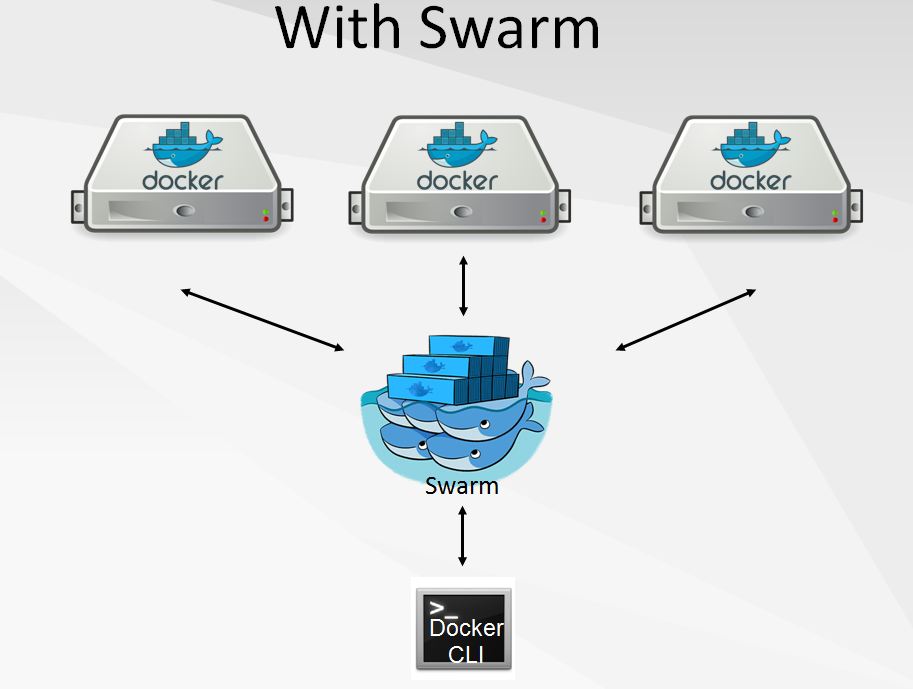
Swarm背景

现实中我们的应用可能会有很多，应用本身也可能很复杂，单个Docker Engine所能提供的资源未必能够满足要求。而且应用本身也会有可靠性的要求，希望避免单点故障，这样的话势必需要分布在多个Docker Engine。在这样一个大背景下，Docker社区就产生了Swarm项目。

Swarm是什么

Swarm这个项目名称特别贴切。在Wiki的解释中，Swarm behavior是指动物的群集行为。比如我们常见的蜂群，鱼群，秋天往南飞的雁群都可以称作Swarm behavior。

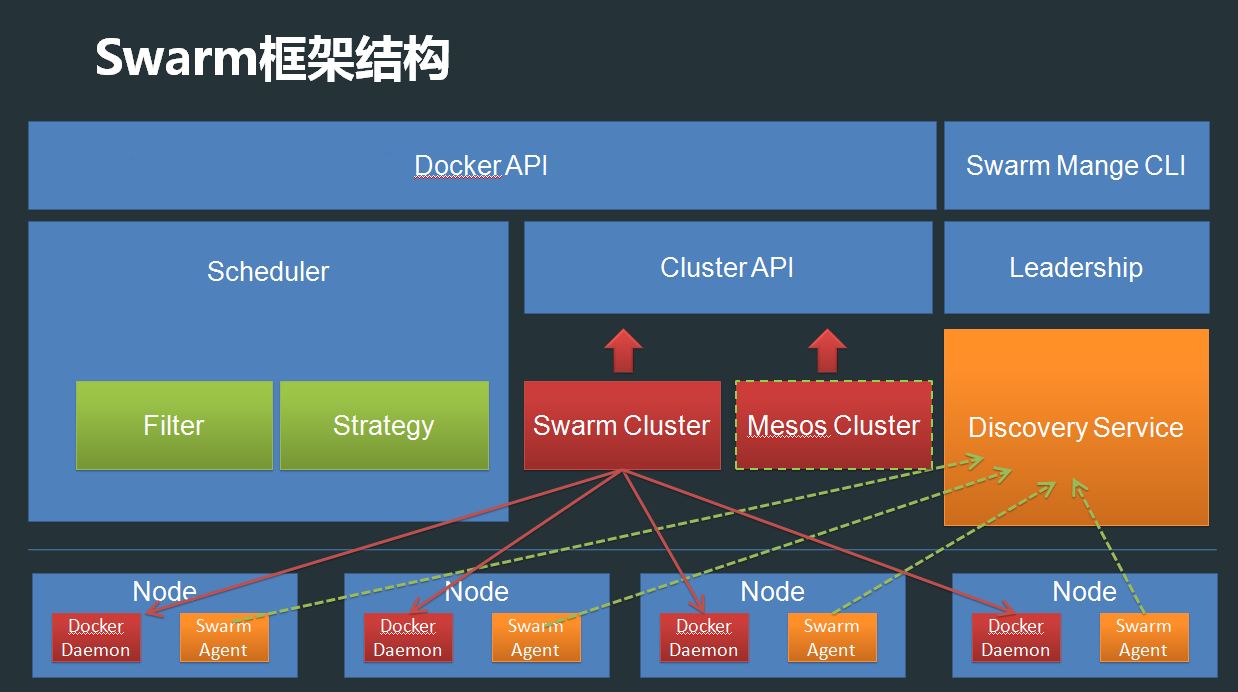
[](http://dockerone.com/uploads/article/20150908/dfc9566b84ce7ca9a2e60c3d6542178e.JPG)

Swarm项目正是这样，通过把多个Docker Engine聚集在一起，形成一个大的docker-engine，对外提供容器的集群服务。同时这个集群对外提供Swarm API，用户可以像使用Docker Engine一样使用Docker集群。

Swarm 特点

* 对外以Docker API接口呈现，这样带来的好处是，如果现有系统使用Docker Engine，则可以平滑将Docker Engine切到Swarm上，无需改动现有系统。
* Swarm对用户来说，之前使用Docker的经验可以继承过来。非常容易上手，学习成本和二次开发成本都比较低。同时Swarm本身专注于Docker集群管理，非常轻量，占用资源也非常少。 \*“Batteries included but swappable”，简单说，就是插件化机制，Swarm中的各个模块都抽象出了API，可以根据自己一些特点进行定制实现。
* Swarm自身对Docker命令参数支持的比较完善，Swarm目前与Docker是同步发布的。Docker的新功能，都会第一时间在Swarm中体现。

Swarm框架结构

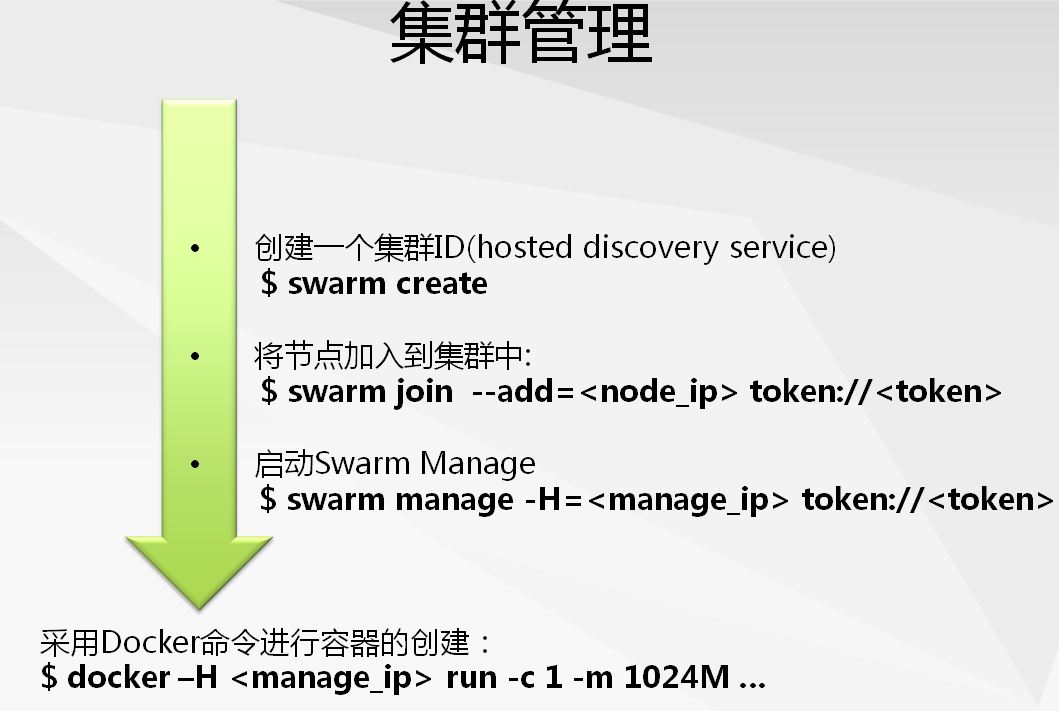
[](http://dockerone.com/uploads/article/20150908/11f6d762768d27db4553924173c83adb.JPG)

* Swarm对外提供两种API， 一种是Docker API，用于负责容器镜像的生命周期管理， 另外一种是Swarm集群管理CLI，用于集群管理。
* Scheduler模块，主要实现调度功能。在通过Swarm创建容器时，会经过Scheduler模块选择出一个最优节点，里面包含了两个子模块，分别是Filter和Strategy， Filter用来过滤节点，找出满足条件的节点（比如资源足够，节点正常等等），Strategy用来在过滤出的节点中根据策略选择一个最优的节点（比如对找出的节点进行对比，找到资源最多的节点等等）, 当然Filter/Strategy用户可以定制。
* Swarm对集群进行了抽象，抽象出了Cluster API，Swarm支持两种集群，一种是Swarm自身的集群，另外一种基于Mesos的集群。
* LeaderShip模块用于Swarm Manager自身的HA，通过主备方式实现。
* Discovery Service 服务发现模块，这个模块主要用来提供节点发现功能。
* 在每一个节点上，都会有一个Agent，用于连接Discovery Service，上报Docker Daemon的IP端口信息，Swarm Manager会直接从服务发现模块中读取节点信息。

Swarm各个模块介绍

集群管理

Swarm Manager CLI用于集群管理。大家可以看这张图，通过三步就可以将集群创建起来。

[](http://dockerone.com/uploads/article/20150908/7b9d0aa76737467b2abfa12341f2221e.JPG)

Swarm容器集群创建完成后，就可以采用Docker命令，像使用Docker Engine一样使用Swarm集群创建容器了。

服务发现

服务发现，在Swarm中主要用于节点发现，每一个节点上的Agent会将docker-egine的IP端口注册到服务发现系统中。Manager会从服务发现模块中读取节点信息。Swarm中服务发现支持已下3种类型的后端：

第一种，是hosted discovery service，是Docker Hub提供的服务发现服务，需要连接外网访问。

第二种，是KV分布式存储系统，现在已支持etcd、ZooKeeper、Consul三种。

第三种，是静态IP。可以使用本地文件或者直接指定节点IP，这种方式不需要启动额外使用其他组件,一般在调试中会使用到。

Scheduler

调度模块主要用户容器创建时，选择一个最优节点。在选择最优节点过程中，分为了两个阶段：

第一个阶段，是过滤。根据条件过滤出符合要求的节点，过滤器有以下5种：

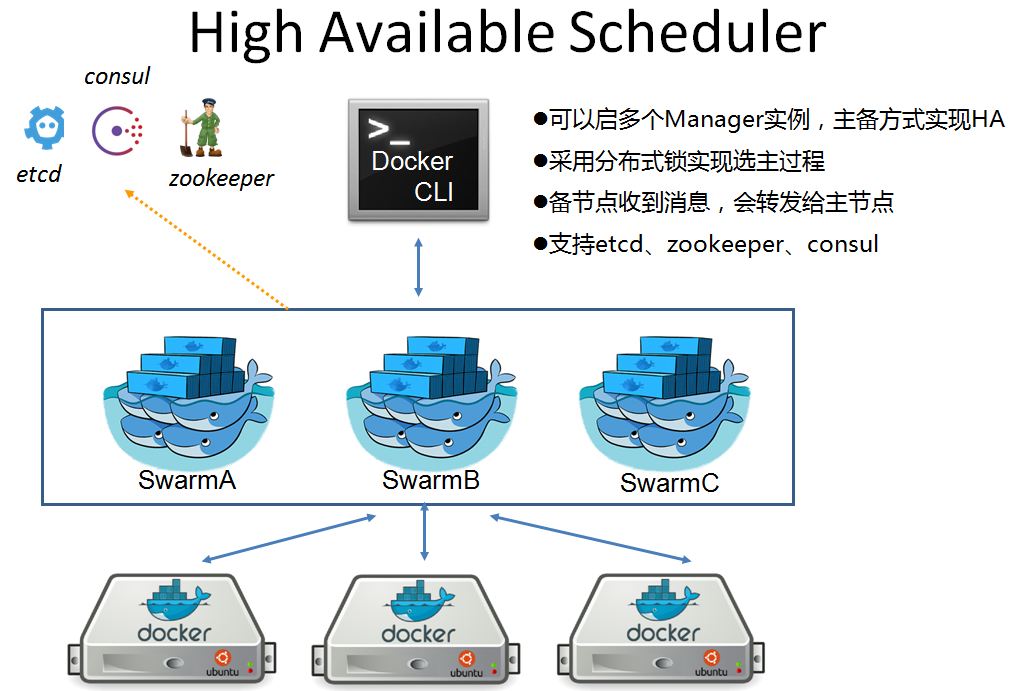
1. Constraints，约束过滤器，可以根据当前操作系统类型、内核版本、存储类型等条件进行过滤，当然也可以自定义约束，在启动Daemon的时候，通过Label来指定当前主机所具有的特点。
2. Affnity，亲和性过滤器，支持容器亲和性和镜像亲和性，比如一个web应用，我想将DB容器和Web容器放在一起，就可以通过这个过滤器来实现。
3. Dependency，依赖过滤器。如果在创建容器的时候使用了**--volume-from/--link/--net**某个容器，则创建的容器会和依赖的容器在同一个节点上。
4. Health filter，会根据节点状态进行过滤，会去除故障节点。
5. Ports filter，会根据端口的使用情况过滤。

调度的第二个阶段是根据策略选择一个最优节点。有以下三种策略：

1. Binpack，在同等条件下，选择资源使用最多的节点，通过这一个策略，可以将容器聚集起来。
2. Spread，在同等条件下，选择资源使用最少的节点，通过这一个策略，可以将容器均匀分布在每一个节点上。
3. Random，随机选择一个节点。

Leadership

Leadership模块，这个模块主要用来提供Swarm Manager自身的HA。

[](http://dockerone.com/uploads/article/20150908/6290a7079c6505b6cb4978802846ab5a.JPG)

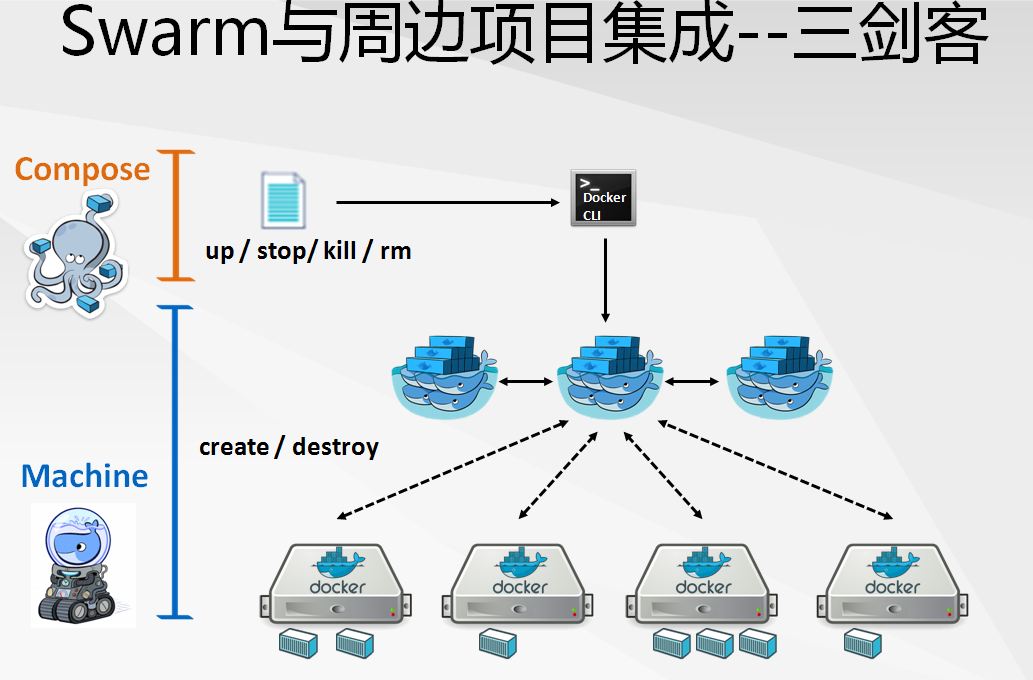
为了防止Swarm Manager单点故障，引入了HA机制，Swarm Manager自身是无状态的，所以还是很容易实现HA的。 实现过程中采用主备方式，当主节点故障以后，会从新选主提供服务，选主过程中采用分布式锁实现，现在支持etcd、ZooKeeper、Consul三种类型的分布式存储，用来提供分布式锁。 当备节点收到消息后，会将消息转发给主节点。

以上就是框架中各个模块的相关介绍，下来和大家一起再看一下，Swarm与周边项目的集成。

首先看一下，与三剑客之间的集成。

Swarm与周边项目集成

三剑客是Docker公司去年底发布的三个项目，这三者是可以紧密协作的。可以看一下这张图：

[](http://dockerone.com/uploads/article/20150908/58074f17fc5aeaf39cc985d38b807c11.JPG)

最下面是Machine，通过Machine可以在不同云平台上创建出包含docker-engine的主机。Machine通过driver机制，目前支持多个平台的docker-egine环境的部署，比如亚马逊、OpenStack等。 Docker Engine创建完以后，就该Swarm上场了，Swarm将每一个主机上的docker-egnine管理起来，对外提供容器集群服务。最上面是Compose项目，Compose项目主要用来提供基于容器的应用的编排。用户通过yml文件描述由多个容器组成的应用，然后由Compose解析yml，调用Docker API，在Swarm集群上创建出对应的容器。

我们知道现在围绕Docker已经产生了很大的一个生态圈。 因此Swarm不仅在和自家兄弟集成，也能积极和周边的一些项目集成。比如，Swarm现在已经可以和Mesos进行集成。Swarm与Mesos集成时，也是以Framework方式集成，实现了Framework所需的接口。这个大特性处在experiment阶段。