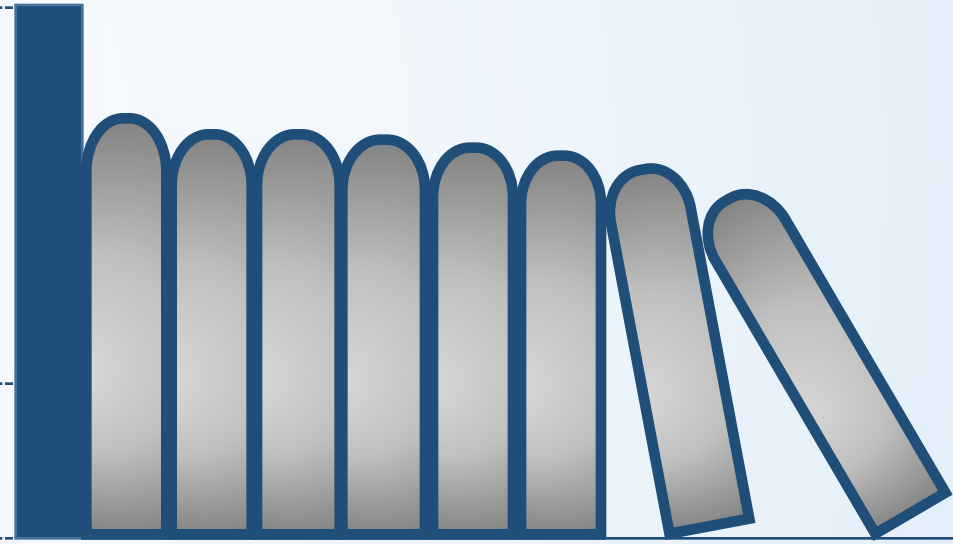
# Kubernetes基础

主讲人：宋小金




# 目录

# 1 预期收获

- 对Kubernetes有初步认识
- 了解Kubernetes的架构
- 了解Kubernetes的组成
- 了解Kubernetes基本术语

# Kubernetes简要介绍

- 首先说下Borg，它是Google内部使用的一个超大规模集群管理系统，它基于容器技术，在Google内部使用已有超过10年的历史，目的是实现资源管理自动化，以及跨多个DC的资源利用率最大化。
- Kubernetes（简称K8s）可理解为Borg的一个开源版本，是一个基于Docker容器技术的分布式架构，于2015年4月首次被公开，其吸取了Borg的经验和教训，开源后迅速成为容器技术领域的领头羊。
- Kubernetes是一个开放的开发平台，无论是Go、Java或Python编写的服务，都可以轻松映射为Kubernetes的Service，并通过标准的TCP通信协议进行交互，因此现有系统可以非常容易迁移到Kubernetes平台上。


# Kubernetes有什么好处？

- 如果系统设计遵循了Kubernetes的设计理念与思想，我们不必费心于*服务监控和故障处理*，使用Kubernetes可以*节约大量的开发成本*，可以使更多精力集中于业务本身。
- 另外Kubernetes提供了强大的*自动化部署与运维功能*，使系统的后期的运维难度和成本大大降低。
- Kubernetes具备完善的集群管理能力，包括多层次的安全防护与准入机制、多租户支撑能力、4/7层服务发现、内置负载均衡器、强大的故障发现与自我修复能力、简单易用的服务部署、升级与回滚、优良的调度策略，以及资源配额管理等。


# Kubernetes的核心 -- 编排

- 弹性资源池，命名空间，多租户，额度管理，资源隔离
- Service服务发现，应用实例副本控制，探针机制保障应用高可用
- Deployment/StatefullSet/DaemonSet等，发布部署机制，支持有/无状态应用上线，更新，回滚，多种发布策略
- 存储自动挂载，资源自动选择，拓扑亲和/反亲和
- 轻量化部署，高部署密度，应用驱逐机制，提升资源利用率

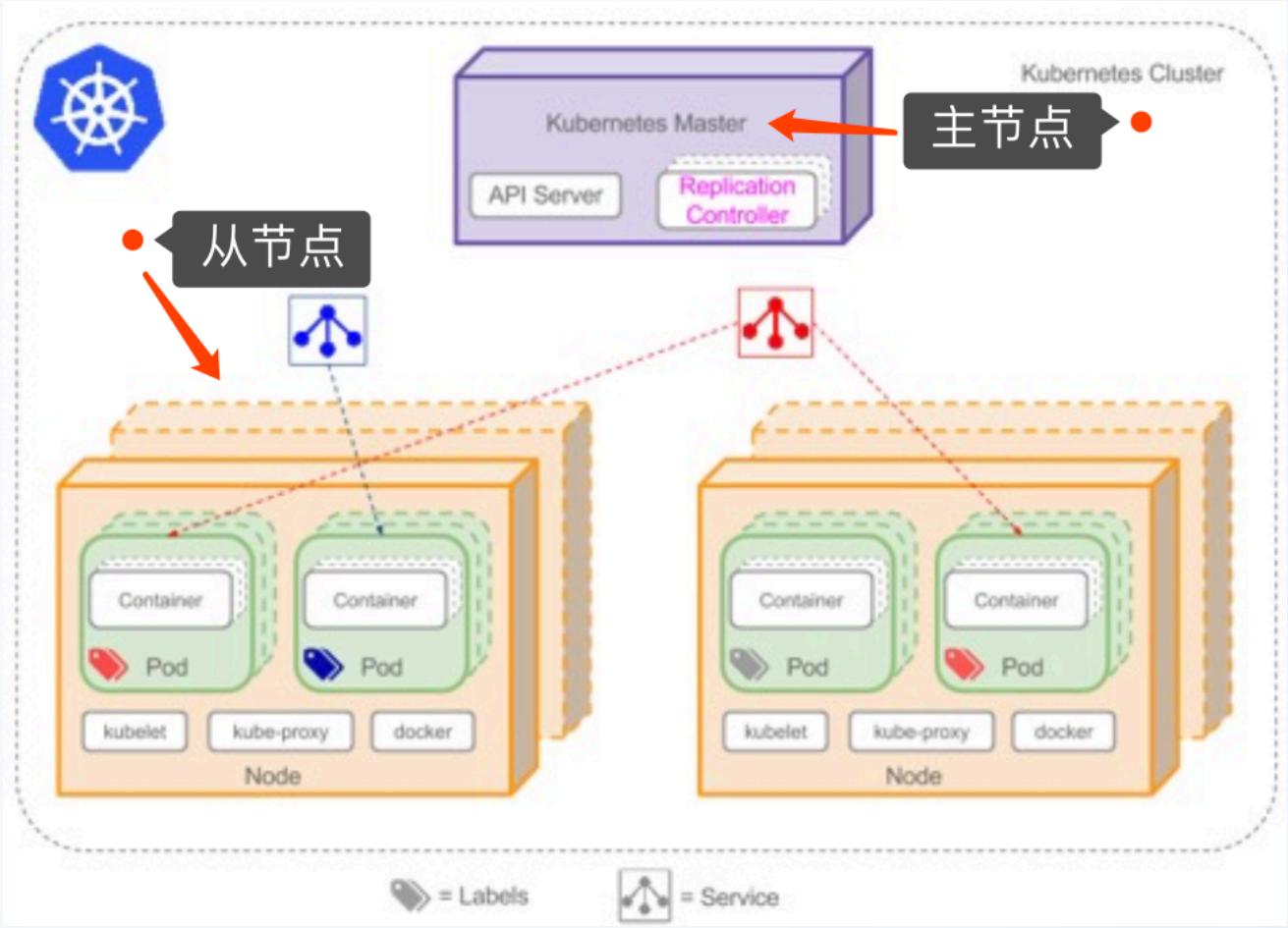

# 为什么用Kubernetes？

Kubernetes*全面拥抱微服务架构*，采用Kubernetes可以轻装上阵的开发一个复杂的系统。

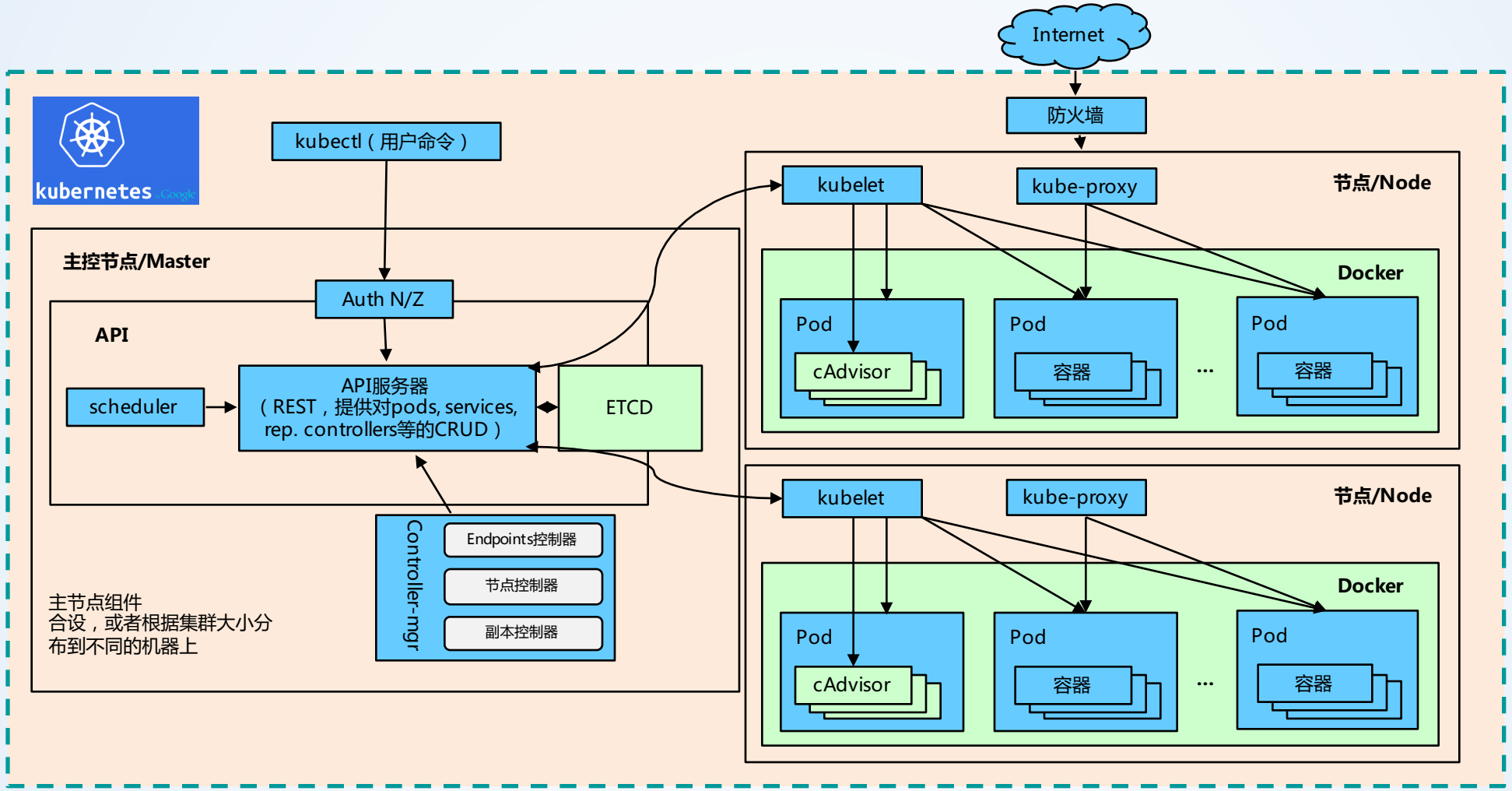
举例：以前一个几十人组成的团队且需要不少技术达人一起分工协作才能实现的和运维的分布式系统，在采用Kubernetes解决方案之后，只需要一个精悍的小团队就可以轻松应对。在这个团队，*一名架构师*负责专注于系统中”服务组件”的提炼，*几名工程师专注于业务代码*的开发，*一名系统兼运维工程师*负责Kubernetes的部署和运维，这并不是我们少做了什么，而是Kubernetes帮我们做了很多。

# Kubernetes架构

Kubernetes架构图示



# Kubernetes架构




# Kubernetes组成（Master）

- **Master Node**
  - Kubernetes集群的管理中心，调度中心，它包括如下几个组件
    - **API Server**：
      - 提供了K8s各种对象(Pod，RS，Service等)的增，删，改，查/ Watch等rest接口
      - 是K8s集群管理的入口，是K8s资源配额控制的入口。
    - **Etcd**：
      - 分布式强一致性的key/value分布式存储，存储k8s的数据和状态信息
        - Jobs信息
        - Pod/service信息,状态信息
        - Namespace和RS信息等


# Kubernetes组成（Master）

- **Master Node**
  - Kubernetes集群的管理中心，调度中心，它包括如下几个组件
    - **Scheduler**：
      - 主要工作：watch apiserver新建Pod信息，一旦发现有待调度的Pod，则按照特定的调度算法和调度策略绑定到合适的Node，并把绑定信息写入到etcd
    - **Controller Manager**：
      - 集群的管理控制中心，通过执行各种控制器，负责整个集群内的Node，Pod，Replica，Endpoint，Namespace，ServiceAccount以及ResourceQuota等管理

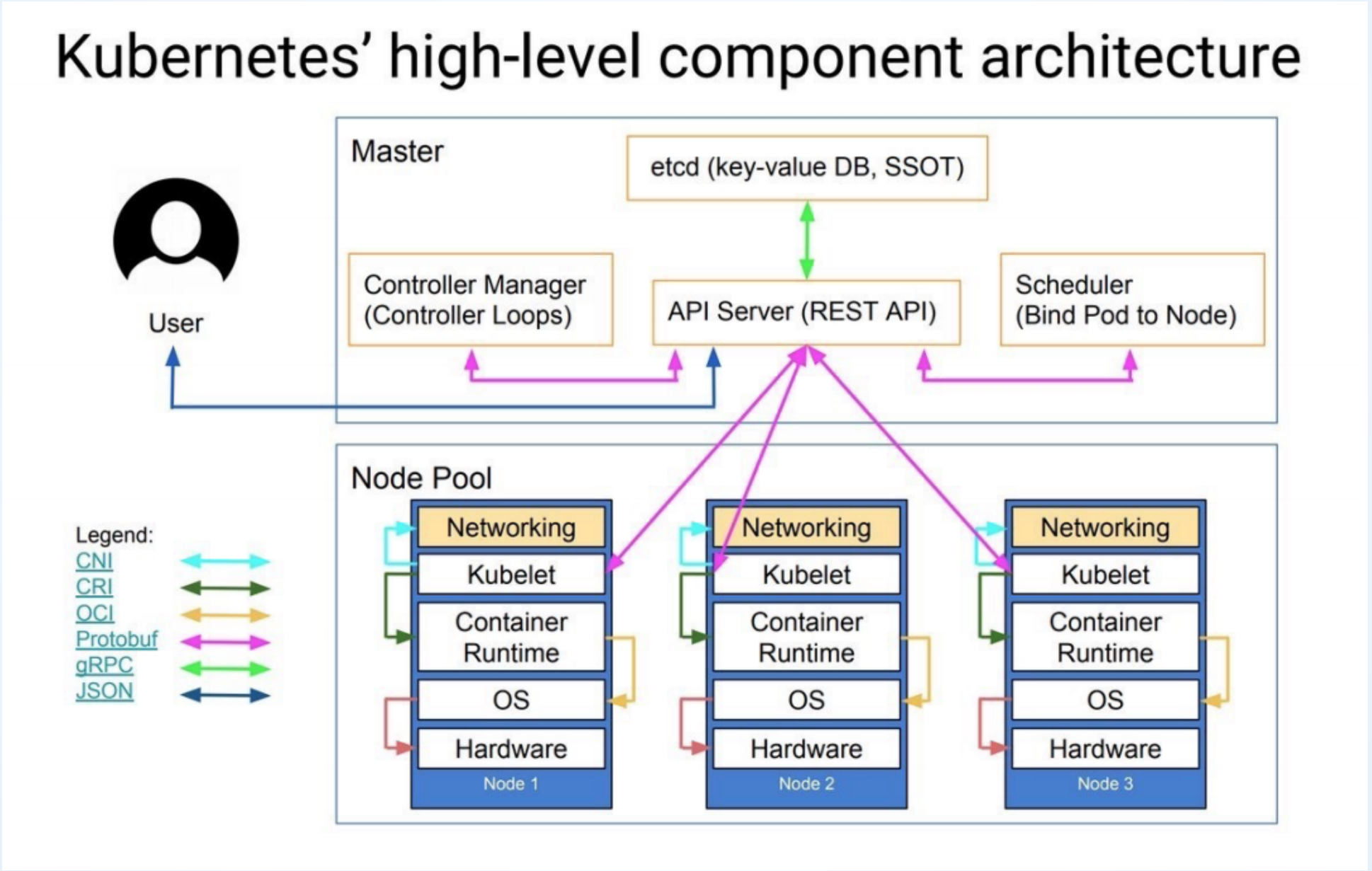

# Kubernetes组成（Worker）

- **Worker node**
  - **Kubelet**：
    - 负责管控容器的生命周期，如启动/停止，创建/销毁、监控运行状态等。它从api server查询node需要创建的Pod信息，进行Pod创建的相关处理(给Pod挂载volumes,下载POD所需要的secrets,Run 容器,周期性进行容器存活检测,汇报容器状态,汇报Node状态，管理网络)
  - **Kubectl**：
    - 命令工具，和api server交互对集群进行管理。
  - **Kube-proxy**：
    - Kube-proxy通过apiserver 拿到service的Endpoints对象信息
    - 给service提供网络代理和负载均衡服务
  - **Docker**：
    - K8s默认支持的容器是Docker，当然K8s也支持Rocket容器技术。

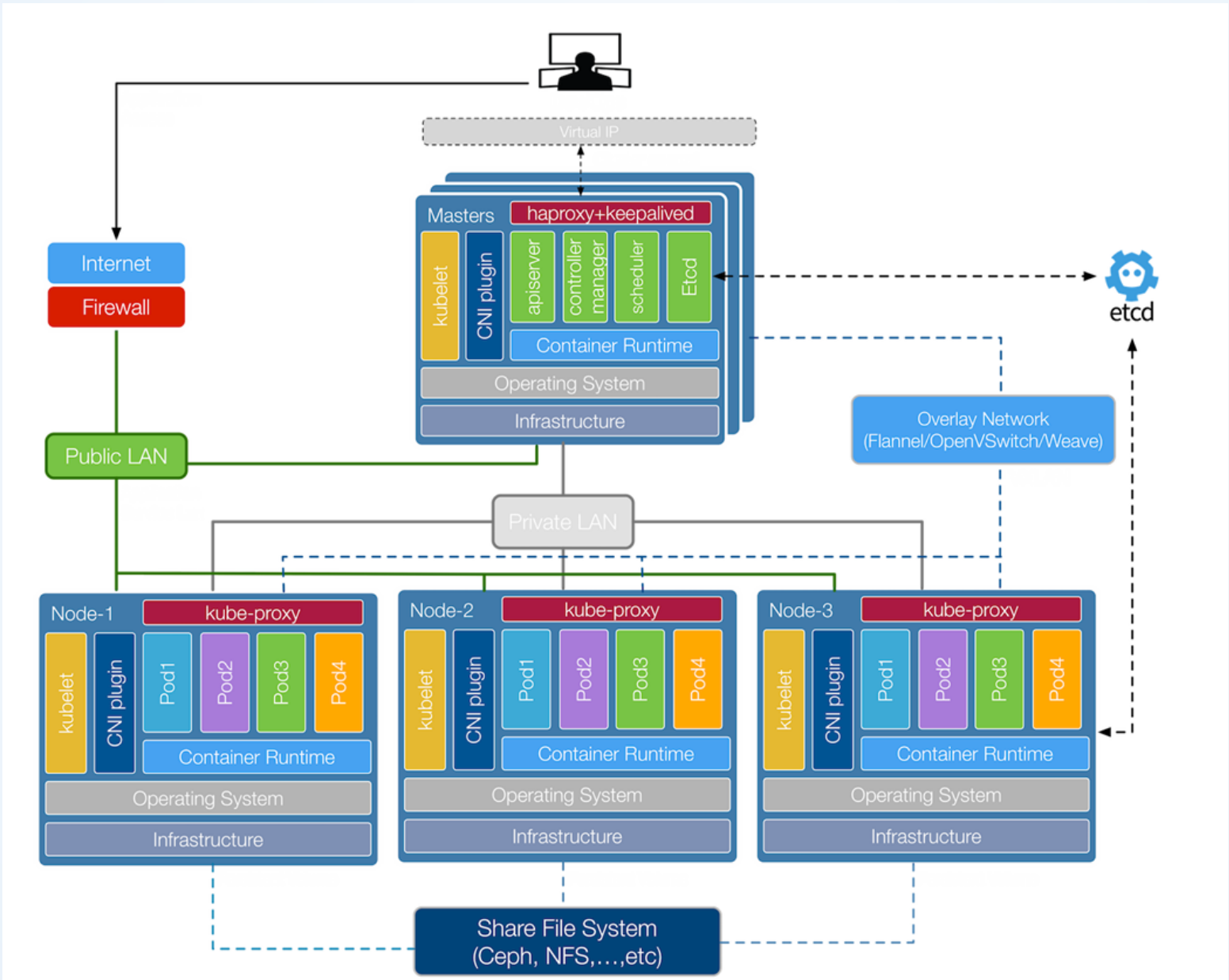

# Kubernetes组成（Addon）

- Dashboard：Kubernetes提供GUI来简化用户与API Server的交互
- CoreDNS：为整个集群提供DNS服务
- Ingress Controller：为服务提供外网暴露入口并保障高可用（traefik/inginx）
- Heapster：资源监控【1.13 later 已废弃】
- Prometheus：资源监控，告警
- Grafana：监控展示面板
- Fluentd-EFK：集群日志采集，存储，查询，分析
- Helm：YAML模板管理

# Kubernetes架构

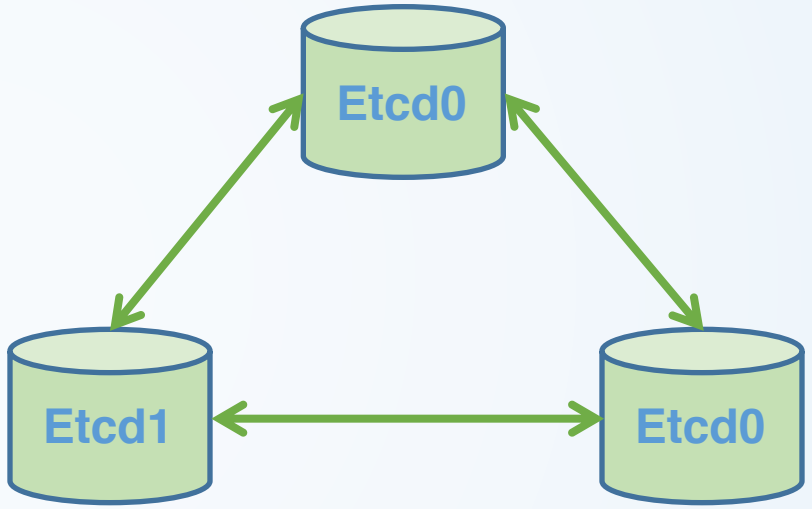


# Kubernetes架构



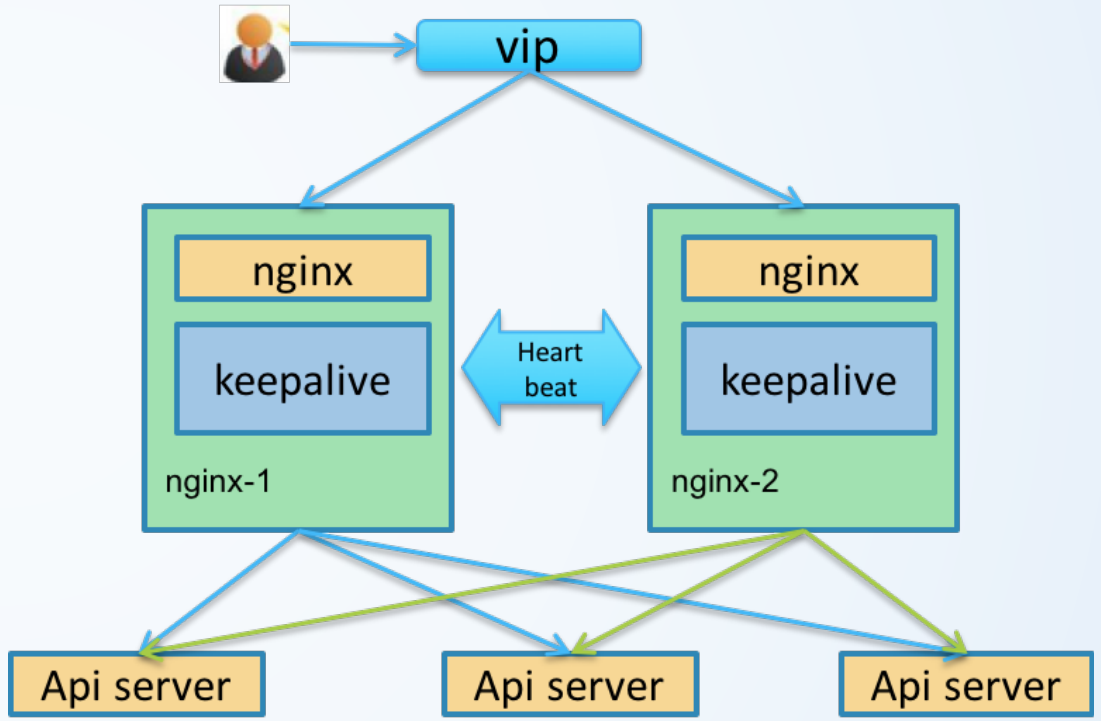
# Kubernetes高可用部署（Etcd）

- 部署3个或5个节点，分别部署在不同的服务器上
- 选举机制，Raft协议，超过一半的节点正常，服务正常
- 可与Kubernetes其它Master组件一起部署
- Etcd的IO能力是瓶颈，可使用SSD
- 使用SSL证书



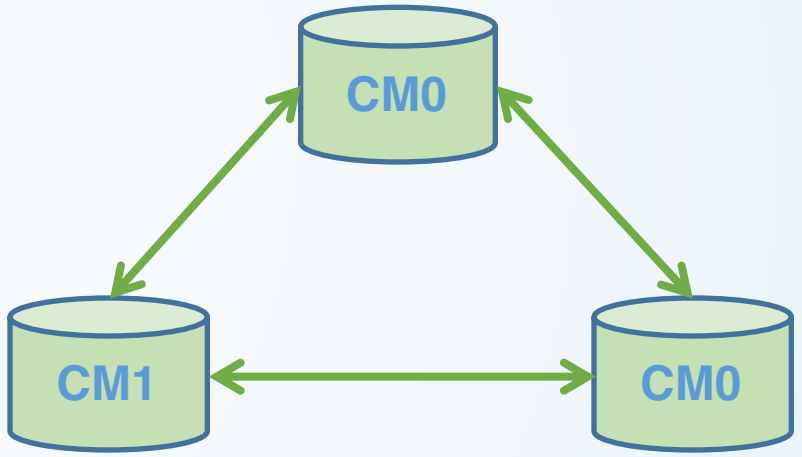
# Kubernetes高可用部署（Api Server）

- 部署3个节点，分别部署在不同的服务器上
- 使用HAProxy/Nginx等，云上部署用ELB，代理，API-Server是无状态服务
- API-Server用VIP提供服务
- 使用SSL证书

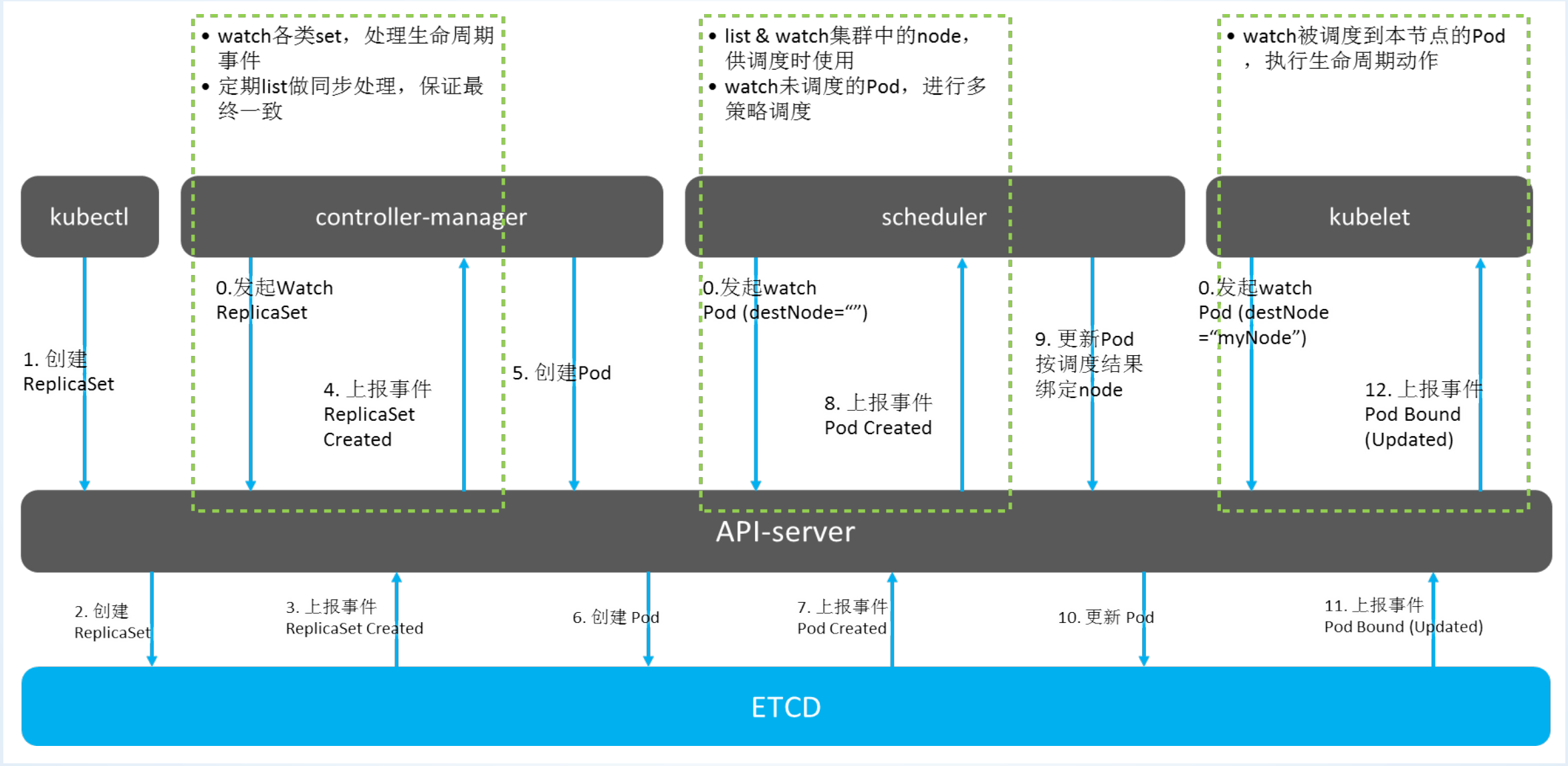


# Kubernetes高可用部署（Scheduler/Controller-Manager）

- 部署3个节点，分别部署在不同的服务器上
- 选举机制，超过一半的节点正常，服务正常，本身通过配置可实现
- 使用SSL证书
- Scheduler与Controller-Manager的高可用方式相同



# Kubernetes工作原理




# Kubernetes基本概念与术语

- Pod
  - 一组容器：一组功能相关的容器的组合
  - 共享存储：同一个Pod内的多个容器可共享存储
  - 最小单位：K8S调度和作业运行的基本单位（Scheduler调度，Kubelet运行）
  - 共享Network Namespace：同一个pod里的容器共享同一个网络命名空间（Other container模式），可通过localhost（或127.0.0.1）互相通信。
- Workloads (ReplicaSet，Deployment, StatefulSet, DaemonSet, Job…)
  - 功能相关的Pod组的封装
- Service
  - 服务访问入口，是pod组的反向代理


# Kubernetes基本概念与术语

Kubernetes 所有资源对象都可以用yaml或JSON格式文件来定义和描述。

下面罗列部分**Kubernetes对象：**

- Label Selector
- StatefulSet
- Job
- DaemonSet
- ConfigMap
- Namespace
- Annotation
- Horizontal Pod Autoscaler

# Kubernetes API 对象的基本构成

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:latest
        ports:
        - containerPort: 80
status: {}
```

typeMeta

objectMeta

spec （期望状态）

status （实际状态）


# 使用kubectl与集群交互

**Basic Commands (Beginner):**

| | |
|---|---|
| **create** | 从文件或stdin创建资源 |
| **expose** | 为deployment，pod创建Service。 |
| **run** | Run a particular image on the cluster |
| **set** | Set specific features on objects |

**Basic Commands (Intermediate):**

| | |
|---|---|
| **get** | 最基本的查询命令。如 kubectl get rs，kubectl get deploy，kubectl get svc，kubectl get rs/foo |
| explain | 查看资源定义。如 kubectl explain replicaset |
| **edit** | 使用系统编辑器编辑资源。如 kubectl edit deploy/foo |
| **delete** | 删除指定资源，支持文件名、资源名、label selector。如 kubectl delete po -l foo=bar |


# 使用kubectl与集群交互

**Deploy Commands:**

| | |
|---|---|
| **rollout** | Deployment，Daemonset的升级过程管理（查看状态、操作历史、暂停升级、恢复升级、回滚等） |
| rolling-update | 客户端滚动升级，仅限ReplicationController |
| **scale** | 修改Deployment，ReplicaSet，ReplicationController，Job的实例数 |
| autoscale | 为Deploy，RS，RC配置自动伸缩规则（依赖heapster和hpa） |

**Cluster Management Commands:**

| | |
|---|---|
| certificate | Modify certificate resources. |
| cluster-info | 查看集群信息 |
| top | 查看资源占用率（依赖heapster） |
| cordon | 标记节点为unschedulable |
| uncordon | 标记节点为schedulable |
| drain | 驱逐节点上的应用，准备下线维护 |
| taint | 修改节点taint标记 |


# 使用kubectl与集群交互

定位问题和调试命名:

| | |
|---|---|
| **describe** | 查看资源详情 |
| **logs** | 查看pod内容器的日志 |
| attach | Attach到pod内的一个容器 |
| **exec** | 在指定容器内执行命令 |
| port-forward | 为pod创建本地端口映射 |
| proxy | 为Kubernetes API server创建代理 |
| **cp** | 容器内外/容器间文件拷贝 |

**Advanced Commands:**

| | |
|---|---|
| **apply** | 从文件或stdin创建/更新资源 |
| patch | 使用strategic merge patch语法更新对象的某些字段 |
| replace | 从文件或stdin更新资源 |
| convert | 在不同API版本之间转换对象定义 |


# 使用kubectl与集群交互

**Settings Commands:**

| | |
|---|---|
| **label** | 给资源设置label |
| **annotate** | 给资源设置annotation |
| completion | 获取shell自动补全脚本(支持bash和zsh) |

**Other Commands:**

| | |
|---|---|
| api-versions | Print the supported API versions on the server, in the form of "group/version" |
| **config** | 修改kubectl配置（kubeconfig文件），如context |
| help | Help about any command |
| version | 查看客户端和Server端K8S版本 |


# kubectl实用技巧

- kubectl命令太多太长记不住？
  - 查看资源缩写

```
kubectl describe ↵
```

  - 配置kubectl自动完成

```
source <(kubectl completion bash)
```

- kubectl写yaml太累，找样例太麻烦?
  - 用run命令生成

```
kubectl run --image=nginx my-deploy -o yaml --dry-run > my-deploy.yaml
```

  - 用get命令导出

```
kubectl get statefulset/foo -o=yaml --export  > new.yaml
```

  - Pod亲和性下面字段的拼写忘记了

```
kubectl explain pod.spec.affinity.podAffinity
```


# Kubernetes集群部署实操演示

参照附件：06　Kubernetes集群部署实操演示.txt


# 课程回顾

**已学知识要点**

- *了解Kubernetes及其架构设计*
- *了解Kubernetes集群的组成*
- *了解Kubernetes的基本概念*