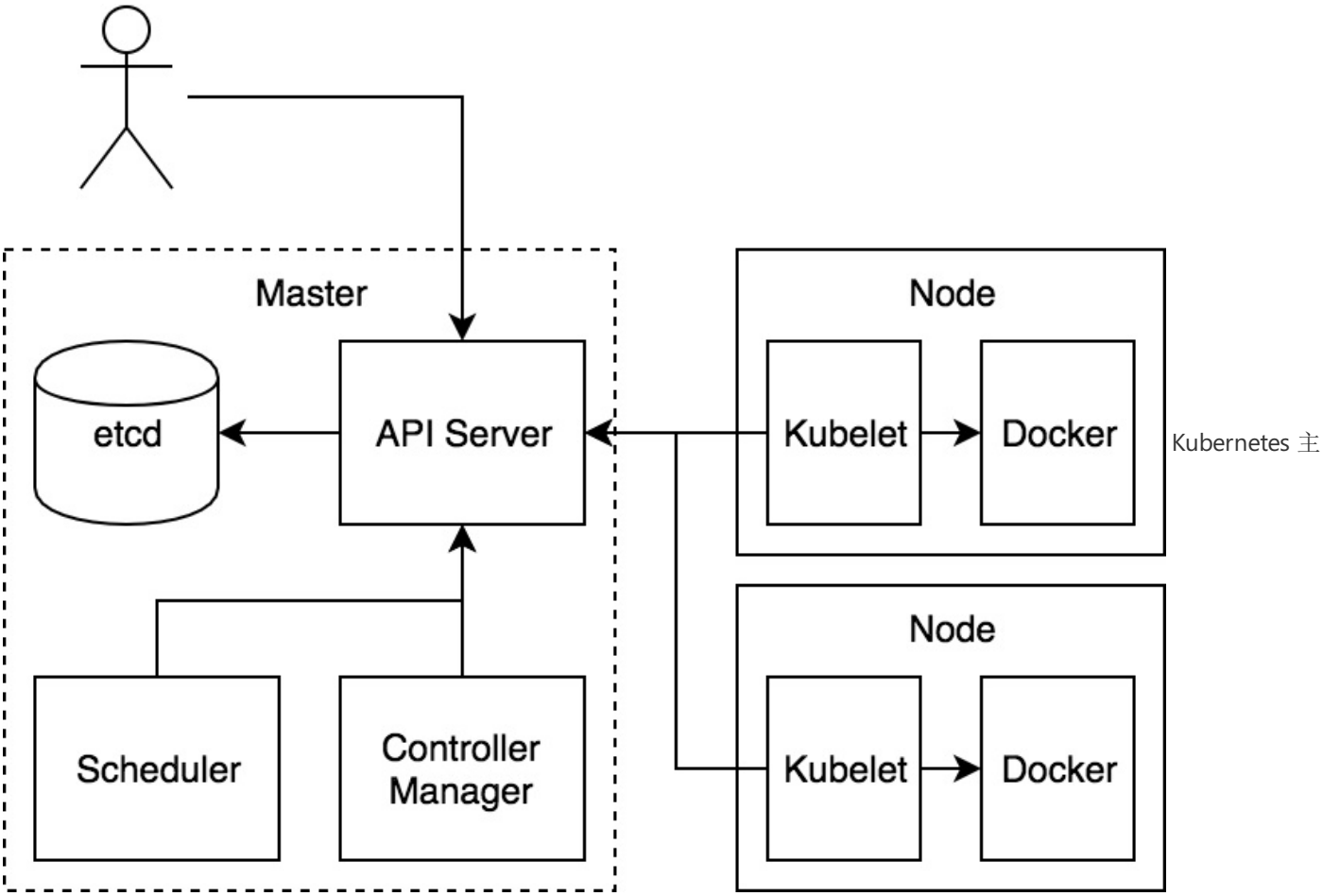
Kubernetes

```
kubeadm init --apiserver-advertise-address 192.168.3.131 --pod-network-cidr 10.244.0.0/16 --kubernetes-version 1.15.0
```

1. 概念

1.0 架构/组件



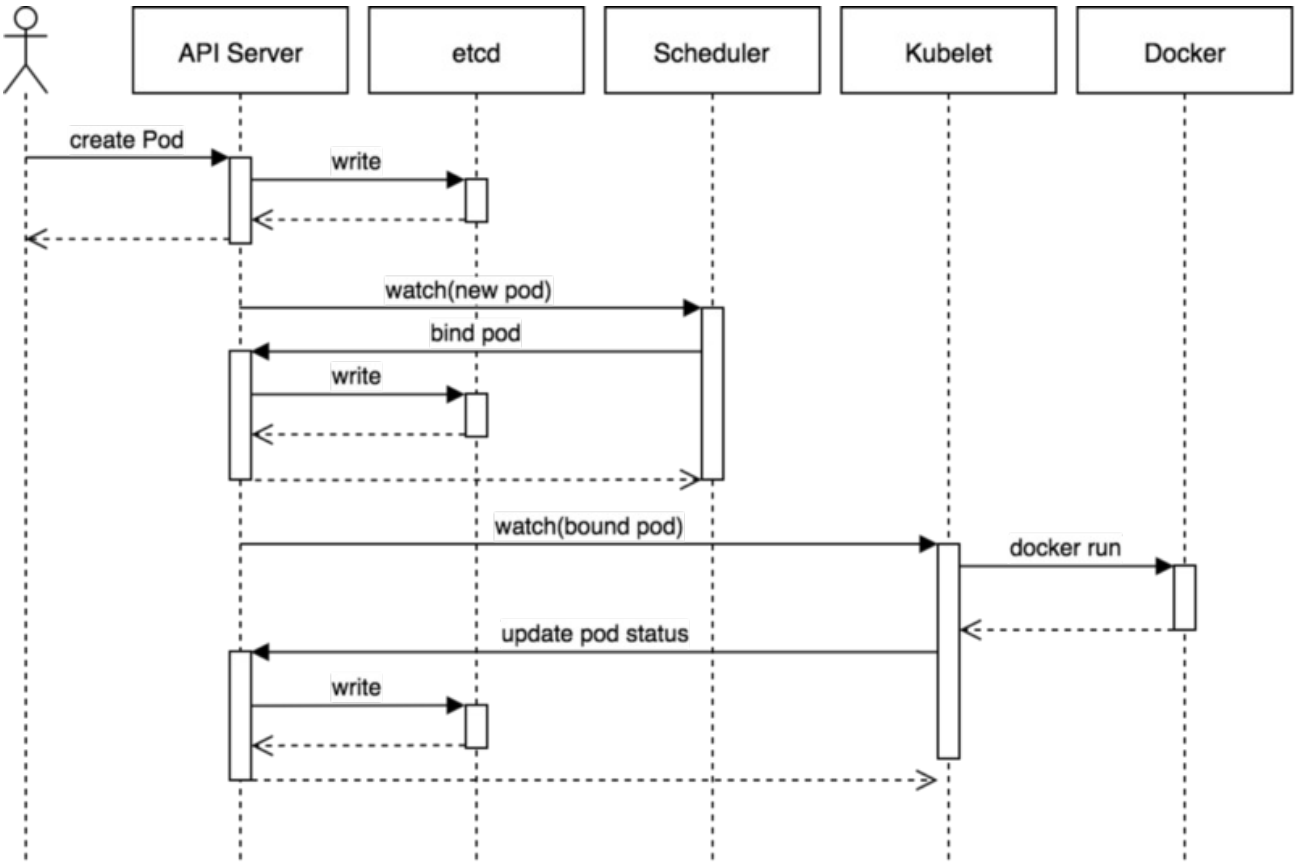
Kubernetes 主要由以下几个核心组件组成:

> `apiserver` 提供了资源操作的唯一入口，并提供认证、授权、访问控制、API 注册和发现等机制；
> `controller manager` 负责维护集群的状态，比如故障检测、自动扩展、滚动更新等； `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； `kubelet` 负责维护容器的生命周期，同时也负责 Volume（CSI）和网络（CNI）的管理； `Container runtime` 负责镜像管理以及 Pod 和容器的真正运行（CRI）；
> `kube-proxy` 负责为 Service 提供 cluster 内部的服务发现和负载均衡；

当然了除了上面的这些核心组件，还有一些推荐的插件：

> `kube-dns` 负责为整个集群提供 DNS 服务 `Ingress Controller` 为服务提供外网入口 `Heapster` 提供资源监控
> `Dashboard` 提供 GUI

pod的创建流程：



1.1 Labels and Selector

参考 k8s-标签和选择器

> 注意：集合选择器 `notin`，如：`tier notin (frontend, backend)` 表示 `有tier且值不等于frontend/backend` 和 `没有tier标签` 的对象；

1.2 namespace

> Node, PersistentVolumes 等则不属于任何 namespace

1.3 RC And RS

1.3.1 区别

RS和RC的功能基本一致，目前唯一的一个区别就是RC只支持基于等式的selector（env=dev或environment!=qa），但RS还支持基于集合的selector（version in (v1.0, v2.0)）；

1.4 Deployment

`Deployment` 和 `RC/RS` 相比：

- RC的全部功能：Deployment具备上面描述的RC的全部功能
- 事件和状态查看：可以查看Deployment的升级详细进度和状态
- 回滚：当升级Pod的时候如果出现问题，可以使用回滚操作回滚到之前的任一版本
- 版本记录：每一次对Deployment的操作，都能够保存下来，这也是保证可以回滚到任一版本的基础
- 暂停和启动：对于每一次升级都能够随时暂停和启动

一个Deployment拥有多个Replica Set，而一个Replica Set拥有一个或多个Pod。一个Deployment控制多个rs主要是为了支持回滚机制，每当Deployment操作时，Kubernetes会重新生成一个Replica Set并保留，以后有需要的话就可以回滚至之前的状态。

1.5 Job And CronJob

> `CronJob`：运行中的 Job 将不会被终止，不会删除 Job 或 它们的 Pod。为了清理那些 Job 和 Pod，需要列出该 Cron Job 创建的全部 Job，然后删除它们;

1.6 flannel

(`Pod IP`是每个Pod的IP地址，它是Docker Engine根据`docker0`网桥的IP地址段进行分配的)，`flannel`网络插件保证所有节点的Pod IP不会冲突

1.7 Cluster IP

`Cluster IP`是一个虚拟的IP，仅仅作用于Kubernetes `Service`这个对象，由Kubernetes自己来进行管理和分配地址，当然我们也无法ping这个地址，他没有一个真正的实体对象来响应，他只能结合Service Port来组成一个可以通信的服务。

1.8 ConfigMap

参考：28.ConfigMap

> 当ConfigMap以数据卷的形式挂载进Pod的时，这时更新ConfigMap（或删掉重建ConfigMap），Pod内挂载的配置信息会热更新。这时可以增加一些监测配置文件变更的脚本，然后reload对应服务。

1.9 RBAC

相关： 30.RBAC、API-GROUPS

1.10 数据卷

数据卷可以使用hostPath自定义数据（在host主机）保存位置

```
volumes:
- name: db
  hostPath:
    path: /var/lib/mysql
```

1.11 HPA

1.11.1 yaml

```
resources:
  limits:
    cpu: 200m
    memory: 200Mi
  requests:
    cpu: 100m
    memory: 100Mi
```

N 其他

1. 命令自动补全

- kubectl:

```
$ source <(kubectl completion bash)  # 临时修改
$ echo "source <(kubectl completion bash)" >> ~/.bashrc  # 永久
```

- 优雅地终止容器 默认所有的优雅退出时间都在30秒内。kubectl delete 命令支持 `--grace-period=<seconds>` 选项，参考Pod Hook

> 值'0'代表 强制删除 pod. 在 kubectl 1.5 及以上的版本里，执行强制删除时必须同时指定 --force --grace-period=0

强制删除并不是立即终止容器：

> 强制删除一个 pod 是从集群状态还有 etcd 里立刻删除这个 pod。 当 Pod 被强制删除时， api 服务器不会等待来自 Pod 所在节点上的 kubelet 的确认信息：pod 已经被终止。在 API 里 pod 会被立刻删除，在节点上， pods 被设置成立刻终止后，在强行杀掉前还会有一个很小的宽限期

2. 常用命令

```
$ kubectl get pods --all-namespaces

# 创建命名空间
$ kubectl create namespace blog

# HPA
$ kubectl autoscale deployment your-deployment --cpu-percent=10 --min=1 --max=10 -n namespace
```

3. 扩展

3.1 命令嵌套

理解下面例子中的 `'pwd'`

```
$ docer run -d -p 5000:5000 \
--restart=always --name registry \
-v \`pwd\`/auth:/auth \
-e "REGISTRY_AUTH=htpasswd" \
-e "REGISTRY_AUTH_HTPASSWD_REALM=Registry Realm" \
-e REGISTRY_AUTH_HTPASSWD_PATH=/auth/htpasswd \
-v \`pwd\`/registry:/var/lib/registry \
registry:2
```

3.2 滚动升级

参考23.使用RC管理Pod、24.Deployment使用

4. 对象文件头（yaml格式）

pod

```
apiVersion: v1
kind: Pod
```

Deployment

```
apiVersion: extensions/v1beta1
kind: Deployment
```

ReplicationController

```
apiVersion: v1
kind: ReplicationController
```

Deployment

```
apiVersion: apps/v1beta1
kind: Deployment
```

Job

```
apiVersion: batch/v1
kind: Job
```

Cron-Job

```
apiVersion: batch/v2alpha1
kind: CronJob
```

Service

```
apiVersion: v1
kind: Service
metadata:
  name: myservice
spec:
  selector:
    app: myapp
  ports:
  - protocol: TCP
    port: 80
    # targetPort 可以是一个字符串，引用了 backend Pod 的一个端口的名称。 因实际指派给该端口名称的端口号，在每个
backend Pod 中可能并不相同，所以对于部署和设计 Service ，这种方式会提供更大的灵活性。
    targetPort: 8080
    name: myapp-http
```

ConfigMap

```
apiVersion: v1
kind: ConfigMap
```

Secret

```
apiVersion: v1
kind: Secret
```

Role

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
```

RoleBinding

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
```

DaemonSet

```
apiVersion: extensions/v1beta1
kind: DaemonSet
```

PV

```
apiVersion: v1
kind: PersistentVolume
```

PVC

```
apiVersion: v1
kind: PersistentVolumeClaim
```