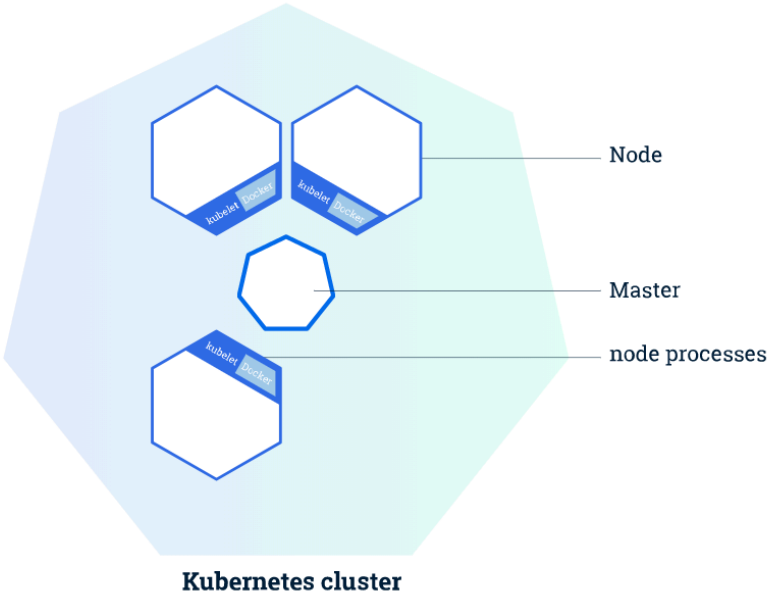
Kubernetes 笔记

----------建立Kubernetes集群---------

Kubernetes包含的节点：

1. Master
2. Nodes



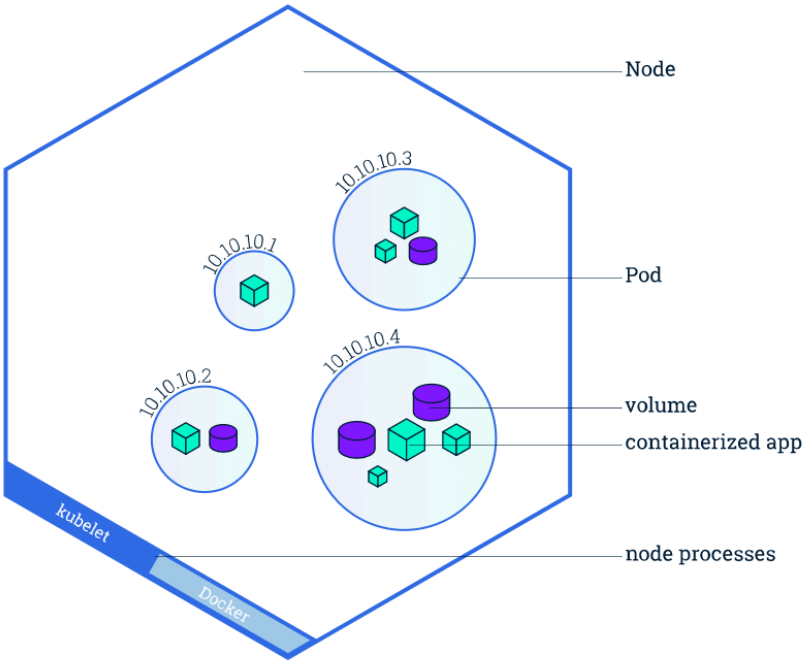
Master作用：

1. 发布应用程序
2. 维护应用程序状态
3. 更改应用程序的规模
4. 更新应用程序

Node：（虚拟机或者物理机，在Kubernetes集群中充当worker的作用）

1. 每一个Node都包含一个Kubelet，Kubelet是一个用来管理node并且和master通信的代理

2. 每一个Node都包含一个Docker工具用来执行容器相关的操作
3. 每一个Kubernetes集群至少要包含三个Node（容错和replica保证）



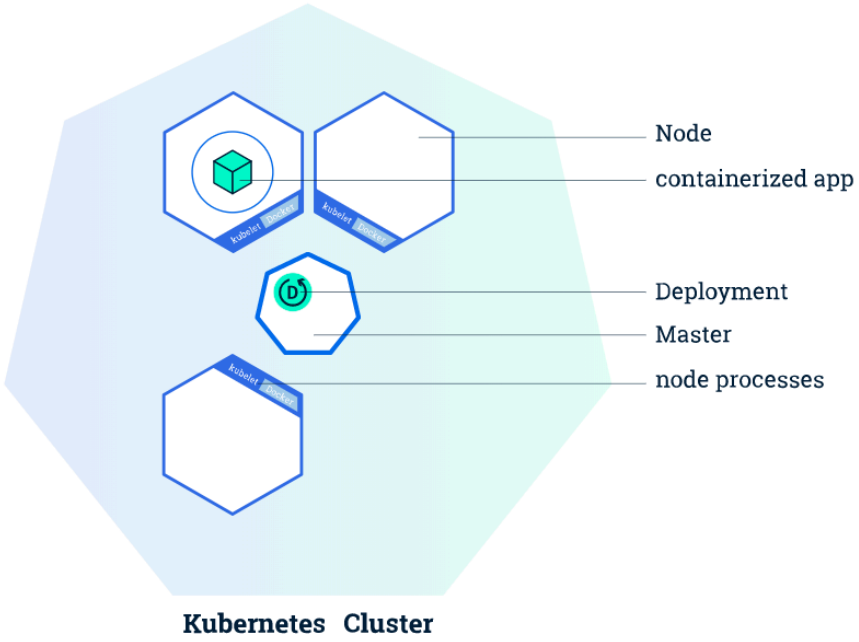
Minikube（Kubernetes在本地的轻量级实现）

Minikube会自动在本地建立一个虚拟机，并且发布一个只包含一个Node的Kubernetes集群

——————在Kubernetes集群上发布应用程序——————

Deployment：

1. Deployment configuration告诉Kubernetes（Master）怎样建立和更新一个容器化的应用程序实例
2. 实例创建之后，使用Deployment Controller监控创建好的实例，一旦有实例挂掉或被删除，Deployment Controller就会创建一个新的实例来替换掉失败的这个实例
3. 在创建Deployment的时候，Kubernetes会创建Pod来装载应用程序
4. 在创建Deployment的时候，需要为应用程序指定所在容器要使用的镜像，和replica的数目（即对Pod的复制）



Kubectl是一个用来管理一个Deployment的CLI，它使用Kubernetes的API来与集群进行交互

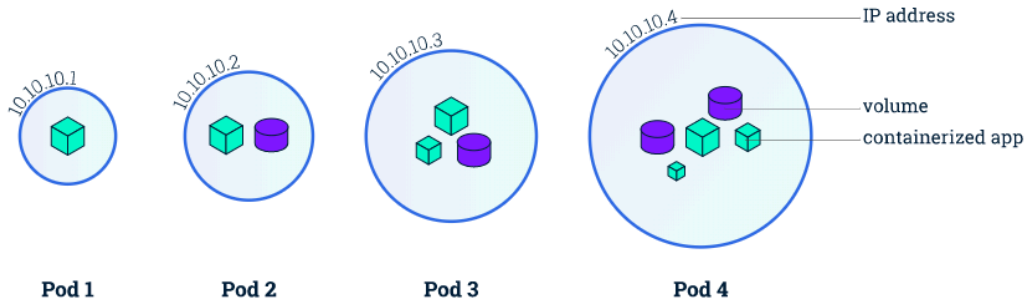
——————————查看应用程序——————————

Kubernetes Pods:

1. **Pod位于Node之中**
2. **一个Node中可以有多个Pod**
3. **Pod在创建Deployment的时候被自动创建**
4. **Pod是一个包含了一个或者多个容器以及被这些容器所共享的资源的小组**

- **被共享的资源包括：**
- **共享存储（卷）**
- **网络，表现为一个共用的IP地址**
- **运行其所包含容器所需的一些信息，例如镜像版本，端口号等**

5. **Pod中可包含运行着相同应用程序的容器或者一些运行着不同应用程序但具有强耦合关系的容器**
6. **为防止Node挂掉，相同的Pod会被安排在集群中的多个其他Node上**
7. **Pods运行在一个被隔离的，私有的网络环境中，所以如果想在本地访问集群中的应用程序，我们可以建立一个proxy，通过这个代理来访问**

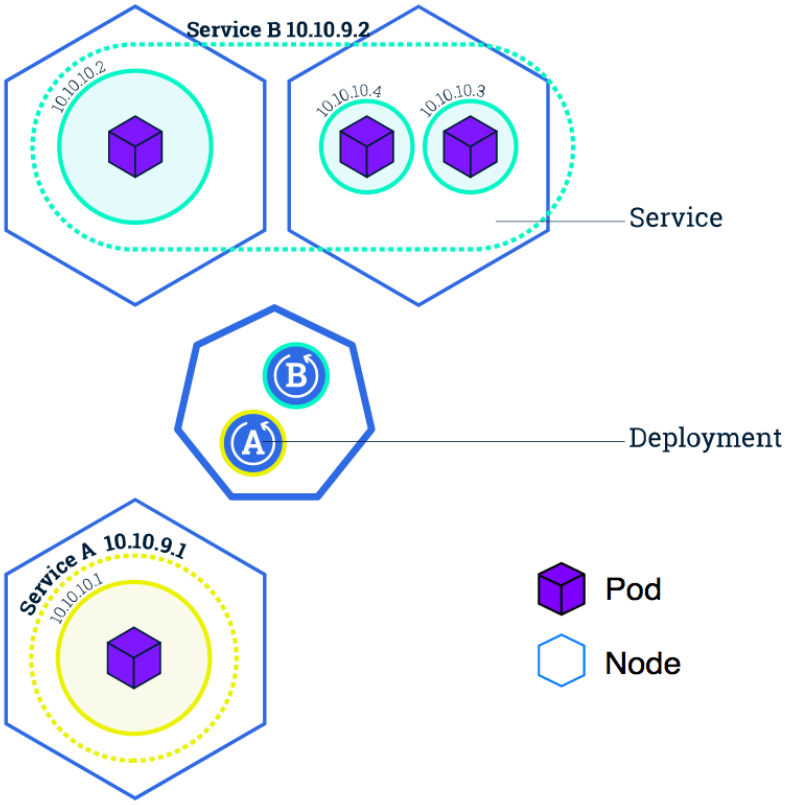


——————使用Service让应用程序对集群外部可见——————

Service:

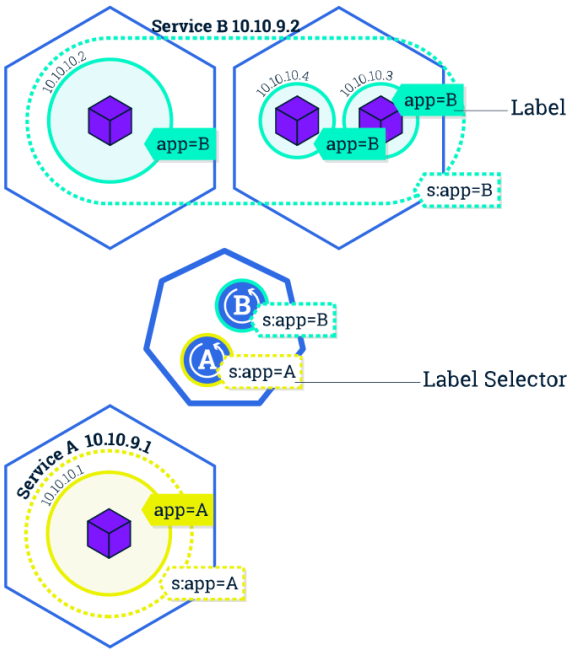
1. **Motivation:** 每一个**Pod**都有一个唯一的**IP**，所以一旦正在使用的**Pod**挂掉，我们需要使用它的备份来继续工作，但此时**Pod**的**IP**已经发生改变，但集群外部并不能及时获取这种变化。
2. **Service**是一种**Kubternetes**的抽象，它定义了一个逻辑上的**Pod**集合以及访问这些**Pod**的原则
3. 使用**YAML**文件来定义**Service**
4. **Service**支持对具有依赖关系的**Pod**进行解除耦合的处理
5. 由于**Pod**的**IP**对集群外不可见，所以在使用**kubectl CLI** 来**expose**时，我们通过设置**type**参数来指定**expose**的方式
6. **expose**的种类：

- **ClusterIP(默认)**，仅对集群内部可见
- **NodePort**，在不同**Node**的同一端口号下暴露**Service**，外部可通过**<NodeIP>:<NodePort>**来进行访问**Service**
- **LoadBalancer**，在当前云平台上为**Service**分布一个外部**IP**，集群外部可通过这个外部**IP**来访问**Service**
- **ExternalName**，为**Service**在**Spec**中设置**externalName**来设置一个外部名



Services and Labels:

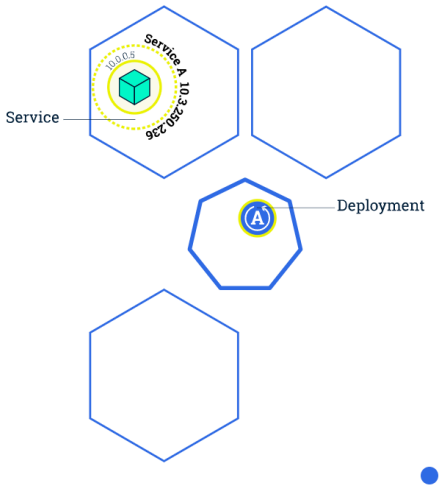
1. 具有依赖关系的**Pods**之间的交互和路由由**Service**负责
2. **Service**所指向的**Pod**集合由**LabelSelector**来确定
3. Label是key/value类型的键值对
4. Label可以在生成对象之时或之后进行创建

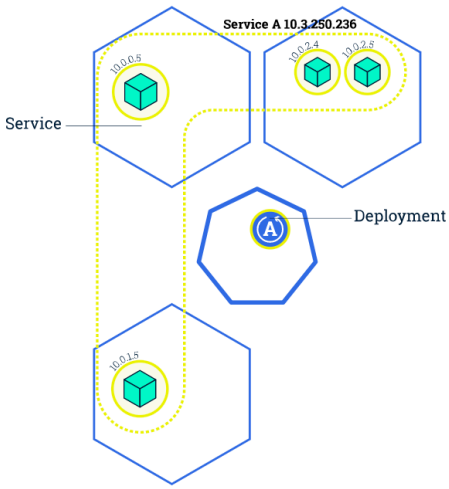


设置Pods规模

Scaling:

1. 通过设置**Deployment**的**replica**参数来对**Pods**规模进行更改
2. 增大**pods**规模时，要保证新的**Pod**所在的**Node**中有足够的资源供使用
3. 在同时运行多个应用程序实例时，集成在Service中的load-balancer会使用endpoints来监控每个运行中的应用程序实例

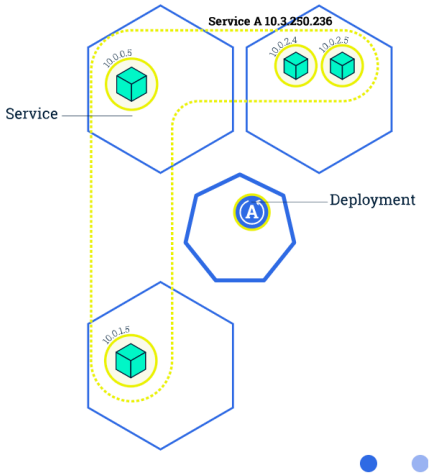


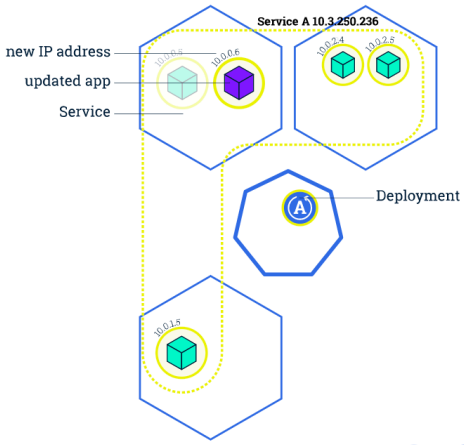


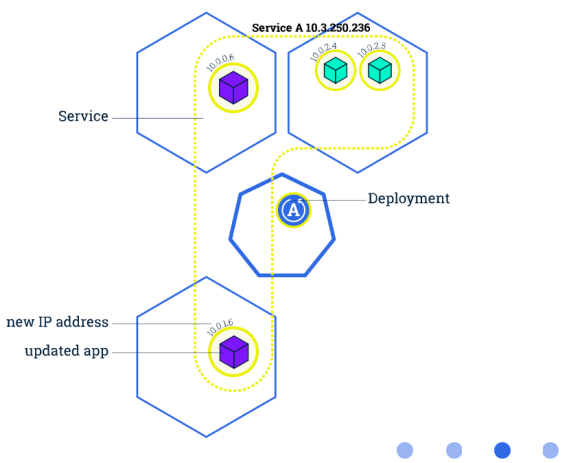
更新应用程序

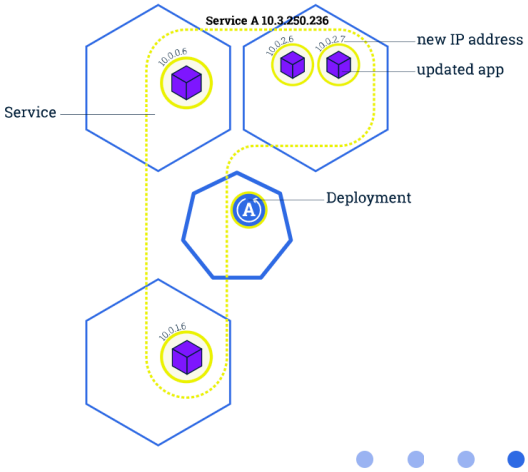
Rolling Update:

1. **Kubernetes**通过版本控制的方法来更新应用程序，更新后的应用程序可以被还原成旧的版本
2. **Rolling Update**可以更新容器镜像






Service A 10.3.250.236
10.0.0.6
10.0.2.4
10.0.2.5
Service
Deployment
new IP address
updated app
10.0.1.6


Service A 10.3.250.236
10.0.0.6
10.0.2.6
10.0.2.7
new IP address
updated app
Service
Deployment
10.0.1.6