# docker和k8s

## 简介

官方定义1：Docker是一个开源的应用容器引擎，开发者可以打包他们的应用及依赖到一个可移植的容器中，发布到流行的Linux机器上，也可实现虚拟化。

官方定义2：k8s是一个开源的容器集群管理系统，可以实现容器集群的自动化部署、自动扩缩容、维护等功能。

Docker守护进程(Docker Daemon)。它是运行在操作系统之上的后台进程，负责管理Docker容器

虚拟机如果你要装软件依赖的话是装在虚拟机里的,docker的话依赖直接给你封镜像里方便,但是虚拟机也是有用的,阿里云这些提供给你的就是虚拟机.

## docker命令

Docker search nginx

Docker pull reg.jianzh5.com/nginx:1.7.9

Docker images 列出所有镜像

Docker rmi 删除镜像

docker run nginx -p 8080:80，新建启动镜像创建容器 将容器的80端口映射到宿主机的8080端口，然后使用localhost:8080就可以查看容器中nginx的欢迎页

Docker ps 列出所有运行的容器

docker stopdocker stop 5d034c6ea010 停止容器

Docker start 5d034c6ea010 启动以前停止的容器

Docker restart 5d034c6ea010 重启容器

Docker exec -it 5d034c6ea010 /bin/bash 进入容器

Docker rm 删除容器

docker build -t 镜像名字 . 用当前目录的dockerfile文件的内容构建出一个镜像.

FROM nginx

RUN echo '这是一个本地构建的nginx镜像' > /usr/share/nginx/html/index.html

上面这就是一个dockerfile文件,首先from nginx就会拉一个nginx的镜像,执行下面的run的内容,就会给这个nginx镜像里面加一个文件类似于这样,首先你都得去pull一个镜像,然后后面的run命令是你定制化的镜像的操作.

dockerfile除了这个简单的from run 还有很多别的命令,比如给镜像加环境变量的env命令.

EXPOSE加端口的 user增加镜像用户的 copy 复制文件的,你可以在build镜像的时候给镜像里做很多的操作.

docker run -d -m 1G --memory-swap 3G -p 9999:80 run的时候可以指定内存

docker run -it --rm --cpus=2 u-stress:latest /bin/bash

限制cpu

docker容器默认的空间是10G,如果想指定默认容器的大小（在启动容器的时候指定），可以在docker配置文件里通过dm.basesize参数指定，比如  
docker -d --storage-opt dm.basesize=20G

## ****Kubernetes****

### ****tips****

K8S，就是基于容器的集群管理平台，它的全称，是kubernetes。

Kubernetes 是一个[自动化部署、伸缩和操作应用程序容器的开源平台](https://link.zhihu.com/?target=http://www.slideshare.net/BrianGrant11/wso2con-us-2015-kubernetes-a-platform-for-automating-deployment-scaling-and-operations" \t "/Users/yangwenshuo/Documents\\x/_blank)

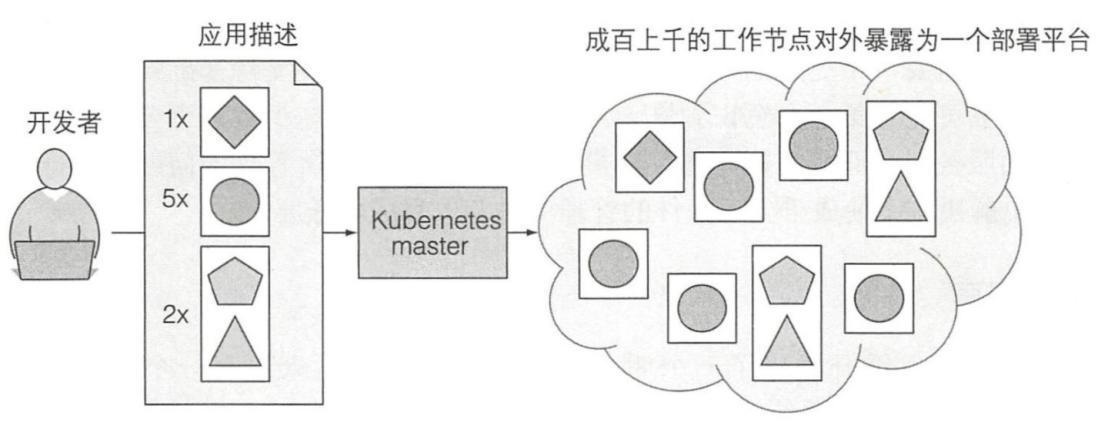
至少，Kubernetes 能在实体机或虚拟机集群上调度和运行程序容器。而且，Kubernetes 也能让开发者斩断联系着实体机或虚拟机的“锁链”，从以主机为中心的架构跃至以容器为中心的架构。该架构最终提供给开发者诸多内在的优势和便利。Kubernetes 提供给基础架构以真正的以容器为中心的开发环境。

自动动发布,调度,配置,监管和故障处理

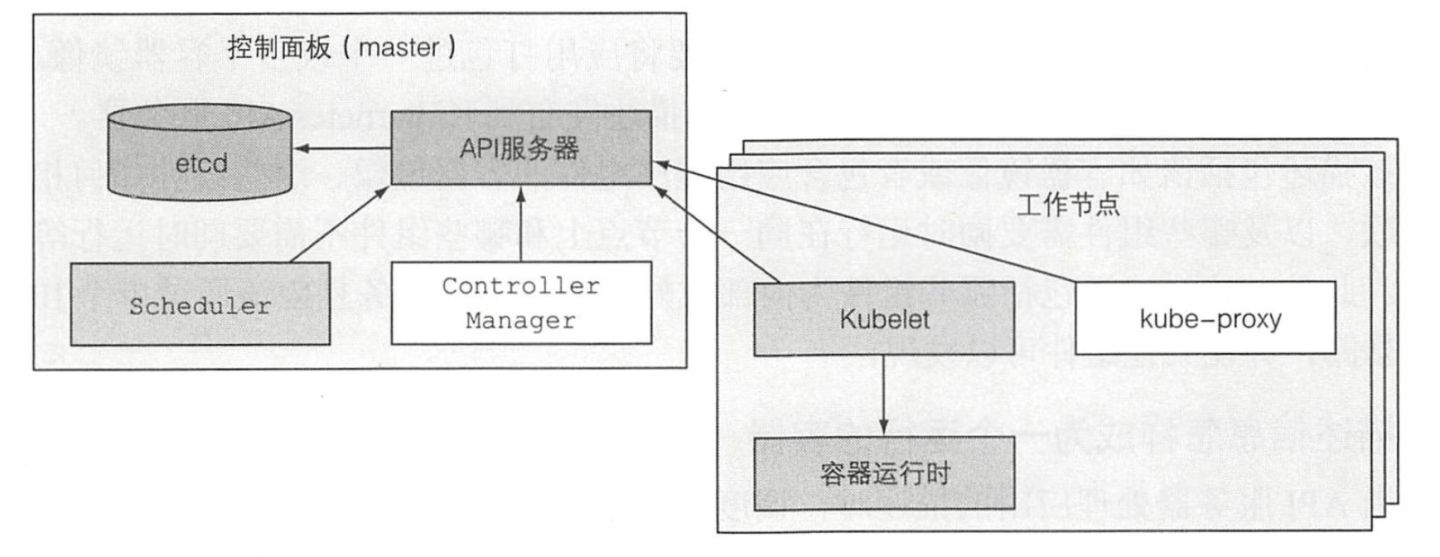
Kubernetes 使开发者可以自主部署应用，并且控制部署的频率，完全脱离运维团队的帮助。

Kubernetes 同时能让运维团队监控整个系统，并且在硬件故障时重新调度应用。系统管理员的工作重心，从监管应用转移到了监管 Kubernetes，以及剩余的系统资源，因为 Kubernetes 会帮助监管所有的应用。

通过对实际硬件做抽象，然后将自身暴露成一个平台，用于部署和运行应用程序。它允许开发者自己配置和部署应用程序，而不需要系统管理员的任何帮助，让系统管理员聚焦于保持底层基础设施运转正常的同时，不需要关注实际运行在平台上的应用程序。



我们已经知道了 Kubernetes 的工作方式，现在让我们近距离看一下 Kubernetes 集群的组成。在硬件层面，Kubernetes 集群由两种节点组成： - 主节点：它承载着 Kubernetes 控制和管理整个集群系统的控制面板 - 工作节点：它们运行用户实际部署的应用



控制面板用于控制集群并使它工作。它包含多个组件，组件可以运行在单个主节点上或者通过副本分别部署在多个主节点以确保高可用性。这些组件是: - Kubernetes API 服务器：你的其他控制面板组件都要和它通信 - Scheduler：它调度你的应用(为应用的每个可部署组件分配一个工作节点〕 - Controller Manager：它执行集群级别的功能，如复制组件、持续跟踪工作节点、处理节点失败等 - etcd：一个可靠的分布式数据存储，它能持久化存储集群配置.

控制面板的组件持有并控制集群状态，但是它们不运行你的应用程序。这是由工作节点完成的。工作节点是运行容器化应用的机器。运行、监控和管理应用服务的任务是由以下组件完成的: - 容器：Docker、rtk 或其他的容器 - Kubelet：它与 API 服务器通信，并管理它所在节点的容器 - Kubernetes Service Proxy (kube-proxy)，它负责组件之间的负载均衡网络流量

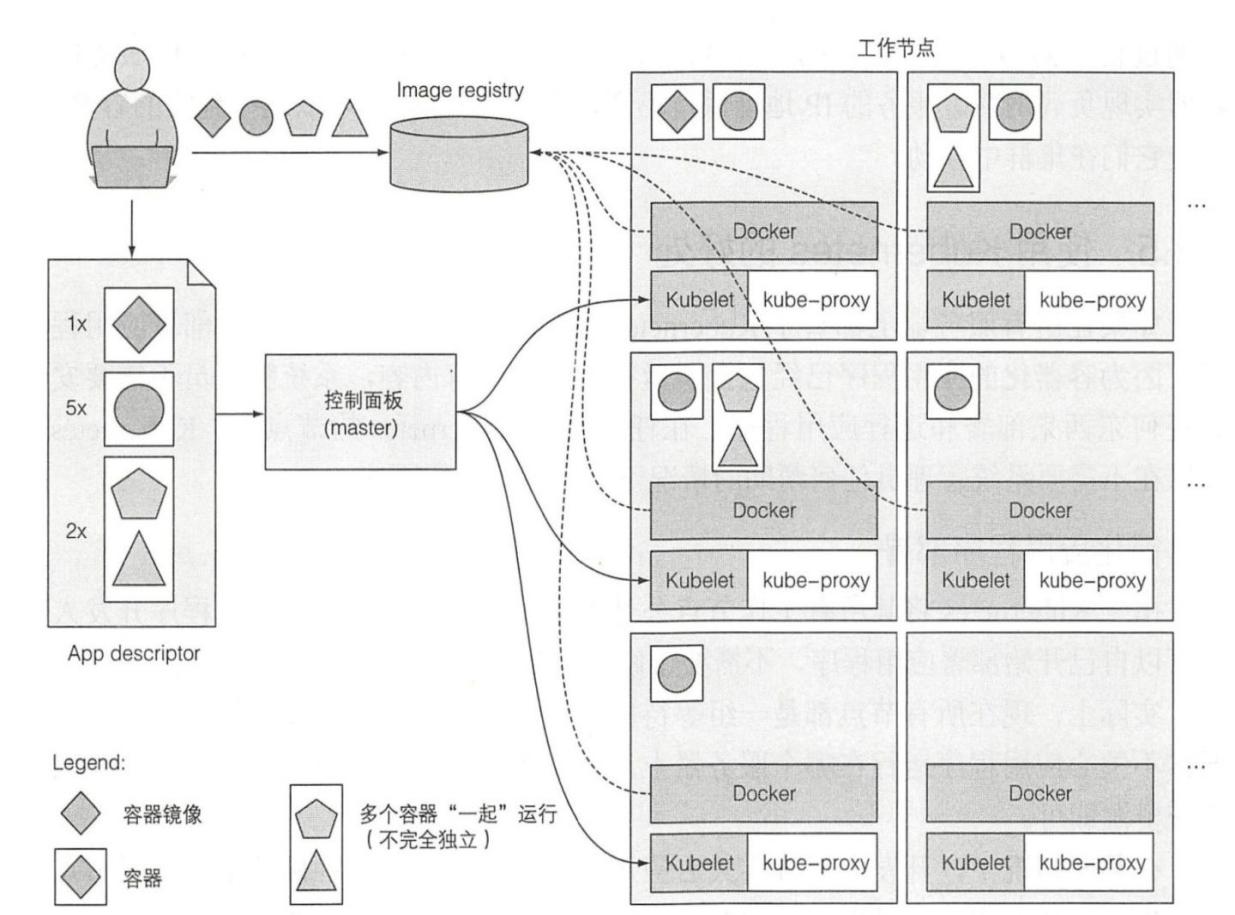
为了在 Kubernetes 中运行应用，首先需要将应用打包进一个或多个容器镜像，再将那些镜像推送到镜像仓库，然后将应用的描述发布到 Kubernetes API 服务器。

应用的描述,看到了么,先在scm上打包镜像,然后这个镜像就是你应用的描述,然后推到master的api服务器,这里就收集到了镜像的信息

该描述包括诸如容器镜像或者包含应用程序组件的容器镜像、这些组件如何相互关联，以及哪些组件需要同时运行在同一个节点上和哪些组件不需要同时运行等信息。此外，该描述还包括哪些组件为内部或外部客户提供服务且应该通过单个 IP 地址暴露，并使其他组件可以发现。 **应用的描述内容有很多**

当 API 服务器处理应用的描述时，调度器调度指定组（Pod）的容器到可用的工作节点上，调度是基于每组（Pod）所需的计算资源，以及调度时每个节点未分配的资源。然后，那些节点上的 Kubelet 指示容器运行时(例如 Docker)拉取所需的镜像并运行容器。这里提到的"组"概念，在 Kubernetes 中被称为 Pod，同一个 Pod 可以包含多个镜像，但是同一个 Pod 中的所有进程就如同运行在同一个物理机一样相互可以看见，它们之间未做隔离。

一旦应用程序运行起来，Kubernetes 就会不断地确认应用程序的部署状态始终与你提供的描述相匹配。例如，如果你指出你需要运行五个 web 服务器实例，那么 Kubernetes 总是保持正好运行五个实例。如果实例之一停止了正常工作，比如当进程崩溃或停止响应时，Kubernetes 将自动重启它。



当应用程序运行时，可以决定要增加或减少副本量，而 Kubernetes 将分别增加附加的或停止多余的副本。甚至可以把决定最佳副本数目的工作交给 Kubernetes。它可以根据实时指标(如 CPU 负载、内存消耗、每秒查询或应用程序公开的任何其他指标)自动调整副本数。

我们已经说过，Kubernetes 中的容器可能是动态的，它随时可能从一台机器上被销毁，然后在另一台机器上重新启动。那么当容器在集群内频繁调度时，它们该如何正确使用这个容器?当这些容器被复制并分布在整个集群中时，客户端如何连接到提供服务的容器呢?

为了让客户能够轻松地找到提供特定服务的容器，可以告诉 Kubernetes 哪些容 器提供相同的服务，而 Kubernetes 将通过一个静态 IP 地址暴露所有容器，并将该地址暴露给集群中运行的所有应用程序。您可以通过环境变量或者 DNS 服务来共享服务 IP。kube-proxy 将确保到服务能连接到对应的容器上。

这一部分存疑,ip可以是很多个,端口可以不同,这些用注册中心维护下就好了,k8s肯定得和consul关联的.

FROM node:7

ADD app.js /app.js

ENTRYPOINT ["node", "app.js"]

测试的dockerfile

*# 定义镜像的新 tag，<docker hub id>/kubia*

docker tag kubia beikejiedeliulangmao/kubia*# 登录 docker hub io，并输入账号密码*

docker login*# 推送本地镜像到 docker 镜像仓库中*

docker push beikejiedeliulangmao/kubia

Push 测试镜像到docker hub

### pod

在Kubernetes集群中，Pod是所有业务类型的基础，也是K8S管理的最小单位级，它是一个或多个容器的组合。这些容器共享存储、网络和命名空间，以及如何运行的规范。在Pod中，所有容器都被同一安排和调度，并运行在共享的上下文中。对于具体应用而言，Pod是它们的逻辑主机，Pod包含业务相关的多个应用容器。

**网络:**每一个Pod都会被指派一个唯一的Ip地址，在Pod中的每一个容器共享网络命名空间，包括Ip地址和网络端口。在同一个Pod中的容器可以同locahost进行互相通信。当Pod中的容器需要与Pod外的实体进行通信时，则需要通过端口等共享的网络资源。

我们看到的那些多个实例同一个ip的其实就是因为在一个pod中.

**存储:**Pod能够被指定共享存储卷的集合，在Pod中所有的容器能够访问共享存储卷，允许这些容器共享数据。存储卷也允许在一个Pod持久化数据，以防止其中的容器需要被重启。

pod的创建是通过模版,调度是通过调度器.

由于不推荐将无关的进程运行在同一个容器中，但是特殊情况下还存在要求多个相关进程运行于同一个容器的需求，kubernetes提供了一种更高级的结构来把容器捆绑在一起，并将这种结构作为调度部署的基本单元，这个结构就是Pod。

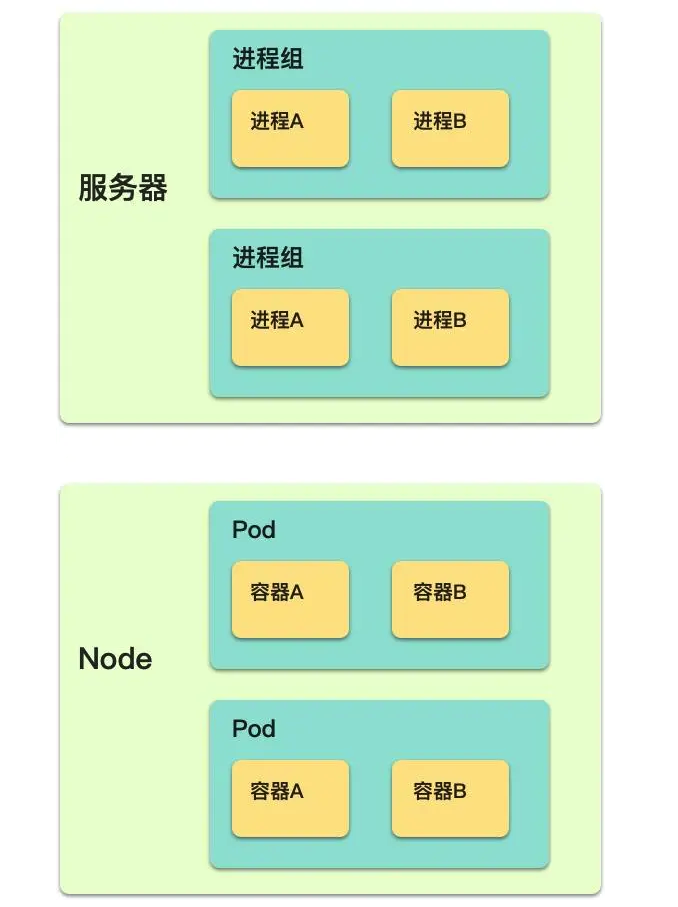
Pod是一组并置的容器，是kubernetes中基本的构建模块。但是这并不意味着一个Pod总是包含多个容器，在实际应用中每个Pod只有一个容器是最常见的部署方式。

相同Pod下运行的容器之间可以共享一些资源，但是并非全部资源（话句话说，这些容器并非完全隔离的），kubernetes通过配置可以让同一个Pod内的容器共享相同的linux命名空间和network等资源，所以这些容器共享相同的主机名和网络接口，话句话说，这些容器在Pod中可以进行IPC通信，就像在局域网中一样。既然共享相同的IP和端口号，那么多个容器就不能绑定到相同的端口，否则会出现端口冲突，所以这也是官方推荐一个Pod只运行一个容器的原因之一。

在多数情况下，我还是建议每个Pod运行一个容器，但是如果你的多个容器有互相依赖关系（比如一个容器的启动依赖于另外一个容器），就需要把多个容器部署到一个Pod。一个Pod中运行一个容器更多的是基于应用分层的考虑，例如：一个应用的容器需要调用一个数据库的容器，这两个容器应该分配到不同的Pod中，不仅仅是为了提高集群机器的利用率，更是为了之后不同层次的扩容。对于kubernetes来说，操作和部署的基本单位是Pod，所以kubernetes扩容的单位是Pod并非容器，如果我们的应用层有性能瓶颈，我们就可以单独的对应用层的Pod进行单独扩容，其他层的Pod保持不变，这不仅仅是节省资金成本的问题了，而是横向扩展的灵活性问题。

### Service

一个Service可以看作一组提供相同服务的Pod的对外访问接口。



### K8s集群搭建

kubeadm是官方社区推出的一个用于快速部署kubernetes集群的工具。

这个工具能通过两条指令完成一个kubernetes集群的部署：

# 创建一个 Master 节点

$ kubeadm init

# 将一个 Node 节点加入到当前集群中

$ kubeadm join <Master节点的IP和端口 >

$ yum install -y kubelet-1.14.0 kubeadm-1.14.0 kubectl-1.14.0

$ systemctl enable kubelet

安装docker kubeadm kubelet kubectl

部署master

kubeadm init

添加新的node,工作节点 join到master里面

kubeadm join 172.16.3.40:6443 --token l79g5t.6ov4jkddwqki1dxe --discovery-token-ca-cert-hash sha256:4f07f9068c543130461c9db368d62b4aabc22105451057f887defa35f47fa076

在Kubernetes集群中创建一个pod，验证是否正常运行：

$ kubectl create deployment nginx --image=nginx

$ kubectl expose deployment nginx --port=80 --type=NodePort

$ kubectl get pod,svc

部署Dashboard

一个管理页面

还可以搭建etcd集群.

### 部署应用

配置好 Kubernetes 集群后，我们就通过之前 Docker 试玩环节使用的 Node Demo kubia 来体验一下如何在 Kubernetes 中部署应用。这里我们先通过 ReplicationController 接口来启动该应用，首先我们要创建一个配置文件 rc.yaml:

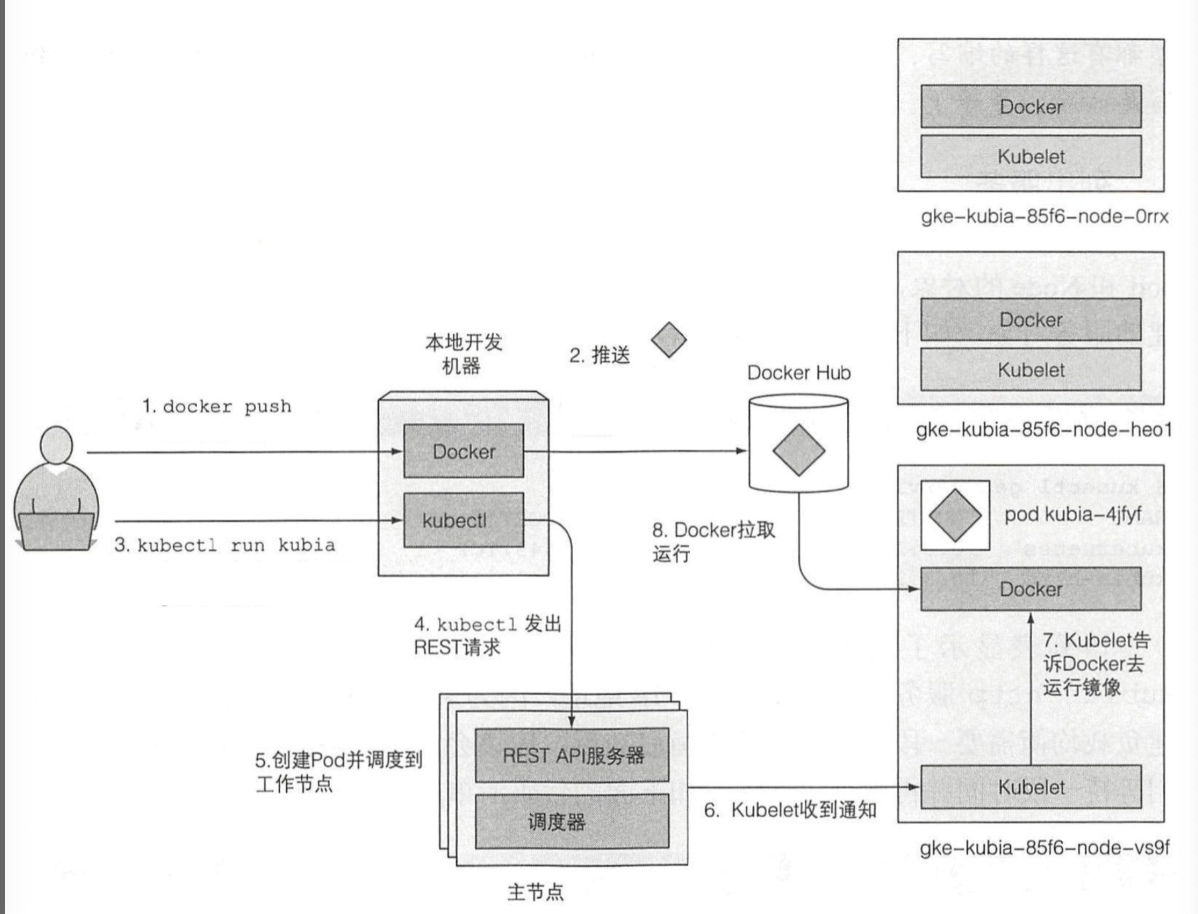


准备好配置文件之后，我们通过命令行工具来注册该 ReplicationController：

kubectl apply -f rc.yaml

如果上述命令正常执行，您将像我一样创建了一个名为 kubia 的 ReplicationController，它会帮助我们维持整个集群恰好有 3 个 kubia 服务（之前的 Node 服务），您可以通过 kubectl get pod -o wide 查看 ReplicationController 为我们创建的 Pod，您可以删除任意一个 Pod 来模拟服务中崩溃，当你再次查看运行中的 pod 时，你会发现 ReplicationController 帮我们又重建了一个新的 Pod。

当运行 kubectl 命令时，它通过向 Kubernetes API 服务器发送一个 REST HTTP 请求来创建一个 ReplicationController，ReplicationController 创建成功后，会根据配置文件中指定的 Pod 个数，在集群中创建一个相应数量 Pod 对象。然后，调度器将其调度到一个工作节点上(node)。随后，该工作节点上的 Kubelet 看到 pod 被调度到自己身上，就告知 Docker 从镜像中心中拉取指定的镜像，因为本地没有该镜像。下载镜像后，Docker 创建并运行容器。



从上图可以看出这个命令发到master的api暴露接口,然后调度器调度到kubelet,然后kubelet发消息到docker,然后docker去docker pull docker run等.

扩容

*# 将 kubia 服务扩容为 6 个 pod*

kubectl scale rc kubia --replicas**=**6

这个命令就可以