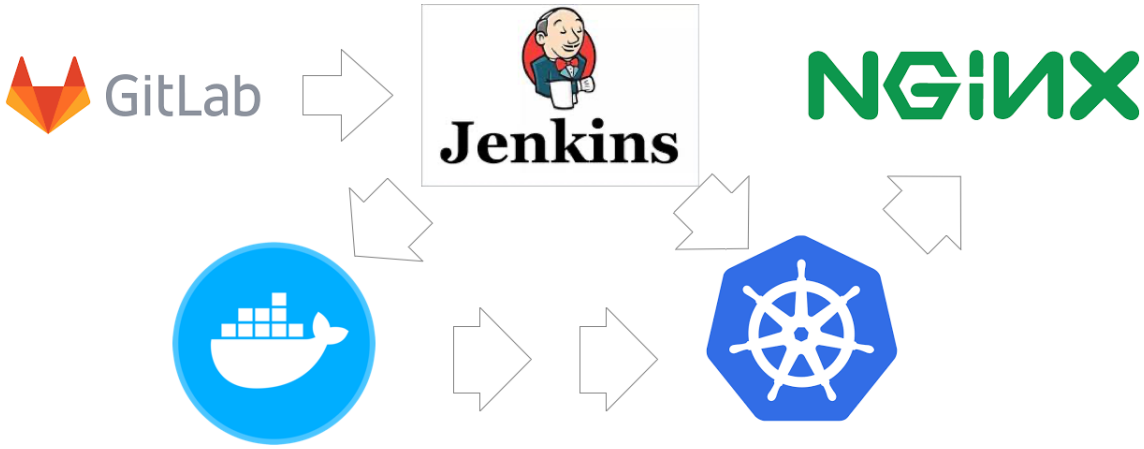
讲师(老司机)

容器虚拟化技术和自动化部署



k8s快速入门之命令行

1.NameSpace介绍

中文名称：命名空间。 你可以认为namespaces是你kubernetes集群中的虚拟化集群。在一个Kubernetes集群中可以拥有多个命名空间，它们在逻辑上彼此隔离。 可以为你提供组织，安全甚至性能方面的帮助！

Namespace是对一组资源和对象的抽象集合，比如可以用来将系统内部的对象划分为不同的项目组或用户组。常见的pods, services, replication controllers和deployments等都是属于某一个namespace的（默认是default），而node, persistentVolumes等则不属于任何namespace。

大多数的Kubernetes中的集群默认会有一个叫default的namespace。实际上，应该是4个：

- default：你的资源默认被创建于default命名空间。
- kube-system：kubernetes系统组件使用。
- kube-node-lease: kubernetes集群节点租约状态,v1.13加入
- kube-public：公共资源使用。但实际上现在并不常用。

这个默认（default）的namespace并没什么特别，但你不能删除它。这很适合刚刚开始使用kubernetes和一些小的产品系统。但不建议应用于大型生产系统。因为，这种复杂系统中，团队会非常容易意外地或者无意识地重写或者中断其他服务service。相反，请创建多个命名空间来把你的service(服务)分割成更容易管理的块。

作用：多租户情况下，实现资源隔离
属于逻辑隔离
属于管理边界
不属于网络边界
可以针对每个namespace做资源配额

1.1查看命名空间

```
kubectl get namespace

查看所有命名空间的pod资源
kubectl get pod --all-namespaces
kubectl get pod -A

简写命令
kubectl get ns
```

1.1.1说明

```
default 用户创建的pod默认在此命名空间
kube-public 所有用户均可以访问，包括未认证用户
kube-node-lease kubernetes集群节点租约状态,v1.13加入
kube-system kubernetes集群在使用
```

1.2创建NameSpace

```
kubectl create namespace lagou

简写命令
kubectl create ns lagou
```

1.3删除NameSpace

```
kubectl delete namespace lagou

简写命令
kubectl delete ns lagou
```

2.Pod

2.0pod简介

Pod是kubernetes集群能够调度的最小单元。Pod是容器的封装 。

在Kubernetes集群中，Pod是所有业务类型的基础，也是K8S管理的最小单位级，它是一个或多个容器的组合。这些容器共享存储、网络和命名空间，以及如何运行的规范。在Pod中，所有容器都被同一安排和调度，并运行在共享的上下文中。对于具体应用而言，Pod是它们的逻辑主机，Pod包含业务相关的多个应用容器。

2.0.1Pod有两个必须知道的特点。

网络:每一个Pod都会被指派一个唯一的Ip地址，在Pod中的每一个容器共享网络命名空间，包括Ip地址和网络端口。在同一个Pod中的容器可以和localhost进行互相通信。当Pod中的容器需要与Pod外的实体进行通信时，则需要通过端口等共享的网络资源。

存储: Pod能够被指定共享存储卷的集合，在Pod中所有的容器能够访问共享存储卷，允许这些容器共享数据。存储卷也允许在一个Pod持久化数据，以防止其中的容器需要被重启。

2.0.2Pod的工作方式

K8s一般不直接创建Pod。 而是通过控制器和模版配置来管理和调度

- Pod模版

我们会在后续章节为大家介绍pod模版。

- Pod重启

在Pod中的容器可能会由于异常等原因导致其终止退出，Kubernetes提供了重启策略以重启容器。重启策略对同一个Pod的所有容器起作用，容器的重启由Node上的kubelet执行。Pod支持三种重启策略，在配置文件中通过*restartPolicy*字段设置重启策略：

1. Always：只要退出就会重启。
2. OnFailure：只有在失败退出（exit code不等于0）时，才会重启。
3. Never：只要退出，就不再重启

注意，这里的重启是指在Pod的宿主Node上进行本地重启，而不是调度到其它Node上。

- 资源限制

Kubernetes通过cgroups限制容器的CPU和内存等计算资源，包括requests（请求，调度器保证调度到资源充足的Node上）和limits（上限）等。

2.1查看Pod

```
查看default命名空间下的pods
kubectl get pods
查看kube-system命名空间下的pods
kubectl get pods -n kube-system
查看所有命名空间下的pods
kubectl get pod --all-namespaces
kubectl get pod -A
```

2.2创建Pod

2.2.1下载镜像

```
K8S集群的每一个节点都需要下载镜像:选择不同的基础镜像，下载镜像的大小也不同。
docker pull tomcat:9.0.20-jre8-alpine        108MB
docker pull tomcat:9.0.37-jdk8-openjdk-slim  305MB
docker pull tomcat:9.0.37-jdk8               531MB

同学们可以自行下载后进行备份。
docker save -o tomcat9.tar tomcat:9.0.20-jre8-alpine
docker load -i tomcat9.tar
```

2.2.2运行pod

```
在default命名空间中创建一个pod副本的deployment

kubectl run tomcat9-test --image=tomcat:9.0.20-jre8-alpine --port=8080

kubectl get pod
kubectl get pod -o wide
使用pod的IP访问容器
crul ***:8080
```

2.3扩容

```
将副本扩容至3个
kubectl scale --replicas=3 deployment/tomcat9-test

kubectl get deployment
kubectl get deployment -o wide
使用deployment的IP访问pod
```

2.4创建服务

```
kubectl expose deployment tomcat9-test --name=tomcat9-svc --port=8888 --target-port=8080 --protocol=TCP --type=NodePort
kubectl get svc
kubectl get svc -o wide
访问服务端口
curl 10.105.225.0:8888
访问集群外端口
http://192.168.198.120:30217
```

```
1.练习
pod deployment 扩容 service 案例

2.自行查找一些资料：k8s集群 NodePort默认的端口号范围。。。

3.扩展技能：如何修改k8s集群NodePort默认的端口号范围
```

2.5kubectl常用命令练习

2.5.1语法规则

```
kubectl [command] [TYPE] [NAME] [flags]
```

其中 `command`、`TYPE`、`NAME` 和 `flags` 分别是：

- `command`：指定要对一个或多个资源执行的操作，例如 `create`、`get`、`describe`、`delete`。
- `TYPE`：指定资源类型。资源类型不区分大小写，可以指定单数、复数或缩写形式。例如，以下命令输出相同的结果：

  ```
  kubectl get pod pod1
  kubectl get pods pod1
  kubectl get po pod1
  ```

- `NAME`：指定资源的名称。名称区分大小写。如果省略名称，则显示所有资源的详细信息 `kubectl get pods`。
  在对多个资源执行操作时，您可以按类型和名称指定每个资源，或指定一个或多个文件：

    - 要按类型和名称指定资源：
        - 要对所有类型相同的资源进行分组，请执行以下操作：`TYPE1 name1 name2 name<#>`。
          例子：`kubectl get pod example-pod1 example-pod2`
        - 分别指定多个资源类型：`TYPE1/name1 TYPE1/name2 TYPE2/name3 TYPE<#>/name<#>`。
          例子：`kubectl get pod/example-pod1 replicationcontroller/example-rc1`
    - 用一个或多个文件指定资源：`-f file1 -f file2 -f file<#>`
        - 使用 `YAML` 而不是 `JSON` 因为 `YAML` 更容易使用，特别是用于配置文件时。
          例子：`kubectl get pod -f ./pod.yaml`
- `flags`：指定可选的参数。例如，可以使用 -s 或 -server 参数指定 Kubernetes API 服务器的地址和端口。

> 注意：
> 从命令行指定的参数会覆盖默认值和任何相应的环境变量。

如果您需要帮助，只需从终端窗口运行 `kubectl help` 即可。

2.5.2get命令

`kubectl get` - 列出一个或多个资源。

```
# 查看集群状态信息
kubectl cluster-info

# 查看集群状态
kubectl get cs

# 查看集群节点信息
kubectl get nodes

# 查看集群命名空间
kubectl get ns

# 查看指定命名空间的服务
kubectl get svc -n kube-system

# 以纯文本输出格式列出所有 pod。
kubectl get pods

# 以纯文本输出格式列出所有 pod，并包含附加信息(如节点名)。
kubectl get pods -o wide

# 以纯文本输出格式列出具有指定名称的副本控制器。提示：您可以使用别名 'rc' 缩短和替换 'replicationcontroller' 资源类型。
kubectl get replicationcontroller <rc-name>

# 以纯文本输出格式列出所有副本控制器和服务。
kubectl get rc,services

# 以纯文本输出格式列出所有守护程序集，包括未初始化的守护程序集。
kubectl get ds --include-uninitialized

# 列出在节点 server01 上运行的所有 pod
```

```
kubectl get pods --field-selector=spec.nodeName=server01
```

2.5.3describe命令

`kubectl describe` - 显示一个或多个资源的详细状态，默认情况下包括未初始化的资源。

```
# 显示名称为 <node-name> 的节点的详细信息。
kubectl describe nodes <node-name>

# 显示名为 <pod-name> 的 pod 的详细信息。
kubectl describe pods/<pod-name>

# 显示由名为 <rc-name> 的副本控制器管理的所有 pod 的详细信息。
# 记住：副本控制器创建的任何 pod 都以复制控制器的名称为前缀。
kubectl describe pods <rc-name>


# 描述所有的 pod，不包括未初始化的 pod
kubectl describe pods --include-uninitialized=false
```

> 说明： `kubectl get` 命令通常用于检索同一资源类型的一个或多个资源。 它具有丰富的参数，允许您使用 `-o` 或 `--output` 参数自定义输出格式。您可以指定 `-w` 或 `--watch` 参数以开始观察特定对象的更新。 `kubectl describe` 命令更侧重于描述指定资源的许多相关方面。它可以调用对 `API 服务器` 的多个 API 调用来为用户构建视图。 例如，该 `kubectl describe node` 命令不仅检索有关节点的信息，还检索在其上运行的 pod 的摘要，为节点生成的事件等。

2.5.4delete命令

kubectl delete` - 从文件、stdin 或指定标签选择器、名称、资源选择器或资源中删除资源。

```
# 使用 pod.yaml 文件中指定的类型和名称删除 pod。
kubectl delete -f pod.yaml

# 删除标签名= <label-name> 的所有 pod 和服务。
kubectl delete pods,services -l name=<label-name>

# 删除所有具有标签名称= <label-name> 的 pod 和服务，包括未初始化的那些。
kubectl delete pods,services -l name=<label-name> --include-uninitialized

# 删除所有 pod，包括未初始化的 pod。
kubectl delete pods --all
```

2.5.5进入容器命令

`kubectl exec` - 对 pod 中的容器执行命令。与docker的exec命令非常类似

```
# 从 pod <pod-name> 中获取运行 'date' 的输出。默认情况下，输出来自第一个容器。
kubectl exec <pod-name> date

# 运行输出 'date' 获取在容器的 <container-name> 中 pod <pod-name> 的输出。
kubectl exec <pod-name> -c <container-name> date

# 获取一个交互 TTY 并运行 /bin/bash <pod-name >。默认情况下，输出来自第一个容器。
kubectl exec -ti <pod-name> /bin/bash
```

2.5.6logs命令

`kubectl logs` - 打印 Pod 中容器的日志。

```
# 从 pod 返回日志快照。
kubectl logs <pod-name>

# 从 pod <pod-name> 开始流式传输日志。这类似于 'tail -f' Linux 命令。
kubectl logs -f <pod-name>
```

2.5.7格式化输出

```
将pod信息格式化输出到一个yaml文件
kubectl get pod web-pod-13je7 -o yaml
```

2.5.8强制删除pod

```
强制删除一个pod
--force --grace-period=0
```

2.6资源缩写

2.6.1缩写汇总

资源名	缩写名	API分组	按命名空间	资源类型
configmaps	cm	_	true	ConfigMap
namespaces	ns		false	Namespace
nodes	no		false	Node
persistentvolumeclaims	pvc		true	PersistentVolumeClaim
persistentvolumes	pv		false	PersistentVolume
pods	po		true	Pod
secrets			true	Secret
serviceaccounts	sa		true	ServiceAccount
services	svc		true	Service
daemonsets	ds	apps	true	DaemonSet
deployments	deploy	apps	true	Deployment
statefulsets	sts	apps	true	StatefulSet
horizontalpodautoscalers	hpa	autoscaling	true	HorizontalPodAutoscaler
cronjobs	cj	batch	true	CronJob
jobs		batch	true	Job
ingresses	ing	extensions	true	Ingress
poddisruptionbudgets	pdb	policy	true	PodDisruptionBudget
clusterrolebindings		rbac.authorization.k8s.io	false	ClusterRoleBinding
clusterroles		rbac.authorization.k8s.io	false	ClusterRole
rolebindings		rbac.authorization.k8s.io	true	RoleBinding
roles		rbac.authorization.k8s.io	true	Role
storageclasses	sc	storage.k8s.io	false	StorageClass