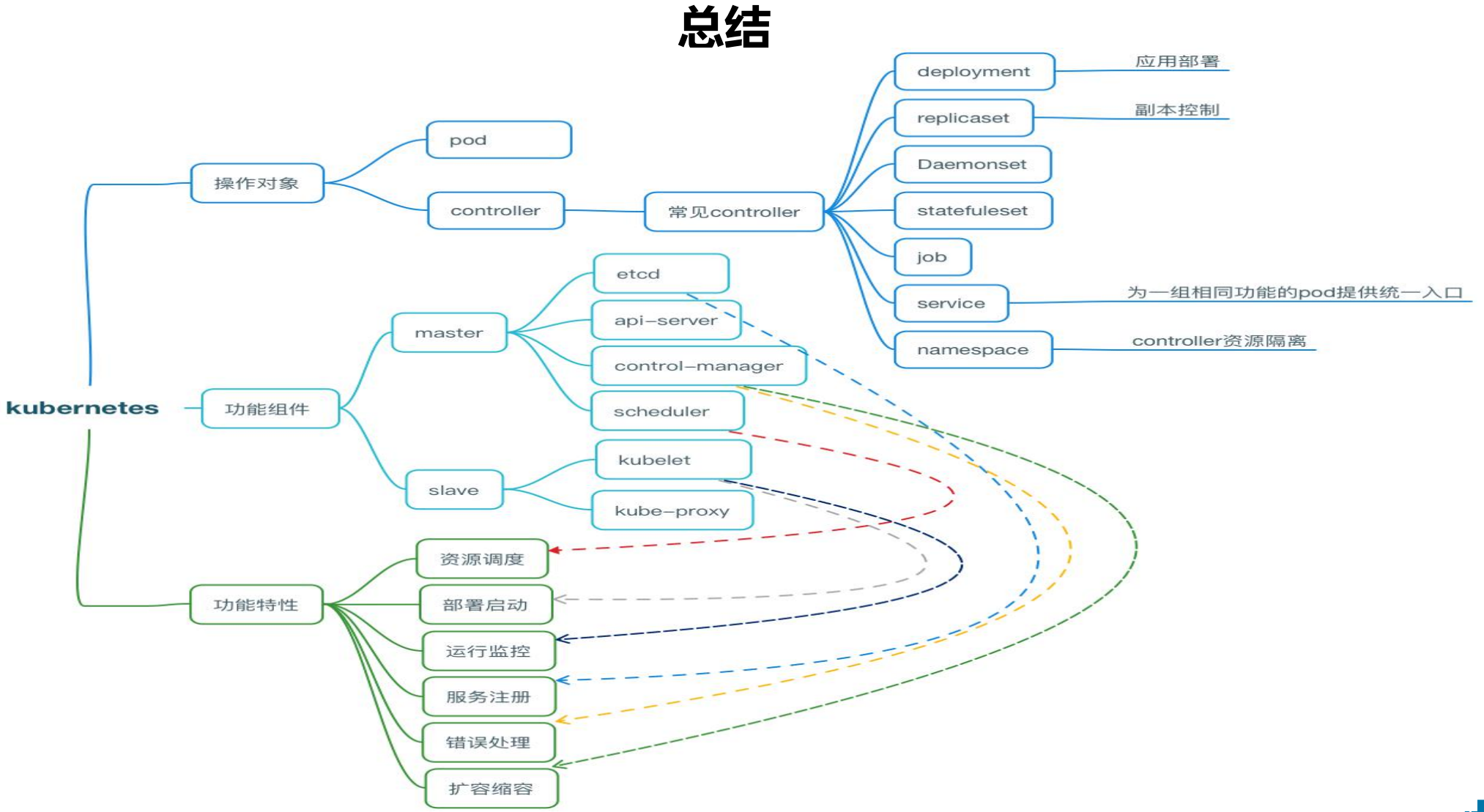
上报node信息如cpu、内存、pod的ip等信息导master节点的api-server。

kube-proxy

用于实现端口映射和负载均衡。Kube-proxy主要有两种模式iptables和ipvs。

RANCHER

3、 kubernetes的架构



4、kubernetes的部署

1、kubeadm

kubeadm为kubernetes官方推荐的自动化部署工具，他将kubernetes的
组件以pod的形式部署在master和node节点上，并自动完成证书认证等操作。
因为kubeadm默认要从google的镜像仓库下载镜像，但目前国内无法访问google镜像仓库，
所以这里需要提前将镜像下好了，只需要将离线包的镜像导入到节点中就可以了。
参考: http://www.bladewan.com/2018/01/02/kubernetes_install/

2、源码安装

https://github.com/kubernetes/kubernetes/tree/release-1.9

github上下载对应版本的源码包，编译，自己生成证书，编写进程启动文件

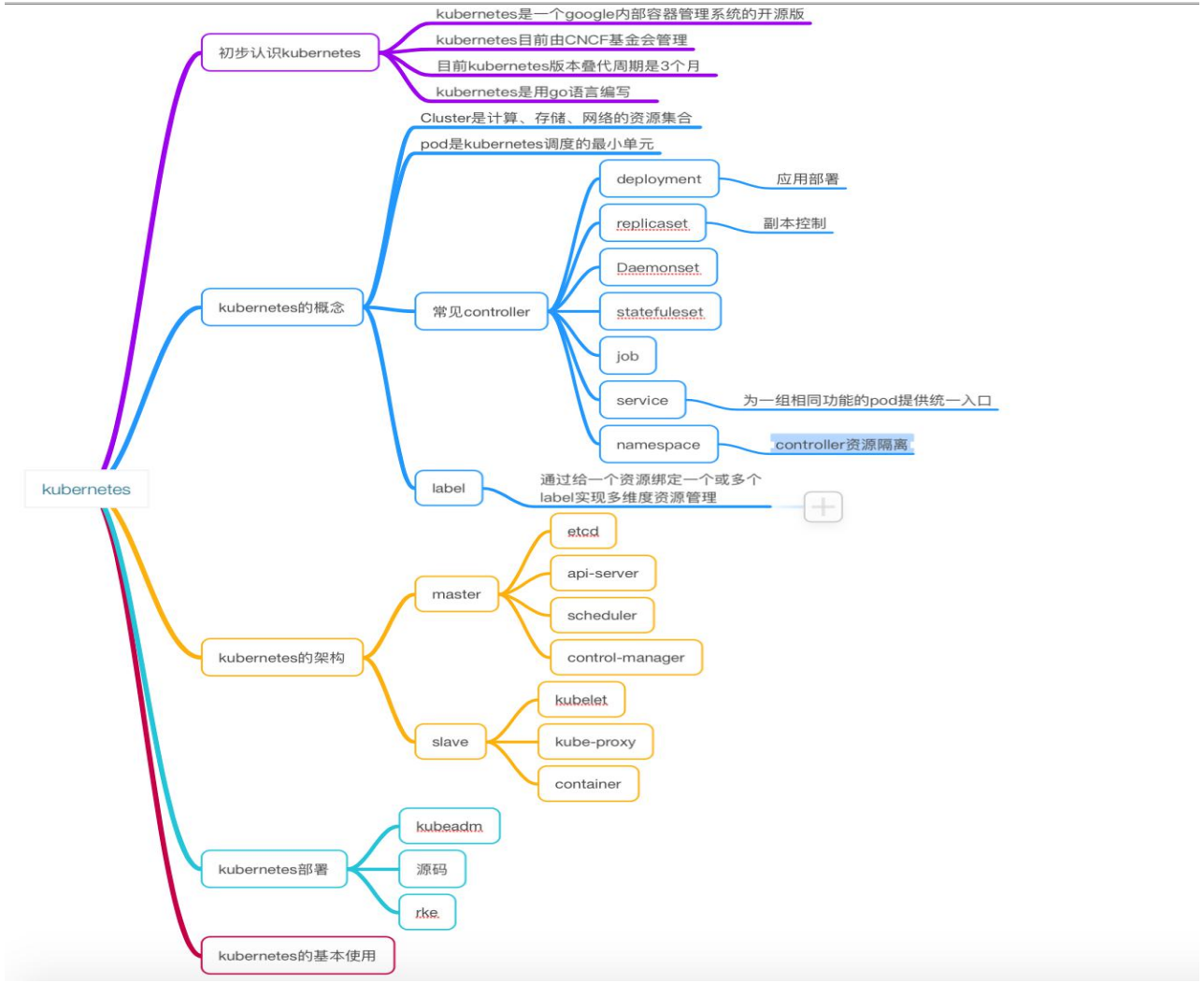
3、rke部署

rke是rancher开发的快速部署kubernetes的工具。

https://www.bladewan.com/2018/03/26/rke部署kubernetes/

RANCHER

4、kubernetes的部署



5、kubernetes的基本操作

查看集群信息

kubectl cluster-info

```
[root@master ~]# kubectl cluster-info
Kubernetes master is running at https://192.168.2.110:6443
KubeDNS is running at https://192.168.2.110:6443/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
[root@master ~]#
```

查看组件健康状态

kubectl get cs(componentstatuses)

```
[root@master ~]# kubectl get cs
NAME                 STATUS    MESSAGE              ERROR
scheduler            Healthy   ok
controller-manager   Healthy   ok
etcd-0               Healthy   {"health": "true"}
[root@master ~]#
```

RANCHER

5、kubernetes的基本操作

1、集群操作
kubectl get pod --all-namespaces

```
[root@master ~]# kubectl get pod --all-namespaces
NAMESPACE     NAME                                READY     STATUS    RESTARTS   AGE
default       httpd-app-5fbccd7c6c-jgmsx          1/1       Running   2          1d
default       kubernetes-bootcamp-6b7849c495-4zgwx   1/1       Running   2          1d
default       kubernetes-bootcamp-6b7849c495-xlnkb   1/1       Running   2          1d
kube-system   etcd-master                         1/1       Running   2          1d
kube-system   kube-apiserver-master               1/1       Running   4          1d
kube-system   kube-controller-manager-master      1/1       Running   5          1d
kube-system   kube-dns-6f4fd4bdf-v5v95            3/3       Running   5          1d
kube-system   kube-flannel-ds-9r5dg               1/1       Running   4          1d
kube-system   kube-flannel-ds-fmcqh               1/1       Running   2          1d
kube-system   kube-flannel-ds-smkjp               1/1       Running   3          1d
kube-system   kube-proxy-9kwl5                    1/1       Running   2          1d
kube-system   kube-proxy-sfxg4                    1/1       Running   2          1d
kube-system   kube-proxy-x6jhs                    1/1       Running   2          1d
kube-system   kube-scheduler-master               1/1       Running   3          1d
[root@master ~]#
```

2、查看指定namespace的pod的状态以kube-system为例
kubectl get pod --namespace=kube-system

```
[root@master ~]# kubectl get pod --namespace=kube-system
NAME                                READY     STATUS    RESTARTS   AGE
etcd-master                         1/1       Running   2          1d
kube-apiserver-master               1/1       Running   4          1d
kube-controller-manager-master      1/1       Running   5          1d
kube-dns-6f4fd4bdf-v5v95            3/3       Running   5          1d
kube-flannel-ds-9r5dg               1/1       Running   4          1d
kube-flannel-ds-fmcqh               1/1       Running   2          1d
kube-flannel-ds-smkjp               1/1       Running   3          1d
kube-proxy-9kwl5                    1/1       Running   2          1d
kube-proxy-sfxg4                    1/1       Running   2          1d
kube-proxy-x6jhs                    1/1       Running   2          1d
kube-scheduler-master               1/1       Running   3          1d
[root@master ~]#
```

RANCHER

5、kubernetes的基本操作

查看pod详细信息

kubectl describe pod kube-apiserver-master --namespace=kube-system

```
[root@master ~]# kubectl describe pod kube-apiserver-master  --namespace=kube-system
Name:         kube-apiserver-master
Namespace:    kube-system
Node:         master/192.168.2.110
Start Time:   Sat, 27 Jan 2018 10:28:34 +0800
Labels:       component=kube-apiserver
              tier=control-plane
Annotations:  kubernetes.io/config.hash=618b9beec70bf782d3b9777d4685a834
              kubernetes.io/config.mirror=618b9beec70bf782d3b9777d4685a834
              kubernetes.io/config.seen=2018-01-25T20:08:05.265438748+08:00
              kubernetes.io/config.source=file
              scheduler.alpha.kubernetes.io/critical-pod=
Status:       Running
IP:           192.168.2.110
```

查看pod详细run信息

kubectl get pods -o wide

```
[root@master ~]# kubectl get pods -o wide
NAME                                   READY     STATUS    RESTARTS   AGE       IP            NODE
httpd-app-5fbccd7c6c-jgmsx             1/1       Running   2          1d        10.244.2.19   node-2
kubernetes-bootcamp-6b7849c495-4zgwx   1/1       Running   2          1d        10.244.1.36   node-1
kubernetes-bootcamp-6b7849c495-xlnkb   1/1       Running   2          1d        10.244.2.20   node-2
[root@master ~]#
```

RANCHER

5、kubernetes的基本操作

查看node节点详细信息

kubectl describe node hostname

```
[[root@master ~]# kubectl describe node master
Name:               master
Roles:              master
Labels:             beta.kubernetes.io/arch=amd64
                    beta.kubernetes.io/os=linux
                    kubernetes.io/hostname=master
                    node-role.kubernetes.io/master=
Annotations:        flannel.alpha.coreos.com/backend-data={"VtepMAC":"e6:23:db:8d:8b:c7"}
                    flannel.alpha.coreos.com/backend-type=vxlan
                    flannel.alpha.coreos.com/kube-subnet-manager=true
                    flannel.alpha.coreos.com/public-ip=192.168.2.110
                    node.alpha.kubernetes.io/ttl=0
                    volumes.kubernetes.io/controller-managed-attach-detach=true
Taints:             node-role.kubernetes.io/master:NoSchedule
CreationTimestamp:  Thu, 25 Jan 2018 20:08:22 +0800
Conditions:
  Type             Status  LastHeartbeatTime                 LastTransitionTime                Reason                       Message
  ----             ------  -----------------                 ------------------                ------                       -------
  OutOfDisk        False   Sat, 27 Jan 2018 11:28:51 +0800   Thu, 25 Jan 2018 20:08:15 +0800   KubeletHasSufficientDisk     kubelet has sufficient disk space available
  MemoryPressure   False   Sat, 27 Jan 2018 11:28:51 +0800   Thu, 25 Jan 2018 20:08:15 +0800   KubeletHasSufficientMemory   kubelet has sufficient memory available
  DiskPressure     False   Sat, 27 Jan 2018 11:28:51 +0800   Thu, 25 Jan 2018 20:08:15 +0800   KubeletHasNoDiskPressure     kubelet has no disk pressure
  Ready            True    Sat, 27 Jan 2018 11:28:51 +0800   Thu, 25 Jan 2018 20:12:43 +0800   KubeletReady                 kubelet is posting ready status
Addresses:
  InternalIP:  192.168.2.110
  Hostname:    master
```

RANCHER

5、kubernetes的基本操作

查看deployment状态
kubectl get deployment

```
[root@master ~]# kubectl get deployment
NAME                  DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
httpd-app             1         1         1            1           1d
kubernetes-bootcamp   2         2         2            2           1d
[root@master ~]#
```

查看指定deployment详细信息
kubectl describe deployment deployment_name

```
[root@master ~]# kubectl describe deployment httpd-app
Name:                   httpd-app
Namespace:              default
CreationTimestamp:      Thu, 25 Jan 2018 20:28:47 +0800
Labels:                 run=httpd-app
Annotations:            deployment.kubernetes.io/revision=1
Selector:               run=httpd-app
Replicas:               1 desired | 1 updated | 1 total | 1 available | 0 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  1 max unavailable, 1 max surge
Pod Template:
  Labels:  run=httpd-app
  Containers:
   httpd-app:
    Image:        httpd
    Port:         <none>
    Environment:  <none>
    Mounts:       <none>
  Volumes:        <none>
Conditions:
  Type           Status  Reason
  ----           ------  ------
  Available      True    MinimumReplicasAvailable
OldReplicaSets:  <none>
NewReplicaSet:   httpd-app-5fbccd7c6c (1/1 replicas created)
Events:          <none>
[root@master ~]#
```

RANCHER

5、kubernetes的基本操作

删除一个deployment

kubectl delete deployment deployment_name

查看创建的service

kubectl get service 或 kubectl get svc

```
[root@master ~]# kubectl get service
NAME                  TYPE        CLUSTER-IP    EXTERNAL-IP   PORT(S)          AGE
kubernetes            ClusterIP   10.96.0.1     <none>        443/TCP          1d
kubernetes-bootcamp   NodePort    10.97.7.225   <none>        8080:30619/TCP   1d
```

kubectl describe service service_name

```
[root@master ~]# kubectl describe service kubernetes-bootcamp
Name:                     kubernetes-bootcamp
Namespace:                default
Labels:                   run=kubernetes-bootcamp
Annotations:              <none>
Selector:                 run=kubernetes-bootcamp
Type:                     NodePort
IP:                       10.97.7.225
Port:                     <unset>  8080/TCP
TargetPort:               8080/TCP
NodePort:                 <unset>  30619/TCP
Endpoints:                10.244.1.36:8080,10.244.2.20:8080
Session Affinity:         None
External Traffic Policy:  Cluster
Events:                   <none>
[root@master ~]#
```

RANCHER

5、kubernetes的基本操作

删除service
kubectl delete service service_name

设置label
kubectl label node node-1 disktype=ssd

查看设置的label
kubectl get node --show-labels

RANCHER

5、kubernetes的基本操作

部署应用（kubernetes-bootcamp)

kubectl run kubernetes-bootcamp --image=docker.io/jocatalin/kubernetes-bootcamp:v1 --port=8080

```
[root@master ~]# kubectl run kubernetes-bootcamp --image=docker.io/jocatalin/kubernetes-bootcamp:v1 --port=8080
deployment "kubernetes-bootcamp" created
[root@master ~]# kubectl run kubernetes-bootcamp --image=docker.io/jocatalin/kubernetes-bootcamp:v1 --port=8080
deployment "kubernetes-bootcamp" created
```

查看部署的应用，这里deployment为kubernetes术语，理解为应用。
kubectl get deployment

```
root@rke-node1:~# kubectl get deployment
NAME                  DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
kubernetes-bootcamp   1         1         1            1           7s
root@rke-node1:~#
```

docker镜像通过--image指定

port设置应用对外服务的端口。

RANCHER

5、kubernetes的基本操作

查看当前pod
kubectl get pods

```
[root@master ~]# kubectl get pods
NAME                                   READY     STATUS    RESTARTS   AGE
kubernetes-bootcamp-5d7f968ccb-5hfq5   1/1       Running   0          51s
[root@master ~]#
```

查看 pod实际运行在哪个节点上
kubectl get pod -o wide

```
[root@master ~]# kubectl get pod  -o wide
NAME                                   READY     STATUS    RESTARTS   AGE       IP            NODE
kubernetes-bootcamp-5d7f968ccb-5hfq5   1/1       Running   0          6m        10.244.2.21   node-2
[root@master ~]#
```

访问

curl 10.244.2.21:8080

```
[root@master ~]# curl 10.244.2.21:8080
Hello Kubernetes bootcamp! | Running on: kubernetes-bootcamp-5d7f968ccb-5hfq5 | v=1
[root@master ~]#
```

RANCHER

5、kubernetes的基本操作

创建Service

默认情况下使用pod只能在集群内部进行访问，为了能在外部访问容器，需要将容器应用的端口映射到node节点端口。
修改kubernetes-bootcamp的端口类型为NodePort，容器内端口为8080，映射的端口会自动从30000~32767中挑选一个。
services可以认为是端口映射。

kubectl expose deployment/kubernetes-bootcamp --type="NodePort" --port=8080

```
[root@master ~]# kubectl expose deployment/kubernetes-bootcamp --type="NodePort" --port=8080
service "kubernetes-bootcamp" exposed
[root@master ~]# kubectl get service
NAME                  TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
kubernetes            ClusterIP   10.96.0.1       <none>        443/TCP          1d
kubernetes-bootcamp   NodePort    10.105.113.24   <none>        8080:31970/TCP   16s
[root@master ~]#
```

RANCHER

5、kubernetes的基本操作

kubernetes-bootcamp分配到了node-2上，可以直接访问node-2的:31079

```
[root@master ~]# kubectl get pods -o wide
NAME                                  READY     STATUS    RESTARTS   AGE       IP             NODE
http-app-6d6d7f55c5-c92x5             1/1       Running   2          23h       10.244.2.41    node-2
http-app-6d6d7f55c5-fxqbb             1/1       Running   2          23h       10.244.1.223   node-1
kubernetes-bootcamp-5d7f968ccb-ztw59  1/1       Running   0          13m       10.244.2.42    node-2
[root@master ~]#
```

```
[root@master ~]# curl node-2:31079
Hello Kubernetes bootcamp! | Running on: kubernetes-bootcamp-5d7f968ccb-ztw59 | v=1
[root@master ~]#
```

RANCHER

5、kubernetes的基本操作

scale应用（弹性伸缩)
当前kubernetes-bootcamp副本数还是为1

```
[root@master ~]# kubectl get deployment
NAME                  DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
http-app              2         2         2            2           23h
kubernetes-bootcamp   1         1         1            1           15m
[root@master ~]#
```

扩展应用为3

kubectl scale deployment/kubernetes-bootcamp --replicas=3

```
[root@master ~]# kubectl get pods
NAME                                  READY     STATUS              RESTARTS   AGE
kubernetes-bootcamp-5d7f968ccb-5hfq5  1/1       Running             0          13m
kubernetes-bootcamp-5d7f968ccb-95rsz  0/1       ContainerCreating   0          3s
kubernetes-bootcamp-5d7f968ccb-b9dzb  0/1       ContainerCreating   0          3s
[root@master ~]#

[root@master ~]# kubectl get deployment
NAME                  DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
kubernetes-bootcamp   3         3         3            3           13m
[root@master ~]#
```

RANCHER

5、kubernetes的基本操作

缩减pod数为2
scale应用（弹性伸缩)

kubectl scale deployment/kubernetes-bootcamp --replicas=2

```
[root@master ~]# kubectl get pods
NAME                                  READY     STATUS        RESTARTS   AGE
kubernetes-bootcamp-5d7f968ccb-5hfq5  1/1       Running       0          16m
kubernetes-bootcamp-5d7f968ccb-95rsz  1/1       Terminating   0          2m
kubernetes-bootcamp-5d7f968ccb-b9dzb  1/1       Running       0          2m
[root@master ~]#
```

```
[root@master ~]# kubectl get deployment
NAME                  DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
kubernetes-bootcamp   2         2         2            2           16m
[root@master ~]#
```

每次访问nodeip:31079，都是不同的pod给回的返回结果。可以发现每次请求发到了不同的pod上从而实现了负载均衡。

```
[root@master ~]# curl node-1:31970
Hello Kubernetes bootcamp! | Running on: kubernetes-bootcamp-5d7f968ccb-95rsz | v=1
[root@master ~]# curl node-1:31970
Hello Kubernetes bootcamp! | Running on: kubernetes-bootcamp-5d7f968ccb-b9dzb | v=1
[root@master ~]# curl node-1:31970
Hello Kubernetes bootcamp! | Running on: kubernetes-bootcamp-5d7f968ccb-95rsz | v=1
[root@master ~]# curl node-1:31970
Hello Kubernetes bootcamp! | Running on: kubernetes-bootcamp-5d7f968ccb-5hfq5 | v=1
[root@master ~]#
```

RANCHER

5、kubernetes的基本操作

滚动更新

当前应用image版本为v1，升级为v2
kubectl set image deployment/kubernetes-bootcamp kubernetes-bootcamp=jocatalin/kubernetes-bootcamp:v2

```
[[root@master ~]# kubectl get pods
NAME                                  READY     STATUS    RESTARTS   AGE
kubernetes-bootcamp-5d7f968ccb-c7bpp  1/1       Running   0          2m
kubernetes-bootcamp-5d7f968ccb-gnjjx  1/1       Running   0          2m
[[root@master ~]# kubectl set image deployment/kubernetes-bootcamp kubernetes-bootcamp=jocatalin/kubernetes-bootcamp:v2
deployment "kubernetes-bootcamp" image updated
[[root@master ~]# kubectl get pods
NAME                                  READY     STATUS              RESTARTS   AGE
kubernetes-bootcamp-5d7f968ccb-c7bpp  1/1       Running             0          2m
kubernetes-bootcamp-5d7f968ccb-gnjjx  1/1       Terminating         0          2m
kubernetes-bootcamp-7689dc585d-2dsss  0/1       ContainerCreating   0          1s
kubernetes-bootcamp-7689dc585d-2zb8b  0/1       ContainerCreating   0          1s
[root@master ~]#
```

RANCHER

5、kubernetes的基本操作

可以看见kubernetes是创建出两个v2版本的镜像，然后在缓慢用v2将v1替换掉

```
[root@master ~]# kubectl get pods
NAME                                   READY     STATUS        RESTARTS   AGE
kubernetes-bootcamp-5d7f968ccb-c7bpp   1/1       Terminating   0          2m
kubernetes-bootcamp-5d7f968ccb-gnjjx   1/1       Terminating   0          2m
kubernetes-bootcamp-7689dc585d-2dsss   1/1       Running       0          9s
kubernetes-bootcamp-7689dc585d-2zb8b   1/1       Running       0          9s
[root@master ~]# kubectl get pods
```

在次访问，版本是v2了

```
[[root@master ~]# curl node-1:31970
Hello Kubernetes bootcamp! | Running on: kubernetes-bootcamp-7689dc585d-2zb8b | v=2
[root@master ~]#
```

RANCHER

5、kubernetes的基本操作

回退版本

kubectl rollout undo deployment/kubernetes-bootcamp

```
[root@master ~]# kubectl rollout undo  deployment/kubernetes-bootcamp
deployment "kubernetes-bootcamp"
[root@master ~]# kubectl get pods
NAME                                  READY     STATUS              RESTARTS   AGE
kubernetes-bootcamp-5d7f968ccb-6fskf  0/1       ContainerCreating   0          3s
kubernetes-bootcamp-5d7f968ccb-jz555  0/1       ContainerCreating   0          3s
kubernetes-bootcamp-7689dc585d-2dsss  1/1       Running             0          2m
kubernetes-bootcamp-7689dc585d-2zb8b  1/1       Terminating         0          2m
[root@master ~]# kubectl get pods
NAME                                  READY     STATUS        RESTARTS   AGE
kubernetes-bootcamp-5d7f968ccb-6fskf  1/1       Running       0          7s
kubernetes-bootcamp-5d7f968ccb-jz555  1/1       Running       0          7s
kubernetes-bootcamp-7689dc585d-2dsss  1/1       Terminating   0          2m
kubernetes-bootcamp-7689dc585d-2zb8b  1/1       Terminating   0          2m
[root@master ~]#
```

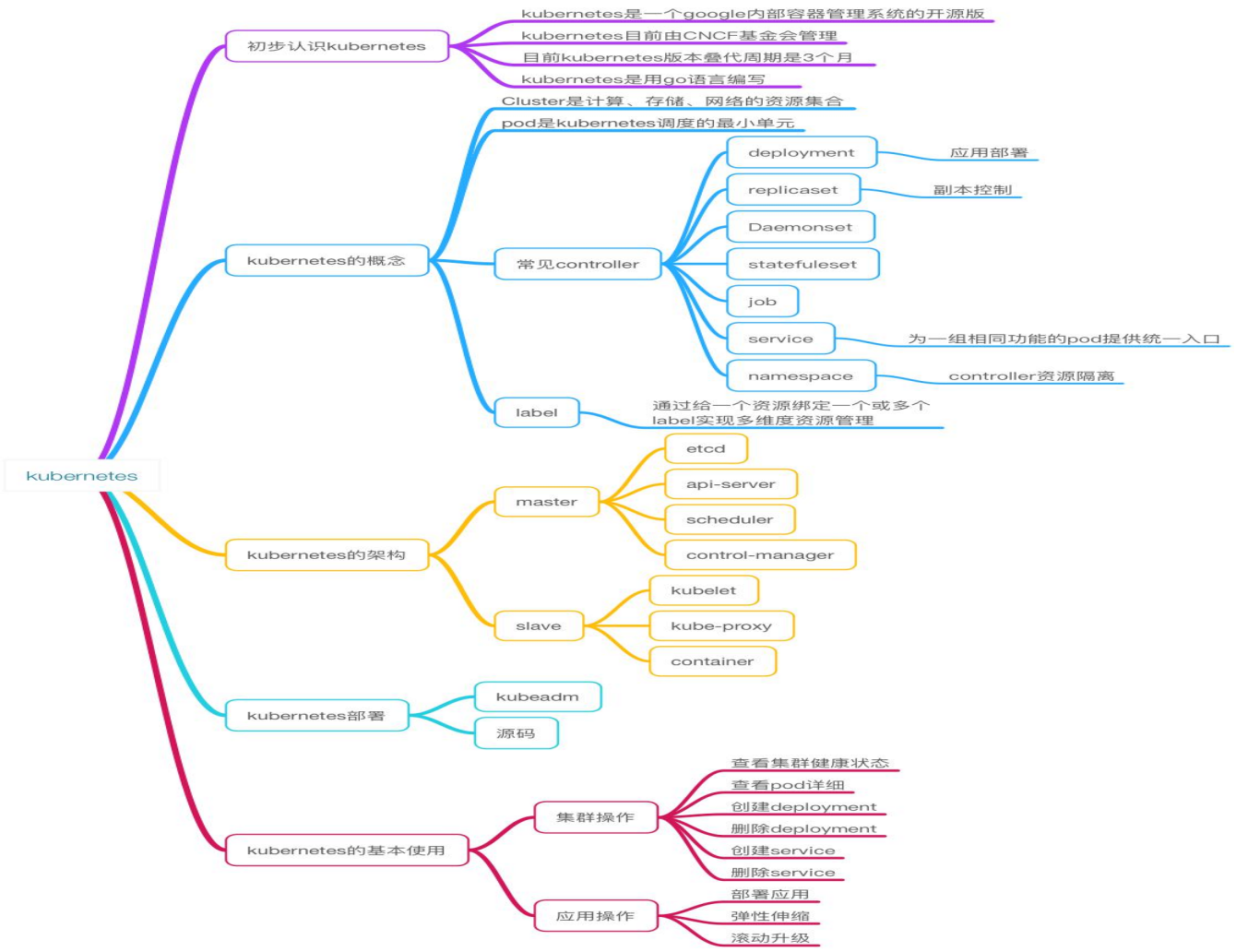
验证结果

```
[root@master ~]# curl node-1:31970
Hello Kubernetes bootcamp! | Running on: kubernetes-bootcamp-5d7f968ccb-6fskf | v=1
[root@master ~]#
```

RANCHER

5、kubernetes的基本操作

总结



Thank You

关注微信公众号
获取第一手干货与资讯

加入官方微信群
与更多同道中人互动

RANCHER®