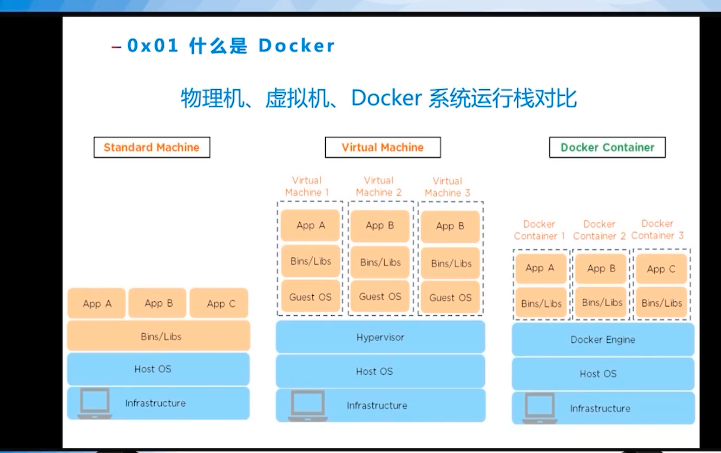
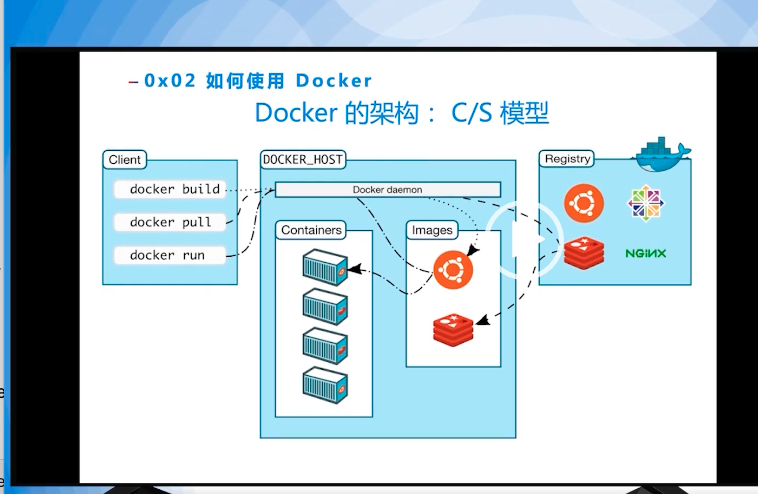
# 云原生

## tips



虚拟机多了个hypervisor 这个是个把os弄成多个硬件接口的技术,还多个了guestos.



<https://hub.docker.com/>

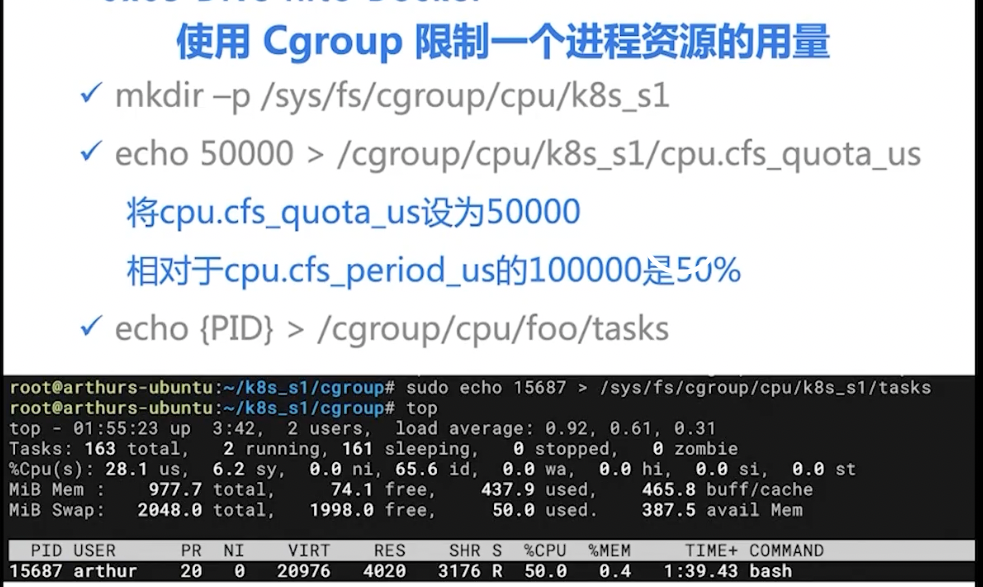
Dockerfile 构建docker镜像的语法. 后面docker build的时候会用到.

## Cgroup

Linux control group 限制一个进程组的资源,包括cpu 内存 网络io等

How much you can use

一组docker进程放到一个控制组里, 通过给这个控制组分配指定的可用资源,达到控制这一组进程可用资源的目的.



这就是个案例,一个死循环的程序占cpu100%, 然后你执行上面三个命令,把这个程序的pid写到cgroup的配置里面,这个程序的cpu就会降到50%

## Namespace

What you can see

主要做访问隔离, 即同一个命名空间的多个资源(cpu mem network)可以互相按到,但是之外的看不到.

就是给你一个单独的隔离空间, 你在这个新空间里创建的进程的pid都是从1开始的,是和其他的namespace的pid不冲突的, 和root namespace的pid都不冲突的.

## AUFS

高性能分层文件系统, 镜像可以通过分层实现继承.大大降低docker image的体积.

## DockerCompose

dockerfile: 构建单个服务镜像，以脚本形式

docker-compose：单机器多镜像编排容器

k8s：跨服务编排

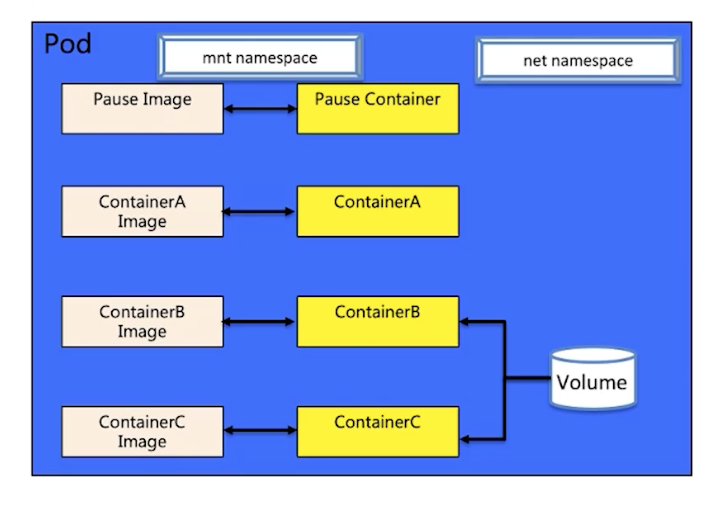
可以通过yml文件描述 一个工程下的多个service, 每个service用那个镜像或者dockerfile创建容器, 容器有多少个.

还可以更新容器配置等.

## POD

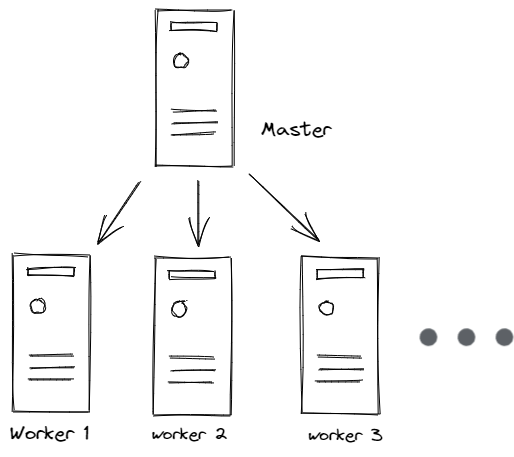
一组容器 共享namespace 原子的可部署单元

豆荚，K8S 调度、管理的最小单位，一个 Pod 可以包含一个或多个容器，每个 Pod 有自己的虚拟IP。一个工作节点可以有多个 pod，主节点会考量负载自动调度 pod 到哪个节点运行



pause是随着pod一起诞生的第一个container.

## 集群架构

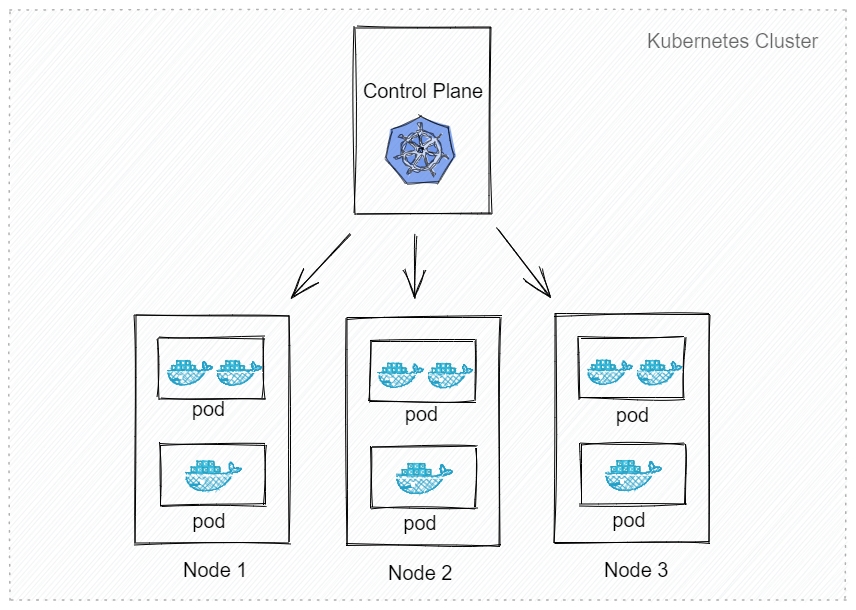


master

主节点，控制平台，不需要很高性能，不跑任务，通常一个就行了，也可以开多个主节点来提高集群可用度。

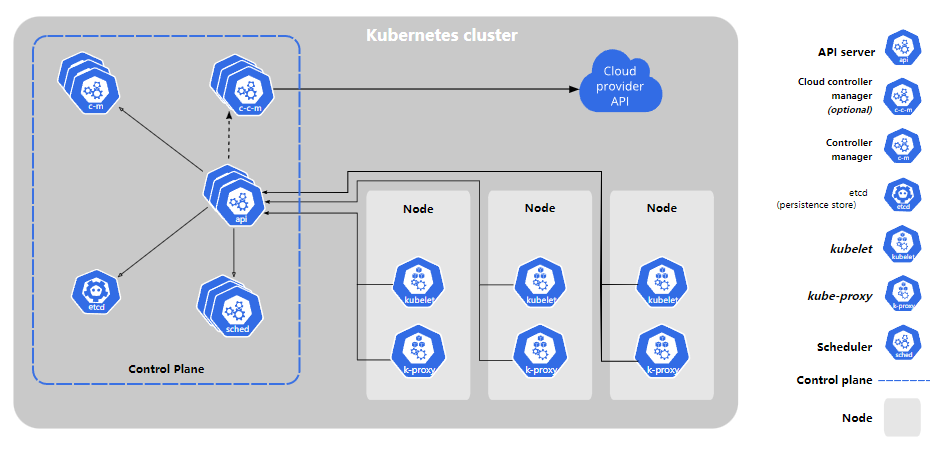
worker

工作节点，可以是虚拟机或物理计算机，任务都在这里跑，机器性能需要好点；通常都有很多个，可以不断加机器扩大集群；每个工作节点由主节点管理.



## 组件

kube-apiserver API 服务器，公开了 Kubernetes API  
etcd 键值数据库，可以作为保存 Kubernetes 所有集群数据的后台数据库  
kube-scheduler 调度 Pod 到哪个节点运行  
kube-controller 集群控制器  
cloud-controller 与云服务商交互



## Service

Service 通过 label 关联对应的 Pod

Servcie 生命周期不跟 Pod 绑定，不会因为 Pod 重创改变 IP

提供了负载均衡功能，自动转发流量到不同 Pod

可对集群外部提供访问端口

集群内部可通过服务名字访问