法律声明

- 本课件包括：演示文稿，示例，代码，题库，视频和声音等，讲师及小象学院拥有完全知识产权的权利；只限于善意学习者在本课程使用，不得在课程范围外向任何第三方散播。任何其他人或机构不得盗版、复制、仿造其中的创意，我们将保留一切通过法律手段追究违反者的权利。

- 课程详情请咨询
  - 微信公众号：小象
  - 新浪微博：ChinaHadoop

Kubernetes基本介绍

目录

1.Minikube的安装使用

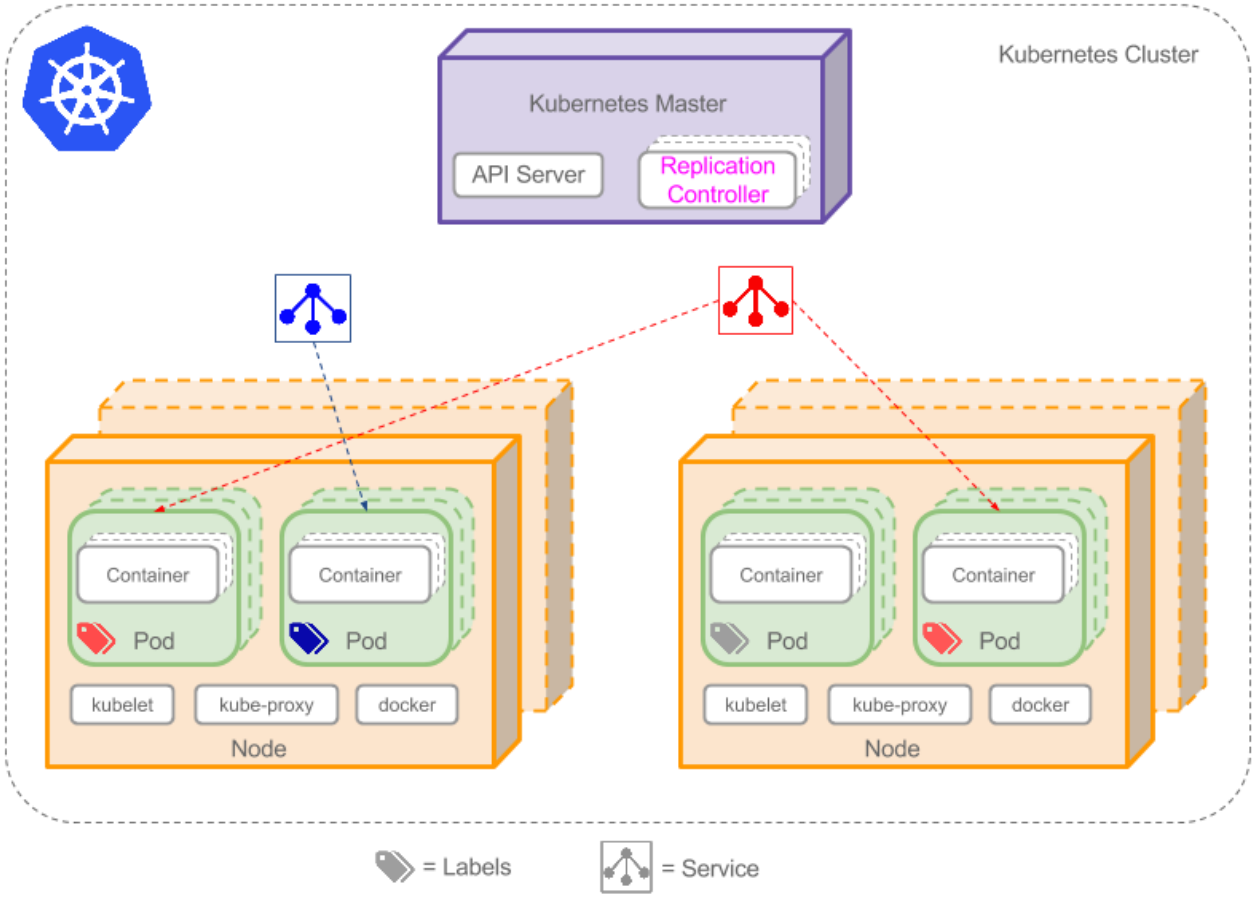
https://github.com/kubernetes/minikube

- **What is Minikube?**
- Minikube is a tool that makes it easy to run Kubernetes locally. Minikube runs a single-node Kubernetes cluster inside a VM on your laptop for users looking to try out Kubernetes or develop with it day-to-day.

1.Minikube的安装使用

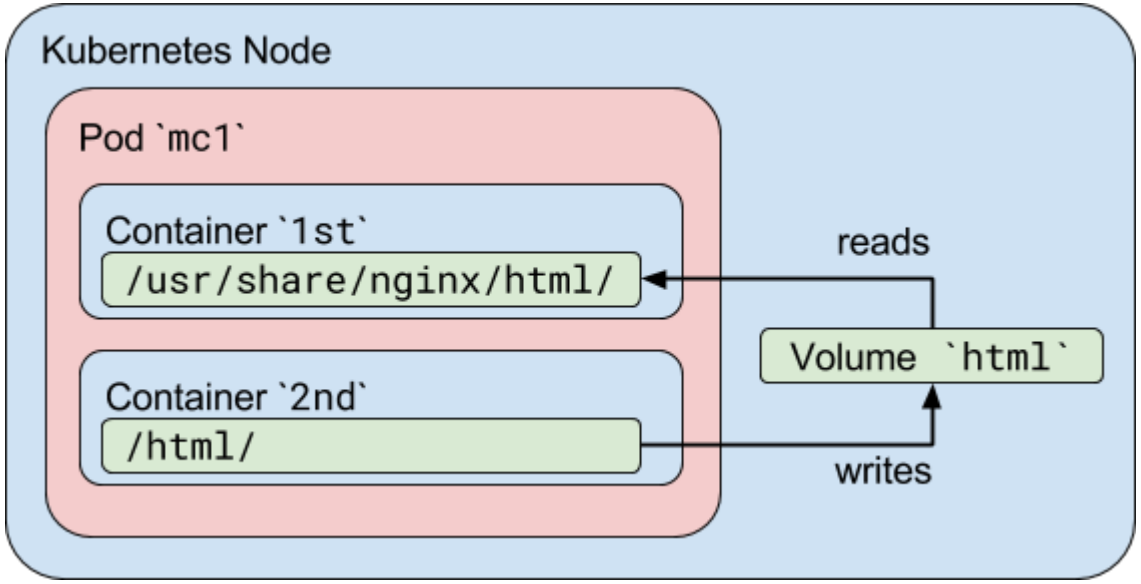
1. 安装docker
2. 从github上下载minikube二进制
3. 墙内安装kubectl
   - wget http://mirrors.ustc.edu.cn/kubernetes/apt/pool/kubectl_1.9.1-00_amd64_bdfb1ad90e0f02a7ae614502079a87ed99209679bdedf0c62873564c186c9f99.deb
   - dpkg-deb -x kubectl.deb tmp
4. 启动minikube，绕过gcr.io下无法下载镜像的问题
   - 观察日志/var/lib/localkube/*.err
   - docker pull kubernetes/pause
   - docker pull grrywlsn/kubernetes-addon-manager:v1.8.0

2.k8s提供的抽象



EN	CN
Container	容器
Pod	容器组
ReplicaSet	复本集合
Service	服务
Label	标签
Node	节点

2.k8s提供的抽象：Pod



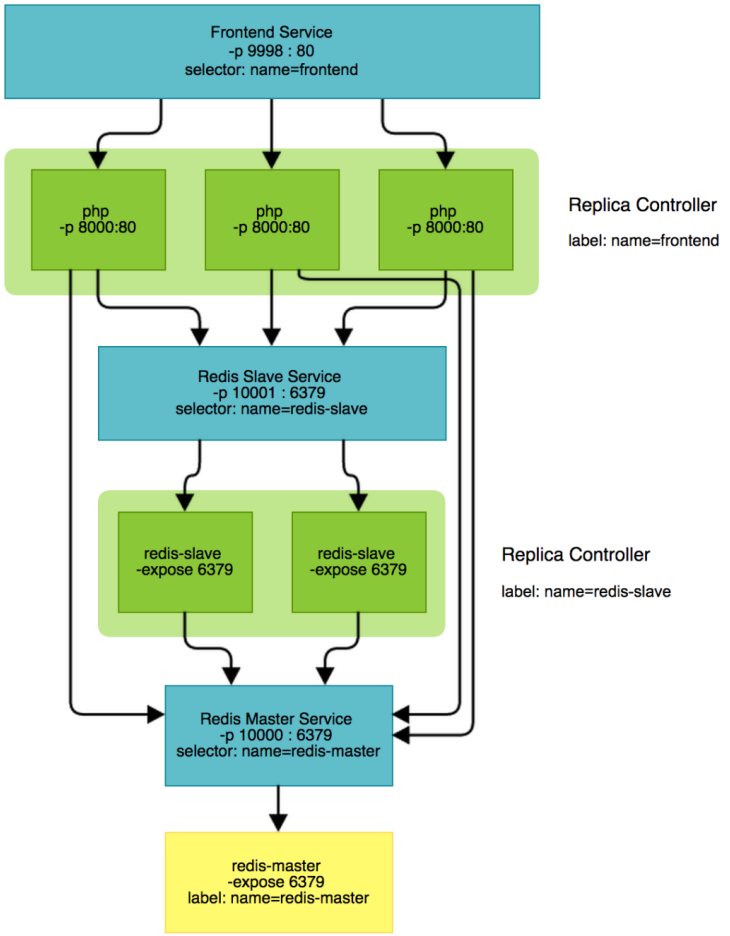
- Pod是在Kubernetes集群中运行部署应用或服务的最小单元，它是可以支持多容器的。Pod的设计理念是支持多个容器在一个Pod中共享网络地址和文件系统，可以通过进程间通信和文件共享这种简单高效的方式组合完成服务。Pod对多容器的支持是K8最基础的设计理念。比如你运行一个操作系统发行版的软件仓库，一个Nginx容器用来发布软件，另一个容器专门用来从源仓库做同步，这两个容器的镜像不太可能是一个团队开发的，但是他们一块儿工作才能提供一个微服务；这种情况下，不同的团队各自开发构建自己的容器镜像，在部署的时候组合成一个微服务对外提供服务。

- Pod是Kubernetes集群中所有业务类型的基础，可以看作运行在K8集群中的小机器人，不同类型的业务就需要不同类型的小机器人去执行。目前Kubernetes中的业务主要可以分为长期伺服型（long-running）、批处理型（batch）、节点后台支撑型（node-daemon）和有状态应用型（stateful application）；分别对应的控制器为Deployment、Job、DaemonSet和PetSet。

2.k8s提供的抽象：副本集（Replica Set，RS）

Replica Set：Kubernetes集群中保证Pod高可用的API对象。通过监控运行中的Pod来保证集群中运行指定数目的Pod副本。指定的数目可以是多个也可以是1个；少于指定数目，RS就会启动运行新的Pod副本；多于指定数目，RS就会杀死多余的Pod副本。即使在指定数目为1的情况下，通过RS运行Pod也比直接运行Pod更明智，因为RS也可以发挥它高可用的能力，保证永远有1个Pod在运行。RS适用于长期伺服型的业务类型，比如提供高可用的Web服务。

2.k8s提供的抽象：服务（Service）



RS只是保证了支撑服务的微服务Pod的数量，但是没有解决如何访问这些服务的问题。一个Pod只是一个运行服务的实例，随时可能在一个节点上停止，在另一个节点以一个新的IP启动一个新的Pod，因此不能以确定的IP和端口号提供服务。要稳定地提供服务需要服务发现和负载均衡能力。服务发现完成的工作，是针对客户端访问的服务，找到对应的的后端服务实例。在K8集群中，客户端需要访问的服务就是Service对象。每个Service会对应一个集群内部有效的虚拟IP，集群内部通过虚拟IP访问一个服务

2.理解 kubernetes 中的对象

```
apiVersion: apps/v1beta1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 3
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9
        ports:
        - containerPort: 80
status:
```

1. Metadata:标识API对象，每个对象都至少有3个元数据：namespace，name和uid；除此以外还有各种各样的标签labels用来标识和匹配不同的对象，例如用户可以用标签env来标识区分不同的服务部署环境，分别用env=dev、env=testing、env=production来标识开发、测试、生产的不同服务

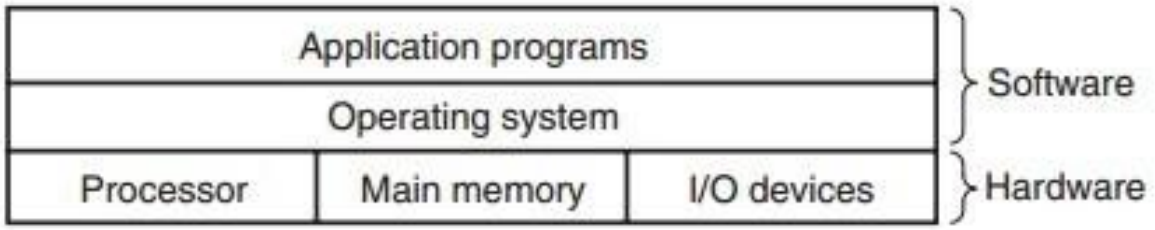
2. Spec：描述了用户期望Kubernetes集群中的分布式系统达到的理想状态（Desired State），例如用户可以通过复制控制器Replication Controller设置期望的Pod副本数为3

3. Status:系统实际当前达到的状态（Status），例如系统当前实际的Pod副本数为2；那么复制控制器当前的程序逻辑就是自动启动新的Pod，争取达到副本数为3

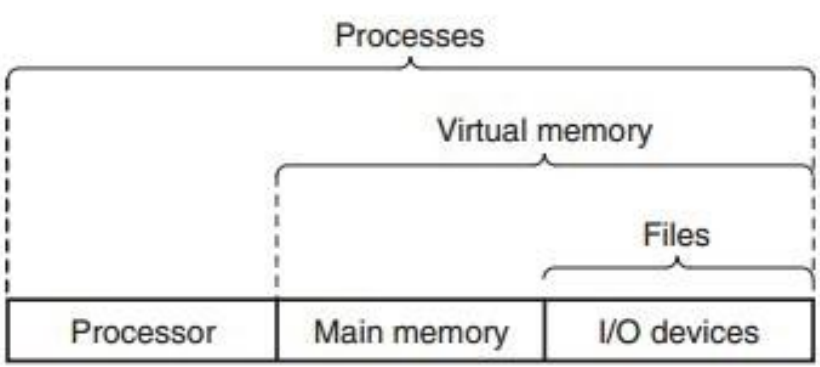
2.对象分类

类别	名称
资源对象	Pod、ReplicaSet、ReplicationController、Deployment、StatefulSet、DaemonSet、Job、CronJob、HorizontalPodAutoscaling
配置对象	Node、Namespace、Service、Secret、ConfigMap、Ingress、Label、ThirdPartyResource、 ServiceAccount
存储对象	Volume、Persistent Volume
策略对象	SecurityContext、ResourceQuota、LimitRange

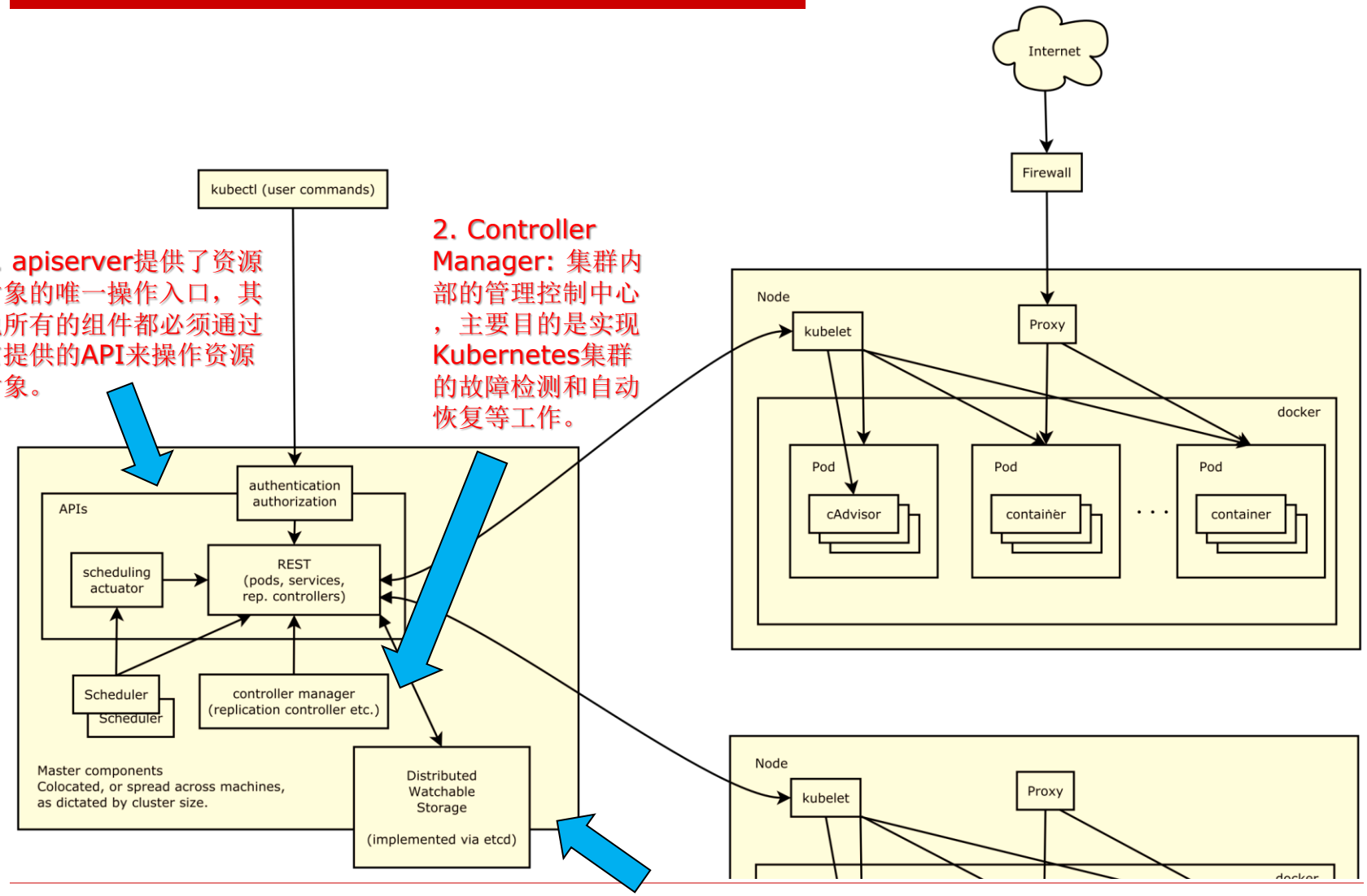
2.k8s提供的对象 vs 操作系统的抽象



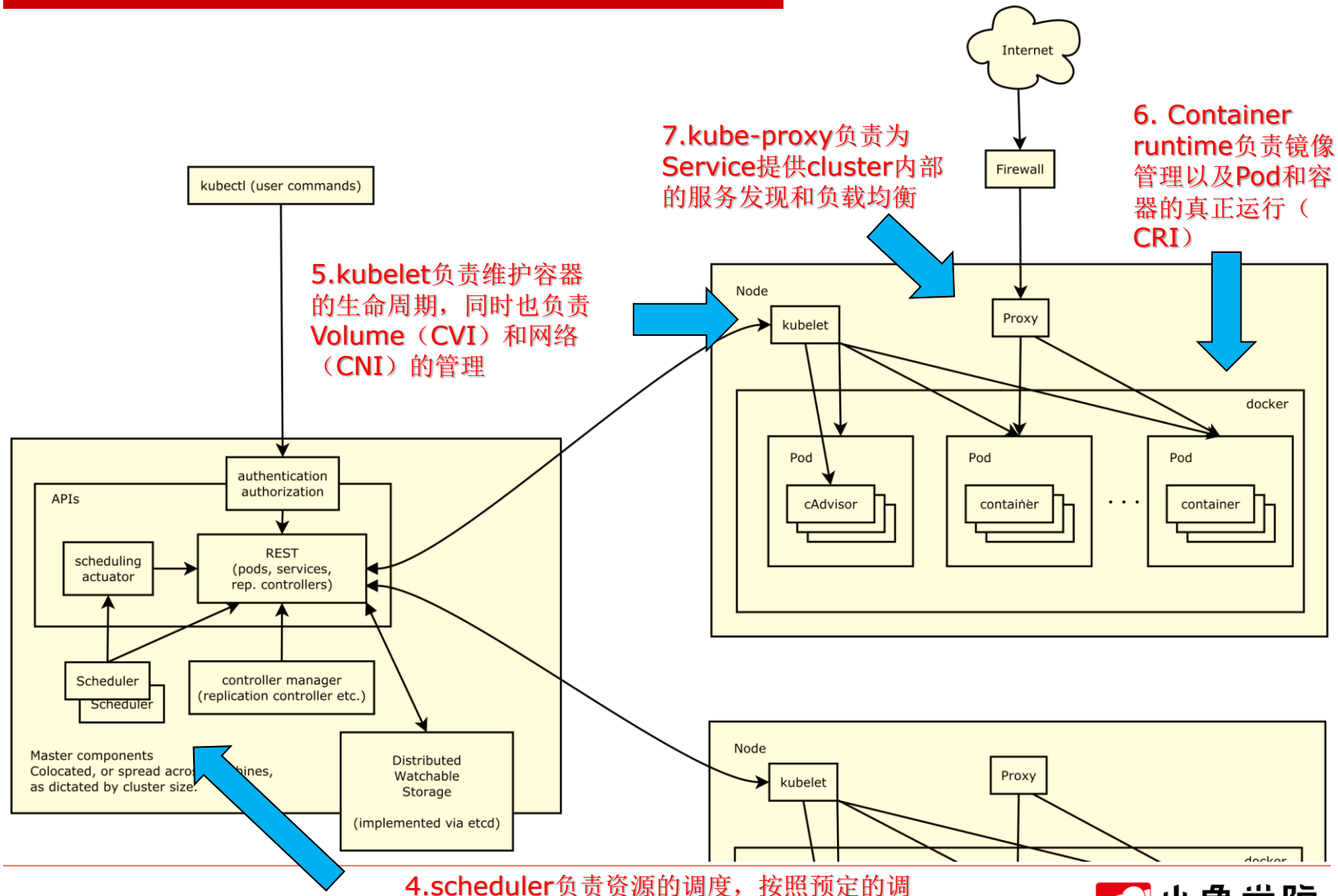
?



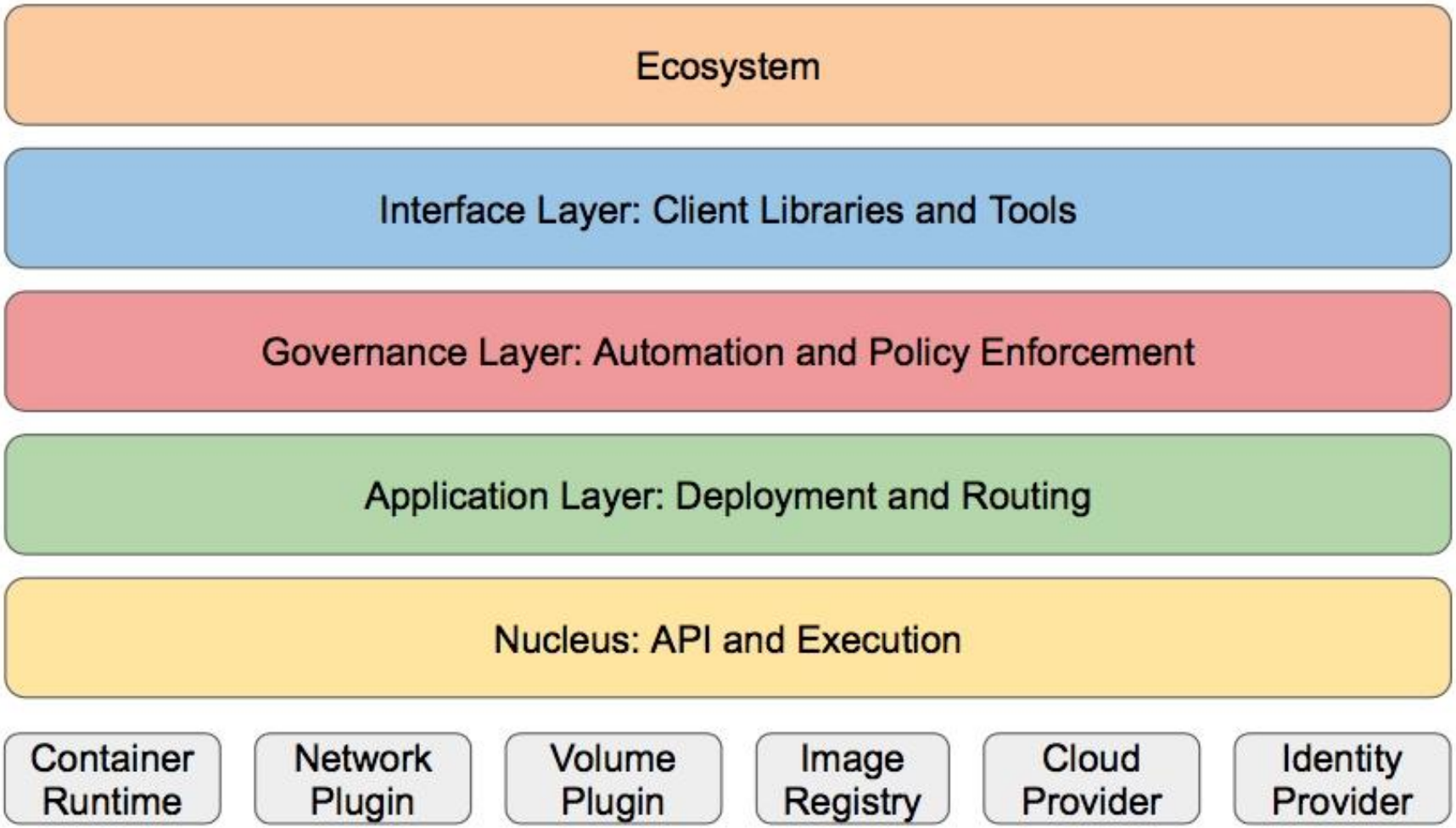
3.k8s 架构简介



3.k8s 架构简介



3.k8s 架构简介：分层架构



4.scheduler负责资源的调度，按照预定的调度策略将Pod调度到相应的机器上

3.k8s 架构简介：分层架构

1. 核心层：Kubernetes最核心的功能，对外提供API构建高层的应用，对内提供插件式应用执行环境
2. 应用层：部署（无状态应用、有状态应用、批处理任务、集群应用等）和路由（服务发现、DNS解析等）
3. 管理层：系统度量（如基础设施、容器和网络的度量），自动化（如自动扩展、动态Provision等）以及策略管理（RBAC、Quota、PSP、NetworkPolicy等）
4. 接口层：kubectl命令行工具、客户端SDK以及集群联邦
5. 生态系统：在接口层之上的庞大容器集群管理调度的生态系统，可以划分为两个范畴
   - Kubernetes外部：日志、监控、配置管理、CI、CD、Workflow、FaaS、OTS应用、ChatOps等
   - Kubernetes内部：CRI、CNI、CVI、镜像仓库、Cloud Provider、集群自身的配置和管理等

4.scheduler负责资源的调度，按照预定的调度策略将Pod调度到相应的机器上

4.k8s 架构设计优缺点：优点

1. 容错性：保证Kubernetes系统稳定性和安全性的基础
2. 易扩展性：保证Kubernetes对变更友好，可以快速迭代增加新功能的基础。
   - API分版本，API可自由扩展(CRD)
   - 插件化，调度器，容器运行时，存储均可扩展
3. 声明式（Declarative）的而不是命令式（Imperative）：声明式操作在分布式系统中的好处是稳定，不怕丢操作或运行多次，例如设置副本数为3的操作运行多次也还是一个结果，而给副本数加1的操作就不是声明式的，运行多次结果就错了。

4.k8s 架构设计优缺点：缺点

1. 配置中心化：所有状态都保存在中心的etcd上，而非分布式存储，性能有一定制约
2. 单体调度：调度一致性好而吞吐低

(a) Monolithic scheduler. (b) Two-level scheduling. (c) Shared-state scheduling. (d) Distributed scheduling. (e) Hybrid scheduling.

Figure 1: Different cluster scheduler architectures. Gray boxes represent cluster machines, circles correspond to tasks and S_i denotes scheduler i.

（a）单体式调度器（b）二级调度（c）共享状态调度（d）分布式调度 (e) 混合式调度

5. 运行第一个应用

```
$ docker pull nginx
$ kubectl run hello-minikube --image=nginx --port=80
deployment "hello-minikube" created
$ kubectl expose deployment hello-minikube --type=NodePort
service "hello-minikube" exposed
```

现场观察Pod/Service/Node等情况。。。。

作业

1. 在自己的环境中部署一套minikube
2. 参考这篇文章，使用kubeadm在3节点上安装一套k8s
3. 翻墙后，使用play with k8s网站，启动k8s环境
4. 使用rancher提供的方法，安装k8s

以上作业可以任意选择一个，作为后续学习k8s的基础环境

联系我们

小象学院：互联网新技术在线教育领航者

- 微信公众号：大数据分析挖掘
- 新浪微博：ChinaHadoop