k8s：Kubernetes
架构图：

核心组件：
etcd：保存了整个集群的状态
apiserver：提供了资源操作的唯一入口，并提供认证、授权、访问控制、API 注册和发现等机制
controller manager：负责维护集群的状态，比如故障检测、自动扩展、滚动更新等
scheduler：负责资源的调度，按照预定的调度策略将 Pod 调度到相应的机器上
kubelet：负责维护容器的生命周期，同时也负责 Volume（CVI）和网络（CNI）的管理
Container runtime：负责镜像管理以及 Pod 和容器的真正运行（CRI）
kube-proxy：负责为 Service 提供 cluster 内部的服务发现和负载均衡
POD：k8s 调度的最小单位
RC(Replication Controller)：POD 应答控件
RS(Replica Set)：RC 集合
Deployment：启动 RS
SVC(Service)：核心的资源对象
Kubeadmin：集成启动命令
流程：外部通过 UI 或者命令行链接 API,由 API 提交给 master,再由 master 将需求转发给 node 执行
分层架构：
核心层：Kubernetes 最核心的功能，对外提供 API 构建高层的应用，对内提供插件式应用执行环境
应用层：部署（无状态应用、有状态应用、批处理任务、集群应用等）和路由（服务发现、DNS 解析等）
管理层：系统度量（如基础设施、容器和网络的度量），自动化（如自动扩展、动态 Provision 等）以及策略管理（RBAC、Quota、PSP、NetworkPolicy 等）
接口层：kubectl 命令行工具、客户端 SDK 以及集群联邦
生态系统：在接口层之上的庞大容器集群管理调度的生态系统
Kubernetes 外部：日志、监控、配置管理、CI、CD、Workflow、FaaS、OTS 应用、ChatOps 等
Kubernetes 内部：CRI、CNI、CVI、镜像仓库、Cloud Provider、集群自身的配置和管理等
ETCD：
开源的分布式键值对存储工具。在每个 coreos 节点上面运行的 etcd，共同组建了 coreos 集群的共享数据总线。
保证 coreos 集群的稳定，可靠。
当集群网络出现动荡，或者当前 master 节点出现异常时，etcd 可以优雅的进行 master 节点的选举工作，同时恢复集群中损失的数据。
分布在各个 coreos 节点中的 app，都可以自由的访问到 etcd 中的数据。
功能：
简单可靠，API 丰富(支持 http，jason)
支持客户端通过 SSL 认证，保证安全性
每个实例可以支持每秒 1000 次写操作
基于 RAFT 协议完成分布式操作

通过 http 轮询，监听网络变化
FLEET：
管理 coreos 和部署 app 的工具。
Fleet 可以把整个 coreos 集群当做一台节点来处理。将应用都封装成轻量级的服务，这些服务很容易在集群中进行管理和部署。
功能：
在当前 coreos 集群中随机部署 docker container
在集群中跨主机进行服务分发
负责维护集群中的服务实例，当服务实例异常时，重新进行任务调度来恢复服务
发现集群中的各个节点
自动 SSH 到其它节点来执行 job
最低配置：2CPU+2G 内存
环境准备：
1.关闭虚拟内存：swapoff -a
2.查看/etc/fstab 是否有虚拟内存的配置
3.在内核中配置网桥，要求 iptables 对 bridge 的数据进行处理
net.bridge.bridge-nf-call-iptables：二层的网桥在转发包时也会被 iptables 的 FORWARD 规则过滤
net.bridge.bridge-nf-call-ip6tables：同上，IPv6 版本
vm.swappiness：配置虚拟内存
[root@client ~]# vim /etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
net.ipv4.ip_forward = 1
vm.swappiness=0
[root@client ~]# sysctl --system
4.配置 docker 和 k8s 的 yum 源
[root@client ~]# cat > /etc/yum.repos.d/kubernetes.repo << EOF
[kubernetes]
name=Kubernetes
baseurl=http://mirrors.aliyun.com/kubernetes/yum/repos/kubernetes-el7-x86_64/
enabled=1
gpgcheck=1
gpgkey=http://mirrors.aliyun.com/kubernetes/yum/doc/yum-key.gpg
        http://mirrors.aliyun.com/kubernetes/yum/doc/rpm-package-key.gpg
EOF
[root@client ~]# wget http://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo
[root@client ~]# mv docker-ce.repo /etc/yum.repos.d/
5.安装 docker，kubeadm，kubelet，kubectl。当前版本为 1.12.2
[root@client ~]# yum install -y docker-ce kubelet kubeadm kubectl kubernetes-cni
[root@client ~]# systemctl start docker
[root@client ~]# systemctl start kubelet
6.kubernetes 集群不允许开启 swap，所以我们需要忽略这个错误
[root@client ~]# vim /etc/sysconfig/kubelet

KUBELET_EXTRA_ARGS="--fail-swap-on=false"

7.使用 yaml 自动配置 kubernetes

apiVersion 可通过查看：https://github.com/kubernetes/kubernetes/tree/release-1.12/cmd/kubeadm/app/apis/kubeadm

```
[root@client ~]# vim kubeadm.yaml
apiVersion: kubeadm.k8s.io/v1alpha2
kind: MasterConfiguration
controllerManagerExtraArgs:
  horizontal-pod-autoscaler-use-rest-clients: "true"
  horizontal-pod-autoscaler-sync-period: "10s"
  node-monitor-grace-period: "10s"
apiServerExtraArgs:
  runtime-config: "api/all=true"
kubernetesVersion: "v1.12.2"
```

8.根据错误提示确定镜像和版本

```
[root@client ~]# kubeadm init --config kubeadm.yaml
[preflight] Some fatal errors occurred:
      [ERROR ImagePull]: failed to pull image k8s.gcr.io/kube-apiserver:v1.12.2: ...
      [ERROR ImagePull]: failed to pull image k8s.gcr.io/kube-controller-manager:v1.12.2: ...
      [ERROR ImagePull]: failed to pull image k8s.gcr.io/kube-scheduler:v1.12.2: ...
      [ERROR ImagePull]: failed to pull image k8s.gcr.io/kube-proxy:v1.12.2: ...
      [ERROR ImagePull]: failed to pull image k8s.gcr.io/etcd:3.2.24: ...
      [ERROR ImagePull]: failed to pull image k8s.gcr.io/coredns:1.2.2: ...
      [ERROR ImagePull]: failed to pull image k8s.gcr.io/pause:3.1: ...
```

9.kubernetes 的镜像托管在 google 云上，无法直接下载。从 https://github.com/anjia0532/gcr.io_mirror 拉取镜像

```
[root@client ~]# vim pull.sh
#!/bin/bash
images=(kube-proxy:v1.12.2 kube-scheduler:v1.12.2 kube-controller-manager:v1.12.2
kube-apiserver:v1.12.2 etcd:3.2.24 coredns:1.2.2 pause:3.1 )
for imageName in ${images[@]}
do
docker pull anjia0532/google-containers.$imageName
docker tag anjia0532/google-containers.$imageName k8s.gcr.io/$imageName
docker rmi anjia0532/google-containers.$imageName
done
```

10.重新执行 yaml，完成 Kubernetes Master 的部署，这个过程需要几分钟

```
[root@client ~]# kubeadm init --config kubeadm.yaml
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
...
kubeadm join 172.16.186.96:6443 --token 8qxmb3.yvrj42ib39a6h1at
--discovery-token-ca-cert-hash
sha256:fe33ff9b62cf7cb6adac85f6bb9247d4bc4e7078c3f1d3065afa78d6531390f5
```

11.配置 kubectl 与 apiserver 的认证

```
[root@client ~]#   mkdir -p $HOME/.kube
[root@client ~]#   sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
[root@client ~]#   sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

12.检查健康状态

```
[root@client ~]# kubectl get cs
```

13.查看节点状态

```
[root@client ~]# kubectl get nodes
```

14.部署网络插件 Weave 并验证

```
[root@client ~]# kubectl apply -f https://git.io/weave-kube-1.6
[root@client ~]# kubectl get pods -n kube-system
NAME                                READY     STATUS       RESTARTS     AGE
coredns-576cbf47c7-kwwk8            1/1       Running      0            42m
coredns-576cbf47c7-p62rv            1/1       Running      0            42m
etcd-client                         1/1       Running      0            41m
kube-apiserver-client               1/1       Running      0            41m
kube-controller-manager-client      1/1       Running      0            41m
kube-proxy-pvsg7                    1/1       Running      0            42m
kube-scheduler-client               1/1       Running      0            41m
weave-net-jwqsk                     2/2       Running      0            78s
```

15.使用 master 调度 pod。注：由于是单机测试，所以没有 client

```
[root@client ~]# kubectl taint nodes --all node-role.kubernetes.io/master-
node/client untainted
```

16.安装可视化插件镜像

```
[root@client ~]# wget
https://raw.githubusercontent.com/kubernetes/dashboard/master/src/deploy/recommended/ku
bernetes-dashboard.yaml
[root@client ~]# vim kubernetes-dashboard.yaml
...
kind: Deployment
apiVersion: apps/v1beta2
...
spec:
  ...
  template:
    ...
    spec:
      containers:
image: k8s.gcr.io/kubernetes-dashboard-amd64:v1.10.0
```

...
#修改最后几行，可以直接使用 token 认证进入
kind: Service
apiVersion: v1
metadata:
  labels:
    k8s-app: kubernetes-dashboard
  name: kubernetes-dashboard
  namespace: kube-system
spec:
  type: NodePort   #添加 Service 的 type 为 NodePort
  ports:
    - port: 443
      targetPort: 8443
      nodePort: 30001 #添加映射到 docker 的端口,k8s 只支持 30000-32767 之间的端口
  selector:
    k8s-app: kubernetes-dashboard
[root@client ~]# docker pull anjia0532/google-containers.kubernetes-dashboard-amd64:v1.10.0
[root@client ~]# docker tag anjia0532/google-containers.kubernetes-dashboard-amd64:v1.10.0 k8s.gcr.io/kubernetes-dashboard-amd64:v1.10.0
[root@client ~]# docker rmi anjia0532/google-containers.kubernetes-dashboard-amd64:v1.10.0
[root@client ~]# kubectl apply -f kubernetes-dashboard.yaml
17.验证 Dashboard 是否正常启动
[root@client ~]# kubectl get pods -n kube-system
...
weave-net-jwqsk   2/2   Running   0   64m
...
18.安装容器存储插件镜像
[root@client ~]# kubectl apply -f https://raw.githubusercontent.com/rook/rook/master/cluster/examples/kubernetes/ceph/operator.yaml
[root@client ~]# kubectl apply -f https://raw.githubusercontent.com/rook/rook/master/cluster/examples/kubernetes/ceph/cluster.yaml
19.查看容器存储插件安装情况。
注：由于通过 deployment 启动 pod，拉镜像需要时间，在第 18 步完成后，需要一段时间后才能查看到结果
[root@client ~]# kubectl get pods -n rook-ceph-system
[root@client ~]# kubectl get pods -n rook-ceph
20.开启服务
[root@client ~]# nohup kubectl proxy --address='0.0.0.0' --accept-hosts='^*$' --disable-filter=true &
21.获取 token 命令
[root@client ~]# kubectl -n kube-system describe $(kubectl -n kube-system get secret -n

kube-system -o name | grep namespace) | grep token
22.网页访问 dashboard
[root@client ~]# curl 127.0.0.1:30001
查看命令：
查看全部节点：kubectl get pods --all-namespaces
查看 pods：kubectl describe pod -n kube-system
查看具体问题：kubectl describe pod kubernetes-dashboard-77fd78f978-m5stn -n kube-system
或 kubectl describe pod kubernetes-dashboard -n kube-system