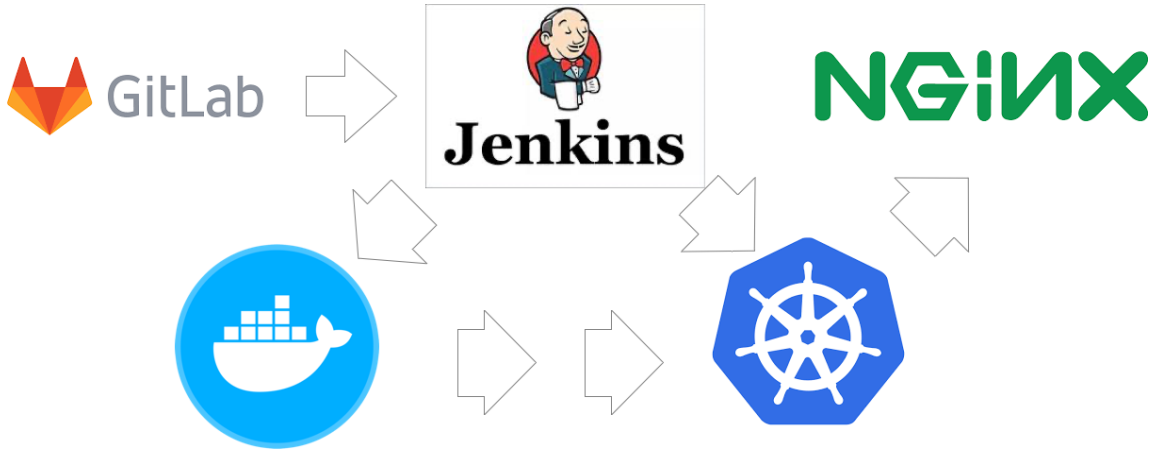
讲师(老司机)

容器虚拟化技术和自动化部署



课程介绍

第1节 学前必备基础

- 安装虚拟机环境

平台	软件
Win	VMware Workstation Pro15
Mac	VMware Workstation Pro15
Linux	VMware Fusion15

- 生成至少三台虚拟机

主机名	主机IP
k8s-master01	192.168.198.156
k8s-node01	192.168.198.157
k8s-node02	192.168.198.158
k8s-node03	192.168.198.159
harbor-159	192.168.198.155

- 远程连接工具

平台	工具
Win	CRT/XShell
Mac	CRT/Item2/终端
Linux	终端

第2节 课程目标

通过学习完本课程，达到以下目标

- 掌握容器特点及应用，了解其应用场景
- 掌握容器编排工具的设计理念及基本使用
- 通过CI/CD工具实现自动化流水线

第3节 课程内容

- Docker
    - Docker快速实战
    - Docker核心原理
    - Docker高级应用
    - Docker的运维管理
- Kubernetes:k8s
    - Kubernetes快速实战
    - Kubernetes核心原理
    - **Kubernetes运维管理**
- DevOps+Jenkins
    - DevOps的概述
    - CI工具之Gitlab
    - CD工具之Jenkins
    - Gitlab+Jenkins实现发布自动化

第一部分 Kubernetes快速实战

前言

1.云平台基础概念

IaaS:基础设施服务

PaaS：平台服务

SaaS：软件服务

2.kubernetes与docker swarm对比

长期以来，Kubernetes 和 Docker Swarm被看做是vs的对手，在接下来的对比中，我们看一下它们应该在何时被使用，以及怎么一起工作的。

关于Kubernetes和Docker有无数的争论和讨论。如果你没有深入研究它们，你会认为这两种开源技术在争夺容器（container）霸权。 让我们来一看清楚，Kubernetes和Docker Swarm不是竞争对手！两者都有各自的优缺点，可以根据你应用程序的需求来选择使用。(择优选用，相互协作)

Docker是一种容器管理服务，它帮助开发人员设计应用程序，使用容器能更容易地创建、部署和运行应用程序。Docker有一个用于集群容器的内置机制，称为“集群模式”（swarm mode）。使用集群模式，你可以使用Docker引擎在多台机器上启动应用程序。

Docker Swarm是Docker自己针对Docker容器的原生集群解决方案，它的优点是紧密集成到Docker的生态系统中，并且使用自己的API。它监视跨服务器集群的容器数量，是创建集群docker应用程序的最方便的方法，不需要额外的硬件。它为Dockerized应用程序提供了一个小型但有用的编排系统。

使用Docker Swarm的优点

> 1. 更快的运行速度
> 2. 完备的相关技术文档
> 3. 快速简单的配置
> 4. 确保程序独立（容器间低耦合）
> 5. 版本控制与组件重用

- 以更快的速度运行： 当您使用虚拟环境时，您可能已经意识到它需要很长时间，并且包含了启动和启动您想要运行的应用程序的冗长过程。对于Docker Swarm来说，这不再是一个问题。Docker Swarm不需要启动一个完整的虚拟机，就可以让应用程序在虚拟和软件定义的环境中快速运行，并有助于DevOps的实现。
- 文档提供了所有的信息： Docker团队在文档方面非常突出! Docker正在快速发展，并且非常受欢迎。如果平台的一个版本在短时间间隔内发布，有些平台可能不维护文档。但是Docker Swarm从不这样，如果一些信息只适用于Docker Swarm的特定版本，那么相应文档将确保更新了所有信息。
- 提供简单快速的配置： Docker Swarm的一个主要优点是它简化了问题。Docker Swarm使用户可以自己配置，将其放入代码中并轻松部署。由于Docker Swarm可以在各种环境中使用，因此需求不受应用程序环境的约束。
- 确保应用程序是隔离的：Docker Swarm负责将每个容器与其他容器隔离，并拥有自己的资源。你可以部署各种容器，以便在不同的堆栈中运行单独的应用程序。除此之外，Docker Swarm将在每个应用程序在自己的容器上运行时清除应用程序的删除。如果不再需要应用程序，可以删除它的容器。它不会在您的主机OS上留下任何临时文件或配置文件。
- 版本控制和组件重用：使用Docker Swarm，您可以跟踪容器的连续版本、检查差异或回滚到前面的版本。容器重用来自前一层的组件，这使得它们非常轻量级。

使用Docker Swarm的缺点

> 1. 跨平台支持效果差
> 2. 不提供存储选项
> 3. 监控信息不足

- Docker依赖于平台: Docker Swarm是一个为Linux设计的平台。虽然Docker支持Windows和Mac OS X，但它使用虚拟机在非linux平台上运行。设计为在Windows上的Docker容器中运行的应用程序不能在Linux上运行，反之亦然。
- 不提供存储选项: Docker Swarm不提供将容器连接到存储的简便方法，这是主要缺点之一。它的数据量需要对主机和手动配置进行大量的改进。如果你想要Docker Swarm解决存储问题，也能办

到，但是方式并不高效，且这种方式对用户并不友好。
- 监控不良: Docker Swarm提供关于容器的基本信息，如果您正在寻找基本的监控解决方案，那么使用Stats命令就足够了。如果您正在寻找先进的监控，那么Docker集群不是好的选择。虽然有像CAdvisor这样的第三方工具可以提供更多的监控，但是使用Docker本身实时收集更多关于容器的数据是不可行的。

为了避免这些不足，可以使用Kubernetes

自动化容器部署、扩展和管理平台
Automated Container Deployment, Scaling and Management Platform

---

当在多台机器上的多个容器中使用不同组件开发应用程序时，需要一个工具来管理和协调容器。这只有在Kubernetes的帮助下才可行。

Kubernetes**是一个用于在集群环境中管理容器化应用程序**的开源系统。以正确的方式使用Kubernetes可以帮助DevOps作为一个服务团队自动扩展应用程序，**并在零停机的情况下进行更新。**

使用Kubernetes的优点

- 它的速度很快：在不停机的情况下持续部署新功能时，Kubernetes是一个完美的选择。Kubernetes的目标是以恒定的正常运行时间更新应用程序。它的速度通过您每小时可以运送的许多功能来衡量，同时保持可用的服务。
- 遵循不可变基础架构的原则: 在传统的方法中，如果多个更新出现错误，您没有任何关于部署了多少个更新以及发生错误的时间点的记录。在不可变的基础架构中，如果想要更新任何应用程序，需要使用新标记构建容器映像并部署它，用旧映像版本销毁旧容器。这样，你就会有一个记录，并了解你做了什么，万一有什么错误;您可以轻松地回滚到前面的映像。
- 提供声明式配置: 用户可以知道系统应该处于什么状态以避免错误。源代码控制、单元测试等传统工具不能与命令式配置一起使用，但可以与声明式配置一起使用。
- 大规模部署和更新软件：由于Kubernetes具有不可变的声明性，因此扩展很容易。Kubernetes提供了一些用于扩展目的的有用功能：
    - 水平基础架构扩展：在单个服务器级别执行操作以应用水平扩展。可以毫不费力地添加或分离atest服务器。
    - 自动扩展：根据CPU资源或其他应用程序指标的使用情况，您可以更改正在运行的容器数
    - 手动扩展：您可以通过命令或界面手动扩展正在运行的容器的数量
    - 复制控制器：复制控制器确保集群在运行状态下具有指定数量的等效pod。如果存在太多pod，则复制控制器可以删除额外的pod，反之亦然。
- 处理应用程序的可用性：Kubernetes检查节点和容器的运行状况，并在由于错误导致的盒中崩溃时提供自我修复和自动替换。此外，它在多个pod之间分配负载，以便在意外流量期间快速平衡资源。
- 存储卷: 在Kubernetes中，数据是在容器之间共享的，但是如果pod被杀死，则自动删除卷。此外，数据是远程存储的，因此如果将pod移动到另一个节点，数据将一直保留，直到用户删除它。

使用Kubernetes的缺点

- 初始进程需要时间: 当创建一个新进程时，您必须等待应用程序启动，然后用户才能使用它。如果您要迁移到Kubernetes，则需要对代码库进行修改，以提高启动流程的效率，这样用户就不会有不好的体验。

- 迁移到无状态需要做很多工作: 如果您的应用程序是集群的或无状态的，那么将不会配置额外的pod，并且必须重新处理应用程序中的配置。
- 安装过程非常单调乏味: 如果不使用Azure、谷歌或Amazon等云提供商，就很难在集群上设置Kubernetes。

Kubernetes vs. Docker Swarm：快速摘要

	Docker Swarm	**Kubernetes**
开发者	Docker 公司	谷歌
发布年份	2013	2014
公司使用	Bugsnag，Bluestem Brands，Hammerhead，Code Picnic，Dial once等。	Asana，Buffer，CircleCI，Evernote，Harvest，Intel，Starbucks，Shopify等
Controller	Manager	Master
Storage	Volumes	Persistent and Epherma
公共云服务提供商	Google，Azure，AWS，OTC	Azure
兼容性	不那么广泛和可定制	更广泛和高度可定制
安装	易于设置	需要时间安装
容差率	低容错性	高容错性
大集群	对于强集群状态考虑速度	在不考虑速度的情况下，即使在大型集群中也提供容器部署和扩展
负载均衡（Loading Balancing）	当容器中的pod定义为服务时，提供负载平衡	通过集群中的任何节点提供自动的内部负载平衡
部署单位	Task	Pod
Port	Published Port	Endpoint
社区	活跃的用户群，定期更新各种应用程序的映像	获得开源社区和谷歌，亚马逊，微软和IBM等大公司的大力支持
弱点	没有供应商的认证计划。大多数组织需要商业认证版本	倾向于开发人员而不是中央IT
优势	主要由可以决定产品方向的单一供应商控制	明确的市场领导者 最高的采用率
Slave	Worker	Nodes
容器设置	功能由Docker API提供并受其限制	客户端API和YAML， 在Kubernetes中是唯一的
可扩展性	即使在大型容器中也可以快速部署容器	以牺牲速度为代价为集群状态提供强有力的保证

Docker和Kubernetes是不同的，但不是竞争对手

正如前面所讨论的，Kubernetes和Docker都在不同的级别上工作，但都可以一起使用。Kubernetes可以集成Docker引擎来执行Docker容器的调度和执行。由于Docker和Kubernetes都是容器编排器，所以它们都可以帮助管理容器的数量，还可以帮助DevOps实现。两者都可以自动化运行容器化的基础架构中涉及的大部分任务，并且都是开源软件项目，由Apache License 2.0管理。除此之外，两者都使用YAML格式的文件来控制工具如何编排容器集群。当两者同时使用时，Docker和Kubernetes都是部署现代云架构的最佳工具。随着Docker Swarm的豁免，Kubernetes和Docker都相互补充。

Kubernetes使用Docker作为主要的容器引擎解决方案，Docker最近宣布它可以支持Kubernetes作为其企业版的编排层，除此之外，Docker还批准了经过认证的Kubernetes程序，该程序可确保所有Kubernetes API按预期运行。Kubernetes使用Docker Enterprise的功能，如安全映像管理，其中Docker EE提供映像扫描，以确保容器中使用的映像是否存在问题。另一个是安全自动化，其中组织可以消除低效率，例如扫描映像是否存在漏洞。

Kubernetes或Docker：哪个是完美的选择?

如果使用Kubernetes：

您正在寻找成熟的部署和监控选项
您正在寻找快速可靠的响应时间
您正在寻求开发复杂的应用程序，并且需要高资源计算而不受限制
你有一个非常大的集群

如果，使用Docker，

您希望在不花费太多时间进行配置和安装的情况下启动工具;
您正在寻找开发一个基本和标准的应用程序，它足够使用默认的docker镜像;
在不同的操作系统上测试和运行相同的应用程序对您来说不是问题;
您需要zdocker API经验和兼容性。

最后的想法：Kubernetes和Docker是朋友

无论你选择Kubernetes还是Docker，两者都被认为是最好的，并且有相当大的差异。在这两者之间做出选择的最好方法可能是考虑哪一个您已经比较熟悉，或者哪一个适合您现有的软件堆栈。如果您需要开发复杂的应用程序，请使用Kubernetes，如果您希望开发小型应用程序，请使用Docker Swarm。此外，选择正确的项目是一项非常全面的任务，完全取决于您的项目需求和目标受众。

k8s 集群快速部署

kubernetes官网地址：国外网站，访问速度较慢。

```
https://kubernetes.io/
```

kubernets中文社区地址：

```
https://www.kubernetes.org.cn/
```

k8s集群部署方式：

> 1. 使用minikube安装单节点集群，用于测试
> 2. 采用工具kubeadm
> 3. 使用kubespray，google官方提供的工具。
> 4. 全手动:二进制方式安装
> 5. 全自动安装:rancher 、kubesphere

快速部署一个 Kubernetes, 即拥有了一个完整的集群。忽略三大步骤

1. centos7.7操作系统配置
2. k8s集群镜像下载
3. k8s集群网络配置

主机名	主机IP
k8s-master01	192.168.198.156
k8s-node01	192.168.198.157
k8s-node02	192.168.198.158
k8s-node03	192.168.198.159

初始化k8s集群

```
kubeadm init --apiserver-advertise-address=192.168.198.156 --kubernetes-
version v1.17.5 --service-cidr=10.1.0.0/16 --pod-network-cidr=10.81.0.0/16
```

```
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 192.168.198.156:6443 --token 8ic4bd.ns2wgycdqx5ey7go \
    --discovery-token-ca-cert-hash
sha256:3b883e6c1f0dcb29834dd08af8eb6e105854d0a475edb3630afc4539fd4f95c8
```

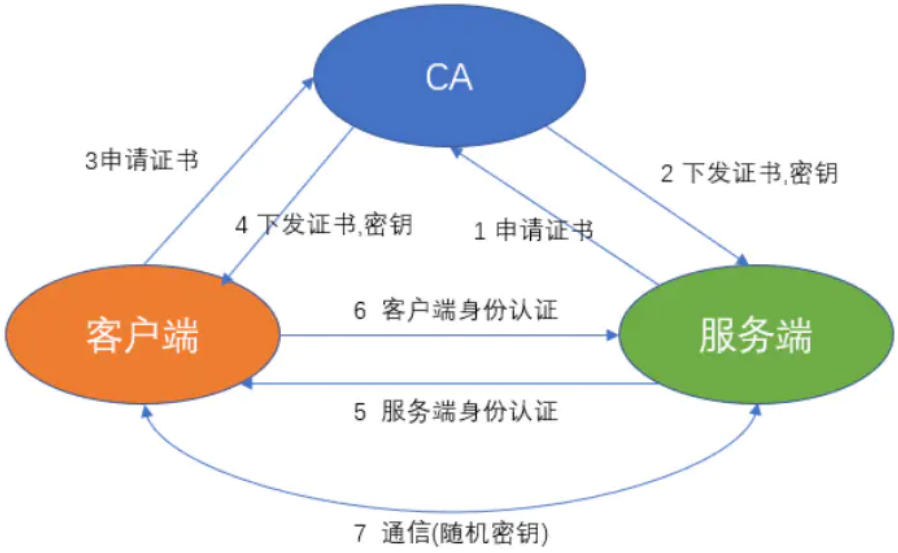
K8S集群安全机制：

Kubernetes 作为一个分布式集群的管理工具，保证集群的安全性是其一个重要的任务。API Server 是集群内部
各个组件通信的中介，也是外部控制的入口。所以 Kubernetes 的安全机制基本就是围绕保护 API Server 来设计
的。Kubernetes 使用了认证（Authentication）、鉴权（Authorization）、准入控制（Admission Control）三步来保证API Server的安全 。

Authentication(认证)

- 第三方授权协议: authenticating proxy
- HTTP Token 认证：通过一个 Token 来识别合法用户
  HTTP Token 的认证是用一个很长的特殊编码方式的并且难以被模仿的字符串 - Token 来表达客户的一
  种方式。Token 是一个很长的很复杂的字符串，每一个 Token 对应一个用户名存储在 API Server 能访
  问的文件中。当客户端发起 API 调用请求时，需要在 HTTP Header 里放入 Token
- HTTP Base 认证：通过 用户名+密码 的方式认证
  用户名+：+密码 用 BASE64 算法进行编码后的字符串放在 HTTP Request 中的 Heather Authorization 域里发送给服务端，服务端收到后进行编码，获取用户名及密码
- 最严格的 HTTPS 证书认证：基于 CA 根证书签名的客户端身份认证方式

  1. HTTPS 证书认证： 采用双向加密认证方式



  2. 证书颁发：

     手动签发：通过 k8s 集群的跟 ca 进行签发 HTTPS 证书
     自动签发：kubelet 首次访问 API Server 时，使用 token 做认证，通过后，Controller Manager 会为
     kubelet 生成一个证书，以后的访问都是用证书做认证了

  3. 安全性说明

     Controller Manager、Scheduler 与 API Server 在同一台机器，所以直接使用 API Server 的非安全端口
     访问， --insecure-bind-address=127.0.0.1

     kubectl、kubelet、kube-proxy 访问 API Server 就都需要证书进行 HTTPS 双向认证

4. kubeconfig 文件包含集群参数（CA证书、API Server地址），客户端参数（上面生成的证书和私钥），集群context 信息（集群名称、用户名）。Kubenetes 组件通过启动时指定不同的 kubeconfig 文件可以切换到不同
的集群

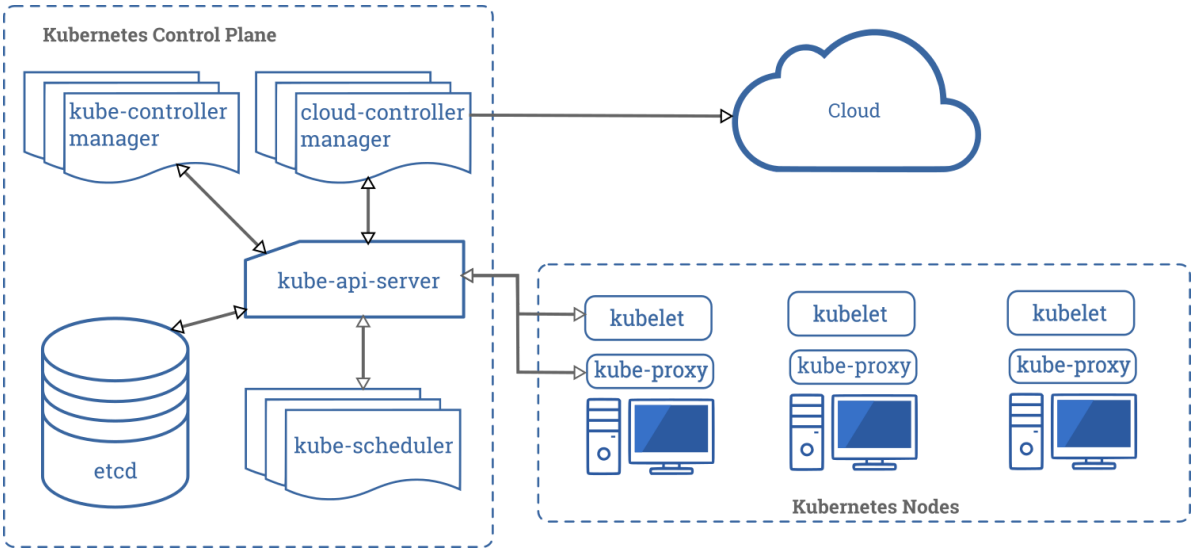
第1节 Kubernetes基础组件

一个 Kubernetes 集群包含 集群由一组被称作节点的机器组成。这些节点上运行 Kubernetes 所管理的容器化应用。集群具有至少一个工作节点和至少一个主节点。

工作节点托管作为应用程序组件的 Pod 。主节点管理集群中的工作节点和 Pod 。多个主节点用于为集群提供故障转移和高可用性。

本章概述交付正常运行的 Kubernetes 集群所需的各种组件。

这张图表展示了包含所有相互关联组件的 Kubernetes 集群。



1.1 控制平面组件（Control Plane Components）

控制平面的组件对集群做出全局决策(比如调度)，以及检测和响应集群事件（例如，当不满足部署的 replicas 字段时，启动新的 pod）。

控制平面组件可以在集群中的任何节点上运行。然而，为了简单起见，设置脚本通常会在同一个计算机上启动所有控制平面组件，并且不会在此计算机上运行用户容器。

1.1.1 kube-apiserver

主节点上负责提供 Kubernetes API 服务的组件；它是 Kubernetes 控制面的前端。

> 1. kube-apiserver是Kubernetes最重要的核心组件之一
> 2. 提供集群管理的REST API接口，包括认证授权，数据校验以及集群状态变更等
> 3. 提供其他模块之间的数据交互和通信的枢纽（其他模块通过API Server查询或修改数据，只有API Server才直接操作etcd）
> 4. 生产环境可以为apiserver做LA或LB。在设计上考虑了水平扩缩的需要。 换言之，通过部署多个实例可以实现扩缩。 参见构造高可用集群。

1.1.2 etcd

> 1. kubernetes需要存储很多东西，像它本身的节点信息，组件信息，还有通过kubernetes运行的pod，deployment，service等等。都需要持久化。etcd就是它的数据中心。生产环境中为了保证数据中心的高可用和数据的一致性，一般会部署最少三个节点。
> 2. 这里只部署一个节点在master。etcd也可以部署在kubernetes每一个节点。组成etcd集群。
> 3. 如果已经有etcd外部的服务，kubernetes直接使用外部etcd服务

etcd 是兼具一致性和高可用性的键值数据库，可以作为保存 Kubernetes 所有集群数据的后台数据库。

Kubernetes 集群的 etcd 数据库通常需要有个备份计划。要了解 etcd 更深层次的信息，请参考 etcd 文档。也可以使用外部的ETCD集群

1.1.3 kube-scheduler

主节点上的组件，该组件监视那些新创建的未指定运行节点的 Pod，并选择节点让 Pod 在上面运行。

> 1. kube-scheduler负责分配调度Pod到集群内的节点上，它监听kube-apiserver，查询还未分配Node的Pod，然后根据调度策略为这些Pod分配节点。

1.1.4 kube-controller-manager

在主节点上运行控制器的组件。

> 1. Controller Manager由kube-controller-manager和cloud-controller-manager组成，是Kubernetes的大脑，它通过apiserver监控整个集群的状态，并确保集群处于预期的工作状态。 kube-controller-manager由一系列的控制器组成，像Replication Controller控制副本，Node Controller节点控制，Deployment Controller管理deployment等等 cloud-controller-manager在Kubernetes启用Cloud Provider的时候才需要，用来配合云服务提供商的控制

1.1.5 云控制器管理器-(cloud-controller-manager)（暂时不考虑）

cloud-controller-manager 运行与基础云提供商交互的控制器。cloud-controller-manager 二进制文件是 Kubernetes 1.6 版本中引入的 alpha 功能。

cloud-controller-manager 仅运行云提供商特定的控制器循环。您必须在 kube-controller-manager 中禁用这些控制器循环，您可以通过在启动 kube-controller-manager 时将 --cloud-provider 参数设置为 external 来禁用控制器循环。

cloud-controller-manager 允许云供应商的代码和 Kubernetes 代码彼此独立地发展。在以前的版本中，核心的 Kubernetes 代码依赖于特定云提供商的代码来实现功能。在将来的版本中，云供应商专有的代码应由云供应商自己维护，并与运行 Kubernetes 的云控制器管理器相关联。

1.1.6kubectl

主节点上的组件

> 1. kubectl是Kubernetes的命令行工具，是Kubernetes用户和管理员必备的管理工具。kubectl提供了大量的子命令，方便管理Kubernetes集群中的各种功能。

1.2 Node 组件

节点组件在每个节点上运行，维护运行的 Pod 并提供 Kubernetes 运行环境。

1.2.1 kubelet

一个在集群中每个节点上运行的代理。它保证容器都运行在 Pod 中。

####

> 1. 一个在集群中每个工作节点上都运行一个kubelet服务进程，默认监听10250端口，接收并执行master发来的指令，管理Pod及Pod中的容器。每个kubelet进程会在API Server上注册节点自身信息，定期向master节点汇报节点的资源使用情况，并通过cAdvisor监控节点和容器的资源。

1.2.2 kube-proxy

> 1. 一个在集群中每台工作节点上都应该运行一个kube-proxy服务，它监听API server中service和endpoint的变化情况，并通过iptables等来为服务配置负载均衡，是让我们的服务在集群外可以被访问到的重要方式。

1.2.3 容器运行环境(Container Runtime)

容器运行环境是负责运行容器的软件。

Kubernetes 支持多个容器运行环境: Docker、 containerd、cri-o、 rktlet 以及任何实现 Kubernetes CRI (容器运行环境接口)。

1.3 插件(Addons)

插件使用 Kubernetes 资源 (DaemonSet, Deployment等) 实现集群功能。因为这些提供集群级别的功能，所以插件的命名空间资源属于 kube-system 命名空间。

所选的插件如下所述：有关可用插件的扩展列表，请参见插件 (Addons)。

1.3.1 KUBE-DNS

> 1. kube-dns为Kubernetes集群提供命名服务，主要用来解析集群服务名和Pod的hostname。目的是让pod可以通过名字访问到集群内服务。它通过添加A记录的方式实现名字和service的解析。普通的service会解析到service-ip。headless service会解析到pod列表。

1.3.2 用户界面(Dashboard)

Dashboard 是 Kubernetes 集群的通用基于 Web 的 UI。它使用户可以管理集群中运行的应用程序以及集群本身并进行故障排除。

1.3.3 容器资源监控

容器资源监控将关于容器的一些常见的时间序列度量值保存到一个集中的数据库中，并提供用于浏览这些数据的界面。

1.3.4 集群层面日志

集群层面日志 机制负责将容器的日志数据保存到一个集中的日志存储中，该存储能够提供搜索和浏览接口

第2节 Kubernetes安装与配置

2.1硬件安装要求：

序号	硬件	要求
1	CPU	至少2核
2	内存	至少3G
3	硬盘	至少50G

临时演示集群节点

主机名	主机IP
k8s-master01	192.168.198.186
k8s-node01	192.168.198.187
k8s-node02	192.168.198.188
k8s-node03	192.168.198.189

centos下载地址:推荐大家使用centos7.6以上版本。

```
http://mirrors.aliyun.com/centos/7/isos/x86_64/
```

查看centos系统版本命令：

```
cat /etc/centos-release
```

配置阿里云yum源

```
1.下载安装wget
yum install -y wget

2.备份默认的yum
mv /etc/yum.repos.d /etc/yum.repos.d.backup

3.设置新的yum目录
mkdir -p /etc/yum.repos.d

4.下载阿里yum配置到该目录中，选择对应版本
wget -O /etc/yum.repos.d/CentOS-Base.repo
http://mirrors.aliyun.com/repo/Centos-7.repo


5.更新epel源为阿里云epel源
mv /etc/yum.repos.d/epel.repo /etc/yum.repos.d/epel.repo.backup
mv /etc/yum.repos.d/epel-testing.repo /etc/yum.repos.d/epel-
testing.repo.backup
```

```
wget -O /etc/yum.repos.d/epel.repo http://mirrors.aliyun.com/repo/epel-7.repo


6.重建缓存
yum clean all
yum makecache

7.看一下yum仓库有多少包
yum repolist
yum update
```

升级系统内核

```
rpm -Uvh http://www.elrepo.org/elrepo-release-7.0-3.el7.elrepo.noarch.rpm
yum --enablerepo=elrepo-kernel install -y kernel-lt
grep initrd16 /boot/grub2/grub.cfg
grub2-set-default 0

reboot
```

查看centos系统内核命令：

```
uname -r
uname -a
```

查看CPU命令：

```
lscpu
```

查看内存命令：

```
free
free -h
```

查看硬盘信息

```
fdisk -l
```

2.2centos7系统配置

2.2.1关闭防火墙

```
systemctl stop firewalld
systemctl disable firewalld
```

2.2.2关闭selinux

```
sed -i 's/SELINUX=enforcing/SELINUX=disabled/g' /etc/sysconfig/selinux
setenforce 0
```

2.2.3网桥过滤

```
vi /etc/sysctl.conf

net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
net.bridge.bridge-nf-call-arptables = 1
net.ipv4.ip_forward=1
net.ipv4.ip_forward_use_pmtu = 0

生效命令
sysctl --system
查看效果
sysctl -a|grep "ip_forward"
```

2.2.4开启IPVS

```
安装IPVS
yum -y install ipset ipvsdm

编译ipvs.modules文件
vi /etc/sysconfig/modules/ipvs.modules

文件内容如下
#!/bin/bash
modprobe -- ip_vs
modprobe -- ip_vs_rr
modprobe -- ip_vs_wrr
modprobe -- ip_vs_sh
modprobe -- nf_conntrack_ipv4

赋予权限并执行
chmod 755 /etc/sysconfig/modules/ipvs.modules && bash
/etc/sysconfig/modules/ipvs.modules &&lsmod | grep -e ip_vs -e
nf_conntrack_ipv4

重启电脑，检查是否生效
reboot
lsmod | grep ip_vs_rr
```

2.2.5同步时间

```
安装软件
yum -y install ntpdate

向阿里云服务器同步时间
ntpdate time1.aliyun.com

删除本地时间并设置时区为上海
rm -rf /etc/localtime
ln -s /usr/share/zoneinfo/Asia/Shanghai /etc/localtime

查看时间
date -R || date
```

2.2.6命令补全

```
安装bash-completion
yum -y install bash-completion bash-completion-extras

使用bash-completion
source /etc/profile.d/bash_completion.sh
```

2.2.7关闭swap分区

```
临时关闭：
swapoff -a

永久关闭：
vi /etc/fstab

将文件中的/dev/mapper/centos-swap这行代码注释掉
#/dev/mapper/centos-swap swap  swap    defaults        0 0

确认swap已经关闭：若swap行都显示 0 则表示关闭成功
free -m
```

2.2.8hosts配置

```
vi /etc/hosts

文件内容如下：
cat <<EOF >>/etc/hosts
192.168.198.186 k8s-master01
192.168.198.187 k8s-node01
192.168.198.188 k8s-node02
192.168.198.189 k8s-node03
EOF
```

2.3安装docker

2.3.0阿里云开发者平台

开发者平台官网地址：可以参考阿里云官网提供的docker安装教程进行安装。

```
https://www.aliyun.com/
```

2.3.1安装docker前置条件

```
yum install -y yum-utils device-mapper-persistent-data lvm2
```

2.3.2添加源

```
yum-config-manager --add-repo http://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo

yum makecache fast
```

2.3.3查看docker更新版本

```
yum list docker-ce --showduplicates | sort -r
```

2.3.4安装docker最新版本

```
 yum -y install docker-ce

 安装指定版本：
 yum -y install docker-ce-18.09.8

 可以通过docker version命令查看
 docker-client版本：当前最新版本
 docker-server版本为：18.09.8
```

2.3.5开启dock而服务

```
systemctl start docker

systemctl status docker
```

2.3.6安装阿里云镜像加速器

```
mkdir -p /etc/docker
tee /etc/docker/daemon.json <<-'EOF'
{
  "registry-mirrors": ["自己的阿里云镜像加速地址"]
}
EOF
systemctl daemon-reload
systemctl restart docker
```

2.3.7设置docker开启启动服务

```
systemctl enable docker
```

2.3.8修改Cgroup Driver

```
修改daemon.json，新增：

"exec-opts": ["native.cgroupdriver=systemd"]

重启docker服务：
systemctl daemon-reload
systemctl restart docker
查看修改后状态：
docker info | grep Cgroup
```

> ***修改cgroupdriver是为了消除安装k8s集群时的告警：***
>
> [WARNING IsDockerSystemdCheck]:
>
> detected “cgroupfs” as the Docker cgroup driver. The recommended driver is “systemd”. Please follow the guide at https://kubernetes.io/docs/setup/cri/.....

2.3.9复习docker常用命令

```
docker -v
docker version
docker info
```

2.4使用kubeadm快速安装

软件	kubeadm	kubelet	kubectl	docker-ce
版本	初始化集群管理集群 版本：1.17.5	用于接收api-server指令，对pod生命周期进行管理版本：1.17.5	集群命令行管理工具 版本：1.17.5	推荐使用版本：19.03.8

2.5安装yum源

2.5.1新建repo文件

```
vi /etc/yum.repos.d/kubernates.repo
```

2.5.2文件内容

```
[kubernetes]
name=Kubernetes
baseurl=https://mirrors.aliyun.com/kubernetes/yum/repos/kubernetes-el7-x86_64
enabled=1
gpgcheck=1
repo_gpgcheck=1
gpgkey=https://mirrors.aliyun.com/kubernetes/yum/doc/yum-key.gpg
       https://mirrors.aliyun.com/kubernetes/yum/doc/rpm-package-key.gpg
```

2.5.3更新缓存

```
yum clean all
yum -y makecache
```

2.5.4验证源是否可用

```
yum list | grep kubeadm

如果提示要验证yum-key.gpg是否可用，输入y。
查找到kubeadm。显示版本
```

2.5.5查看k8s版本

```
yum list kubelet --showduplicates | sort -r
```

2.5.6安装k8s-1.17.5

```
yum install -y kubelet-1.17.5 kubeadm-1.17.5 kubectl-1.17.5
```

2.6设置kubelet

2.6.1增加配置信息

```
如果不配置kubelet，可能会导致K8S集群无法启动。为实现docker使用的cgroupdriver与kubelet使用的cgroup的一致性。
vi /etc/sysconfig/kubelet

KUBELET_EXTRA_ARGS="--cgroup-driver=systemd"
```

2.6.2设置开机启动

```
systemctl enable kubelet
```

2.7初始化镜像（选学部分）

如果是第一次安装k8s，手里没有备份好的镜像，可以执行如下操作。也可以使用资料包中的镜像备份跳过本章节学习内容。

教学目标：本章节重点为下载镜像和备份镜像。

2.7.1查看安装集群需要的镜像

```
kubeadm config images list
```

2.7.2编写执行脚本

```
mkdir -p /data
cd /data
vi images.sh

#!/bin/bash
# 下面的镜像应该去除"k8s.gcr.io"的前缀，版本换成kubeadm config images list命令获取到的版本
images=(
    kube-apiserver:v1.17.5
    kube-controller-manager:v1.17.5
    kube-scheduler:v1.17.5
    kube-proxy:v1.17.5
    pause:3.1
    etcd:3.4.3-0
    coredns:1.6.5
)
for imageName in ${images[@]} ;
do
    docker pull registry.cn-hangzhou.aliyuncs.com/google_containers/$imageName
    docker tag registry.cn-hangzhou.aliyuncs.com/google_containers/$imageName k8s.gcr.io/$imageName
    docker rmi registry.cn-hangzhou.aliyuncs.com/google_containers/$imageName
done


```

2.7.3执行脚本

```
mkdir -p /data
cd /data
给脚本授权
chmod +x images.sh
执行脚本
./images.sh
```

2.7.4保存镜像:

idea的列编辑模式：alt+鼠标左键

```
docker save -o k8s.1.17.5.tar \
k8s.gcr.io/kube-proxy:v1.17.5  \
k8s.gcr.io/kube-apiserver:v1.17.5 \
k8s.gcr.io/kube-controller-manager:v1.17.5 \
k8s.gcr.io/kube-scheduler:v1.17.5 \
k8s.gcr.io/coredns:1.6.5 \
k8s.gcr.io/etcd:3.4.3-0 \
k8s.gcr.io/pause:3.1 \
```

2.8导入镜像

2.81导入master节点镜像tar包

```
master节点需要全部镜像

docker load -i k8s.1.17.5.tar
```

2.8.2导入node节点镜像tar包

```
node节点需要kube-proxy:v1.17.5和pause:3.1,2个镜像

docker load -i k8s.1.17.5.node.tar
```

2.9初始化集群

配置k8s集群网络

2.9.0calico官网地址

```
官网下载地址：
https://docs.projectcalico.org/v3.14/manifests/calico.yaml

github地址：
https://github.com/projectcalico/calico

镜像下载：
docker pull calico/cni:v3.14.2
docker pull calico/pod2daemon-flexvol:v3.14.2
docker pull calico/node:v3.14.2
docker pull calico/kube-controllers:v3.14.2
```

```
配置hostname：
hostnamectl set-hostname k8s-master01
配置ip地址：
```

2.9.1初始化集群信息:calico网络

```
kubeadm init --apiserver-advertise-address=192.168.198.156 --kubernetes-version v1.17.5 --service-cidr=10.1.0.0/16 --pod-network-cidr=10.81.0.0/16
```

2.9.2执行配置命令

```
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

2.9.3node节点加入集群信息

```
kubeadm join 192.168.198.186:6443 --token kksfgq.b9bhf82y35ufw4np \
    --discovery-token-ca-cert-hash sha256:e1e347e6db1db5c13fcdc2c7d51a2f9029100a4cc13c2d89a2dbfa5077f5b07f
```

2.9.4kubectl命令自动补全

```
echo "source <(kubectl completion bash)" >> ~/.bash_profile
source ~/.bash_profile
```

2.9.5发送邮件问题

```
在 bash 中设置当前 shell 的自动补全，要先安装 bash-completion 包。
echo "unset MAILCHECK">> /etc/profile
source /etc/profile

在你的 bash shell 中永久的添加自动补全
```

2.9.6yum-key.gpg验证未通过

```
wget https://mirrors.aliyun.com/kubernetes/yum/doc/yum-key.gpg
wget https://mirrors.aliyun.com/kubernetes/yum/doc/rpm-package-key.gpg
rpm --import yum-key.gpg
rpm --import rpm-package-key.gpg
```