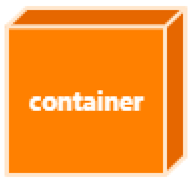
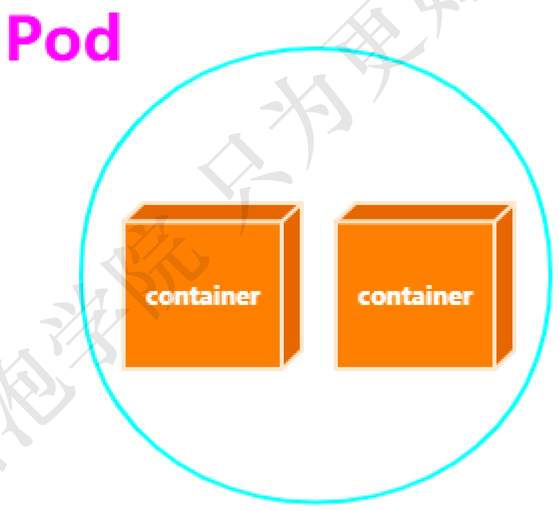
K8S核心组件和架构图

> K8S Docs Concepts：https://kubernetes.io/docs/concepts/

> (1)先以container为起点，k8s既然是容器编排工具，那么一定会有container



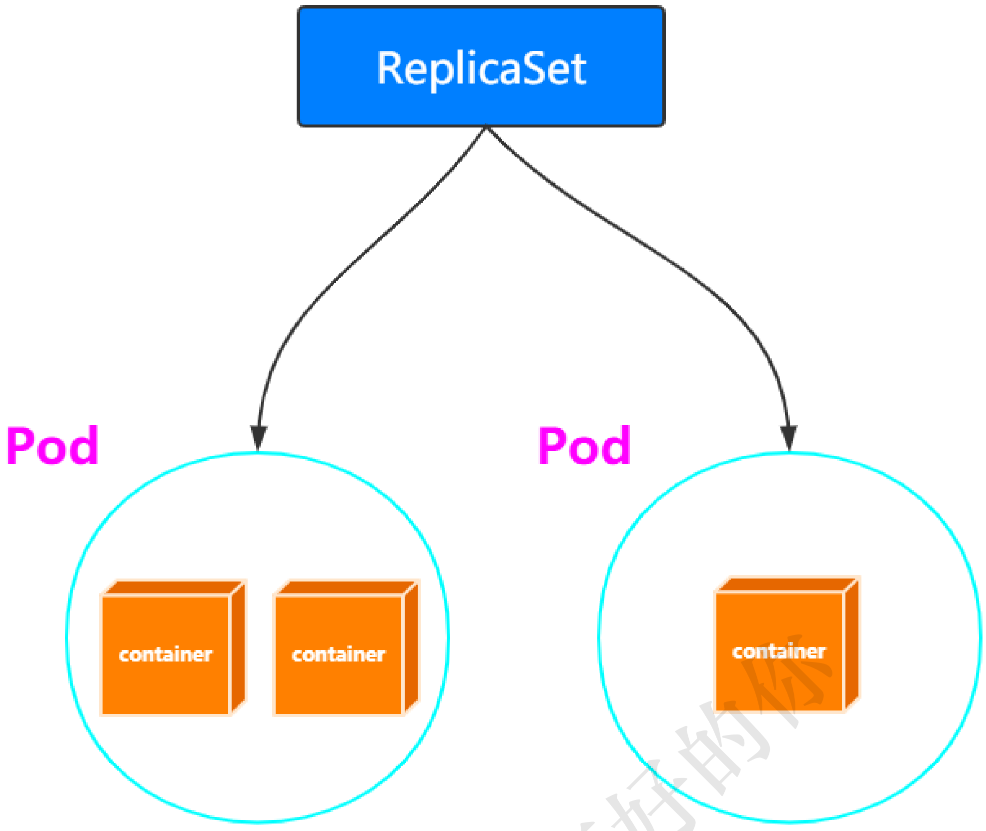
> (2)那k8s如何操作这些container呢？从感性的角度来讲，得要有点逼格，k8s不想直接操作container，因为操作container的事情是docker来做的，k8s中要有自己的最小操作单位，称之为Pod
>
> 说白了，Pod就是一个或多个Container的组合



`看看官网怎么描述的`：https://kubernetes.io/docs/concepts/workloads/pods/pod/

```
A Pod (as in a pod of whales or pea pod) is a group of one or more containers
(such as Docker containers),
with shared storage/network, and a specification for how to run the containers.
```

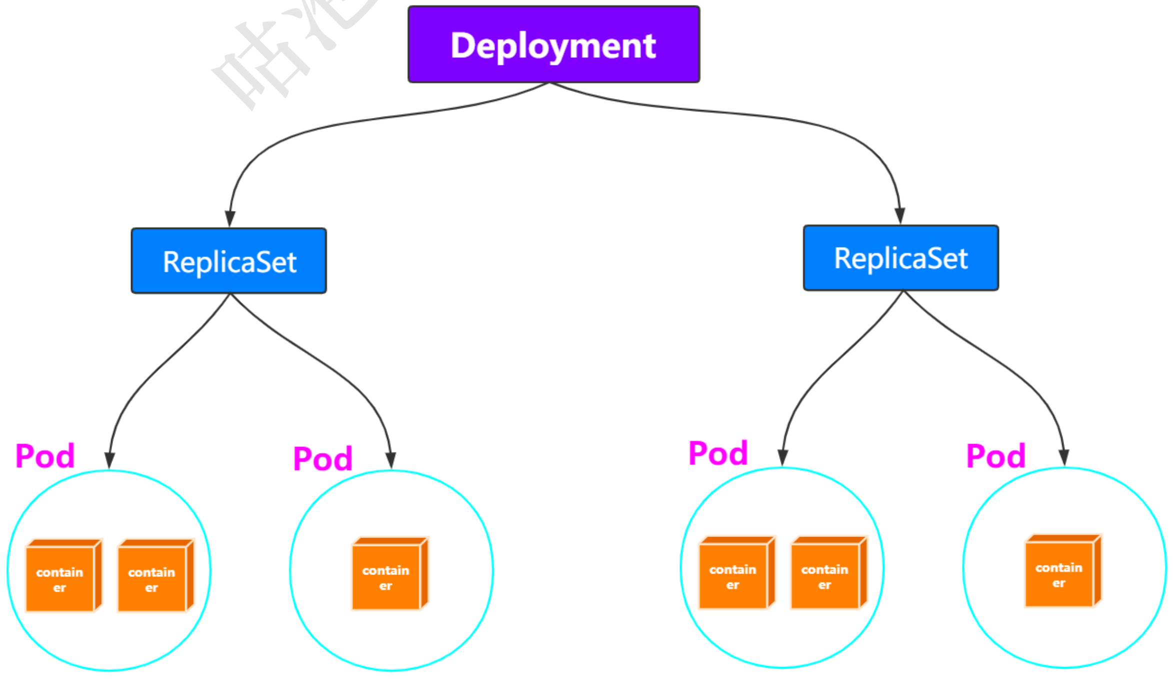
> (3)那Pod的维护谁来做呢？那就是ReplicaSet，通过selector来进行管理



看看官网怎么描述的： https://kubernetes.io/docs/concepts/workloads/controllers/replicaset/

```
A ReplicaSet is defined with fields, including a selector that specifies how to
identify Pods it can acquire, a number of replicas indicating how many Pods it
should be maintaining, and a pod template specifying the data of new Pods it
should create to meet the number of replicas criteria.
```

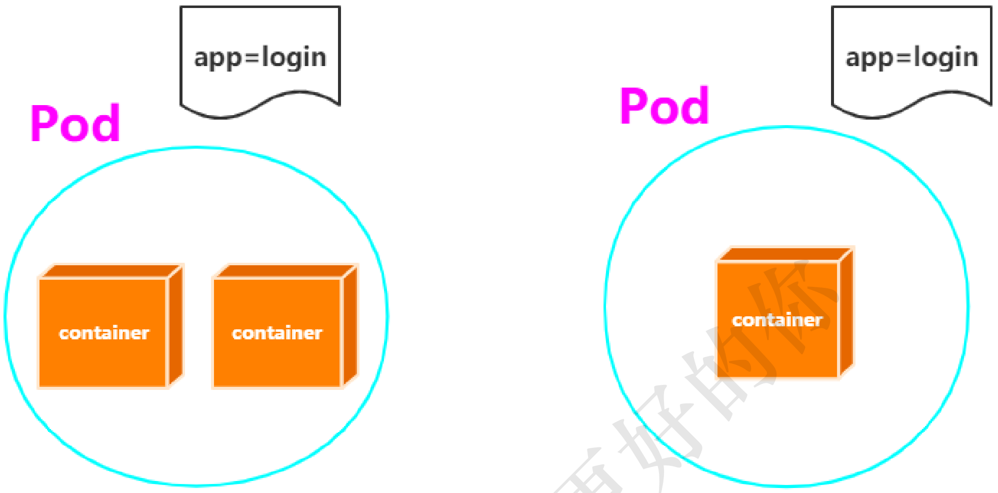
> (4)Pod和ReplicaSet的状态如何维护和监测呢？Deployment



官网是如何描述的： https://kubernetes.io/docs/concepts/workloads/controllers/deployment/

```
A Deployment controller provides declarative updates for Pods and ReplicaSets.

You describe a desired state in a Deployment, and the Deployment controller
changes the actual state to the desired state at a controlled rate. You can
define Deployments to create new ReplicaSets, or to remove existing Deployments
and adopt all their resources with new Deployments.
```

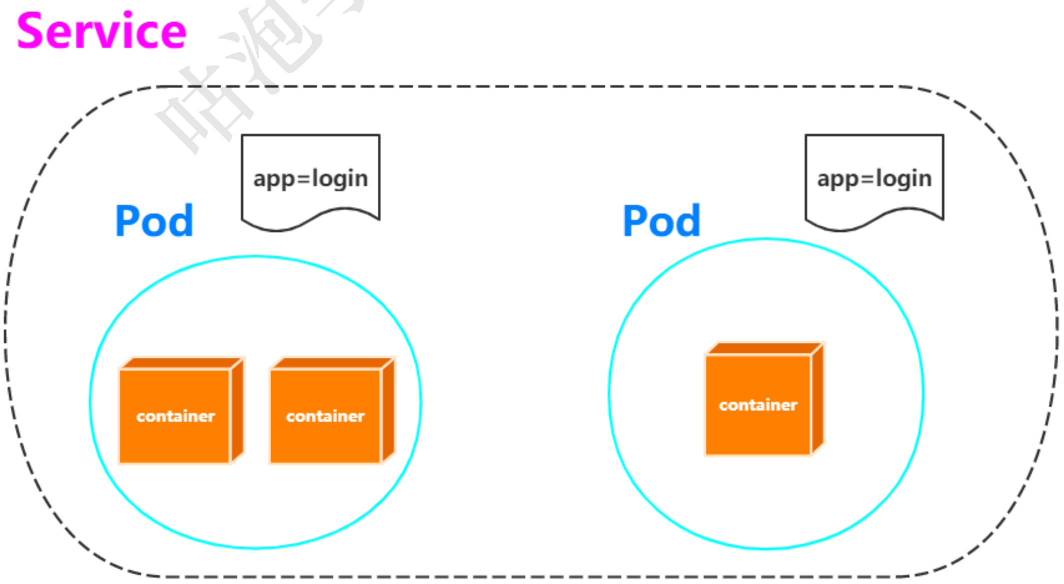
> (5)不妨把相同或者有关联的Pod分门别类一下，那怎么分门别类呢？Label



`官网是如何描述的`：https://kubernetes.io/docs/concepts/overview/working-with-objects/labels/

```
Labels are key/value pairs that are attached to objects, such as pods.
```

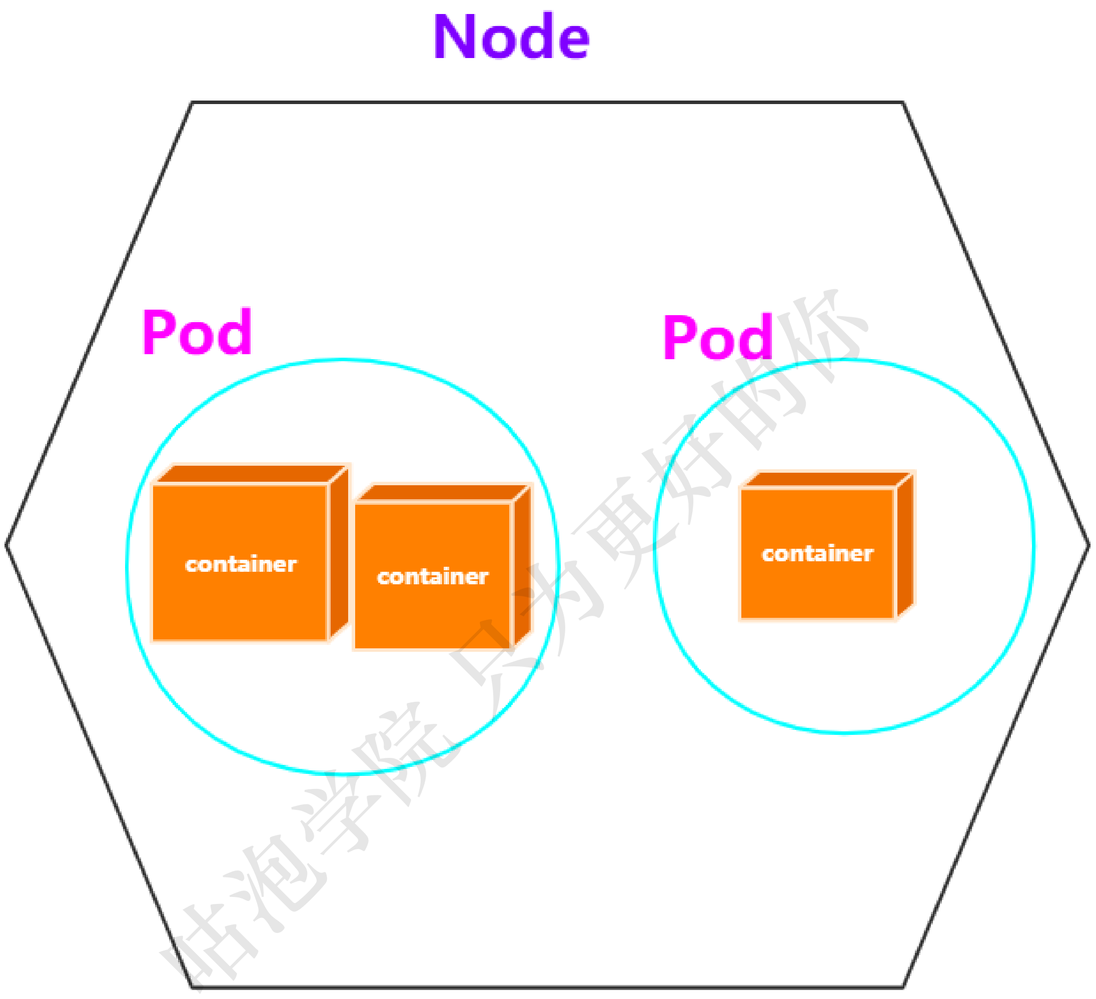
> (6)具有相同label的service要是能够有个名称就好了，Service



`看官网上怎么说`：https://kubernetes.io/docs/concepts/services-networking/service/

```
An abstract way to expose an application running on a set of Pods as a network
service.

With Kubernetes you don't need to modify your application to use an unfamiliar
service discovery mechanism. Kubernetes gives Pods their own IP addresses and a
single DNS name for a set of Pods, and can load-balance across them.
```

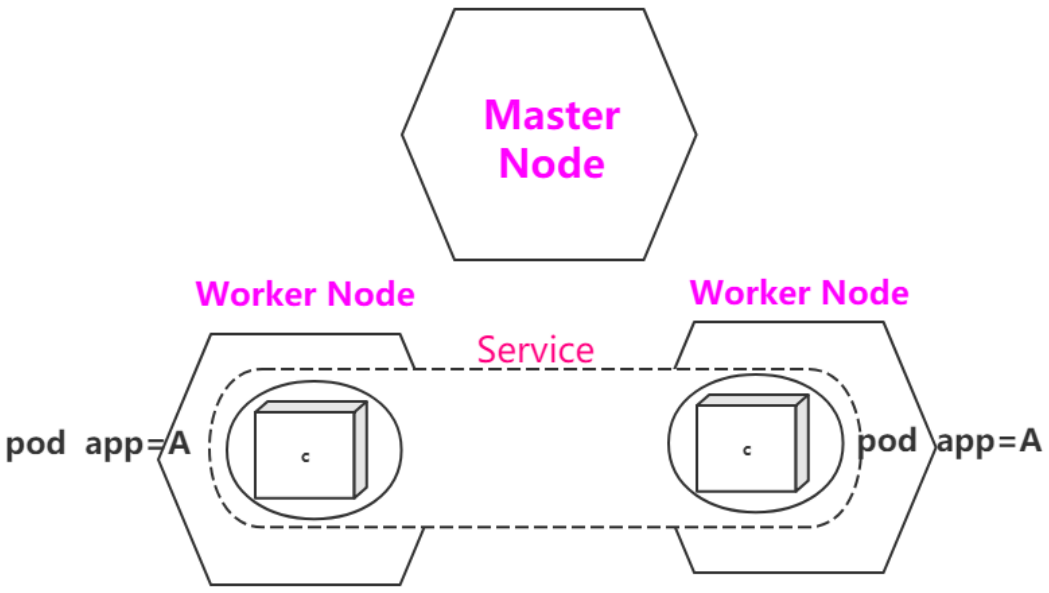
> (7)上述说了这么多，Pod运行在哪里呢？当然是机器咯，比如一台centos机器，我们把这个机器称作为Node



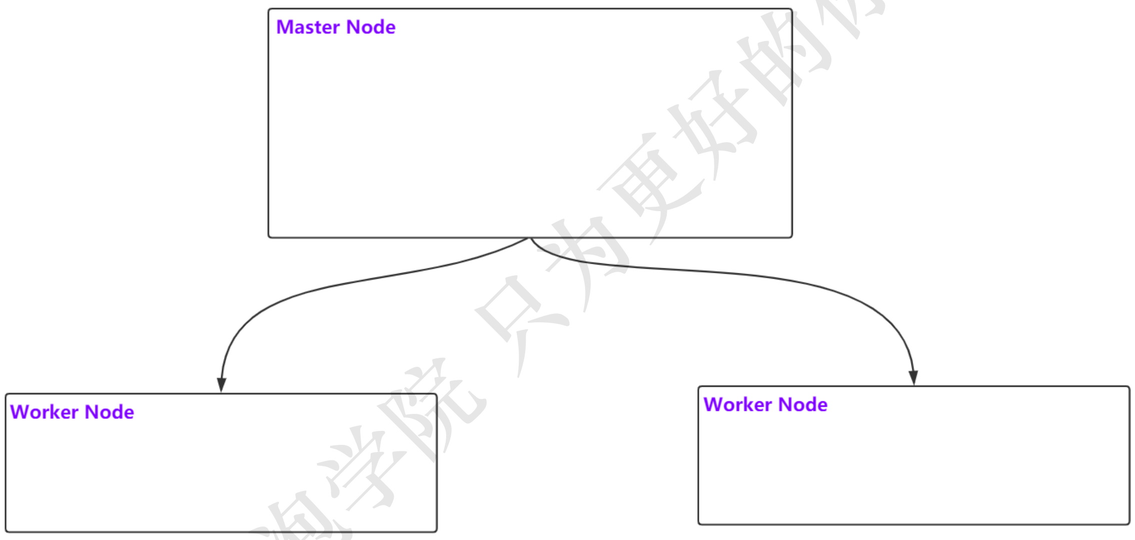
`看看官网怎么说`：https://kubernetes.io/docs/concepts/architecture/nodes/

```
A node is a worker machine in Kubernetes, previously known as a minion. A node
may be a VM or physical machine, depending on the cluster. Each node contains
the services necessary to run pods and is managed by the master components.
```

> (8)难道只有一个Node吗？显然不太合适，多台Node共同组成集群才行嘛
>
> 画个图表示一下咯，最好能把之前的Label，Service也一起画上去，整体感受一下

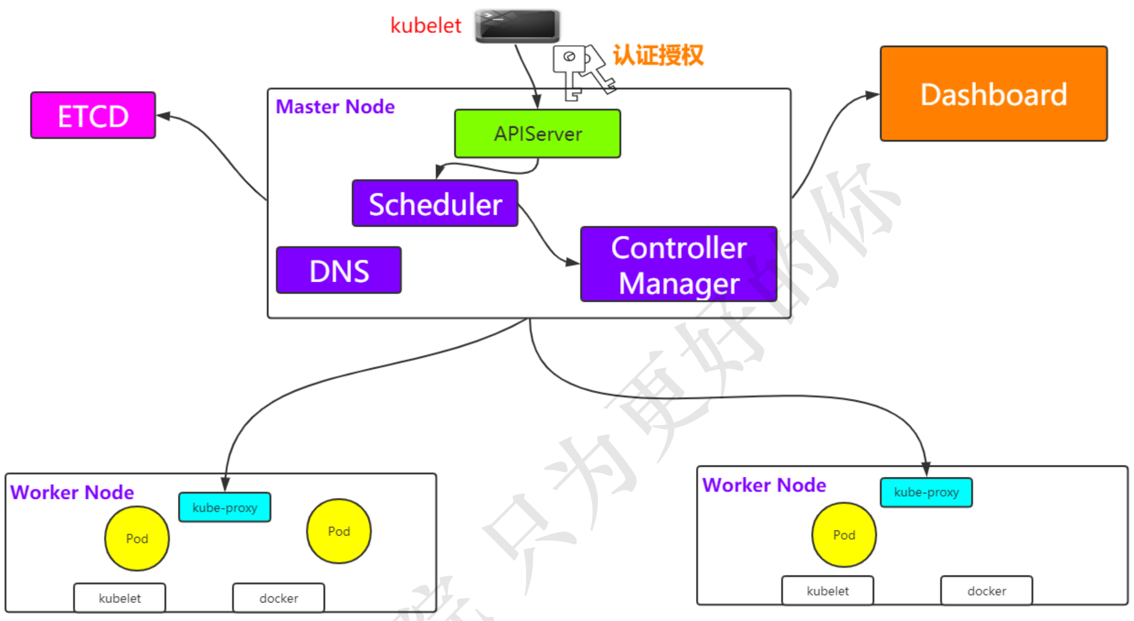


> (9)此时，我们把目光转移到由3个Node节点组成的Master-Node集群

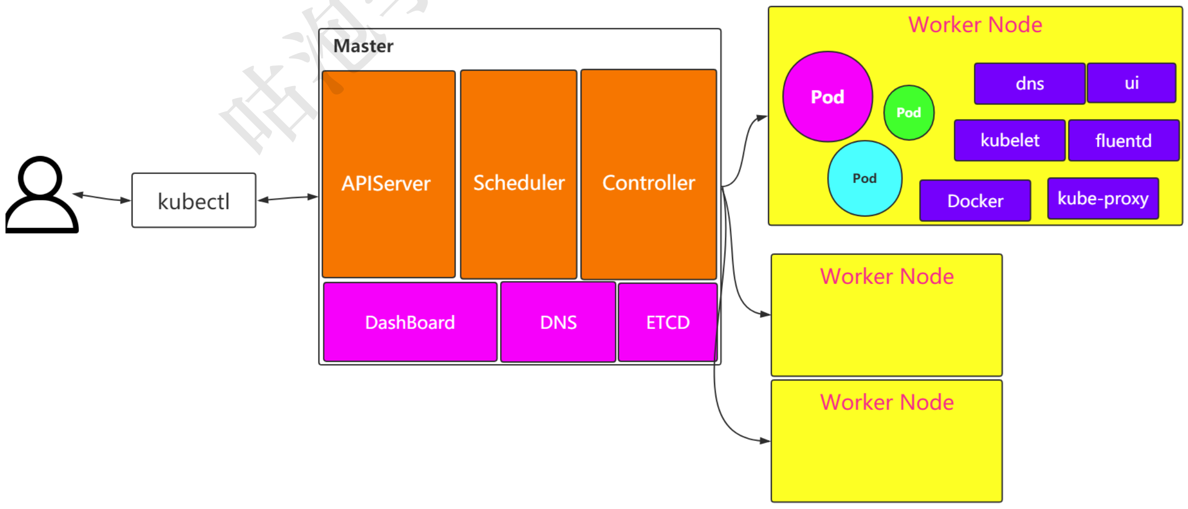


> (10)这个集群要配合完成一些工作，总要有一些组件的支持吧？接下来我们来想想有哪些组件，然后画一个相对完整的架构图
>
> ```
> 01-总得要有一个操作集群的客户端，也就是和集群打交道
>     kubectl
>
> 02-请求肯定是到达Master Node，然后再分配给Worker Node创建Pod之类的
>     关键是命令通过kubectl过来之后，是不是要认证授权一下？
>
> 03-请求过来之后，Master Node中谁来接收？
>     APIServer
>
> 04-API收到请求之后，接下来调用哪个Worker Node创建Pod，Container之类的，得要有调度策略
>     Scheduler
>     [https://kubernetes.io/docs/concepts/scheduling/kube-scheduler/]
>
>
> 05-Scheduler通过不同的策略，真正要分发请求到不同的Worker Node上创建内容，具体谁负责？
>     Controller Manager
>
> 06-Worker Node接收到创建请求之后，具体谁来负责
> ```

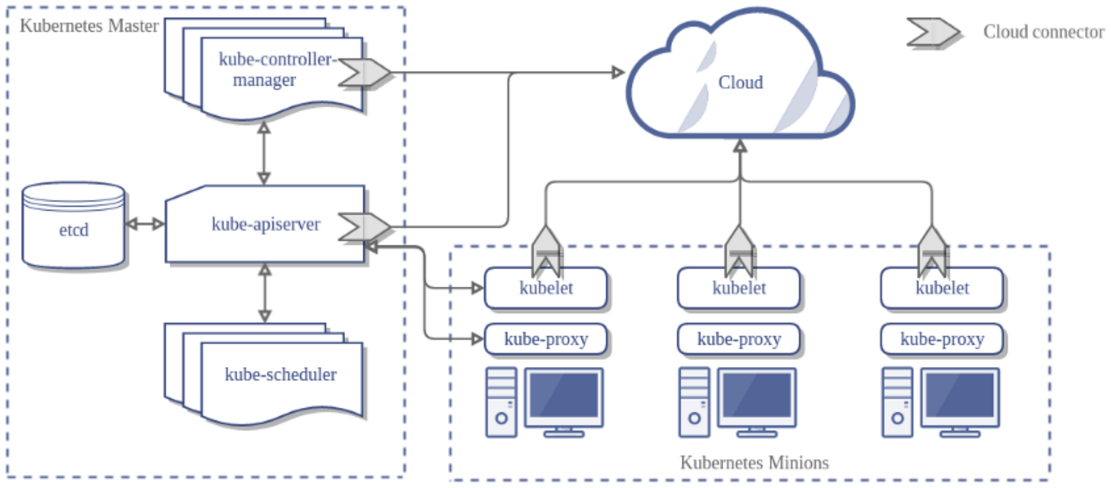
```
    Kubelet服务，最终Kubelet会调用Docker Engine，创建对应的容器[这边是不是也反应出一
点，在Node上需要有Docker Engine，不然怎么创建维护容器？]

07-会不会涉及到域名解析的问题？
    DNS

08-是否需要有监控面板能够监测整个集群的状态？
    Dashboard

09-集群中这些数据如何保存？分布式存储
    ETCD

10-至于像容器的持久化存储，网络等可以联系一下Docker中的内容
```



> (11)不妨把这个图翻转一下方便查看



> (12)官网K8S架构图
>
> https://kubernetes.io/docs/concepts/architecture/cloud-controller/



`小结`：有些小伙伴可能会想，有些组件知道它的作用了，有些有些模糊。没关系，目前我们只是停留在理论层面的推断和分析，毕竟一点实际操作都没有，现在能够有个感性的认知就已经很不错了。

The Common Ways of Installing Kubernetes

The hard way

> `Kelsey Hightower`：https://github.com/kelseyhightower

在线play-with-k8s

> `网址`：https://labs.play-with-k8s.com/

```
 This is a sandbox environment. Using personal credentials
 is HIGHLY! discouraged. Any consequences of doing so, are
 completely the user's responsibilites.

 You can bootstrap a cluster as follows:

 1. Initializes cluster master node:
 kubeadm init --apiserver-advertise-address $(hostname -i)

 2. Initialize cluster networking:
 kubectl apply -n kube-system -f \
    "https://cloud.weave.works/k8s/net?k8s-version=$(kubectl version | base64
|tr -d '\n')"

 3. (Optional) Create an nginx deployment:
 kubectl apply -f
https://raw.githubusercontent.com/kubernetes/website/master/content/en/examples/
application/nginx-app.yaml
```

Cloud上搭建

> `GitHub`：https://github.com/kubernetes/kops

企业级解决方案CoreOS

> `coreos`：https://coreos.com/tectonic/

Minikube[Y]

> `K8S单节点，适合在本地学习使用`
>
> `官网`：https://kubernetes.io/docs/setup/learning-environment/minikube/
>
> `GitHub`：https://github.com/kubernetes/minikube

kubeadm[Y]

> `本地多节点`
>
> `GitHub`：https://github.com/kubernetes/kubeadm

使用Minikube搭建单节点K8s

Windows

> `kubectl官网`：https://kubernetes.io/docs/tasks/tools/install-kubectl/#install-kubectl-on-windows
>
> `minikube官网`：https://kubernetes.io/docs/tasks/tools/install-minikube/

- 选择任意一种虚拟化的方式

```
• Hyper-V
• VirtualBox[课上选择的]
```

- 安装kubectl

```
(1)根据官网步骤  [或]  直接下载:
https://storage.googleapis.com/kubernetes-
release/release/v1.16.2/bin/windows/amd64/kubectl.exe

(2)配置kubectl.exe所在路径的环境变量，使得cmd窗口可以直接使用kubectl命令

(4)kubectl version检查是否配置成功
```

- 安装minikube

```
(1)根据官网步骤  [或]  直接下载:
https://github.com/kubernetes/minikube/releases/download/v1.5.2/minikube-
windows-amd64.exe

(2)修改minikube-windows-amd64.exe名称为minikube.exe

(3)配置minikube所在路径的环境变量，使得cmd窗口可以直接使用minikube命令

(4)minikube version检查是否配置成功
```

- 使用minikube创建单节点的k8s

```
minikube start --vm-driver=virtualbox --image-repository=gcr.azk8s.cn/google-
containers
```

- 小结

> **其实就是通过minikube创建一个虚拟机**
>
> **这个虚拟机中安装好了单节点的K8S环境然后通过kubectl进行交互**

```
# 创建K8S
minikube start

# 删除K8S
minikube delete

# 进入到K8S的机器中
minikube ssh

# 查看状态
minikube status

# 进入dashboard
minikube dashboard
```

CentOS

> `kubectl官网`：https://kubernetes.io/docs/tasks/tools/install-kubectl/#install-kubectl-on-linux
>
> `minikube官网`：https://kubernetes.io/docs/tasks/tools/install-minikube/

- 安装docker
- 安装kubectl

```
# 01 下载[这边我给大家下载好了，在网盘kubectl&minikube中，大家上传到自己的centos7机器中。]

# 02 授权
chmod +x ./kubectl

# 03 添加到环境变量
sudo mv ./kubectl /usr/local/bin/kubectl

# 04 检查
kubectl version
```

- 安装minikube

```
# 01 下载[这边我给大家下载好了，在网盘kubectl&minikube中，大家上传到自己的centos7机器中。]
wget https://github.com/kubernetes/minikube/releases/download/v1.5.2/minikube-linux-amd64

# 02 配置环境变量
sudo mv minikube-linux-amd64 minikube && chmod +x minikube && mv minikube /usr/local/bin/

# 03 检查
minikube version
```

- 使用minikube创建单节点的k8s

```
minikube start --vm-driver=none --image-repository=gcr.azk8s.cn/google-
containers
```

Mac OS

也是下载安装kubectl和minikube，选择virtualbox，然后minikube start，就可以通过kubectl操作咯

先感受一下Kubernetes

> 既然已经通过Minikube搭建了单节点的Kubernetes，不妨先感受一些组件的存在以及操作咯

查看连接信息

```
kubectl config view
kubectl config get-contexts
kubectl cluster-info
```

体验Pod

> (1)创建pod_nginx.yaml
>
> resources/basic/pod_nginx.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  labels:
    app: nginx
spec:
  containers:
  - name: nginx
    image: nginx
    ports:
    - containerPort: 80
```

> (2)根据pod_nginx.yaml文件创建pod

```
kubectl apply -f pod_nginx.yaml
```

> (3)查看pod

```
kubectl get pods
kubectl get pods -o wide
kubectl describe pod nginx
```

> (4)进入nginx容器

```
# kubectl进入
kubectl exec -it nginx bash

# 通过docker进入
minikube ssh
docker ps
docker exec -it containerid bash
```

> (5)访问nginx，端口转发

```
# 若在minikube中，直接访问

# 若在物理主机上，要做端口转发
    kubectl port-forward nginx 8080:80
```

> (6)删除pod

```
kubectl delete -f pod_nginx.yaml
```

`小结`：通过Minikube，我们使用kubectl操作单节点的K8S，而且也能感受到pod的创建和删除，包括pod中对应的容器，一切才刚刚开始，具体细节咱们先不聊，后面慢慢说。