# Kubernetes基础

K8s 谷歌开源的，borg。 K8s一开始的贡献者都是borg开发者。

官方文档: <https://kubernetes.io/>

Kubelet：负责监听节点上Pod的状态，同时负责上报节点和节点上面的Pod的状态。

负责与Master节点通讯，管理该节点上的Pod。

Kube-proxy：负责Pod之间的通信和负载均衡。将指定的流量分发到后端正确的机器上。

Iptables、IPVS

Iptables：监听Master节点增加和删除service以及endpoint的消息，对于每一个service，他都会建议一个iptables规则，将service的clusterIP代理到后端对应的Pod上，iptables的流量是在内核态进行的。性能稍高，也比较可靠。

Ipvs：监听Master节点增加和删除service以及endpoint的消息，调用netlink接口创建相应的IPVS规则。访问service地址时，通过ipvs规则，将流量转发至相对的Pod上。

目前新版本默认是用的是IPVS模式，

查看kube-proxy的代理模式：

[root@k8s-master01 ~]# curl 127.0.0.1:10249/proxyMode

Ipvs

主机的30000端口—》到了k8s service的443端口—》容器的8443端口

ETCD：CoreOS开发的，键值数据库，zookeeper，保存的是一些的集群的信息，一般再生产环境中高可用部署，奇数个，

Kube-apiserver：控制中枢，各个模块之间的信息交互都需要经过kube-apiserver。他也是集群管理、资源配额，也是整个集群安全机制的入口。

Controller-manager：集群的状态管理器。保证Pod或其他资源达到期望值。这个控制器也是和APIServer进行通信，在需要的时候创建、更新或删除它所管理的资源。

Scheduler：集群的调度中心。他会根据指定的一系列条件，选择一个最佳的节点，然后将Pod部署上去。监听Node节点上的Pod，NodeName为空。

CoreDNS：可以让Pod解析k8s集群内部的service。

Calico：符合CNI标准的一个网络插件，他会负责给Pod分配一个不会重复的IP。

Flannel：

Calico Flannel 区别：

纯三层网络 vxlan或者UDP

部署简单、性能强大。 手动配置。

高 一般

私有网络，支持网络策略 不支持网络策略

Docker是一个容器，里面运行的是我们的应用程序，java、nodejs、Go，他们都是前台运行的。

Pod，里面是一个或多个容器的集合，共享存储、IP地址、Port。

Pod：一个是nginx 的容器，一个是PHP-FPM的容器。

Nginx是80端口，php是9000。127.0.0.1:9000,127.0.0.1:80

Pod里面就只有一个Docker。Java 、python、Go。

Nginx+Php，同一个Pod。两个docker是比较好的。

Java：控制台，Java的日志自定义一个目录，文件。Pod附加一个filebeat的容器，sidecar。

Istio：注入Istio容器管理Pod的出入流量。

Pause容器：Pod的父容器，用于负责僵尸进程的回收管理。

Pod：在生产环境中，很少直接部署，我们一般都是用Deployment、daemonset、statuefulset

定义一个Pod：

apiVersion: v1 # apps/v1

kind: Pod #Deployment/Service

metadata:

annotations: # 注释信息

cni.projectcalico.org/podIP: 10.244.122.129/32

cni.projectcalico.org/podIPs: 10.244.122.129/32

name: busybox # Pod的名字

namespace: default # 所在命名空间

spec:

containers:

- command: # 容器的命令，docker里面的entrypoint

- sleep

- "3600"

image: busybox:1.28 # 镜像地址，不写域名默认的是官方的。Hub.docker.com

imagePullPolicy: IfNotPresent # 如果服务器没有这个镜像，就去拉，如果有就不在拉取，Always：总是拉取。Never：不管镜像有没有，都不拉。

name: busybox # 容器的名字

resources: {} # 资源限制的

volumeMounts: # 挂载路径，

- mountPath: /var/run/secrets/kubernetes.io/serviceaccount

name: default-token-nr9s4

readOnly: true

dnsPolicy: ClusterFirst #Deployment再讲

nodeName: k8s-master02 # 当前Pod所在的节点，一般不指定

restartPolicy: Always # 重启策略，容器失效时，就重启这个容器。默认的策略。 OnFailure：容器不为0的状态码终止，自动重启该容器。Never：无论何种状态都不重启。

securityContext: {} # 内核配置，高权限

tolerations: #容忍配置

- effect: NoExecute

key: node.kubernetes.io/not-ready

operator: Exists

tolerationSeconds: 300

- effect: NoExecute

key: node.kubernetes.io/unreachable

operator: Exists

tolerationSeconds: 300

volumes: # pv pvc

- name: default-token-nr9s4

secret:

defaultMode: 420

secretName: default-token-nr9s4

RC：Replication Controller的缩写，复制控制器，现在已经不用了。

RS：ReplicaSet，支持基于集合的标签选择器的下一代RC。和Deployment协调，创建、删除和更新Pod。

定义一个Deployment：

apiVersion: apps/v1 #1.16以上，extensions/v1beta2已经被废弃，

kind: Deployment

metadata:

annotations: # 注释

deployment.kubernetes.io/revision: "1"

labels: # 定义这个Deployment的标签。

k8s-app: kube-dns

kubernetes.io/name: CoreDNS

name: coredns

namespace: kube-system

spec:

replicas: 1 #Pod的副本数

revisionHistoryLimit: 10 #保留历史版本的个数

selector: #管理的是哪些Pod

matchLabels:

k8s-app: kube-dns

strategy: # 更新策略

rollingUpdate:

maxSurge: 25% # 最大启用的个数

maxUnavailable: 1 # 最大不可用

type: RollingUpdate

template:

metadata:

labels:

k8s-app: kube-dns

spec:

affinity:

podAntiAffinity:

requiredDuringSchedulingIgnoredDuringExecution:

- labelSelector:

matchExpressions:

- key: k8s-app

operator: In

values:

- kube-dns

topologyKey: kubernetes.io/hostname

containers:

- args: # command-ENTRYPOINT， args-CMD

- -conf

- /etc/coredns/Corefile

image: coredns/coredns:1.6.6

tty: true

imagePullPolicy: IfNotPresent

livenessProbe: # Pod的健康检查，过滤你的进程有没有起来

failureThreshold: 5 失败多少次，kubelet可以重启该Pod

httpGet:# exec：执行一个命令，tcpSocket，检查端口。

path: /health #接口健康检查的地址

port: 8080 #端口 127.0.0.1:80/health

scheme: HTTP #协议

httpHeaders:

* name: xxx

value: bbb

initialDelaySeconds: 30

periodSeconds: 10 # 检查的间隔

successThreshold: 1 # 成功的次数

timeoutSeconds: 5 # 1-5

name: coredns

env:

* name: a

value: b

ports:

- containerPort: 53

name: dns

protocol: UDP

- containerPort: 53

name: dns-tcp

protocol: TCP

- containerPort: 9153

name: metrics

protocol: TCP

readinessProbe: # 检查容器进程的状态，Ready，程序初始化完成以后，

failureThreshold: 3

httpGet:

path: /ready

port: 8181

scheme: HTTP

periodSeconds: 10

successThreshold: 1

timeoutSeconds: 1

resources: #资源限制

limits: # 最大的使用量

memory: 170Mi #1Gi

cpu: 1 #1000m

requests: # 资源请求量

cpu: 100m

memory: 70Mi

securityContext:

allowPrivilegeEscalation: false #

capabilities:

add:

- NET\_BIND\_SERVICE

drop:

- all

readOnlyRootFilesystem: true

volumeMounts:

- mountPath: /etc/coredns

name: config-volume

readOnly: true

dnsPolicy: Default # Pod从宿主机上继承解析，None，忽略Pod DNS配置，dnsConfig，ClusterFirst：转发到宿主机去进行解析。ClusterFirstWithHostNet：Ingress，hostNetwork: true,

nodeSelector:

beta.kubernetes.io/os: linux

restartPolicy: Always

securityContext: {}

tolerations:

- key: CriticalAddonsOnly

operator: Exists

volumes:

- configMap:

defaultMode: 420

items:

- key: Corefile

path: Corefile

name: coredns

name: config-volume

Exec:

exec:

command:

* wget 127.0.0.1:80/health
* ps aux |grep xxx

TcpSocket:

tcpSocket:

port: 80

host: 127.0.0.1

Deployment：

1. Deployment 无状态的应用。
2. 需要支持弹性扩缩容

创建一个Deployment：

NAME READY UP-TO-DATE AVAILABLE AGE

demo-nginx 0/2 2 0 6s

READY Pod起来的个数

UP-TO-DATE： 显示已达到期望状态的副本数

AVAILABLE ： 可用的副本数

AGE：运行的时间

查看不是Running状态的Pod的状态：

kubectl describe pod xxx

Running： Pod起来了，但是不代表容器里面的进程起来了。需要Ready字段

ImagePullBackOff： 镜像无法下载，网络原因，镜像不存在

Crash：镜像已经下载完成，但是容器的里面的进程无法启动。Logs describe

Pending：Pod已经被k8s系统接收，describe。

Succeeded：所有的容器执行成功，并终止。

Failed：所有的容器都已经被禁止，但是至少有一个容器是以失败的方式终止的。

Unknown：宿主机的问题。

SysctlForbidden：Pod设置了内核参数，但是宿主机不支持该项内核配置，describe。

更新Deployment：

1. 直接更改创建deployment时的yaml文件，然后执行kubectl replace -f example-deployment.yaml -n kube-public进行更改
2. 通过命令行更改[root@k8s-master01 ~]# kubectl set image deploy demo-nginx nginx=nginx -n kube-public

deployment.apps/demo-nginx image updated

1. 直接通过edit进行更新deployment
   1. [root@k8s-master01 ~]# kubectl edit deploy demo-nginx
   2. Edit cancelled, no changes made.
   3. [root@k8s-master01 ~]# kubectl get po -n kube-public
   4. NAME READY STATUS RESTARTS AGE
   5. demo-nginx-84cbf96454-74v8q 1/1 Running 0 48s
   6. [root@k8s-master01 ~]# kubectl edit deploy demo-nginx -n kube-public
   7. deployment.apps/demo-nginx edited
   8. [root@k8s-master01 ~]# kubectl get deploy demo-nginx -nkube-public -w
   9. NAME READY UP-TO-DATE AVAILABLE AGE
   10. demo-nginx 1/5 5 1 3h37m
   11. demo-nginx 2/5 5 2 3h37m
   12. demo-nginx 3/5 5 3 3h37m
   13. demo-nginx 4/5 5 4 3h38m
2. 查看更新状态[root@k8s-master01 ~]# kubectl rollout status deploy demo-nginx -n kube-public

Waiting for deployment "demo-nginx" rollout to finish: 4 of 8 updated replicas are available...

回滚deployment：

[root@k8s-master01 ~]# kubectl -n kube-public set image deploy demo-nginx nginx=nginx:4444 --record

deployment.apps/demo-nginx image updated

查看历史版本

[root@k8s-master01 ~]# kubectl rollout history deploy demo-nginx -n kube-public

deployment.apps/demo-nginx

REVISION CHANGE-CAUSE

1 <none>

2 <none>

3 <none>

1. kubectl set image deploy demo-nginx nginx=nginx:4444 --namespace=kube-public --record=true

回滚到上一个版本

[root@k8s-master01 ~]# kubectl rollout undo deploy demo-nginx -n kube-public

deployment.apps/demo-nginx rolled back

查看指定版本的详细信息

[root@k8s-master01 ~]# kubectl rollout history deploy demo-nginx -n kube-public --revision=8

deployment.apps/demo-nginx with revision #8

Pod Template:

Labels: app=demo-nginx

pod-template-hash=84cbf96454

Annotations: kubernetes.io/change-cause: kubectl set image deploy demo-nginx nginx=nginx --namespace=kube-public --record=true

Containers:

nginx:

Image: nginx

Port: 80/TCP

Host Port: 0/TCP

Limits:

cpu: 100m

memory: 270Mi

Requests:

cpu: 100m

memory: 70Mi

Liveness: tcp-socket :80 delay=10s timeout=5s period=10s #success=1 #failure=2

Readiness: http-get http://:80/ delay=20s timeout=1s period=10s #success=1 #failure=3

Environment: <none>

Mounts: <none>

Volumes: <none>

回滚到指定版本

[root@k8s-master01 ~]# kubectl rollout undo deploy demo-nginx -n kube-public --to-revision=8

deployment.apps/demo-nginx rolled back

暂停一个deployment的更新：

[root@k8s-master01 ~]# kubectl rollout pause deploy demo-nginx -n kube-public

deployment.apps/demo-nginx paused

恢复更新：

[root@k8s-master01 ~]# kubectl rollout resume deploy demo-nginx -n kube-public

deployment.apps/demo-nginx resumed

更新deployment注意事项：

清理策略：spec.revisionHistoryLimit 保留就的ReplicaSet的数量

0：不保留历史记录

更新策略：

rollingUpdate:

maxSurge: 2

maxUnavailable: 1

type: RollingUpdate

Recreate：重建，旧的Pod手动删除后，才会创建新的。

Ready参数：

Spec.minReadySeconds:可选参数

StatefulSet：有状态集，主要用于部署有状态的应用，或者有启动顺序要求的应用程序。

也是管理基于相同容器规范的Pod。

StatefulSet： redis-sentinel 副本数是3

redis-sentinel-0 redis-sentinel-1 redis-sentinel-2。

StatefulSet创建的时候。必须要制定一个Service名称。无头Service Headless Service，没有ClusterIP，Endpoint。

Headless Service：

statefulSetName-{0-N-1}.serviceName.namespace.svc.cluster.local

什么情况下使用StatefulSet：

1. 需要稳定的独一无二的网络标识符
2. 需要持久化数据
3. 需要有序的、优雅的部署和扩展

terminationGracePeriodSeconds 不要设置成0。

创建Pod的流程：

3个副本的sts，按顺序从0创建N-1

删除，反向删除

分段更新：

Partition： 更新大于等于该数字的Pod。

简单实现金丝雀发布/灰度发布。

删除StatefulSet：

级联删除和非级联删除。

DaemonSet：守护进程集，和Linux上的守护进程类似，确保全部（或者部分符合条件的）节点上运行一个指定的Pod。

Ingress、fluentd、Prometheus Node Exporter、Glusterd。

Label：

对系统的任何API对象进行分组，key=value

Selector:

匹配过滤符合条件的资源对象。

Env，project， region，type