法律声明

- 本课件包括：演示文稿，示例，代码，题库，视频和声音等，讲师及小象学院拥有完全知识产权的权利；只限于善意学习者在本课程使用，不得在课程范围外向任何第三方散播。任何其他人或机构不得盗版、复制、仿造其中的创意，我们将保留一切通过法律手段追究违反者的权利。

- 课程详情请咨询
  - 微信公众号：小象
  - 新浪微博：ChinaHadoop

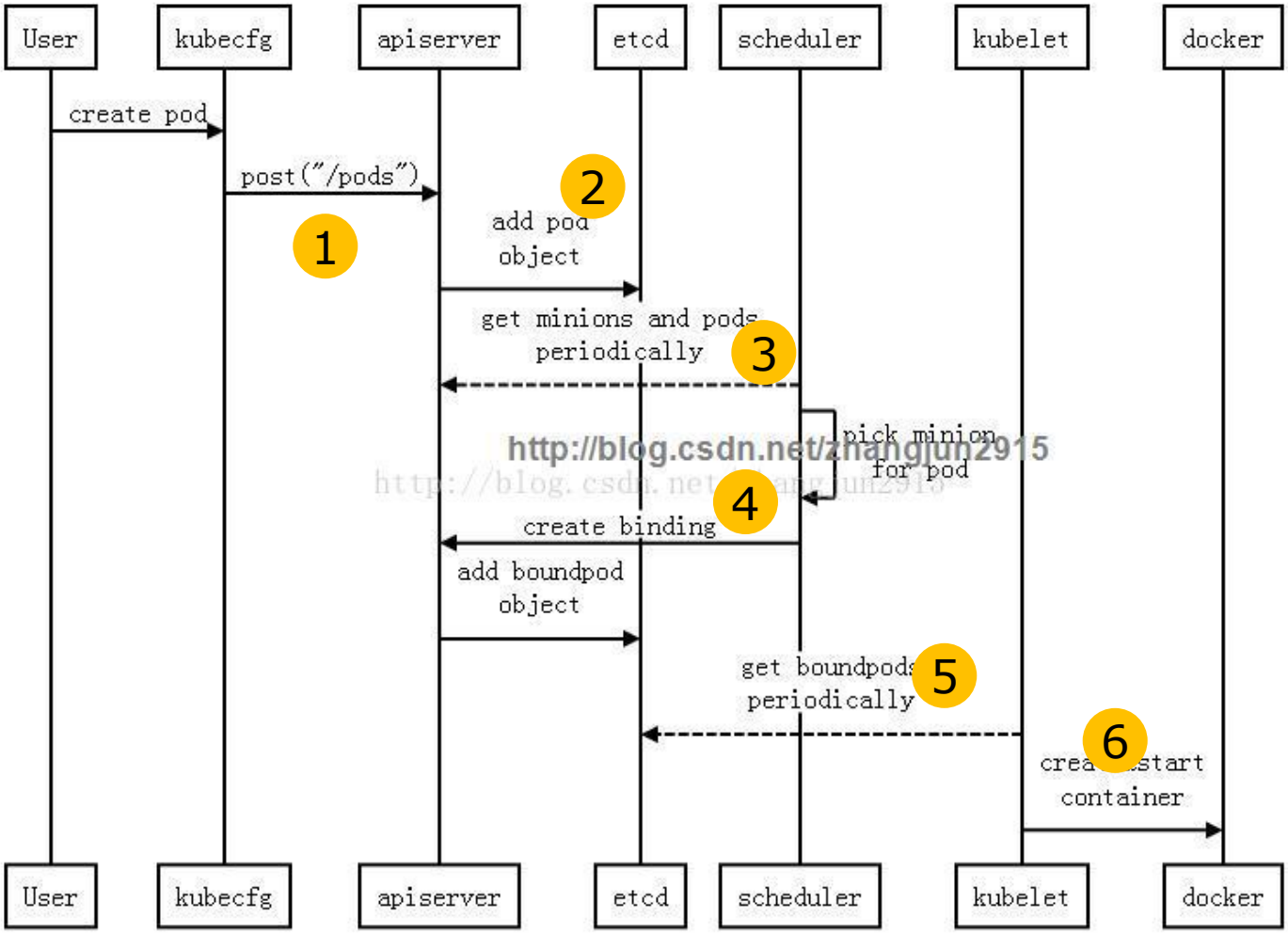
Kubernetes集群搭建及基本操作

目录

1. 业内各种安装方式对比

	安装前准备	适用范围及特点	高可用	安装复杂度
kubeadm	kubectl/kubelet	不对接节点，自己准备节点环境，容易被集成到其他工具链中	HA(aplha)	高
kops	kubectl	对接AWS/GCE/Vmware，帮助你管理虚机	HA	中
minikube	kubectl	单机对接VM	无	低
rancher	rancher	墙内加速，跨云能力	不清楚	中
手工安装	11+组件	任意集群	HA	疯狂

2.当我们谈论创建pod时，我们在谈论什么



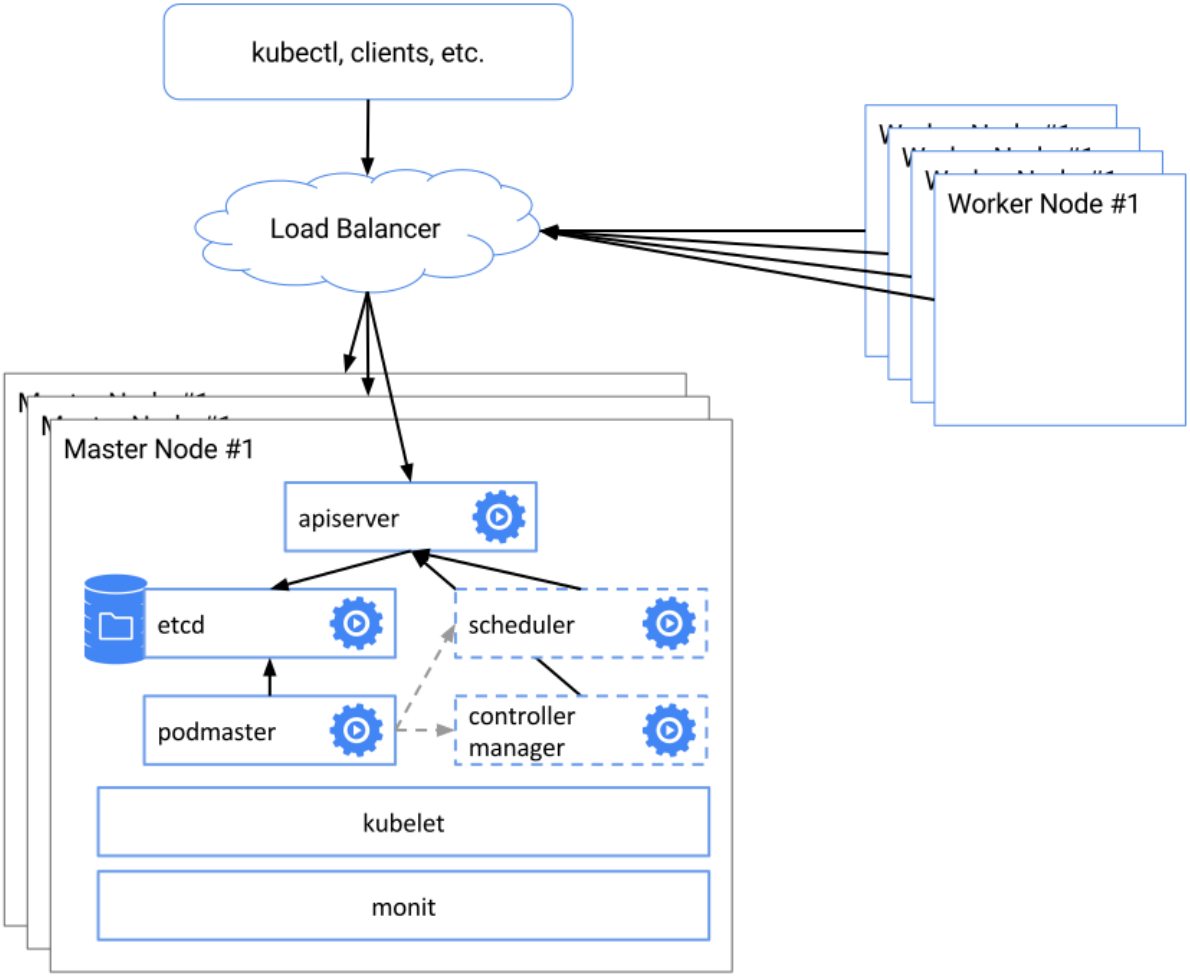
2.当我们谈论创建pod时，我们在谈论什么

1. kubectl提交创建请求，可以通过API Server的Restful API，也可以使用kubectl命令行工具。支持的数据类型包括JSON和YAML。
2. kube-apiserver处理用户请求，存储Pod数据到etcd。
3. kube-scheduler通过API Server查看未绑定的Pod。尝试为Pod分配主机。
   - 过滤主机 (调度预选)：调度器用一组规则过滤掉不符合要求的主机。比如Pod指定了所需要的资源量，那么可用资源比Pod需要的资源量少的主机会被过滤掉。
   - 主机打分(调度优选)：对第一步筛选出的符合要求的主机进行打分，在主机打分阶段，调度器会考虑一些整体优化策略，比如把容一个Replication Controller的副本分布到不同的主机上，使用最低负载的主机等。
4. kube-scheduler选择主机：选择打分最高的主机，进行binding操作，这个操作本质是通过kube-apisever修改Pod的字段，结果存储到etcd中。
5. kubelet根据调度结果执行Pod创建操作： 绑定成功后，pod.spec.nodeName有值了。运行在每个工作节点上的kubelet也会定期与etcd同步pod信息（属于自己这个node的）
6. docker接受到kubelet下发的命令，启动相应容器，至此，一个Pod启动完毕

2.当我们谈论创建pod时，我们在谈论什么

1. kubectl提交创建请求，可以通过API Server的Restful API，也可以使用kubectl命令行工具。支持的数据类型包括JSON和YAML。
2. kube-apiserver处理用户请求，存储Pod数据到etcd。
3. kube-scheduler通过API Server查看未绑定的Pod。尝试为Pod分配主机。
   - 过滤主机 (调度预选)：调度器用一组规则过滤掉不符合要求的主机。比如Pod指定了所需要的资源量，那么可用资源比Pod需要的资源量少的主机会被过滤掉。
   - 主机打分(调度优选)：对第一步筛选出的符合要求的主机进行打分，在主机打分阶段，调度器会考虑一些整体优化策略，比如把容一个Replication Controller的副本分布到不同的主机上，使用最低负载的主机等。
4. kube-scheduler选择主机：选择打分最高的主机，进行binding操作，这个操作本质是通过kube-apisever修改Pod的字段，结果存储到etcd中。
5. kubelet根据调度结果执行Pod创建操作： 绑定成功后，pod.spec.nodeName有值了。运行在每个工作节点上的kubelet也会定期与etcd同步pod信息（属于自己这个node的）
6. docker接受到kubelet下发的命令，启动相应容器，至此，一个Pod启动完毕

3.k8s高可用集群方案介绍



3.k8s高可用集群方案介绍

- Establishing a redundant, reliable data storage layer
  - Clustering etcd
    - Validating your cluster
  - Even more reliable storage
- Replicated API Servers
  - Installing configuration files
  - Starting the API Server
  - Load balancing
  - Endpoint reconciler
- Master elected components
  - Installing configuration files

4. kubectl自动补全

$ source <(kubectl completion bash) #bash下自动补全

$ source <(kubectl completion zsh) # zsh下自动补全

4. kubectl上下文和配置

$ kubectl config view # 显示合并后的 kubeconfig 配置

$ kubectl config current-context # 显示当前的上下文

$ kubectl config use-context my-cluster-name # 设置默认上下文为 my-cluster-name

使用指定的用户名和 namespace 设置上下文

$ kubectl config set-context gce --user=cluster-admin --namespace=foo \

&& kubectl config use-context gce

kubectl config current-context –h 帮助及例子

4. kubectl创建对象-A

```
$ kubectl create -f ./my-manifest.yaml           # 创建资源
$ kubectl create -f ./my1.yaml -f ./my2.yaml     # 使用多个文件创建资源
$ kubectl create -f ./dir                        # 使用目录下的所有清单文件来创建资源
$ kubectl create -f https://git.io/vPieo         # 使用 url 来创建资源
$ kubectl run nginx --image=nginx                # 启动一个 nginx 实例
$ kubectl explain pods                           # 获取 pod 和 svc 的文档
```

4. kubectl创建对象-B

```
# 创建包含几个 key 的 Secret
$ cat <<EOF | kubectl create -f -
apiVersion: v1
kind: Secret
metadata:
  name: mysecret
type: Opaque
data:
  password: $(echo "s33msi4" | base64)
  username: $(echo "jane" | base64)
EOF
```

4. kubectl显示和查找资源-A

```
# Get commands with basic output
$ kubectl get services                          # 列出所有 namespace 中的所有 service
$ kubectl get pods --all-namespaces             # 列出所有 namespace 中的所有 pod
$ kubectl get pods -o wide                      # 列出所有 pod 并显示详细信息
$ kubectl get deployment my-dep                 # 列出指定 deployment
$ kubectl get pods --include-uninitialized      # 列出该 namespace 中的所有 pod 包括未初始化的
```

4. kubectl显示和查找资源-B

使用详细输出来描述命令

$ kubectl describe nodes my-node

$ kubectl describe pods my-pod

$ kubectl get services --sort-by=.metadata.name # List Services Sorted by Name

4. kubectl显示和查找资源-C

```
# 根据重启次数排序列出 pod
$ kubectl get pods --sort-by='.status.containerStatuses[0].restartCount'

# 获取所有具有 app=cassandra 的 pod 中的 version 标签
$ kubectl get pods --selector=app=cassandra rc -o \
  jsonpath='{.items[*].metadata.labels.version}'

# 获取所有节点的 ExternalIP
$ kubectl get nodes -o
jsonpath='{.items[*].status.addresses[?(@.type=="ExternalIP")].address}'
```

4. kubectl显示和查找资源-D

```
# 根据重启次数排序列出 pod
$ kubectl get pods --sort-by='.status.containerStatuses[0].restartCount'

# 获取所有具有 app=cassandra 的 pod 中的 version 标签
$ kubectl get pods --selector=app=cassandra rc -o \
  jsonpath='{.items[*].metadata.labels.version}'

# 获取所有节点的 ExternalIP
$ kubectl get nodes -o
jsonpath='{.items[*].status.addresses[?(@.type=="ExternalIP")].address}'
```

4. kubectl显示和查找资源-E

```
# 列出属于某个 PC 的 Pod 的名字

#  “jq”命令用于转换复杂的 jsonpath，参考 https://stedolan.github.io/jq/

$ sel=${$(kubectl get rc my-rc --output=json | jq -j '.spec.selector | to_entries | .[] |
"\(.key)=\(.value),"')%?}

$ echo $(kubectl get pods --selector=$sel --output=jsonpath={.items..metadata.name})


# 查看哪些节点已就绪

$ JSONPATH='{range .items[*]}{@.metadata.name}:{range
@.status.conditions[*]}{@.type}={@.status};{end}{end}' \

 && kubectl get nodes -o jsonpath="$JSONPATH" | grep "Ready=True"


# 列出当前 Pod 中使用的 Secret

$ kubectl get pods -o json | jq '.items[].spec.containers[].env[]?.valueFrom.secretKeyRef.name'
| grep -v null | sort | uniq
```

4. kubectl编辑资源

```
$ kubectl edit svc/docker-registry                # 编辑名为 docker-registry 的 service
$ KUBE_EDITOR="nano" kubectl edit svc/docker-registry   # 使用其它编辑器
```

4. kubectl Scale 资源

$ kubectl scale --replicas=3 rs/foo # Scale a replicaset named 'foo' to 3

$ kubectl scale --replicas=3 -f foo.yaml # Scale a resource specified in "foo.yaml" to 3

$ kubectl scale --current-replicas=2 --replicas=3 deployment/mysql # If the deployment named mysql's current size is 2, scale mysql to 3

$ kubectl scale --replicas=5 rc/foo rc/bar rc/baz # Scale multiple replication controllers

4. kubectl 删除资源

$ kubectl scale --replicas=3 rs/foo # Scale a replicaset named 'foo' to 3

$ kubectl scale --replicas=3 -f foo.yaml # Scale a resource specified in "foo.yaml" to 3

$ kubectl scale --current-replicas=2 --replicas=3 deployment/mysql # If the deployment named mysql's current size is 2, scale mysql to 3

$ kubectl scale --replicas=5 rc/foo rc/bar rc/baz # Scale multiple replication controllers

4. kubectl与运行中的 Pod 交互

$ kubectl logs my-pod # dump 输出 pod 的日志（stdout）

$ kubectl logs my-pod -c my-container # dump 输出 pod 中容器的日志（stdout，pod 中有多个容器的情况下使用）

$ kubectl logs -f my-pod # 流式输出 pod 的日志（stdout）

$ kubectl logs -f my-pod -c my-container # 流式输出 pod 中容器的日志（stdout，pod 中有多个容器的情况下使用）

$ kubectl run -i --tty busybox --image=busybox -- sh # 交互式 shell 的方式运行 pod

$ kubectl attach my-pod -i # 连接到运行中的容器

$ kubectl port-forward my-pod 5000:6000 # 转发 pod 中的 6000 端口到本地的 5000 端口

$ kubectl exec my-pod -- ls / # 在已存在的容器中执行命令（只有一个容器的情况下）

$ kubectl exec my-pod -c my-container -- ls / # 在已存在的容器中执行命令（pod 中有多个容器的情况下）

$ kubectl top pod POD_NAME --containers # 显示指定 pod 和容器的指标度量

4.kubectl与节点和集群交互

```
$ kubectl cordon my-node                                  # 标记 my-node 不可调度
$ kubectl drain my-node                                   # 清空 my-node 以待维护
$ kubectl uncordon my-node                                # 标记 my-node 可调度
$ kubectl top node my-node                                # 显示 my-node 的指标度量
$ kubectl cluster-info                                    # 显示 master 和服务的地址
$ kubectl cluster-info dump                               # 将当前集群状态输出到 stdout
$ kubectl cluster-info dump --output-directory=/path/to/cluster-state   # 将当前集群状态输出到 /path/to/cluster-state

# 如果该键和影响的污点（taint）已存在，则使用指定的值替换
$ kubectl taint nodes foo dedicated=special-user:NoSchedule
```

4.kubectl学习捷径

详见：kubectl Cheat Sheet

或者：kubectl * -h

或者：pip install kube-shell

作业

- 熟悉kubectl的基本操作，完成上面实践课中60%的内容

联系我们

小象学院：互联网新技术在线教育领航者

- 微信公众号：大数据分析挖掘
- 新浪微博：ChinaHadoop