

kubernetes中的基本概念及术语

1. Cluster

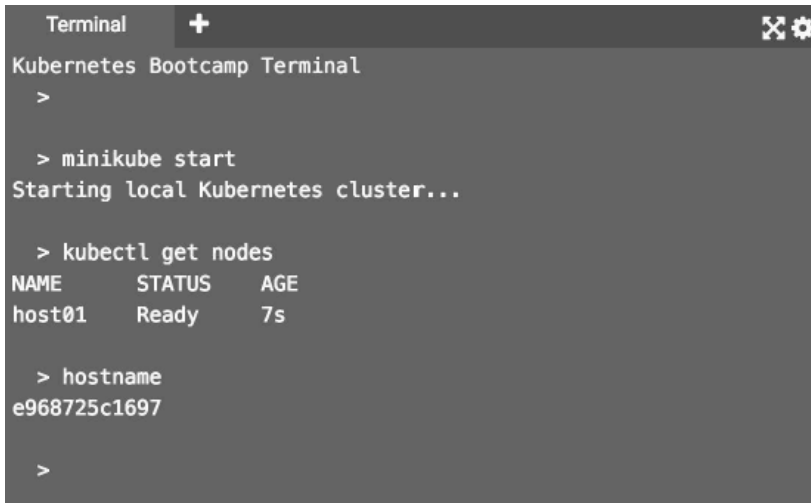
Cluster是计算、存储和网络资源的集合，Kubernetes利用这些资源运行各种基于容器的应用。

2. Master

Master是Cluster的大脑，它的主要职责是调度，即决定将应用放在哪里运行。Master运行Linux操作系统，可以是物理机或者虚拟机。为了实现高可用，可以运行多个Master。

3. Node

Node的职责是运行容器应用。Node由Master管理，Node负责监控并汇报容器的状态，同时根据Master的要求管理容器的生命周期。Node运行在Linux操作系统上，可以是物理机或者是虚拟机。在前面交互式教程中，我们创建的Cluster只有一个主机host01，它既是Master也是Node

A terminal window titled "Terminal" with a "+" icon and window controls. The text inside shows the command "minikube start" being executed, followed by "Starting local Kubernetes cluster...". Then, the command "kubectl get nodes" is executed, resulting in a table with columns NAME, STATUS, and AGE. The table shows one node named "host01" with status "Ready" and age "7s". Finally, the command "hostname" is executed, returning "e968725c1697".

```
Terminal +  
Kubernetes Bootcamp Terminal  
>  
  
> minikube start  
Starting local Kubernetes cluster...  
  
> kubectl get nodes  
NAME      STATUS    AGE  
host01    Ready     7s  
  
> hostname  
e968725c1697  
  
>
```

4. Pod

Pod是Kubernetes的最小工作单元。每个Pod包含一个或多个容器。Pod中的容器会作为一个整体被Master调度到一个Node上运行。Kubernetes引入Pod主要基于下面两个目的：

（1）可管理性。

有些容器天生就是需要紧密联系，一起工作。Pod提供了比容器更高层次的抽象，将它们封装到一个部署单元中。Kubernetes以Pod为最小单位进行调度、扩展、共享资源、管理生命周期。

（2）通信和资源共享。

Pod中的所有容器使用同一个网络namespace，即相同的IP地址和Port空间。它们可以直接用localhost通信。同样的，这些容器可以共享存储，当Kubernetes挂载volume到Pod，本质上是将volume挂载到Pod中的每一个容器。

Pods有两种使用方式：

（1）运行单一容器。

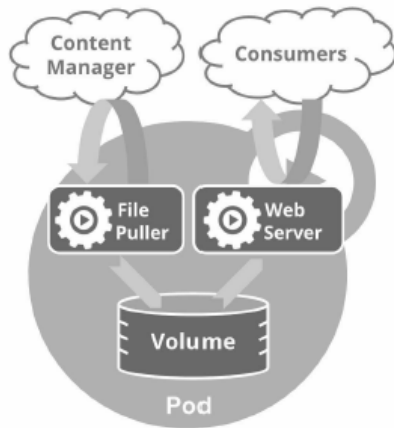
one-container-per-Pod是Kubernetes最常见的模型，这种情况下，只是将单个容器简单封装成Pod。即便是只有一个容器，Kubernetes管理的也是Pod而不是直接管理容器。

(2) 运行多个容器。

问题在于：哪些容器应该放到一个Pod中？

答案是：这些容器联系必须非常紧密，而且需要直接共享资源。

举个例子，如图所示，这个Pod包含两个容器：一个是File Puller，一个是Web Server。



这两个容器是紧密协作的，它们一起为Consumer提供最新的数据；同时它们也通过volume共享数据，所以放到一个Pod是合适的。

再来看一个反例：是否需要将Tomcat和MySQL放到一个Pod中？

Tomcat从MySQL读取数据，它们之间需要协作，但还不至于需要放到一个Pod中一起部署、一起启动、一起停止。同时它们之间是通过JDBC交换数据，并不是直接共享存储，所以放到各自的Pod中更合适。

5. Controller

Kubernetes通常不会直接创建Pod，而是通过Controller来管理Pod的。Controller中定义了Pod的部署特性，比如有几个副本、在什么样的Node上运行等。为

了满足不同的业务场景，Kubernetes提供了多种Controller，包括Deployment、ReplicaSet、DaemonSet、StatefulSet、Job等，我们逐一讨论。

(1) Deployment是最常用的Controller，比如在线教程中就是通过创建Deployment来部署应用的。Deployment可以管理Pod的多个副本，并确保Pod按照期望的状态运行。

(2) ReplicaSet实现了Pod的多副本管理。使用Deployment时会自动创建ReplicaSet，也就是说Deployment是通过ReplicaSet来管理Pod的多个副本的，我们通常不需要直接使用ReplicaSet。

(3) DaemonSet用于每个Node最多只运行一个Pod副本的场景。正如其名称所揭示的，DaemonSet通常用于运行daemon。

(4) StatefulSet能够保证Pod的每个副本在整个生命周期中名称是不变的，而其他Controller不提供这个功能。当某个Pod发生故障需要删除并重新启动时，Pod的名称会发生变化，同时StatefulSet会保证副本按照固定的顺序启动、更新或者删除。

(5) Job用于运行结束就删除的应用，而其他Controller中的Pod通常是长期持续运行。

6. Service

Deployment可以部署多个副本，每个Pod都有自己的IP，外界如何访问这些副本呢？

通过Pod的IP吗？

要知道Pod很可能会被频繁地销毁和重启，它们的IP会发生变化，用IP来访问不太现实。
答案是Service。

Kubernetes Service定义了外界访问一组特定Pod的方式。Service有自己的IP和端口，Service为Pod提供了负载均衡。

Kubernetes运行容器（Pod）与访问容器（Pod）这两项任务分别由Controller和Service执行。

7. Namespace

如果有多个用户或项目组使用同一个Kubernetes Cluster，如何将他们创建的Controller、Pod等资源分开呢？

答案就是Namespace。

Namespace可以将一个物理的Cluster逻辑上划分成多个虚拟Cluster，每个Cluster就是一个Namespace。不同Namespace里的资源是完全隔离的。

Kubernetes默认创建了两个Namespace

```
> kubectl get namespace
NAME          STATUS    AGE
default       Active    17s
kube-system   Active    17s
>
```

default：创建资源时如果不指定，将被放到这个Namespace中。

kube-system：Kubernetes自己创建的系统资源将放到这个Namespace中。