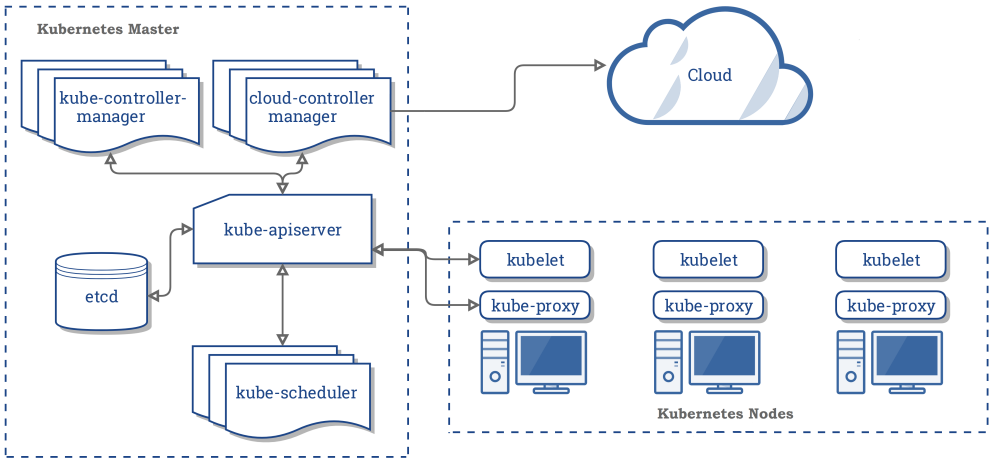
kubernetes 概览

`k8s主要是对容器进行编排/管理，被誉为云端操作系统`

kubernetes组件及运行原理

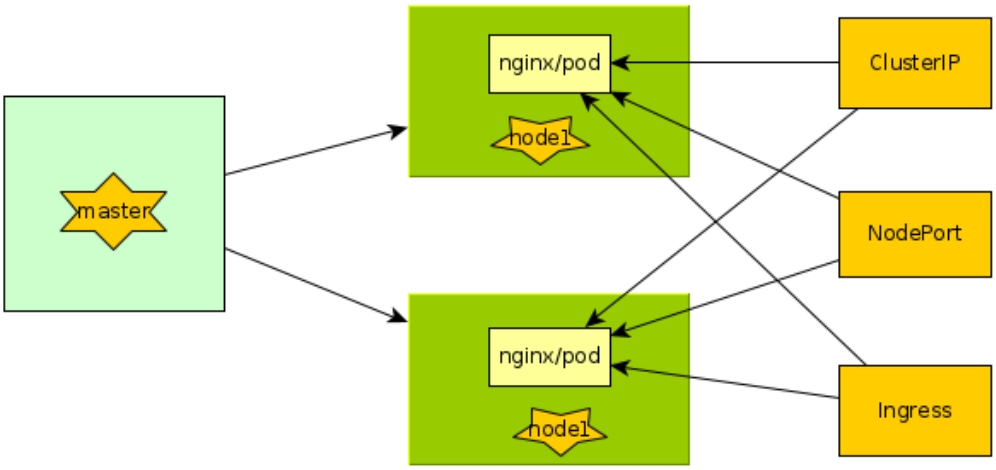
kubernetes集群：至少由一台master节点机台和至少一台node节点机台(真正干活的机台)组成。



具体组件的介绍写在：../k8s概览.emmx

一个实例阐明k8s运行原理

部署一个nginx web应用到k8s集群中，并实现三种访问方式(clusterIP/NodePort/Ingress)。



- 先准备镜像：reg.swharbor.com/tidc/nginx:v-ping并上传到harbor镜像仓库(私有仓库)
- 按照上图的例子，须建立一个Deployment(用来申明Pod的个数)，nginx的个数为2

  定义deployment.yaml资源文件来描述nginx的deployment：

```
apiVersion: apps/v1              #api 版本
kind: Deployment                 #资源种类
```

```
metadata:
  name: nginx-deployment-show   #资源的名称
spec:
  replicas: 2                   #设定Pod副本的数目
  selector:
    matchLabels:
      app: show                 #选择管理pod的标签
  template:                     #期望的模板
    metadata:
      labels:
        app: show               #pod的label标签
    spec:                       #pod期望的状态
      containers:
      - name: deploy-nginx-show #容器的名字
        image: reg.swharbor.com/tidc/nginx:v-ping    #pod中所用镜像的名字
        command: ["nginx","-g","daemon off;"]        #运行容器执行的命令
        ports:
        - containerPort: 80                          #容器释放的端口
```

kubelet 命令创建资源

```
[root@k8s-master01 show]# kubectl apply -f deployment.yaml
deployment.apps/nginx-deployment-show created
```

执行如上命令之后，会创建一个deployment资源与两个Pod资源：

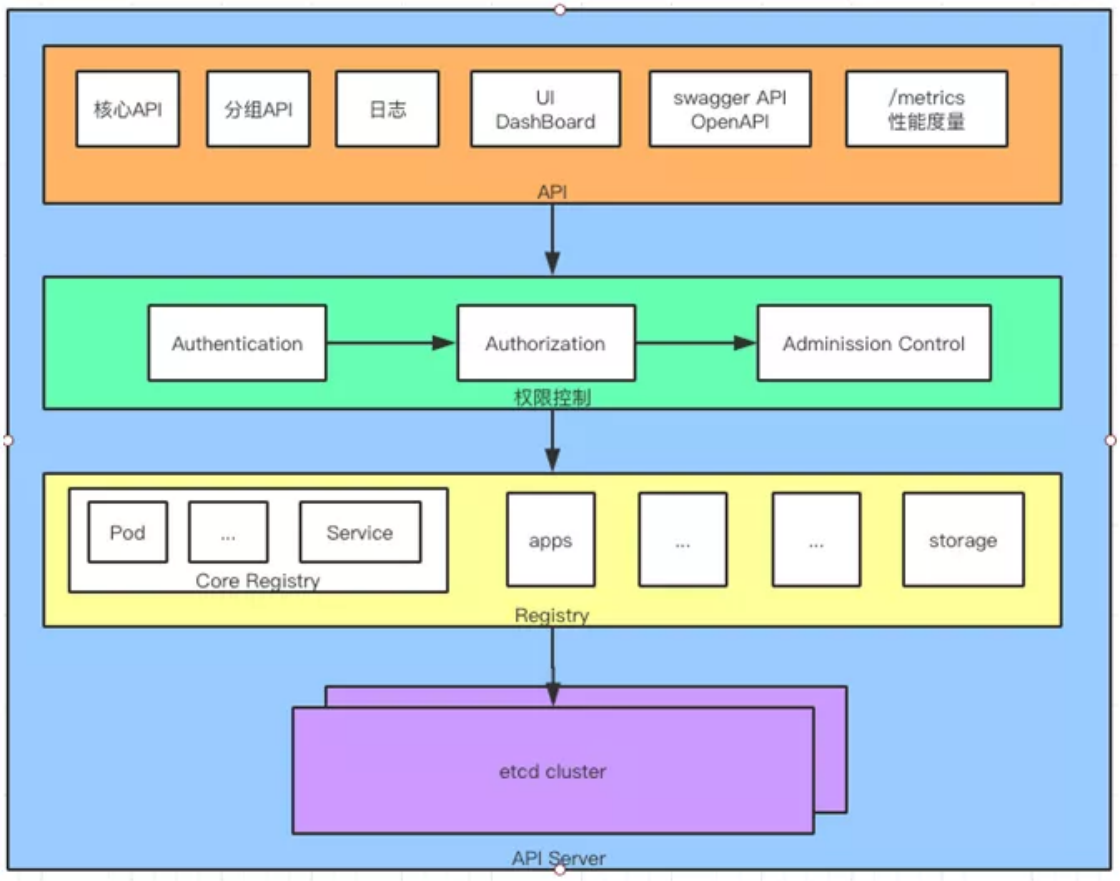
```
[root@k8s-master01 show]# kubectl get deployment
NAME                   READY   UP-TO-DATE   AVAILABLE   AGE
nginx-deployment-show   2/2     2            2           19s

[root@k8s-master01 show]# kubectl get pod -o wide
NAME                                    READY   STATUS    RESTARTS   AGE     IP
           NODE         NOMINATED NODE   READINESS GATES
nginx-deployment-show-8656b7958b-spx2q   1/1     Running   0          122m
10.244.2.64   k8s-node02   <none>           <none>
nginx-deployment-show-8656b7958b-wbtz9   1/1     Running   0          122m
10.244.1.92   k8s-node01   <none>           <none>
```

下面让我们探究一下这条命令完成的背后，组件究竟做了哪些工作。

api-server的四层架构

先探究一下api-server(集群内模块之间数据交换的枢纽)的架构。

- **API 层：**主要以 REST 方式提供各种 API 接口，针对 Kubernetes 资源对象的 CRUD 和 Watch 等主要 API，还有健康检查、UI、日志、性能指标等运维监控相关的 API。
- **访问控制层：**负责身份鉴权，核准用户对资源的访问权限，设置访问逻辑（Admission Control）。
- **注册表层：**选择要访问的资源对象。PS：Kubernetes 把所有资源对象都保存在注册表（Registry）中，例如：Pod，Service，Deployment 等等。
- **etcd 数据库：**保存创建副本的信息。用来持久化 Kubernetes 资源对象的 Key-Value 数据库。



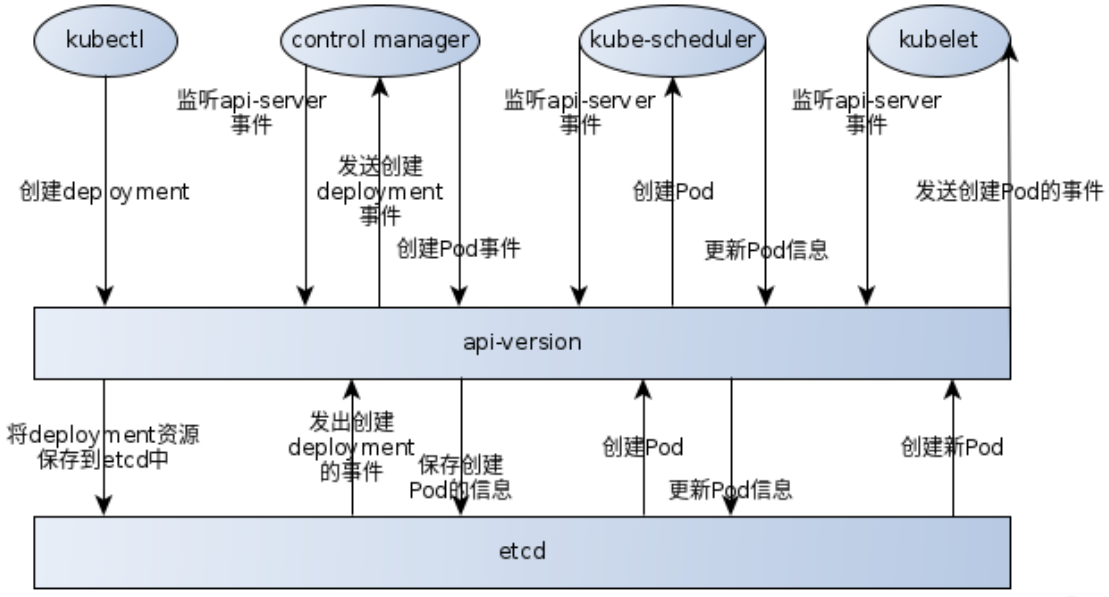
kubectl命令运行时，先通过api-server的API层调用对应的RESTAPI方法。

之后会到达权限控制层，Authentication会获取用户信息，Authorization会获取此用户的权限信息，AdmissionControl 中可配置权限认证插件，通过插件来检查请求约束。这一层主要用于实现安全策略。

接着到达Registry层，这一层会解析deployment.yaml文件中所包含的资源，然后将node/Pod/container等一系列资源保存到etcd中，用于持久化保存k8s资源对象。

各组件之间的通信(监听接口)

Pod的创建并不是由api-server创建，而是control manager/kube-scheduler/kubelet三者联合创建并维持资源的状态的，下图为创建资源时组件之间的通信：



k8s就是使用上图所示的List-Watch(监听)的机制保持数据同步的。

service与kube-proxy

至此，Pod已经创建完成。但我们创建的nginx服务如何被外部访问，亦或集群内部访问呢。

现在我们创建三种类型的svc，提供集群内部/集群外部(IP)/集群外部(域名)三种访问方式：

- clusterIP

  service-clusterIP.yaml：

```
apiVersion: v1      #api 版本
kind: Service       #资源类型
metadata:
  name: svc-show
spec:
  ports:
  - port: 80        #svc暴露的端口
    targetPort: 80  #访问的目标端口，指pod的端口
    protocol: TCP   #这个端口的IP协议，支持TCP/UDP/SCTP
  selector:         #根据标签选择确定SVC的范围
    app: show       #label标签
```

```
#运行如下命令创建SVC
[root@k8s-master01 show]# kubectl create -f service-clusterIP.yaml
service/svc-show created

[root@k8s-master01 show]# kubectl get svc
NAME         TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.96.0.1        <none>        443/TCP   11d
svc-show     ClusterIP   10.109.139.230   <none>        80/TCP    71s
```

  创建完成之后，可以在集群内部通过10.109.139.230访问服务：

```
[root@k8s-master01 show]# curl 10.109.139.230
<title>Welcome to nginx!</title>
```

- NodePort

  service-nodeport.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: nodeport-show
spec:
  type: NodePort
  ports:
    - name: http
      port: 80
      targetPort: 80
      protocol: TCP
  selector:
    app: show
```

```
#运行如下命令创建SVC
[root@k8s-master01 show]# kubectl create -f service-nodeport.yaml
service/nodeport-show created

[root@k8s-master01 show]# kubectl get svc
NAME            TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)
AGE
kubernetes      ClusterIP   10.96.0.1        <none>        443/TCP
11d
nodeport-show   NodePort    10.100.116.39    <none>        80:30423/TCP   9s
svc-show        ClusterIP   10.109.139.230   <none>        80/TCP
14m

#可以通过Node的IP加上端口访问
[root@k8s-master01 show]# curl 10.41.95.99:30423
<title>Welcome to nginx!</title>
```

- Ingress

  ingress.yaml：

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: ingress-show
spec:
  rules:                        #ingress 的规则
  - host: www.ingress.show.com  #设定访问的域名
    http:
      paths:
      - path: /                 #匹配的是当前域名的根
        backend:
          serviceName: svc-show #匹配SVC的名字
          servicePort: 80       #SVC暴露的端口
```

```
#先安装ingress并让ingress以 Nodeport的方式运行
[root@k8s-master01 show]# kubectl get svc -n ingress-nginx
NAME            TYPE       CLUSTER-IP    EXTERNAL-IP   PORT(S)
       AGE
ingress-nginx   NodePort   10.97.89.38   <none>
 80:30913/TCP,443:32063/TCP   6d23h

[root@k8s-master01 show]# kubectl create -f ingress.yaml
ingress.extensions/ingress-show created
[root@k8s-master01 show]# kubectl get ingress
NAME           HOSTS                 ADDRESS   PORTS   AGE
ingress-show   www.ingress.show.com             80      12s

[root@k8s-master01 show]# curl www.ingress.show.com:30913
<title>Welcome to nginx!</title>
```

在 Kubernetes 集群的每个 Node 上都会运行一个 kube-proxy 服务进程，我们可以把这个进程看作 Service 的负载均衡器，其核心功能是将到 Service 的请求转发到后端的多个 Pod 上。

此外，Service 的 Cluster-IP 与 NodePort 是 kube-proxy 服务通过 iptables 的 NAT 转换实现的。kube-proxy 在运行过程中动态创建与 Service 相关的 iptables 规则。

由于 iptables 机制针对的是本地的 kube-proxy 端口，所以在每个 Node 上都要运行 kube-proxy 组件。

因此在 Kubernetes 集群内部，可以在任意 Node 上发起对 Service 的访问请求。