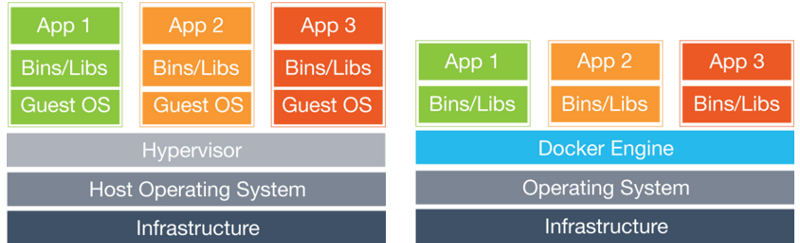
Docker容器编排

# 1.Docker容器编排概念

Docker是一个主流容器管理工具，它是第一个基于Linux容器（LXC），但是现在被runC所取代了（runC是是一个由Open Containers Initiative开发的CLI工具，它能够创建和运行容器）。Docker容器支持分层的文件系统，因此它能够和宿主机共享系统内核。这个特性意味着即便一个Docker镜像基于一个1GB的操作系统，在同一个主机上运行10个容器实例并不需要消耗10GB的空间，相比之下，每一台虚拟机都需要一个完整的1GB操作系统。

[](http://dockone.io/uploads/article/20160320/e91fe0f42ffee5a1b2d1b7c2fa6e4e39.png)

Docker的镜像可以理解为一个操作系统的快照。如果你想要创建一个新的镜像，你需要启动一个基础镜像，然后做一些修改，最后提交修改，形成新的镜像。这些镜像能够发布在私有或者公有的库上[10]供其他开发者使用，开发者只需要将镜像pull下来即可。

使用镜像可以非常方便的创建操作系统的快照，并且使用它们来创建新的容器，这些功能非常的轻量和易用，这些都是Docer CLI和商业模式的核心。

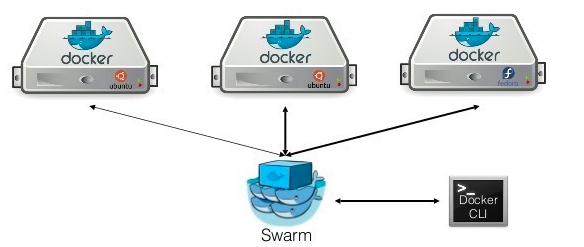
容器编排工具的主要任务就是负责在最合适的主机上启动容器，并且将它们关联起来。它必须能够通过自动的故障转移（fail-overs）来处理错误，并且当一个实例不足以处理/计算数据时，它能够扩展容器来解决问题。其包含了一系列任务：

* 调度（包括部署、复制、扩展、复活、重新调度、升级、降级等）
* 资源管理（内存、CPU、存储空间、端口、IP、镜像等）
* 服务管理（即使用标签、分组、命名空间、负载均衡和准备就绪检查将多个容器编排在一起）

目前常见的容器编排工具有Docker Swarm、Apache Mesos( Running Marathon)、Google kuberntes及Rancher等。

# 2.Docker Swarm

Docker Swarm是一个由Docker开发的调度框架。由Docker自身开发的好处之一就是标准Docker API的使用。Swarm的架构由两部分组成：

[](http://dockone.io/uploads/article/20160320/c2d12aaca08bf2b3270089600f8b1c4d.png)

其中一个机器运行了一个Swarm的镜像（就像运行其他Docker镜像一样），它负责调度容器，在图片上鲸鱼代表这个机器。Swarm使用了和Docker标准API一致的API，这意味着在Swarm上运行一个容器和在单一主机上运行容器使用相同的命令。尽管有新的flags可用，但是开发者在使用Swarm的同时并不需要改变他的工作流程。

Swarm由多个代理（agent）组成，把这些代理称之为节点（node）。这些节点就是主机，这些主机在启动Docker daemon的时候就会打开相应的端口，以此支持Docker远程API[5]。其中三个节点显示在了图上。这些机器会根据Swarm调度器分配给它们的任务，拉取和运行不同的镜像。

当启动Docker daemon时，每一个节点都能够被贴上一些标签（label），这些标签以键值对的形式存在，通过标签就能够给予每个节点对应的细节信息。当运行一个新的容器时，这些标签就能够被用来过滤集群。

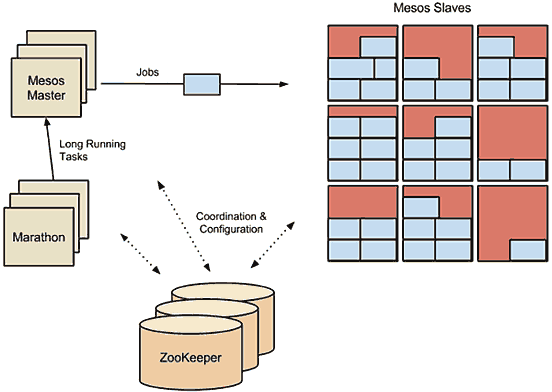
# 3.Apache Mesos & Mesosphere Marathon

Mesos的目的就是建立一个高效可扩展的系统，并且这个系统能够支持很多各种各样的框架，不管是现在的还是未来的框架，它都能支持。这也是现今一个比较大的问题：类似Hadoop和MPI这些框架都是独立开的，这导致想要在框架之间做一些细粒度的分享是不可能的。

因此Mesos的提出就是为了在底部添加一个轻量的资源共享层（resource-sharing layer），这个层使得各个框架能够适用一个统一的接口来访问集群资源。Mesos并不负责调度而是负责委派授权，毕竟很多框架都已经实现了复杂的调度。

取决于用户想要在集群上运行的作业类型，共有四种类型的框架可供使用[52]。其中有一些支持原生的Docker，比如说Marathon[39]。Docker容器的支持自从Mesos 0.20.0就已经被加入到Mesos中了。

我们接下来将会重点关注如何在让Mesos和Marathon一起工作，毕竟Marathon主要是由Mesosphere维护，并且提供了很多关于调度的功能，比如说约束（constraints），健康检查（health checks），服务发现（service discovery）和负载均衡（load balancing）。



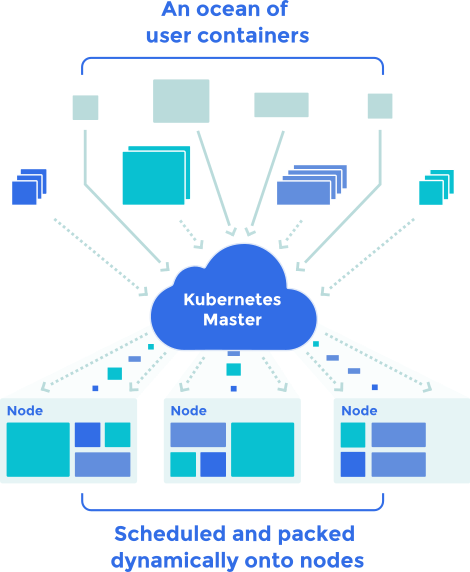
可以从图上看到，集群中一共出现了4个模块。ZooKeeper帮助Marathon查找Mesos master的地址，同时它具有多个实例可用，以此应付故障的发生。Marathon负责启动，监控，扩展容器。Mesos maser则给节点分配任务，同时如果某一个节点有空闲的CPU/RAM，它就会通知Marathon。Mesos slave运行容器，并且报告当前可用的资源。

# 4.Google Kubernetes

Kubernetes是一个Docker容器的编排系统，它使用label和pod的概念来将容器换分为逻辑单元。Pods是同地协作（co-located）容器的集合，这些容器被共同部署和调度，形成了一个服务，这是Kubernetes和其他两个框架的主要区别。相比于基于相似度的容器调度方式（就像Swarm和Mesos），这个方法简化了对集群的管理.

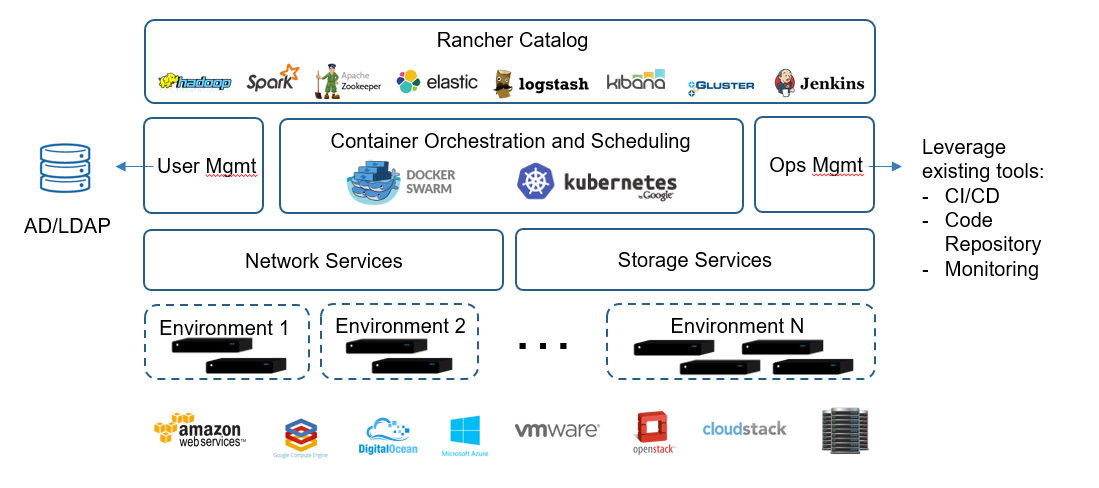
Kubernetes调度器的任务就是寻找那些PodSpec.NodeName为空的pods，然后通过对它们赋值来调度对应集群中的容器。相比于Swarm和Mesos，Kubernetes允许开发者通过定义PodSpec.NodeName来绕过调度器。调度器使用谓词（predicates）和优先级（priorites）[30]来决定一个pod应该运行在哪一个节点上。通过使用一个新的调度策略配置可以覆盖掉这些参数的默认值。

命令行参数plicy-config-file可以指定一个JSON文件（见附录A）来描述哪些predicates和priorities在启动Kubernetes时会被使用，通过这个参数，调度就能够使用管理者定义的策略了。



# 5.Rancher

Rancher 提供了一套完整的 Docker 编排解决方案（重点是[开源](https://github.com/rancher/rancher)的）。功能上包括网络，存储，负载均衡，安全，服务发现和资源管理等。 Rancher 可以管理[DigitalOcean](cloud.digitalocean.com)、[AWS](aws.amazone.com)、[OpenStack](http://openstack.org/)等云主机，自动创建 Docker 运行环境，实现跨云管理。使用上可以通过 Web 界面或者命令行方式进行操作。



# 6.总结

下面的表格是四种编排工具的对比：



Docker Swarm是最简单的调取器，它拥有易于理解的策略和过滤器，但是由于它不能处理节点的失败问题，所以在实际的生产环境中，不推荐使用。Swarm和Docker环境很好的结合在一起，它使用了Docker引擎一样的API，并且能够和Docker Compose很好的一起工作，因此它非常适合那些对其他调度器不太了解的开发者。

Swarm非常轻量，并且提供了多个驱动，使得它它能够和未来所有的集群解决方式一起工。Swarm是一个调度解决方案，非常易于使用，比如它为开发者提供了高纬度的配置方式，让他们能够快速实现具体的工作流。Docker Swarm并没有绑定到某一个具体的云服务提供商，它是完全开源的，并且拥有一个非常强劲的社区。

如果你已经拥有一个Mesos 集群，Mesos & Marathon将会是一个完美的组合方案。它能够像其他Mesos框架一样调度行任务，同时拥有一个类似于Docker Compose的描述文件来制定任务，这些特性使得它成为在集群上运行容器的极佳方案。Mesosphere[45]提供的完整解决方案同样也是一个适合生产环节的、简单而强大的方式。

尽管Kubernetes的逻辑和标准的Docker哲学不同，但是它关于pod和service的概念让开发者在使用容器的同时思考这些容器的组合是什么，真是非常有趣的。Google在它的集群解决方案提供了非常简单的方式来使用Kubernetes，这使得Kubernetes对于那些已经使用了Google生态环境的开发者来说，是一个合理的选择。

在容器编排上，Kubernetes,Mesos和Docker Swarm一直处于竞争状态，但是Docker官方将支持Kubernetes，内置完整的Kubernetes发行版本。 Kubernetes 俨然赢得了编排框架市场的胜利

http://www.infoq.com/cn/news/2017/10/Docker-Kubernetes-Swarm

http://dockone.io/article/1138