# Kubernetes（k8s）实习学习笔记

## 一、Kubernetes 基础概念

### 什么是 Kubernetes

Kubernetes，简称 k8s，是一个开源的容器编排平台，用于自动化部署、扩展和管理容器化应用程序（Automation system）。

### Kubernetes 的作用

如果应用由多个容器组成，且需要在不同的服务器上运行时，手动管理这些容器会变得非常复杂。Kubernetes 可以实现容器的自动部署、弹性伸缩、负载均衡、自愈能力等，简化了容器化应用的管理流程，K8S管理pod，pod管理容器应用。

### Kubernetes 的核心特性

自动部署：能够按照定义的规则自动将容器部署到集群中的节点上。

弹性伸缩：根据应用的负载情况，自动增加或减少容器的数量，以保证应用的性能和资源利用率。

负载均衡：将请求均匀地分配到多个容器实例上，避免单个容器负载过高。

自愈能力：当容器或节点出现故障时，能够自动重启容器或在其他健康节点上重新部署容器确保应用的可用性。

滚动更新：在不中断应用服务的情况下，对应用进行更新，降低更新风险。

**K8S部署方式**

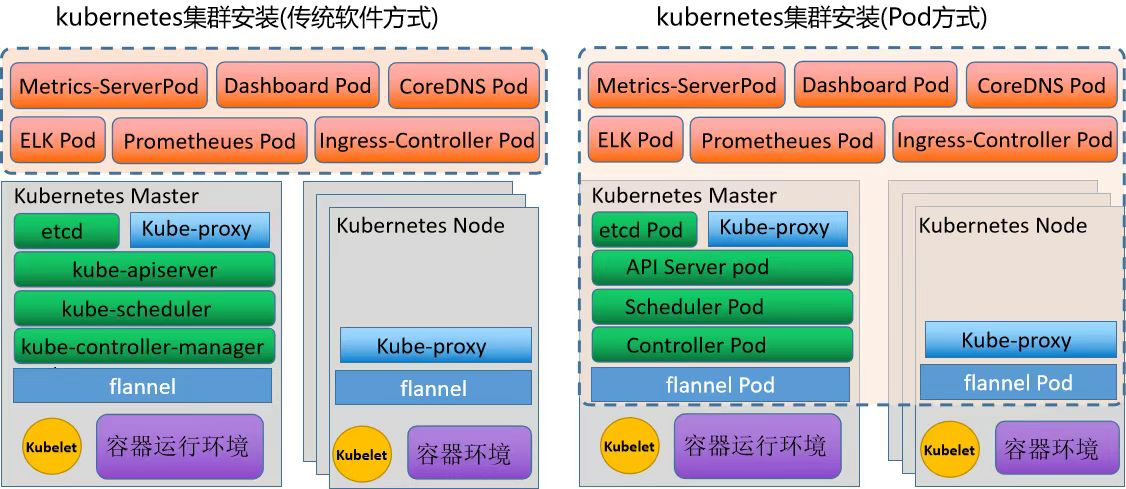
kubeadm部署，单主式分布（一个master）

【也可以多主式分布（多个master） kubeadm 或工具搭建多 Master 】

1.物理节点网络（根基，本地网络）

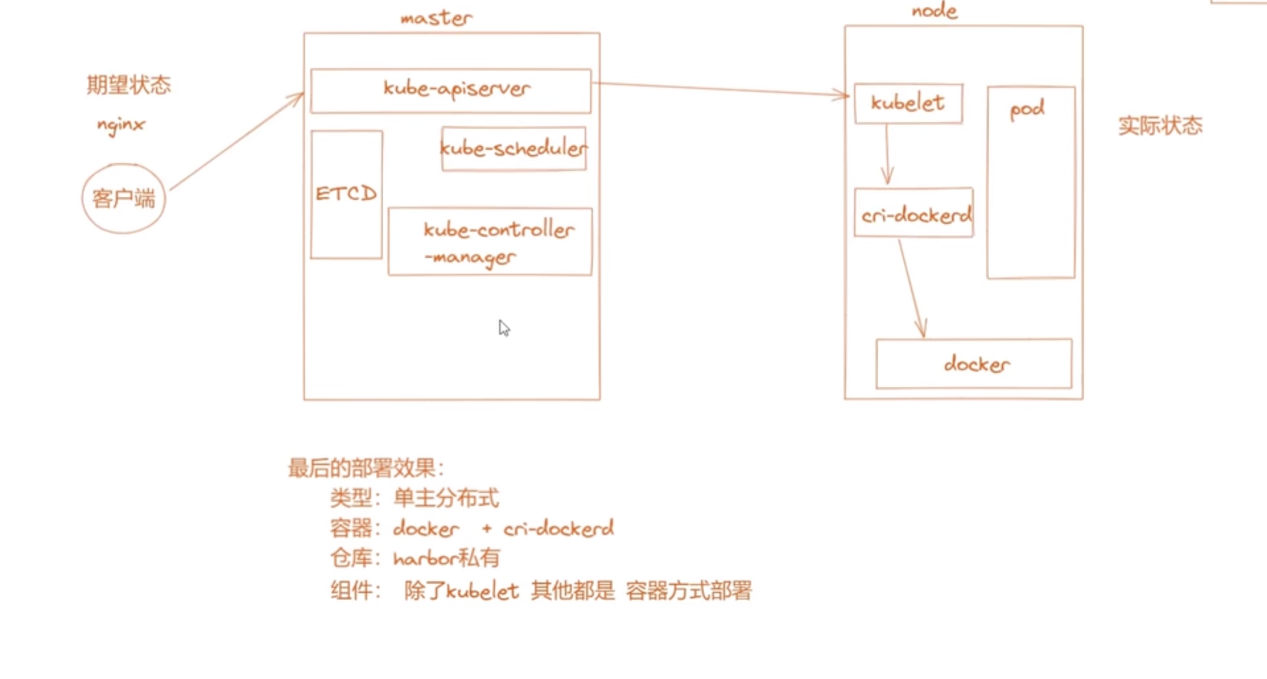
2.容器应用网络（pod网络）

3.应用服务网络（service网络）



## 二、Kubernetes 核心组件

Kubernetes 集群由控制平面和节点组成，每个部分都包含多个核心组件，它们协同工作以实现集群的各项功能。



注：客户端向 kube - apiserver 提交期望状态（如部署 nginx ），apiserver 存到 ETCD 。kube - scheduler 选节点，目标节点 kubelet 借 cri - dockerd 调用 docker 启动 Pod ，kube - controller - manager 保障状态符合期望 ，实现从声明到运行及维持的流程 。

### 控制平面组件

API Server：是 Kubernetes 集群的入口，所有的操作都通过 API Server 进行。它负责接收和处理来自用户、其他组件的请求，并将相关信息存储到 ETCD 中。

ETCD：是 Kubernetes 的数据库，专门存储集群的核心配置和状态信息，如 Pod 的定义、服务的路由规则、节点的状态等。它具有强一致性和高可用性的特点，确保集群信息的准确性和可靠性。

schedule：为pod选择合适的node。

Controller Manager：包含多种控制器，如 Deployment Controller、StatefulSet Controller、DaemonSet Controller 等。这些控制器负责维护集群的期望状态，当集群的实际状态与期望状态不一致时，控制器会采取相应的措施进行调整(管理node,一个control管理多个节点，一起就叫cluster)。

### 节点组件

Kubelet：运行在每个节点上，负责确保容器按照 Pod 的定义在节点上正常运行。它会定期向 API Server 汇报节点和容器的状态。

Kube-proxy：负责为 Pod 提供网络代理服务，实现 Pod 之间的通信以及外部网络与 Pod 之间的通信。它通过维护网络规则来实现负载均衡和服务发现等功能。

容器运行时：是运行容器的软件，如 Docker、containerd 等。Kubernetes 通过容器运行时接口（CRI）与容器运行时进行交互，实现对容器的管理。

## 三、Pod 相关知识

### 为什么创建 Pod 要先有镜像

Pod 是 Kubernetes 中最小的部署单元，里面运行的是应用程序。而 Kubernetes 并不知道应用的具体内容和运行环境，它需要一个包含应用代码、依赖和运行环境的 “打包好的应用模板”，这个模板就是容器镜像（拉取 Harbor 私有仓库的 nginx 镜像)。就像在电脑上安装软件需要安装包一样，Kubernetes 运行应用也需要容器镜像作为基础。

### Pod 的定义

Pod 是由一个或多个容器组成的集合，这些容器共享网络命名空间、存储卷等资源。它是 Kubernetes 进行部署和管理的基本单位。

### Pod 的特点

短暂性：Pod 是临时性的，当 Pod 所在的节点出现故障、资源不足或 Pod 本身出现问题时，Kubernetes 可能会销毁该 Pod 并在其他节点上重新创建一个新的 Pod。

共享资源：Pod 内的容器共享网络和存储资源，它们可以通过[localhost](https://localhost" \t "/Users/justice/Documents\\x/_blank)进行通信，并且可以访问相同的存储卷。

最小部署单元：Kubernetes 不直接部署容器，而是通过部署 Pod 来管理容器。

## 四、Harbor 相关知识

### Harbor 的作用

Harbor 是一个用于存储和管理容器镜像的工具，它就像一个容器镜像的仓库。用户可以将打包好的应用镜像上传到 Harbor 进行保存，当需要在 Kubernetes 集群中运行应用时，再从 Harbor 中拉取镜像。

### Harbor 的功能

镜像存储：提供了安全可靠的镜像存储机制，支持大量镜像的存储。

镜像管理：支持镜像的版本控制，用户可以对不同版本的镜像进行管理，如查看、删除等。同时，还可以对镜像设置访问权限，控制谁可以上传和下载镜像。

安全扫描：能够对镜像进行安全扫描，检测镜像中存在的漏洞和安全隐患，提高应用的安全性，还可以控制什么样的人具有访问权。

镜像复制：支持将镜像在不同的 Harbor 实例之间进行复制，方便在多集群环境中进行镜像分发。

## 五、控制器

控制器是 Kubernetes 中用于管理 Pod 的重要组件，它们通过监控 Pod 的状态，确保 Pod 按照期望的状态运行。

### Deployment

用于管理无状态应用，它可以确保指定数量的 Pod 副本始终运行。当需要更新应用时，可以通过 Deployment 实现滚动更新，在不中断服务的情况下完成应用的升级。同时，它还支持回滚功能，当更新出现问题时，可以快速回滚到之前的版本（控制器维持状态）。

## 六、服务发现与负载均衡

在 Kubernetes 集群中，Pod 的 IP 地址是动态变化的，当 Pod 被重新创建时，其 IP 地址可能会发生改变。为了解决这个问题，Kubernetes 提供了 Service 资源，实现了服务发现和负载均衡的功能。

### Service 的作用截屏2025-07-12 下午2.17.10

Service 为 Pod 提供了一个固定的访问入口，无论 Pod 的 IP 地址如何变化，通过 Service 都可以访问到对应的 Pod。同时，Service 还可以将请求负载均衡到多个 Pod 副本上，提高应用的处理能力。（service用来映射外部请求，通过service找到pod标签，将外部信息传输给pod）

**Pod：**运行容器的最小单元，有独立 pod\_ip:pod\_port（即 endpoint，端点）。

**Deployment：**管理 Pod 的 “控制器”，确保 Pod 数量、版本符合期望（比如维持 3 个 nginx 副本）。

**Service：**给 Pod 绑定统一访问入口（如 ClusterIP、NodePort），通过 label（标签） 关联 Pod（标签是 Kubernetes 中 “分组匹配” 的核心）。

**kube-proxy：**运行在每个节点的组件，通过 iptables 或 ipvs 规则，实现 Service 的流量转发（把访问 Service 的请求，转发到后端 Pod）。

**集群网络**：clusterIP 是集群内部虚拟 IP，仅集群内应用可访问；若要暴露到外部，需用 NodePort（映射到宿主机端口）或 Ingress

### Service 的类型

NodePort：在每个节点上开放一个静态端口，通过节点的 IP 地址和该端口可以访问 Service，适用于在集群外部访问应用。

Service，适用于在云环境中暴露服务。

ExternalName：将 Service 映射到外部的域名，通过域名可以访问外部服务。

七 最新部署方式是HELM

Helm 是 Kubernetes 的包管理器，就像是 Linux 系统中的 apt 或 yum。它能够将一组相关的 Kubernetes 资源（如 Deployment、Service、ConfigMap 等）打包成一个 Chart（类似于软件包），方便进行应用的安装、升级、回滚和管理。使用 Helm 可以极大地简化 Kubernetes 应用的部署流程，提高部署效率，同时降低因手动配置资源而出现错误的概率。