

docker-swarm 集群搭建和使用

Name : 曲中岭

Email : zlingqu@126.com

Q Q : 441869115

第一章 部署准备

1.1 目的

使用官方提供的 Swarm 工具搭建 Docker 集群，实现小规模 docker 集群的管理。

1.2 规划

OS	: CentOS_7.5 x64
Host1	: 172.16.6.31(docker01), manager 节点, 监听 TCP2377
Host1	: 172.16.6.31(docker01), worker 节点, 监听 TCP2375 (manager 节点, 默认也是 worker 节点, 也会监听 2375)
Host2	: 172.16.6.32(docker02), worker 节点, 监听 TCP2375
Host3	: 172.16.6.33(docker03), worker 节点, 监听 TCP2375
Docker-ce-cli	: 18.09.0
Docker-ce	: 18.09.0
Swarm	: leatest(>1.2.8)

第二章 docker 安装

操作对象：

172.16.6.31

172.16.6.32

172.16.6.33

安装方法有很多，这里选择其中一种，rpm 方式。

2.1 安装

添加 docker 源：

```
yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo
```

或者使用国内阿里云或者清华的源：

```
yum-config-manager --add-repo https://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo
```

```
yum-config-manager --add-repo  
https://mirrors.tuna.tsinghua.edu.cn/docker-ce/linux/centos/docker-ce.repo
```

从指定源安装 docker-ce：

```
yum install docker-ce --enablerepo=docker-ce-stable -y
```

```
systemctl start docker
```

```
systemctl enable docker
```

查看是否开机运行：

```
systemctl list-unit-files|grep docker
```

2.2 确认

```
docker version
```

```
[root@docker01 ~]# docker version
Client:
Version:      18.09.0
API version:  1.39
Go version:   gol.10.4
Git commit:   4d60db4
Built:        Wed Nov  7 00:48:22 2018
OS/Arch:      linux/amd64
Experimental: false

Server: Docker Engine - Community
Engine:
Version:      18.09.0
API version:  1.39 (minimum version 1.12)
Go version:   gol.10.4
Git commit:   4d60db4
Built:        Wed Nov  7 00:19:08 2018
OS/Arch:      linux/amd64
Experimental: false
```

2.3 ubuntu 安装（补充）

方法有很多，这里只说一种。

```
curl -sSL https://get.docker.com/ | sh
service start docker
sysv-rc-conf --list|grep docker
update-rc.d docker start 90 3 4 5 . stop 20 0 1 2 6 .
sysv-rc-conf --list|grep docker
docker version
```

第三章 修改配置

操作对象：

172.16.6.31

172.16.6.32

172.16.6.33

3.1 修改 hostname（可选）

按照规划，修改三台机器的 hostname 为 docker01、docker02、docker03

3.2 host 文件修改

三台机器的 host 文件分别添加如下内容

```
172.16.6.31 docker01
```

```
172.16.6.32 docker02
```

```
172.16.6.33 docker03
```

3.3 网络配置

Swarm 集群通信端口：

```
TCP2377
```

```
TCP7946
```

```
UDP 7946
```

```
UDP 4789
```

网络层要保证集群内部这些端口是相通的，比如 iptables 等配置。

第四章 swarm 启动

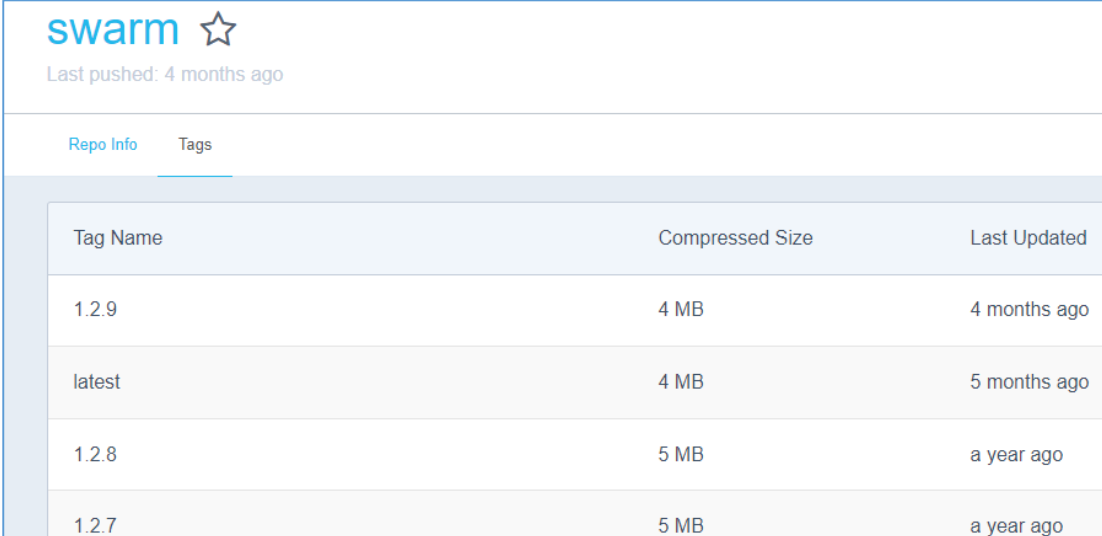
4.1 安装 swarm

在 manage 节点 (172.16.6.31) 上, 执行如下命令安装 swarm

```
docker pull swarm
```

默认安装最新版 (latest), 我这里安装的是比 1.2.8 更新的一个版本, 可到下面地址查看。

<https://hub.docker.com/r/library/swarm/tags/>



Tag Name	Compressed Size	Last Updated
1.2.9	4 MB	4 months ago
latest	4 MB	5 months ago
1.2.8	5 MB	a year ago
1.2.7	5 MB	a year ago

4.2 初始化

```
docker swarm init --advertise-addr 172.16.6.31
```

```
[root@docker01 ~]# docker swarm init --advertise-addr 172.16.6.31
Swarm initialized: current node (j4737k9s35rvj09tij88ihuag) is now a manager.

To add a worker to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-3ywwxc1twcr3mqm3qhkvdqf5xo1z2j37ej67p0v42pen4xil6h-27hwdp7hnwgdmtcwgpcy6q85 172.16.6.31:2377

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions.

[root@docker01 ~]#
[root@docker01 ~]#
[root@docker01 ~]# docker node list
ID                HOSTNAME          STATUS    AVAILABILITY    MANAGER STATUS   ENGINE VERSION
j4737k9s35rvj09tij88ihuag *  docker01        Ready     Active           Leader            18.09.0
[root@docker01 ~]#
```

上面命令执行后, 该机器自动加入到 swarm 集群。同时创建一个集群 token, 获取全球唯一的 token, 作为集群唯一标识。如图, token 是

SWMTKN-1-3ywwxc1twcr3mqm3qhkvdqf5xo1z2j37ej67p0v42pen4xil6h-27hwdp7hnwgdmtcwgpcy6q85

提示中还给出了:

其他 worker 节点如何加入这个集群的命令;

其他 manager 节点如何加入这个集群的命令。

--advertise-addr 参数表示其它 swarm 中的 worker 节点使用此 ip 地址与 manager 联系。

使用

```
docker node list
```

```
docker node ls
```

可以查看当前集群中已经有一个节点了，角色是 leader，即 manager 节点。

4.3 worker 节点加入集群

使用如下命令，将 worker 节点加入集群：

```
docker swarm join --token SWMTKN-1-3ywwxc1twer3mqm3qhkvdqf5xo1z2j37ej67p0v42pen4xil6h-27hwdp7hnwgdmtcwgpcy6q85 172.16.6.31:2377
```

```
[root@docker02 ~]# docker swarm join --token SWMTKN-1-3ywwxc1twer3mqm3qhkvdqf5xo1z2j37ej67p0v42pen4xil6h-27hwdp7hnwgdmtcwgpcy6q85 172.16.6.31:2377
This node joined a swarm as a worker.
[root@docker02 ~]#
```

```
[root@docker03 ~]# docker swarm join --token SWMTKN-1-3ywwxc1twer3mqm3qhkvdqf5xo1z2j37ej67p0v42pen4xil6h-27hwdp7hnwgdmtcwgpcy6q85 172.16.6.31:2377
This node joined a swarm as a worker.
[root@docker03 ~]#
```

在 manager 节点查看，集群有三个节点，其中 docker01 带有星号*标记，表示是 manager 节点

```
[root@docker01 ~]# docker node list
```

ID	HOSTNAME	STATUS	AVAILABILITY	MANAGER STATUS	ENGINE VERSION
j4737k9s35rvj09tij88ihuag *	docker01	Ready	Active	Leader	18.09.0
hbl8xfar2b909dy3hkukasavk	docker02	Ready	Active		18.09.0
pxlaq0t136a9izegt4i738yaj	docker03	Ready	Active		18.09.0

```
[root@docker01 ~]#
```

第五章 常用操作

5.1 节点离开集群

5.1.1 work 节点主动离开集群

worker 节点上执行命令，使 worker 节点离开集群

```
docker swarm leave
```

```
[root@docker02 ~]# docker swarm leave
Node left the swarm.
[root@docker02 ~]#
```

如图，使 docker02 离开集群，在 manage 节点查看，发现 docker 处于 down 状态

```
[root@docker01 ~]# docker node ls
ID                HOSTNAME        STATUS      AVAILABILITY    MANAGER STATUS  ENGINE VERSION
j4737k9s35rvj09tij88ihuag *  docker01      Ready       Active           Leader          18.09.0
hbl8xfar2b909dy3hkukasavk    docker02      Down        Active           -              18.09.0
pxlaq0tl36a9izegt4i738yaj    docker03      Ready       Active           -              18.09.0
[root@docker01 ~]#
```

5.1.2 从 swarm 删除 work 节点

如下如，可以删除一个处于 down 状态的节点

```
docker node rm ID|HOSTNAME
```

```
[root@docker01 ~]# docker node ls
ID                HOSTNAME        STATUS      AVAILABILITY    MANAGER STATUS  ENGINE VERSION
c7mp3eugwzxrhrsz5joq97ter    Unknown       Active           -              18.09.0
j4737k9s35rvj09tij88ihuag *  docker01      Ready       Active           Leader          18.09.0
hbl8xfar2b909dy3hkukasavk    docker02      Down        Active           -              18.09.0
un3vwmkgtgtdxnflvjkl93r3yp    docker02      Ready       Active           -              18.09.0
pxlaq0tl36a9izegt4i738yaj    docker03      Down        Active           -              18.09.0
[root@docker01 ~]#
[root@docker01 ~]#
[root@docker01 ~]# docker node rm hbl8xfar2b909dy3hkukasavk
hbl8xfar2b909dy3hkukasavk
[root@docker01 ~]#
[root@docker01 ~]# docker node ls
ID                HOSTNAME        STATUS      AVAILABILITY    MANAGER STATUS  ENGINE VERSION
c7mp3eugwzxrhrsz5joq97ter    Unknown       Active           -              18.09.0
j4737k9s35rvj09tij88ihuag *  docker01      Ready       Active           Leader          18.09.0
un3vwmkgtgtdxnflvjkl93r3yp    docker02      Ready       Active           -              18.09.0
pxlaq0tl36a9izegt4i738yaj    docker03      Down        Active           -              18.09.0
[root@docker01 ~]#
```

不能删除一个处于 ready 状态的集群，需要先执行 5.1.1 步骤，如果一定要删除，会报如下错误

```
[root@docker01 ~]# docker node rm un3vwmkgtgtdxnflvjkl93r3yp
Error response from daemon: rpc error: code = FailedPrecondition desc = node un3vwmkgtgtdxnflvjkl93r3yp is not down and can't be removed
[root@docker01 ~]#
[root@docker01 ~]#
```

可以使用 -f 参数，强制删除一个处于 ready 状态的集群

```
docker node rm -f $ID
```

```
[root@docker01 ~]# docker node rm -f un3vwkkgtdxnflvjkl193r3yp
un3vwkkgtdxnflvjkl193r3yp
[root@docker01 ~]#
[root@docker01 ~]#
[root@docker01 ~]# docker node ls
```

ID	HOSTNAME	STATUS	AVAILABILITY	MANAGER STATUS	ENGINE VERSION
c7mp3eugwzxrhrz5joq97ter		Unknown	Active		
j4737k9s35rvj09ti188ihuag *	docker01	Ready	Active	Leader	18.09.0
pxlaq0t136a9izegt4i738ya	docker03	Down	Active		18.09.0

```
[root@docker01 ~]#
[root@docker01 ~]#
```

如果使用 -f 强制删除一个处于 ready 状态的 worker 节点，如果要想再次加入，需要重新执行如下命令

```
docker swarm leave
```

测试效果如下图：

```
[root@docker02 ~]# docker swarm join --token SWMTKN-1-3ywwxcltwer3mqm3qhkvdqf5x0lz2j37ej67p0v42pen4xil6h-27hwdp7hnwgdmtcwgpcy6q85 172.16.6.31:2377
Error response from daemon: This node is already part of a swarm. Use "docker swarm leave" to leave this swarm and join another one.
[root@docker02 ~]#
[root@docker02 ~]#
[root@docker02 ~]# docker swarm leave
Node left the swarm.
[root@docker02 ~]#
[root@docker02 ~]# docker swarm join --token SWMTKN-1-3ywwxcltwer3mqm3qhkvdqf5x0lz2j37ej67p0v42pen4xil6h-27hwdp7hnwgdmtcwgpcy6q85 172.16.6.31:2377
This node joined a swarm as a worker.
[root@docker02 ~]#
[root@docker02 ~]#
```

5.1.3 manager 节点离开集群

当集群中只有最后一个 manager 节点时，使用如下命令可以强制离开集群，即销毁集群。

```
docker swarm leave -f
```

```
[root@docker03 ~]# docker node ls
```

ID	HOSTNAME	STATUS	AVAILABILITY	MANAGER STATUS	ENGINE VERSION
l6oixj8yww1pk5ngfmhwryci *	docker03	Ready	Active	Leader	18.09.0

```
[root@docker03 ~]#
[root@docker03 ~]#
[root@docker03 ~]# docker node rm -f l6oixj8yww1pk5ngfmhwryci
Error response from daemon: rpc error: code = FailedPrecondition desc = node l6oixj8yww1pk5ngfmhwryci is a cluster manager and is a member of the raft cluster. It must be demoted before removal
[root@docker03 ~]#
[root@docker03 ~]# docker-
docker-init docker-proxy
[root@docker03 ~]# docker-
docker-init docker-proxy
[root@docker03 ~]#
[root@docker03 ~]#
[root@docker03 ~]# docker swarm leave -f
Node left the swarm.
[root@docker03 ~]#
```

注意，这时不可以使用如下命令删除

```
docker node rm -f l6oixj8yww1pk5ngfmhwryci
```

5.2 添加节点

5.2.1 添加 work 节点

在 manager 节点上使用

```
docker swarm join-token worker
```

获取 worker 的 token 及加入 worker 节点的命令

```
[root@docker01 ~]# docker swarm join-token worker
To add a worker to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-3ywwxcltwer3mqm3qhkvdqf5x0lz2j37ej67p0v42pen4xil6h-27hwdp7hnwgdmtcwgpcy6q85 172.16.6.31:2377

[root@docker01 ~]#
```

然后在 docker02 上运行改命令

```
[root@docker02 ~]# docker swarm join --token SWMTKN-1-3ywwxcltwer3mqm3qhkvdqf5x0lz2j37ej67p0v42pen4xil6h-27hwdp7hnwgdmtcwgpcy6q85 172.16.6.31:2377
This node joined a swarm as a worker.
[root@docker02 ~]#
```

如下图，再次加入后，会重新分配一个 ID，原有的还是处于 down 状态


```
[root@docker01 ~]# docker node ls
ID                HOSTNAME        STATUS        AVAILABILITY        MANAGER STATUS        ENGINE VERSION
c7mp3eugwzxrhrz5joq97ter   Unknown        Active
j4737k9s35rvj09t1j88ihuag *   docker01       Ready        Active              Leader                18.09.0
hbl8xfar2b909dy3hkukasavk   docker02       Down         Active              -                    18.09.0
un3vwwkkgtdxnflvj193r3yp     docker02       Ready        Active              -                    18.09.0
pxlaq0t136a9izegt4i738yaj    docker03       Ready        Active              -                    18.09.0
[root@docker01 ~]#
[root@docker01 ~]#
```

对于如 5.1 主动离开的 worker 节点也可以使用这种方法，只是加入后会重新生成一个新的 ID，原有 ID 仍处于 down 状态，如上图。

5.2.2 添加 manager 节点

与 5.2 类似，不再演示，唯一的区别是，查询命令不同，如下。

```
docker swarm join-token manager
```

5.3 修改节点

5.3.1 节点的可用性

availability 参数有

active：调度器能够安排任务到该节点

drain：调度器不能够安排任务到该节点，而且会停止已存在的任务，并将这些任务分配到其他 Active 状态的节点

pause：调度器不能够安排任务到该节点，但是已经存在的任务会继续运行

```
[root@docker01 ~]# docker node ls
ID                HOSTNAME        STATUS        AVAILABILITY        MANAGER STATUS        ENGINE VERSION
j4737k9s35rvj09t1j88ihuag *   docker01       Ready        Drain               Leader                18.09.0
b10wegkpw9nr2t0aokzj804       docker02       Ready        Pause               -                    18.09.0
r8ofwf9gjjqa49gmh5dyihv1      docker03       Down         Active               -                    18.09.0
[root@docker01 ~]#
[root@docker01 ~]#
```

可使用如下命令进行状态转换：

```
docker node update --availability drain $ID|HOSTNAME
```

```
docker node update --availability active $ID|HOSTNAME
```

```
docker node update --availability pause $ID|HOSTNAME
```

5.3.1 状态转换

MANAGER STATUS 的三种状态

Leader：为群体做出所有群管理和编排决策的主要管理者节点

Reachable：如果 Leader 节点变为不可用，该节点有资格被选举为新的 Leader

Unavailable：该节点不能和其他 Manager 节点产生任何联系，这种情况下，应该添加一个新的 Manager 节点到集群，或者将一个 Worker 节点提升为 Manager 节点

提升节点：

```
docker node promote b10wegkpwtw9nr2t0aokzj804
```

```
docker node promote docker02
```

提升后变成 Reachable 状态

```
[root@docker01 ~]# docker node promote docker02
Node docker02 promoted to a manager in the swarm.
[root@docker01 ~]#
[root@docker01 ~]#
[root@docker01 ~]# docker node ls
```

ID	HOSTNAME	STATUS	AVAILABILITY	MANAGER STATUS	ENGINE VERSION
t9k2bso0tre4tj2fr74ia50k *	docker01	Ready	Active	Leader	18.09.1
cjk3tie9qoameh8mldf8tv85r	docker02	Ready	Active	Reachable	18.09.1
jkwbdl68rlxzqfa4ee517ryzg	docker03	Ready	Active		18.09.1
gmhhlir23oc3kstibf441sk5	docker04	Ready	Active		18.09.1

```
[root@docker01 ~]#
```

降低节点：

```
docker node demote b10wegkpwtw9nr2t0aokzj804
```

```
[root@docker01 ~]#
[root@docker01 ~]# docker node ls
```

ID	HOSTNAME	STATUS	AVAILABILITY	MANAGER STATUS	ENGINE VERSION
t9k2bso0tre4tj2fr74ia50k *	docker01	Ready	Active	Leader	18.09.1
cjk3tie9qoameh8mldf8tv85r	docker02	Ready	Active		18.09.1
jkwbdl68rlxzqfa4ee517ryzg	docker03	Ready	Active		18.09.1
gmhhlir23oc3kstibf441sk5	docker04	Ready	Active		18.09.1

```
[root@docker01 ~]#
[root@docker01 ~]#
[root@docker01 ~]# docker node promote docker02
Node docker02 promoted to a manager in the swarm.
[root@docker01 ~]#
[root@docker01 ~]# docker node ls
```

ID	HOSTNAME	STATUS	AVAILABILITY	MANAGER STATUS	ENGINE VERSION
t9k2bso0tre4tj2fr74ia50k *	docker01	Ready	Active	Leader	18.09.1
cjk3tie9qoameh8mldf8tv85r	docker02	Ready	Active	Reachable	18.09.1
jkwbdl68rlxzqfa4ee517ryzg	docker03	Ready	Active		18.09.1
gmhhlir23oc3kstibf441sk5	docker04	Ready	Active		18.09.1

```
[root@docker01 ~]#
[root@docker01 ~]#
[root@docker01 ~]# docker node demote docker02
Manager docker02 demoted in the swarm.
[root@docker01 ~]#
[root@docker01 ~]#
[root@docker01 ~]# docker node ls
```

ID	HOSTNAME	STATUS	AVAILABILITY	MANAGER STATUS	ENGINE VERSION
t9k2bso0tre4tj2fr74ia50k *	docker01	Ready	Active	Leader	18.09.1
cjk3tie9qoameh8mldf8tv85r	docker02	Ready	Active		18.09.1
jkwbdl68rlxzqfa4ee517ryzg	docker03	Ready	Active		18.09.1
gmhhlir23oc3kstibf441sk5	docker04	Ready	Active		18.09.1

```
[root@docker01 ~]#
[root@docker01 ~]#
```

第六章 服务管理

6.1 创建服务

manager 节点上操作:

```
docker service create --name my_web --replicas 3 --publish 8080:80 nginx:1.13
#创建 nginx 容器
docker service create --replicas 1 --name myping alpine ping www.baidu.com
#创建 alpine 容器, 执行 ping www.baidu.com 命令
```

docker service create 创建服务, 以下是常见的选项:

- name my_web 服务的名字, 如果有多个副本, 容器的名字以此为前缀
- replicas 3 指定生成 3 个容器分片
- mode global 在所有 worker 节点上都启动一个副本, 和 --replicas 参数冲突
- publish 指定端口映射情况, 类似于 docker run -p 参数, 建议使用如下形式
- publish published=8080,target=80 这种形式是新版风格
- with-registry-auth 从私有仓库下载镜像而不是从官方, 需要首先登陆, 如下官网的例子

```
$ docker login registry.example.com
$ docker service create \
  --with-registry-auth \
  --name my_service \
  registry.example.com/acme/my_image:latest
```

nginx 镜像名称, 默认 latest 版本

注意:

启用服务的各个 worker 节点本地都会下载这个 nginx 镜像

6.2 查看服务

查看正在运行的服务列表:

```
docker service ls
```

```
[root@docker01 ~]# docker service ls
ID                NAME           MODE           REPLICAS        IMAGE           PORTS
y0na80bujty5     helloworld     replicated     1/1             alpine:latest
ln4ofudxz5zz     my_web        replicated     3/3             nginx:latest
no2ueg017wqf     myping        replicated     1/1             alpine:latest
[root@docker01 ~]#
```

查看服务的详细信息:

```
docker service inspect --pretty my_web
```

```
[root@docker01 ~]# docker service inspect --pretty my_web
```

```

ID:          1n4ofudxz5zzw9bpqldw7tgrp
Name:        my_web
Service Mode: Replicated
  Replicas:   3
Placement:
UpdateConfig:
  Parallelism: 1
  On failure:  pause
  Monitoring Period: 5s
  Max failure ratio: 0
  Update order: stop-first
RollbackConfig:
  Parallelism: 1
  On failure:  pause
  Monitoring Period: 5s
  Max failure ratio: 0
  Rollback order: stop-first
ContainerSpec:
  Image:       nginx:latest@sha256:5d32f60db294b5deb55d078cd4feb410ad88e6fe77500c87d3970eca97f54dba
  Init:        false
Resources:
Endpoint Mode: vip
Ports:
  PublishedPort = 9000
  Protocol = tcp
  TargetPort = 80
  PublishMode = ingress

```

查看正在运行服务的节点：

```
docker service ps my_web
```

```
[root@docker01 ~]# docker service ps my_web
```

ID	NAME	IMAGE	NODE	DESIRED STATE	CURRENT STATE	ERROR
hlpxts5rzse6	my_web.1	nginx:latest	docker03	Running	Running about an hour ago	
wprhf6jx0ca3	my_web.2	nginx:latest	docker01	Running	Running about an hour ago	
fg7ph0kxtzt6	my_web.3	nginx:latest	docker02	Running	Running about an hour ago	

6.3 扩容/缩容服务

即扩大或减少服务运行的副本数量，如下，扩容 my_web 的副本数从 3 到 5

```
docker service scale my_web=5
```

```
[root@docker01 ~]# docker service ps my_web
```

ID	NAME	IMAGE	NODE	DESIRED STATE	CURRENT STATE	ERROR
hlpxts5rzse6	my_web.1	nginx:latest	docker03	Running	Running about an hour ago	
wprhf6jx0ca3	my_web.2	nginx:latest	docker01	Running	Running about an hour ago	
fg7ph0kxtzt6	my_web.3	nginx:latest	docker02	Running	Running about an hour ago	

```

[root@docker01 ~]#
[root@docker01 ~]#
[root@docker01 ~]#
[root@docker01 ~]#
[root@docker01 ~]# docker service scale my_web=5
my_web scaled to 5
overall progress: 5 out of 5 tasks
1/5: running [=====>]
2/5: running [=====>]
3/5: running [=====>]
4/5: running [=====>]
5/5: running [=====>]
verify: Service converged
[root@docker01 ~]#
[root@docker01 ~]# docker service ps my_web

```

ID	NAME	IMAGE	NODE	DESIRED STATE	CURRENT STATE	ERROR
hlpxts5rzse6	my_web.1	nginx:latest	docker03	Running	Running about an hour ago	
wprhf6jx0ca3	my_web.2	nginx:latest	docker01	Running	Running about an hour ago	
fg7ph0kxtzt6	my_web.3	nginx:latest	docker02	Running	Running about an hour ago	
xd86tcmgluh7	my_web.4	nginx:latest	docker01	Running	Running 7 seconds ago	
b4wl6x7wybtf	my_web.5	nginx:latest	docker03	Running	Running 7 seconds ago	

同样也可以使用此命令较少服务的副本数量。

6.4 删除服务

```
docker service rm my_web
```

```
[root@dock01 ~]# docker service rm my_web
my_web
```

6.5 更新服务

例如使用如下命令创建 redis 服务，版本是 3.0.6

```
docker service create \
  --replicas 3 \
  --name redis \
  --update-delay 30s \
  redis:3.0.6
```

可以使用如下命令，对 redis 服务进行滚动更新。

```
docker service update --image redis:3.0.7 redis
```

默认情况下，调度程序一次更新 1 个任务。您可以传递该 `--update-parallelism` 标志以配置调度程序同时更新的最大服务任务数。

默认情况下，当对单个任务的更新返回状态时 `RUNNING`，调度程序会安排另一个任务进行更新，直到所有任务都更新为止。如果在更新任务期间的任何时间返回 `FAILED`，则调度程序会暂停更新。您可以使用或 `--update-failure-action` 标志来控制行为。

更新完成之后，查看服务，发现 3.0.6 都停止了，运行的都是 3.0.7。

```
[root@dock01 ~]# docker service ps redis
[root@dock01 ~]# docker service ps redis
ID            NAME      IMAGE          NODE      DESIRED STATE   CURRENT STATE           ERRORS
gsc91lp44geq  redis.1   redis:3.0.7    docker03  Running         Running 59 seconds ago
0ap6j2pycjdd  \_redis.1 redis:3.0.6    docker03  Shutdown        Shutdown about a minute ago
krhuydp3rit   redis.2   redis:3.0.7    docker01  Running         Running 8 seconds ago
wlcrs29bqdr1  \_redis.2 redis:3.0.6    docker01  Shutdown        Shutdown 27 seconds ago
jzllf0p0dz6v  redis.3   redis:3.0.7    docker02  Running         Running about a minute ago
00ntvizgwj4d  \_redis.3 redis:3.0.6    docker02  Shutdown        Shutdown 2 minutes ago
```

6.6 查看日志

```
docker service logs NAME
```

```
docker service logs -f NAME
```

6.7 回滚配置

比如修改了服务的副本数 4—>3，使用如下命令进行回滚操作

```
docker service rollback my_web
```

```
[root@dock01 ~]#
[root@dock01 ~]# docker service rollback my_web
my_web
service rolled back: rollback completed
[root@dock01 ~]#
```

回滚副本数量这个操作，注意 4—>3—>4，新的第四个节点相当于重新部署，可能与其他三个节点不一致（比如修改了 html 页面）。

6.8 转移副本

接着上一节，使用如下命令

```
docker node update --availability drain docker01
```

将 docker01 节点设置为 drain 状态，则正如 5.3.1 所说，该节点将不再接受调度器的任务指派，并将已经指派过来（使用 swarm 分发而不是 docker run 启动的）的任务分配到其他 active 节点。

```
[root@docker01 ~]# docker service ps redis
ID                NAME      IMAGE          NODE      DESIRED STATE   CURRENT STATE
gsc91lp44qeq      redis.1    redis:3.0.7    docker03   Running          Running 7 minutes ago
0ap6j2pycjdd      \_ redis.1  redis:3.0.6    docker03   Shutdown        Shutdown 7 minutes ago
y3hpjmu5io4k      redis.2    redis:3.0.7    docker03   Running          Running about a minute ago
krhuiypd3rit      \_ redis.2    redis:3.0.7    docker01   Shutdown        Shutdown about a minute ago
wlcrs29bqdr1      \_ redis.2    redis:3.0.6    docker01   Shutdown        Shutdown 6 minutes ago
jz1lf0p0dz6v      redis.3    redis:3.0.7    docker02   Running          Running 7 minutes ago
00ntvizgwj4d      \_ redis.3    redis:3.0.6    docker02   Shutdown        Shutdown 8 minutes ago
[root@docker01 ~]#
```

此时，即使再次将其设置为 active 节点，只会对以后新的任务指派生效，而不会将已经调度出的任务再调度回来，如上图，不会将 docker03 上的 redis，再返回给一个给 docker01。

6.9 负载均衡测试

创建 4 个节点的 nginx，提供四个不同的首页内容

```
echo docker—01 >index.html
```

```
docker container cp index.html e2f62887015c:/usr/share/nginx/html
```

测试负载均衡，发送 20 次请求到某个节点：

```
for i in {1..20}; do curl 172.16.6.33:8080 ;done
```

```
[root@docker06 ~]#
[root@docker06 ~]# for i in {1..20}; do curl 172.16.6.33:8080 ;done
docker--02
docker--01
docker--03
docker--04
docker--02
docker--01
docker--03
docker--04
docker--02
docker--01
docker--03
docker--04
docker--02
docker--01
docker--03
docker--04
docker--02
docker--01
docker--03
docker--04
docker--02
docker--01
docker--03
docker--04
```

返回内容符合前期规划。

测试发现，启动了四个 nginx 节点，只有两个可以正常接受请求，具体原因还待分析。

第七章 使用 swarm 模式路由

<https://docs.docker.com/engine/swarm/ingress/#publish-a-port-for-a-service>