Docker(六)：Docker 三剑客之 Docker Swarm

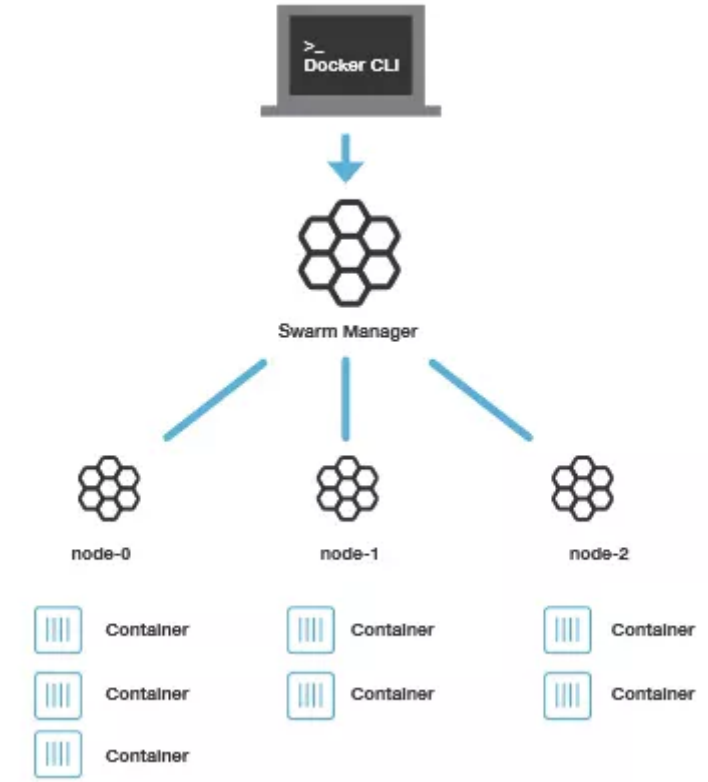
原创： 纯洁的微笑 纯洁的微笑 2018-04-18

实践中会发现，生产环境中使用单个 Docker 节点是远远不够的，搭建 Docker 集群势在必行。然而，面对 Kubernetes, Mesos 以及 Swarm 等众多容器集群系统，我们该如何选择呢？它们之中，Swarm 是 Docker 原生的，同时也是最简单，最易学，最节省资源的，比较适合中小型公司使用。

Docker Swarm 介绍

Swarm 在 Docker 1.12 版本之前属于一个独立的项目，在 Docker 1.12 版本发布之后，该项目合并到了 Docker 中，成为 Docker 的一个子命令。目前，Swarm 是 Docker 社区提供的唯一一个原生支持 Docker 集群管理的工具。它可以把多个 Docker 主机组成的系统转换为单一的虚拟 Docker 主机，使得容器可以组成跨主机的子网网络。

Docker Swarm 是一个为 IT 运维团队提供集群和调度能力的编排工具。用户可以把集群中所有 Docker Engine 整合进一个「虚拟 Engine」的资源池，通过执行命令与单一的主 Swarm 进行沟通，而不必分别和每个 Docker Engine 沟通。在灵活的调度策略下，IT 团队可以更好地管理可用的主机资源，保证应用容器的高效运行。



Docker Swarm 优点

任何规模都有高性能表现

对于企业级的 Docker Engine 集群和容器调度而言，可拓展性是关键。任何规模的公司——不论是拥有五个还是上千个服务器——都能在其环境下有效使用 Swarm。 经过测试，Swarm 可拓展性的极限是在 1000 个节点上运行 50000 个部署容器，每个容器的启动时间为亚秒级，同时性能无减损。

灵活的容器调度

Swarm 帮助 IT 运维团队在有限条件下将性能表现和资源利用最优化。Swarm 的内置调度器（scheduler）支持多种过滤器，包括：节点标签，亲和性和多种容器部策略如 binpack、spread、random 等等。

服务的持续可用性

Docker Swarm 由 Swarm Manager 提供高可用性，通过创建多个 Swarm master 节点和制定主 master 节点宕机时的备选策略。如果一个 master 节点宕机，那么一个 slave 节点就会被升格为 master 节点，直到原来的 master 节点恢复正常。 此外，如果某个节点无法加入集群，Swarm 会继续尝试加入，并提供错误警报和日志。在节点出错时，Swarm 现在可以尝试把容器重新调度到正常的节点上去。

和 Docker API 及整合支持的兼容性 Swarm 对 Docker API 完全支持，这意味着它能为使用不同 Docker 工具（如 Docker CLI，Compose，Trusted Registry，Hub 和 UCP）的用户提供无缝衔接的使用体验。

Docker Swarm 为 Docker 化应用的核心功能（诸如多主机网络和存储卷管理）提供原生支持。开发的 Compose 文件能（通过 docker-compose up ）轻易地部署到测试服务器或 Swarm 集群上。Docker Swarm 还可以从 Docker Trusted Registry 或 Hub 里 pull 并 run 镜像。

综上所述，Docker Swarm 提供了一套高可用 Docker 集群管理的解决方案，完全支持标准的 Docker API，方便管理调度集群 Docker 容器，合理充分利用集群主机资源。

> * 并非所有服务都应该部署在Swarm集群内。数据库以及其它有状态服务就不适合部署在Swarm集群内。*

相关概念

节点

运行 Docker 的主机可以主动初始化一个 Swarm 集群或者加入一个已存在的 Swarm 集群，这样这个运行 Docker 的主机就成为一个 Swarm 集群的节点 (node) 。节点分为管理 (manager) 节点和工作 (worker) 节点。

管理节点用于 Swarm 集群的管理，docker swarm 命令基本只能在管理节点执行（节点退出集群命令 docker swarm leave 可以在工作节点执行）。一个 Swarm 集群可以

有多个管理节点，但只有一个管理节点可以成为 leader，leader 通过 raft 协议实现。工作节点是任务执行节点，管理节点将服务 (service) 下发至工作节点执行。管理节点默认也作为工作节点。你也可以通过配置让服务只运行在管理节点。下图展示了集群中管理节点与工作节点的关系。

服务和任务

任务（Task）是 Swarm 中的最小的调度单位，目前来说就是一个单一的容器。 服务（Services）是指一组任务的集合，服务定义了任务的属性。服务有两种模式：

- replicated services 按照一定规则在各个工作节点上运行指定个数的任务。
- global services 每个工作节点上运行一个任务

两种模式通过 docker service create 的 --mode 参数指定。下图展示了容器、任务、服务的关系。

创建 Swarm 集群

我们知道 Swarm 集群由管理节点和工作节点组成。我们来创建一个包含一个管理节点和两个工作节点的最小 Swarm 集群。

初始化集群

查看虚拟主机，现在没有

```
1. docker-machine ls
2. NAME    ACTIVE    DRIVER    STATE    URL    SWARM    DOCKER    ERRORS
```

使用 virtualbox 创建管理节点

```
1. docker-machine create --driver virtualbox manager1
2. #进入管理节点
3. docker-machine ssh manager1
```

执行 sudo -i 可以进入Root 权限

我们使用 docker swarm init 在 manager1 初始化一个 Swarm 集群。

```
docker@manager1:~$ docker swarm init --advertise-addr 192.168.99.100
Swarm initialized: current node (j0o7sykkvi86xpc00w71ew5b6) is now a manager.
To add a worker to this swarm, run the following command:
    docker swarm join --token SWMTKN-1-47z6jld2o465z30dl7pie2kqe4oyug4fxdtbgkfjqgybsy4esl-8r55lxhxs7ozfil45gedd5b8a 192.168.99.100:2377
To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions.
```

如果你的 Docker 主机有多个网卡，拥有多个 IP，必须使用 --advertise-addr 指定 IP。 执行 docker swarm init 命令的节点自动成为管理节点。

命令 `docker info` 可以查看 swarm 集群状态:

```
Containers: 0
Running: 0
Paused: 0
Stopped: 0
 ...snip...
Swarm: active
  NodeID: dxn1zf6l61qsb1josjja83ngz
  Is Manager: true
  Managers: 1
  Nodes: 1
  ...snip...
```

命令 `docker node ls` 可以查看集群节点信息:

```
docker@manager1:~$ docker node ls
ID                           HOSTNAME            STATUS              AVAILABILITY        MANAGER STATUS
1ipck4z2uuwf11f4b9mnon2ul *  manager1            Ready               Active              Leader
```

退出虚拟主机

```
docker@manager1:~$ exit
```

增加工作节点

上一步初始化了一个 Swarm 集群，拥有了一个管理节点，在 Docker Machine 一节中我们了解到 Docker Machine 可以在数秒内创建一个虚拟的 Docker 主机，下面我们使用它来创建两个 Docker 主机，并加入到集群中。

创建虚拟主机 worker1

创建主机

```
$ docker-machine create -d virtualbox worker1
```

进入虚拟主机 worker1

```
$ docker-machine ssh worker1
```

加入 swarm 集群

```
docker@worker1:~$ docker swarm join \
      --token SWMTKN-1-47z6jld2o465z30dl7pie2kqe4oyug4fxdtbgkfjqgybsy4esl-8r55lxhxs7ozfil45gedd5b8a \
      192.168.99.100:2377
This node joined a swarm as a worker.
```

退出虚拟主机

```
docker@worker1:~$ exit
```

创建虚拟主机 worker2

创建

```
$ docker-machine create -d virtualbox worker2
```

入虚拟主机worker2

```
$ docker-machine ssh worker2
```

加入 swarm 集群

```
docker@worker2:~$ docker swarm join \
      --token SWMTKN-1-47z6jld2o465z30dl7pie2kqe4oyug4fxdtbgkfjqgybsy4esl-8r55lxhxs7ozfil45gedd5b8a \
      192.168.99.100:2377
This node joined a swarm as a worker.
```

退出虚拟主机

```
docker@worker2:~$ exit
```

两个工作节点添加完成

查看集群

进入管理节点：

```
docker-machine ssh manager1
```

宿主机子上查看虚拟主机

```
docker@manager1:~$ docker-machine ls
NAME           ACTIVE     DRIVER           STATE       URL                         SWARM     DOCKER              ERRORS
manager1     *            virtualbox     Running     tcp://192.168.99.100:2376             v17.12.1-ce
worker1        -            virtualbox     Running     tcp://192.168.99.101:2376             v17.12.1-ce
```

```
5. worker2      -            virtualbox    Running     tcp://192.168.99.102:2376
                v17.12.1-ce
```

在主节点上面执行 docker node ls 查询集群主机信息

```
1. docker@manager1:~$ docker node ls
2. ID                                                  HOSTNAME                  STATUS
                    AVAILABILITY            MANAGER STATUS
3. 1ipck4z2uuwf11f4b9mnon2ul *     manager1                  Ready
     Active                         Leader
4. rtcpqgcn2gytnvufwfveukgrv       worker1                   Ready
       Active
5. te2e9tr0qzbetjju5gyahg6f7       worker2                   Ready
       Active
```

这样我们就创建了一个最小的 Swarm 集群，包含一个管理节点和两个工作节点。

部署服务

我们使用 docker service 命令来管理 Swarm 集群中的服务，该命令只能在管理节点运行。

新建服务

进入集群管理节点：

```
1. docker-machine ssh manager1
```

使用 docker 中国镜像

```
1. docker search alpine
2. docker pull registry.docker-cn.com/library/alpine
```

现在我们在上一节创建的 Swarm 集群中运行一个名为 helloworld 服务。

```
1. docker@manager1:~$ docker service create --replicas 1 --name helloworld
alpine ping ityouknow.com
2. rwpw7eij4v6h6716jvqvpxbyv
3. overall progress: 1 out of 1 tasks
4. 1/1: running   [==================================================>]
5. verify: Service converged
```

命令解释：

- `docker service create` 命令创建一个服务
- `--name` 服务名称命名为 `helloworld`
- `--replicas` 设置启动的示例数
- `alpine`指的是使用的镜像名称， `ping ityouknow.com`指的是容器运行的bash

使用命令 `docker service ps rwpw7eij4v6h6716jvqvpxbyv` 可以查看服务进展

```
1. docker@manager1:~$ docker service ps rwpw7eij4v6h6716jvqvpxbyv
2. ID                  NAME                IMAGE
      NODE                DESIRED STATE       CURRENT STATE
              ERROR               PORTS
3. rgroe3s9qa53        helloworld.1        alpine:latest
worker1             Running             Running about a minute ago
```

使用 docker service ls 来查看当前 Swarm 集群运行的服务。

```
1. docker@manager1:~$ docker service ls
2. ID                  NAME                MODE
       REPLICAS            IMAGE               PORTS
3. yzfmyggfky8c        helloworld          replicated          0/1
                    alpine:latest
```

监控集群状态

登录管理节点 manager1

```
1. docker-machine ssh manager1
```

运行 `docker service inspect--pretty<SERVICE-ID>` 查询服务概要状态，以 helloworld 服务为例：

```
1. docker@manager1:~$   docker service inspect --pretty helloworld
2. ID:                  rwpw7eij4v6h6716jvqvpxbyv
3. Name:                helloworld
4. Service Mode:    Replicated
5.  Replicas:           1
6. Placement:
7. UpdateConfig:
8.  Parallelism:    1
9.  On failure:         pause
10.  Monitoring Period: 5s
11.  Max failure ratio: 0
12.  ...
13.  Rollback order:        stop-first
14. ContainerSpec:
15.  Image:
alpine:latest@sha256:7b848083f93822dd21b0a2f14a110bd99f6efb4b838d499df6d04a49d0debf8b
16.  Args:              ping ityouknow.com
17. Resources:
18. Endpoint Mode:   vip
```

运行 `docker service inspect helloworld` 查询服务详细信息。

运行 `docker service ps<SERVICE-ID>` 查看那个节点正在运行服务:

```
docker@manager1:~$  docker service ps helloworld
NAME                                                 IMAGE     NODE
DESIRED STATE   LAST STATE
helloworld.1.8p1vev3fq5zm0mi8g0as41w35   alpine    worker1   Running
  Running 3 minutes
```

在工作节点查看任务的执行情况

```
docker-machine ssh  worker1
```

在节点执行 `docker ps` 查看容器的运行状态。

```
docker@worker1:~$  docker ps
CONTAINER ID            IMAGE                         COMMAND
    CREATED                  STATUS                      PORTS
     NAMES
96bf5b1d8010            alpine:latest          "ping ityouknow.com"    4
minutes ago          Up 4 minutes
  helloworld.1.rgroe3s9qa53lf4u4ky0tzcb8
```

这样的话，我们在 Swarm 集群中成功的运行了一个 helloworld 服务，根据命令可以看出在 worker1 节点上运行。

弹性伸缩实验

我们来做一组实验来感受 Swarm 强大的动态水平扩展特性，首先动态调整服务实例个数。

调整实例个数

增加或者减少服务的节点数

调整 helloworld 的服务实例数为2个

```
docker service update --replicas 2 helloworld
```

查看那个节点正在运行服务:

```
docker service ps helloworld
ID                                   NAME                                IMAGE
    NODE                        DESIRED STATE          CURRENT STATE
                ERROR                                                PORTS
rgroe3s9qa53            helloworld.1            alpine:latest
manager1                Running                 Running 8 minutes ago
a61nqrmfhyrl            helloworld.2            alpine:latest
worker2                 Running                 Running 9 seconds ago
```

调整 helloworld 的服务实例数为1个

```
docker service update --replicas 1 helloworld
```

再次查看节点运行情况:

```
docker service ps helloworld
ID                  NAME                IMAGE               NODE                DESIRED STATE       CURRENT STATE            ERROR               PORTS
a61nqrmfhyrl        helloworld.2        alpine:latest       worker2             Running             Running about a minute ago
```

再次调整 helloworld 的服务实例数为3个

```
docker service update --replicas 3 helloworld
helloworld
overall progress: 3 out of 3 tasks
1/3: running   [==================================================>]
2/3: running   [==================================================>]
3/3: running   [==================================================>]
verify: Service converged
```

查看节点运行情况：

```
docker@manager1:~$ docker service ps helloworld
ID                  NAME                IMAGE               NODE                DESIRED STATE       CURRENT STATE            ERROR               PORTS
mh7ipjn74o0d        helloworld.1        alpine:latest       worker2             Running             Running 40 seconds ago
1w4p9okvz0xw        helloworld.2        alpine:latest       manager1            Running             Running 2 minutes ago
snqrbnh4k94y        helloworld.3        alpine:latest       worker1             Running             Running 32 seconds ago
```

删除集群服务

```
docker service rm helloworld
```

调整集群大小

动态调整 Swarm 集群的工作节点。

添加集群

创建虚拟主机 worker3

```
$ docker-machine create -d virtualbox worker3
```

入虚拟主机 worker3

```
$ docker-machine ssh worker3
```

加入swarm 集群

```
docker@worker3:~$ docker swarm join \
      --token SWMTKN-1-47z6jld2o465z30dl7pie2kqe4oyug4fxdtbgkfjqgybsy4esl-8r55lxhxs7ozfil45gedd5b8a \
      192.168.99.100:2377
This node joined a swarm as a worker.
```

退出虚拟主机

```
docker@worker3:~$exit
```

在主节点上面执行 docker node ls 查询集群主机信息

登录主节点

```
docker-machine ssh   manager1
```

查看集群节点

```
docker@manager1:~$   docker node ls
ID                                            HOSTNAME                  STATUS
                    AVAILABILITY            MANAGER STATUS
j0o7sykkvi86xpc00w71ew5b6 *    manager1                  Ready
     Active                        Leader
xwv8aixasqraxwwpox0d0bp2i       worker1                   Ready
         Active
ij3z1edgj7nsqvl8jgqelrfvy       worker2                   Ready
         Active
i31yuluyqdboyl6aq8h9nk2t5       worker3                   Ready
         Active
```

可以看出集群节点多了 worker3

退出 Swarm 集群

如果 Manager 想要退出 Swarm 集群， 在 Manager Node 上执行如下命令：

```
docker swarm leave
```

就可以退出集群，如果集群中还存在其它的 Worker Node，还希望 Manager 退出集群，则加上一个强制选项，命令行如下所示：

```
docker swarm leave --force
```

在 Worker2 上进行退出测试，登录 worker2 节点

```
docker-machine ssh   worker2
```

执行退出命令

```
docker swarm leave
```

查看集群节点情况：

```
docker@manager1:~$ docker node ls
```

```
2. ID                                            HOSTNAME                  STATUS
                    AVAILABILITY            MANAGER STATUS
3. j0o7sykkvi86xpc00w71ew5b6 *    manager1                  Ready
     Active                        Leader
4. xwv8aixasqraxwwpox0d0bp2i      worker1                   Ready
       Active
5. ij3z1edgj7nsqvl8jgqelrfvy      worker2                   Down
      Active
6. i31yuluyqdboyl6aq8h9nk2t5      worker3                   Ready
       Active
```

可以看出集群节点 worker2 状态已经下线

也可以再次加入

```
1. docker@worker2:~$ docker swarm join \
2. >       --token SWMTKN-1-47z6jld2o465z30dl7pie2kqe4oyug4fxdtbgkfjqgybsy4esl-
8r55lxhxs7ozfil45gedd5b8a \
3. >       192.168.99.100:2377
4. This node joined a swarm as a worker.
```

再次查看

```
1. docker@manager1:~$ docker node ls
2. ID                                            HOSTNAME                  STATUS
                    AVAILABILITY            MANAGER STATUS
3. j0o7sykkvi86xpc00w71ew5b6 *    manager1                  Ready
     Active                        Leader
4. xwv8aixasqraxwwpox0d0bp2i      worker1                   Ready
       Active
5. 0agpph1vtylm421rhnx555kkc      worker2                   Ready
       Active
6. ij3z1edgj7nsqvl8jgqelrfvy      worker2                   Down
      Active
7. i31yuluyqdboyl6aq8h9nk2t5      worker3                   Ready
       Active
```

可以看出集群节点 worker2 又重新加入到了集群中

重新搭建命令

使用 VirtualBox 做测试的时候，如果想重复实验可以将实验节点删掉再重来。

```
1. //停止虚拟机
2. docker-machine stop [arg...]   //一个或多个虚拟机名称
3. docker-machine stop    manager1 worker1 worker2
```

```
1. //移除虚拟机
2. docker-machine rm [OPTIONS] [arg...]
3. docker-machine rm manager1 worker1 worker2
```

停止、删除虚拟主机后，再重新创建即可。

总结

通过对 Swarm 的学习，强烈感觉到自动化水平扩展的魅力，这样在公司流量爆发的时候，只需要执行一个命令就可以完成实例上线。如果再根据公司的业务流量做自动化控制，那就真正实现了完全自动的动态伸缩。

举个例子，我们可以利用脚本监控公司的业务流量，当流量是某个级别的时候我们启动对应的N个节点数，当流量减少的时候我们也动态的减少服务实例个数，既可以节省公司资源，也不用操心业务爆发被流量击垮。Docker 能发展的这么好还是有原因的，容器化是 DevOps 最重要的一个环节，未来容器化的技术会越来越丰富和完善，智能化运维可期待。

推荐阅读：Docker(五)：Docker 三剑客之 Docker Machine