Day02 kubernetes笔记

今日课程主题

#1、认识kubernetes(k8s)在企业中实际的应用场景 --- 为什么要学习k8s?

#2、云技术 --- 虚拟化及虚拟机基本理念和原理

#3、云技术 --- 容器技术及容器基本原理

#4、云原生 --- 云计算、私有云、公用云、混合云

# 1、为什么要学习k8s？

1）、互联网企业可以利用k8s构建一套自动化的运维平台（运维工作：自愈、自动伸缩……） ---- 降本增效

2）、互联网项目拆分为多个服务（soa,微服务架构），这么多服务利用非常多的服务器资源（物理机，虚拟机），可以使用k8s构建一套环境，进行服务器部署，充分利用服务器资源

3）、互联网项目需求不断变更，迭代，使得项目不停进行发布版本，进行测试，进行部署 – k8s构建一套环境，实现项目无缝迁移。

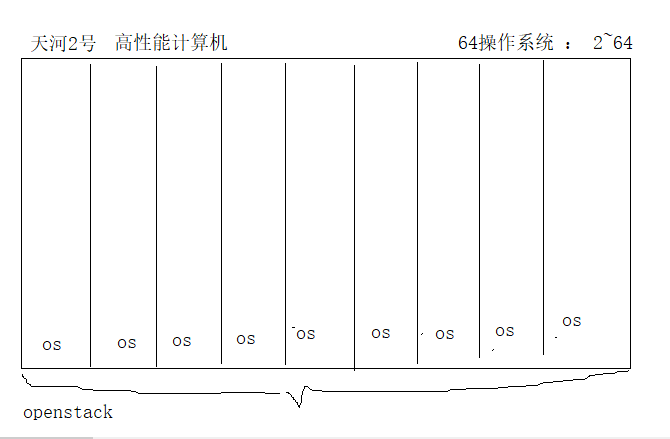
# 2、虚拟化及虚拟机基本理念和原理

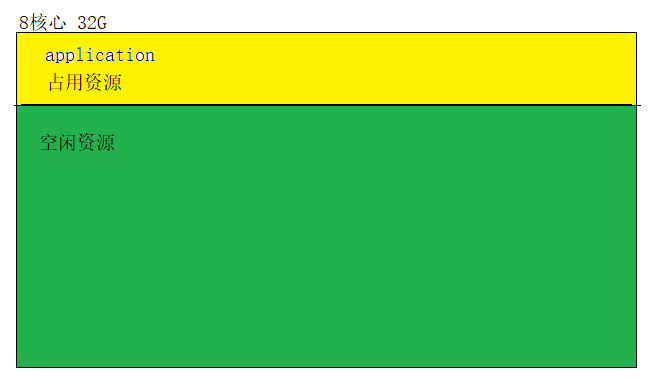
## 2.1、什么是虚拟化技术

虚拟化技术就是对物理资源（网络、硬件、CPU、内存）进行隔离一种技术。叫做虚拟化技术。可以充分利用计算机的计算资源。

## 2.2、虚拟化作用

充分利用高性能计算机资源

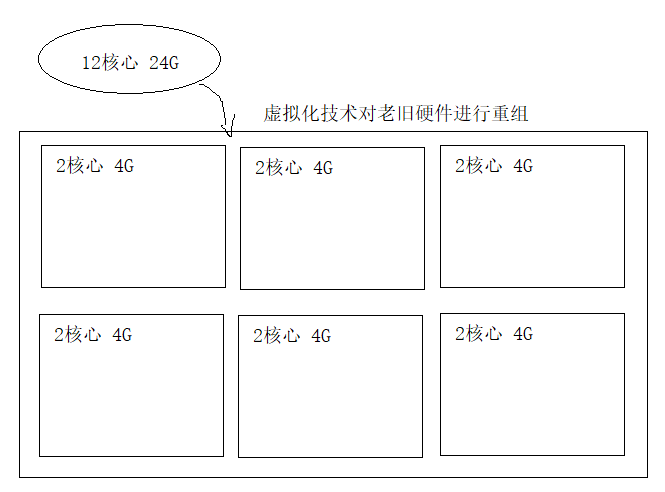




如果没有虚拟机化技术，在使用高性能计算机时候，存在大量的计算资源的浪费。使用虚拟化技术对物理资源进行隔离，把隔离的资源进行分配使用。

虚拟化技术实际上就是对物理硬件资源的一种调度。

利用老旧服务器重组再利用



总结： 虚拟化技术其实在物理硬件上安装多个操作系统，使用操作系统对资源进行隔离。（前提条件是：硬件必须支持虚拟化技术）

# 3、Docker&KVM&Openstack

## 3.1、Docker应用

Docker : 进程级别隔离

1、服务应用开发，测试，部署 --- docker容器技术解决服务从开发，测试，部署 实现 可持续部署，可持续交付 （DevOps）

2、应用开发模式是纯粹应用开发模式，可以使用容器技术

虚拟化KVM: 物理硬件隔离，隔离的更彻底

1、构建安全级别更高的私有云环境，必须使用kvm虚拟化技术

2、公有云 虚拟机技术 –- 内部服务（paas ，saas: es,mq –-- 容器化）

3、混合云 虚拟机技术

公司中：只是用于业开发，实现可持续服务交付，部署 ------- 不需要使用虚拟化技术，直接使用容器化方式，对项目进行测试，部署即可。

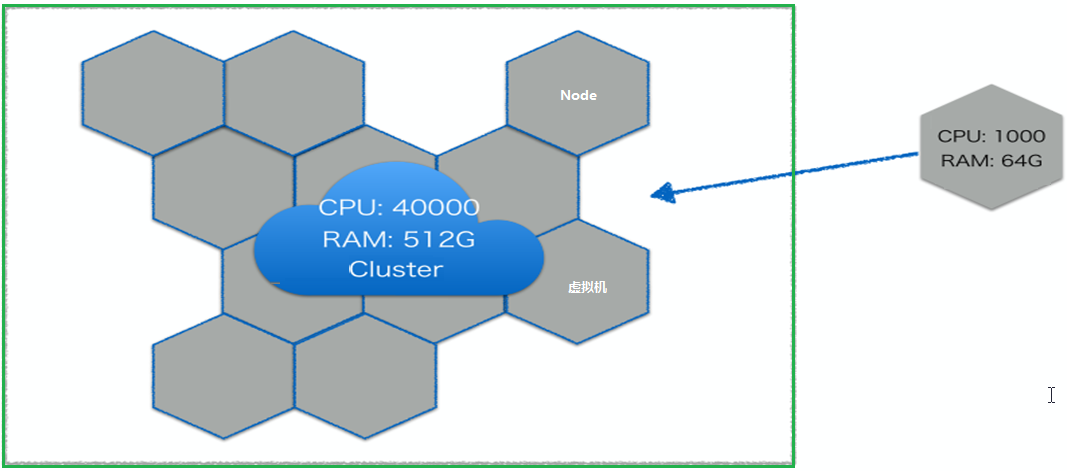
问题1：物理硬件进行虚拟化处理，使用了很多虚拟机？ 庞大的虚拟化集群网络如何管理？？

Openstack

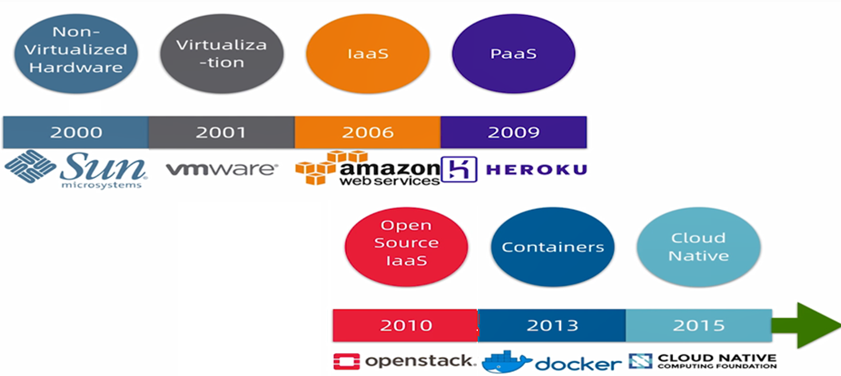
问题2：

docker部署服务（微服务架构：按照function拆分，服务非常之多………）,docker容器越来越多………,同时面临很严重的问题，管理庞大容器资源非常困难。

可以k8s对容器进行编排管理。



## 3.2、云计算平台



云计算3层结构：

1、iaas [infrastructure as a service] 基础设施即服务

公司： 服务器购买，建设机房，DNS,路由器，硬件，存储………… -- 抽象成服务提供给公司（用户）使用

云计算提供厂商：阿里云，腾讯云，私有云平台 --- 提供：存储，网络，dns，服务器硬件资源 ………

2、paas[platform as a service] 平台即服务

在iaas基础上进行了更高层次的抽象，iaas提供硬件层服务，paas提供了基础软件服务（mysql，k8s，mq）

3、saas[software as as service ]

钉钉 （saas）

微信企业版 （saas）

OA

财务

# 4、云原生

## 4.1、何为云原生

云原生：

架构：软件开发思想（软件架构思想）

应用：就是为了让应用程序（项目，MySQL、elasticsearch……）都运行在云上容器中，这样技术就叫做云原生。

思考：云原生架构 ---- 项目上云 适合云原生架构

云原生特点：

1、容器化 ： 所有都必须跑在容器中

容器部署项目，起到了隔离的作用 （k8s编排容器）

2、微服务 ： 实现云原生最好采用微服务架构

微服务按照function拆分后，可以做到高内聚，低耦合。实现CI/CD

3、DevOps

开发+运维 – 开发和运维的结合体。DevOps是一种敏捷思维，开发一种组织形式，可以实现项目可持续交付，部署。

4、CI/CD

持续交付：不停机更新

云扩展思维：

1、caas[container as a service] 容器就是一个服务，所有的软件服务都运行在容器中

2、faas[function as a service] 函数即是一个服务---微服务架构（按照function拆分）

3、service mesh[服务网格架构：服务治理—服务限流、服务降级、服务监控…… istio]

4、serverless [服务，无 🡺 无服务] 无服务架构 ，程序员开发完全不需要关系服务器的事情，只需要开发业务代码即可（CURD）

## 4.2、如何云原生

1、本地部署应用可能需要停机更新，而云原生就不需要，始终是最新的状态，支持频繁的变更。

2、本地部署应用无法进行动态扩展（动态伸缩容），云原生可以利用云资源的弹性进行自动伸缩容，从而为企业降本增效。

3、本地部署应用对物理硬件IP,网络端口有强依赖，云原生就不需要了

4、本地部署应用需要人肉运维方式，云原生实现自动化运维 （运维： 使用k8s, iaas层是运维）

实现云原生：

采用微服务架构 是 云原生环境的标配，使用云原生模式部署架构项目，项目必须采用微服务架构。微服务架构的项目离开了云原生，部署，运维的效率就大大折扣。

微服务架构：

Springcloud Alibaba

Nacos

Sentinel

dubbo

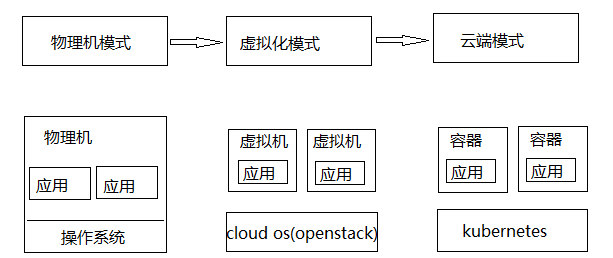
Springcoud netflix

## 4.3、服务部署演变

物理机模式： 服务部署在物理机

虚拟机模式： 为了充分利用物理机的计算资源，使用虚拟机方式部署服务

云原生模式： 应用部署在容器中



问题：虚拟机部署服务，容器部署服务 ，微服务架构（function拆分），拆分服务规模比较庞大，因此不行的虚拟机非常多，容器非常多，如何管理。

虚拟化： openstack

容器化： Kubernetes

问题：为什么要管理容器（虚拟机）？

1、怎么扩容？ (自动)

2、容器宕机了，怎么恢复？ （自动）

3、更新容器会不会影响业务？ （不影响）

4、如何监控 （自动）

5、如何调度 （自动）

6、数据安全 （自动）

……………………

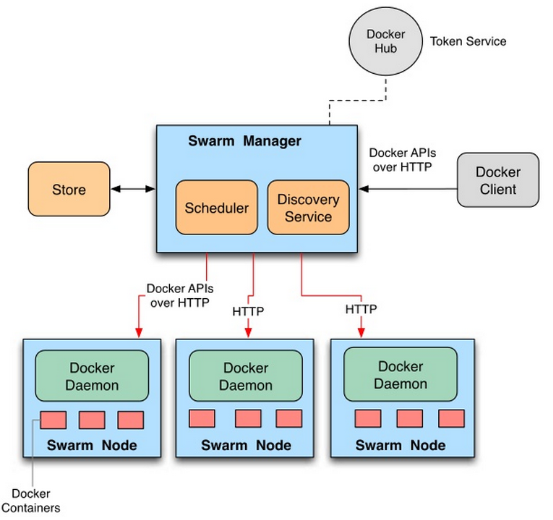
# 5、容器编排技术

## 5.1、docker-compose

Docker-compose组件可以批量的创建容器，管理容器。粗颗粒度

## 5.2、Swarm

Swarm容器编排工具是docker公司自己的开发，但是docker公司都不使用，docker公司使用的kubernetes (k8s),阿里云切换为k8s.



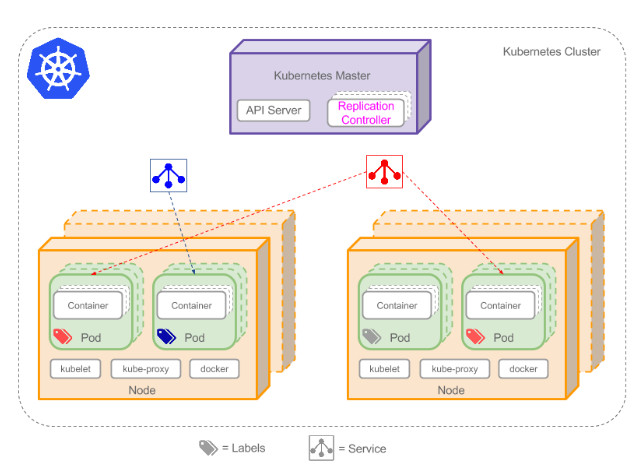
## 5.3、mesos

Mesos针对不同的运行框架采用相对独立的调度系统，其框架提供了Docker容器的原生支持。 Mesos并不负责调度而是负责委派授权，毕竟很多框架都已经实现了复杂的调度。

Mesos是Apache企业的开源项目，是一个资源管理，用来进行容器的管理。

## 5.4、kubernetes

Kubernetes则采用了Pod和Label这样的概念把容器组合成一个个的互相存在依赖关系的逻辑单元。相关容器被组合成Pod后被共同部署和调度，形成服务（Service）



Kubernetes基本结构：

1、master节点： 负责调度，存储集群状态（服务注册发现），提供统一API入口…,一个master对应一群node节点

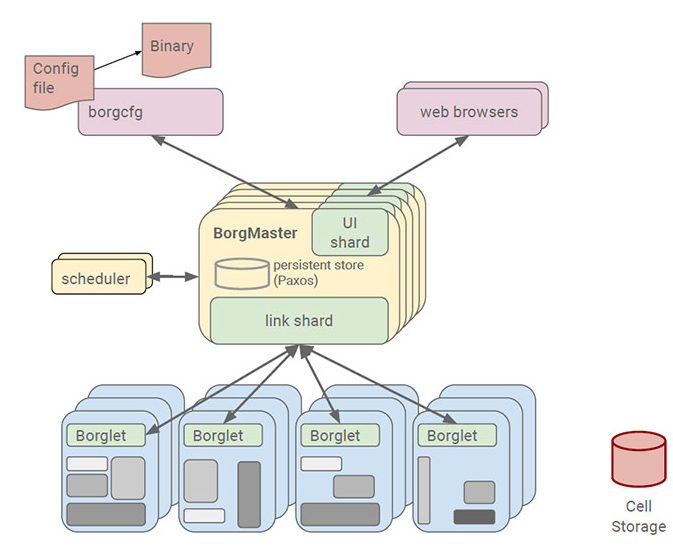
2、node节点：node节点存储pod(pod内部封装容器的)，一个node节点理论上可以存储无数个pod,但是node节点存储pod的数量受限于硬件资源的限制，同时受限于pod内部服务运行所占用的资源。

# 6、Kubernetes

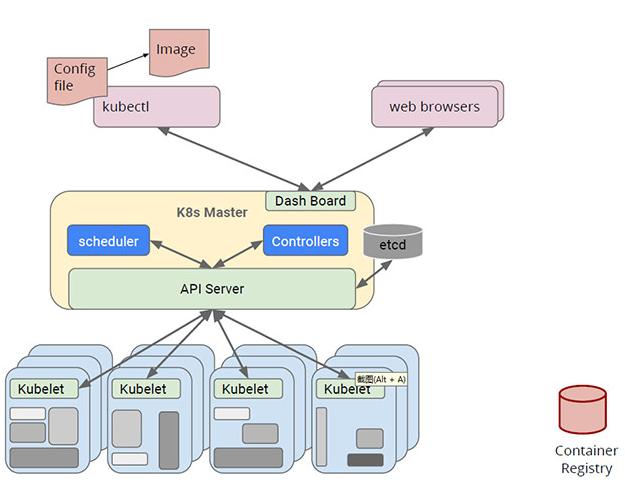
## 6.1、Borg系统

Borg系统是google公司研发的一套服务管理软件，borg系统是kubernetes的前身，kubernetes也是google公司开发的，架构设计思想是参考borg系统来件架构设计的。

Google在十几年前已经开发应用：borg系统，容器化管理方式。



## 6.2、K8s



发送请求： kubectl 客户端指令 ，浏览器（可视化方式：rancher,dashboard..）

Master节点： scheduler 调度器 ，负责计算该把pod调度到哪一个node节点

Controllers ： 控制器 ，负责维护node节点资源对象（deployment,rs,pod）

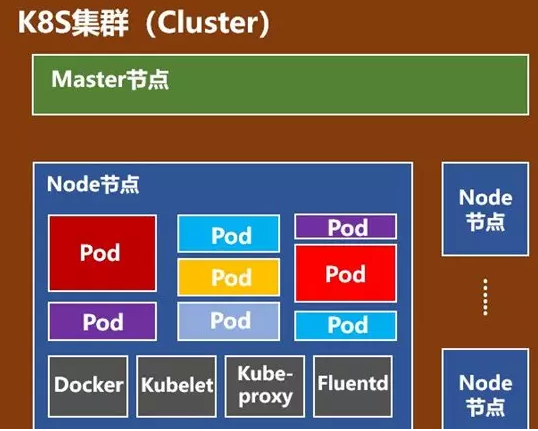
apiServer : 网关，所有请求都必须要经过网关

etcd: 服务发现，注册，集群状态信息，调度信息

node节点： 每一个node节点都运行一个kubelet进程，此进程负责本机服务的pod的创建维护。

Regitry: 镜像仓库—阿里镜像仓库—harbor构建一套自己的私有仓库

## 6.3、Node节点



Pod: 是k8s管理的最小的基本单元。Pod内部可以运行一个或者多个容器。一般情况下，pod内部只允许一个容器运行，便于管理。

Docker： docker引擎，pod内部运行的都是容器，这个容器是由docker引擎创建的，docker引擎是node节点基础服务。

Kubelet: node节点代理，kubelet代理master节点请求，在本地node节点执行；

Kube-proxy: 网络代理 ，主要是用来生成网络规则，创建访问路由，创建service网络访问规则，负载均衡规则。

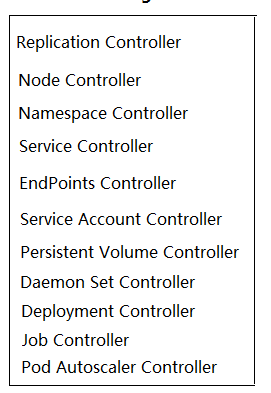
Fluentd: 日志，收集日志

## 6.4、Master节点

1、ApiServer

集群的统一入口，各组件协调者，以HTTP API提供接口服务，所有对象资源的增删改查和监听操作都 交给APIServer处理后再提交给Etcd存储

2、controllers

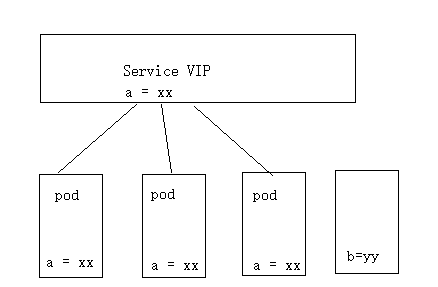


1）replication Controller : 副本控制器 --- 实现副本数量和预期设定的数量永远保持一致。

例如： 构建集群 ： 6台服务器集群 --- k8s部署：预期设定数量=6 ，k8s能永远保证副本数据量一致等于6.

2）Service Controller: 管理维护Service（虚拟IP），提供负载以及服务代理。

3）EndPoints Controller: 管理维护Endpoints，关联Service和Pod



4）Persistent Volume Controller ： 持久化数据卷控制器

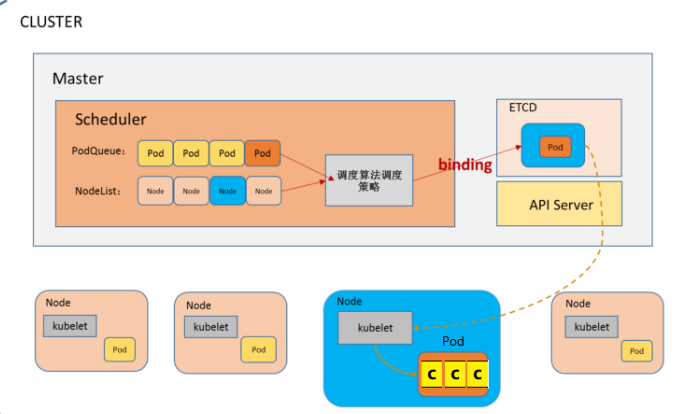
有状态服务部署控制器

5）Daemon Set Controller：让每一个node节点都运行相同的服务

6）Deployment Controller ： 无状态服务部署的控制器（项目）

3、scheduler

Scheduler就是k8s的调度器。



创建pod的流程：

1、kubeclt 发送创建的pod的指令，此时这个指令被apiserver拦截，把创建的pod存储在etcd

2、schduler 发起调用请求，此时这个指令被apiserver拦截，获取etcd中 podQueue.NodeList

调度算法：

1）预选调度

2）优选策略

选择出一个合适的node节点

3、把选择合适的node,pod存储在etcd

4、node节点上有一个kubelet进程，发送请求获取pod,node对应创建资源

5、此时如果kubelet发现pod是本机节点需要创建的，kubelet就开始创建pod