DOCKER 高级特性

本次分享给大家介绍Docker 的高级特性与相应的工具。 它们就是Docker 三剑客，Compose、Machine和Swarm

COMPOSE

介绍

Docker Compose 是Docker 官方编排（Orchestration）项目之一，负责快速的部署分布式应用。

Compose 定位是 「定义和运行多个Docker 容器的应用（Defining and running multi-container Docker applications）」

其前身是开源项目Fig。其代码目前在https://github.com/docker/compose 上开源。

安装

```
pip install -U docker-compose
```

或

```
sudo curl -L
"https://github.com/docker/compose/releases/download/1.24.1/docker-
compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
```

使用

`Dockerfile`

```
FROM python:3.7-slim

WORKDIR /app

COPY . /app
```

```
RUN pip install flask -i https://mirrors.aliyun.com/pypi/simple  --
trusted-host mirrors.aliyun.com

EXPOSE 80

ENV NAME World

CMD ["python", "app.py"]

```

`app.py`

```python
from flask import Flask
import os
import socket

app = Flask(__name__)

@app.route("/")
def hello():
    html = "<h3>Hello {name}!</h3>" \
           "<b>Hostname:</b> {hostname}"
    return html.format(name=os.getenv("NAME", "world"),
                       hostname=socket.gethostname())

if __name__ == "__main__":
    app.run(host='0.0.0.0', port=80)

```

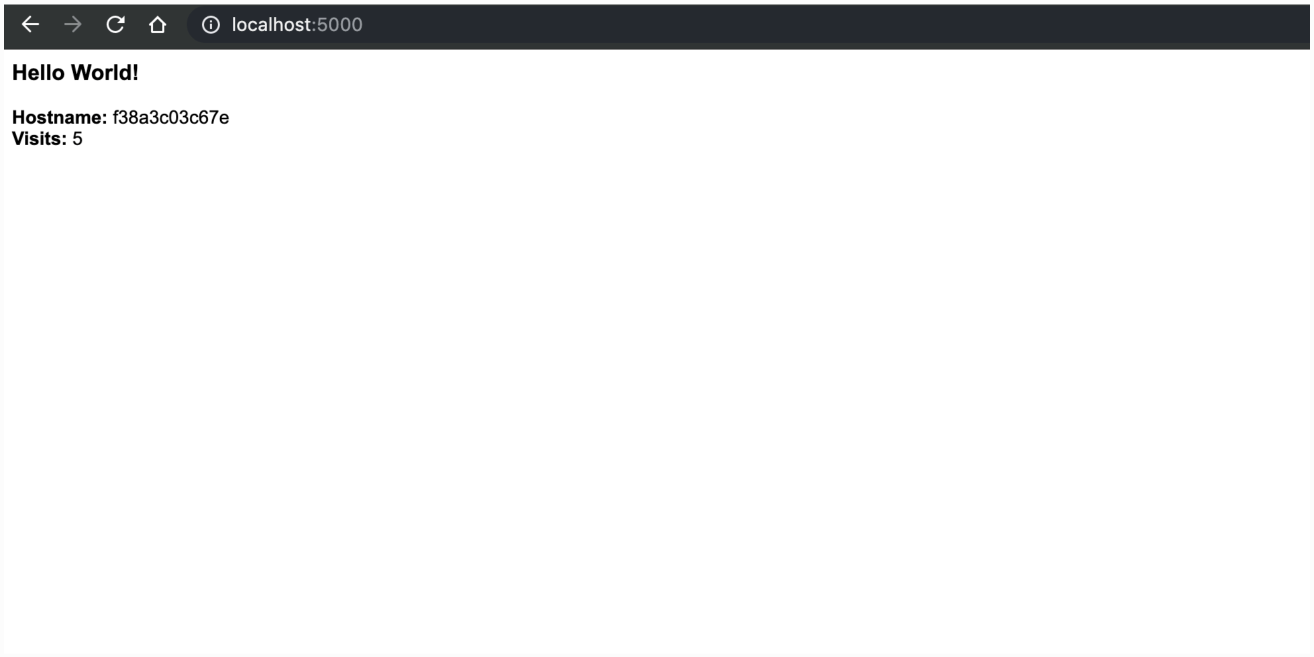
`docker-compose.yml`

```yaml
version: "3"
services:
  myapp:
    # build: .
    image: friendlyhello:v2
    container_name: myapp
    ports:
      - "5000:80"
    environment:
      NAME: World

  redis:
    image: redis
    container_name: web
```

执行 `docker-compose build` 可生成镜像

执行 `docker-compose up` 启动容器运行

浏览器访问



Hello World!

Hostname: f38a3c03c67e
Visits: 5

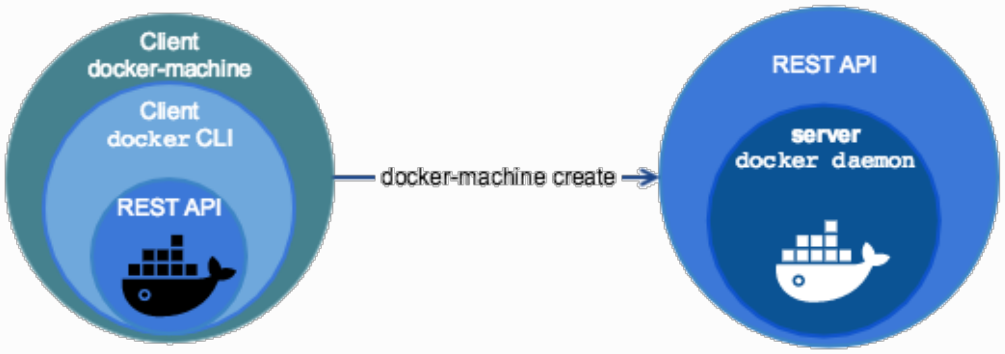
命令说明

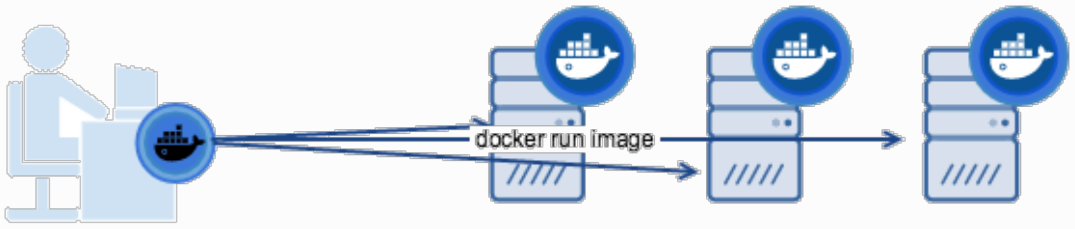
```
Commands:
  build              Build or rebuild services
  bundle             Generate a Docker bundle from the Compose file
  config             Validate and view the Compose file
  create             Create services
  down               Stop and remove containers, networks, images, and volumes
  events             Receive real time events from containers
  exec               Execute a command in a running container
  help               Get help on a command
  images             List images
  kill               Kill containers
  logs               View output from containers
  pause              Pause services
  port               Print the public port for a port binding
  ps                 List containers
  pull               Pull service images
  push               Push service images
  restart            Restart services
  rm                 Remove stopped containers
  run                Run a one-off command
  scale              Set number of containers for a service
  start              Start services
  stop               Stop services
  top                Display the running processes
  unpause            Unpause services
  up                 Create and start containers
  version            Show the Docker-Compose version information
```

MACHINE

介绍

Docker Machine 是 `Docker` 官方编排（Orchestration）项目之一，负责在多种平台上快速安装 `Docker` 环境。

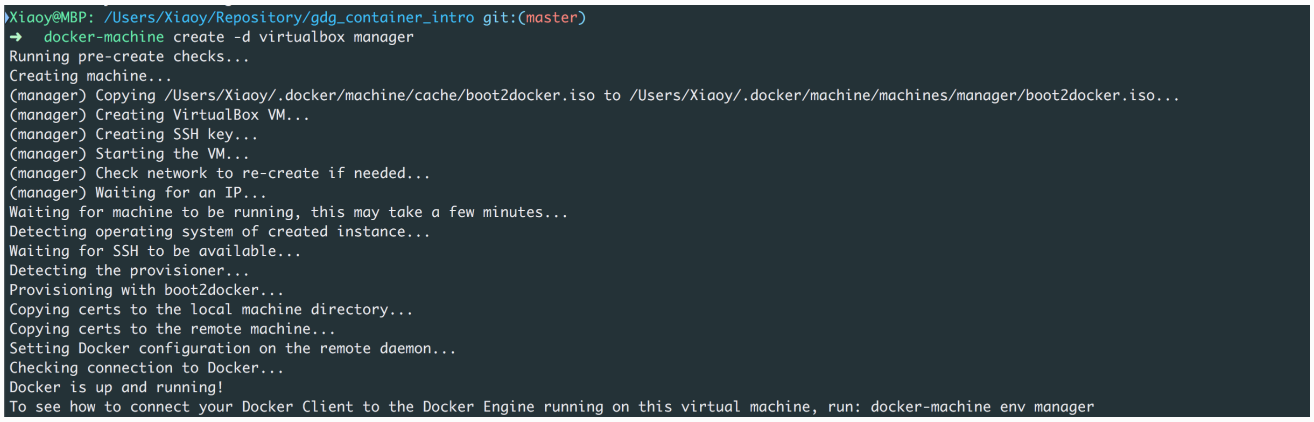




使用

使用 `virtualbox` 类型的驱动，创建一台 `Docker` 主机，命名为 `manager` 。

```
docker-machine create -d virtualbox manager
```



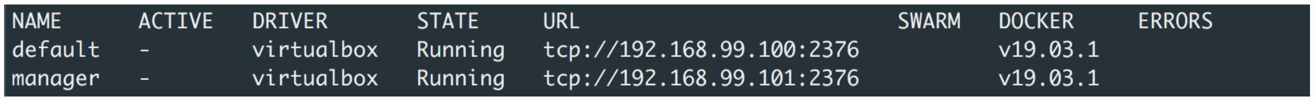
```
Xiaoy@MBP: /Users/Xiaoy/Repository/gdg_container_intro git:(master)
➜  docker-machine create -d virtualbox manager
Running pre-create checks...
Creating machine...
(manager) Copying /Users/Xiaoy/.docker/machine/cache/boot2docker.iso to /Users/Xiaoy/.docker/machine/machines/manager/boot2docker.iso...
(manager) Creating VirtualBox VM...
(manager) Creating SSH key...
(manager) Starting the VM...
(manager) Check network to re-create if needed...
(manager) Waiting for an IP...
Waiting for machine to be running, this may take a few minutes...
Detecting operating system of created instance...
Waiting for SSH to be available...
Detecting the provisioner...
Provisioning with boot2docker...
Copying certs to the local machine directory...
Copying certs to the remote machine...
Setting Docker configuration on the remote daemon...
Checking connection to Docker...
Docker is up and running!
To see how to connect your Docker Client to the Docker Engine running on this virtual machine, run: docker-machine env manager
```

可以在创建时加上如下参数，来配置主机或者主机上的 `Docker` 。

```
--engine-opt dns=114.114.114.114 配置Docker 的默认DNS

--engine-registry-mirror https://registry.docker-cn.com 配置Docker 的仓库镜像

--virtualbox-memory 2048 配置主机内存

--virtualbox-cpu-count 2 配置主机CPU
```

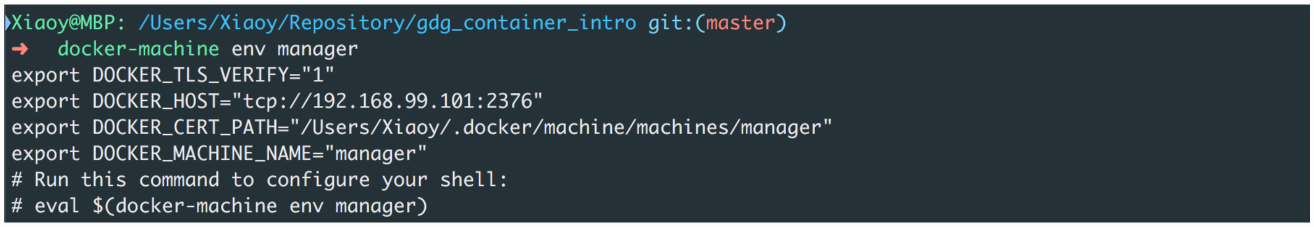
更多参数请使用 `docker-machine create –help` 命令查看。

`docker-machine ls` 查看主机



```
NAME      ACTIVE   DRIVER       STATE     URL                         SWARM   DOCKER     ERRORS
default   -        virtualbox   Running   tcp://192.168.99.100:2376           v19.03.1
manager   -        virtualbox   Running   tcp://192.168.99.101:2376           v19.03.1
```

`docker-machine env manager` 查看环境变量



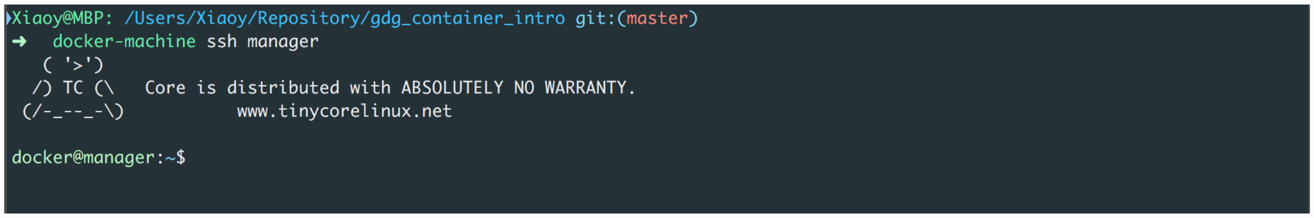
```
Xiaoy@MBP: /Users/Xiaoy/Repository/gdg_container_intro git:(master)
➜  docker-machine env manager
export DOCKER_TLS_VERIFY="1"
export DOCKER_HOST="tcp://192.168.99.101:2376"
export DOCKER_CERT_PATH="/Users/Xiaoy/.docker/machine/machines/manager"
export DOCKER_MACHINE_NAME="manager"
# Run this command to configure your shell:
# eval $(docker-machine env manager)
```

切换 `docker` 主机 `manager` 为操作对象

```
eval $(docker-machine env manager)
```

或者可以 `ssh` 登录到 `docker` 主机

```
docker-machine ssh manager
```



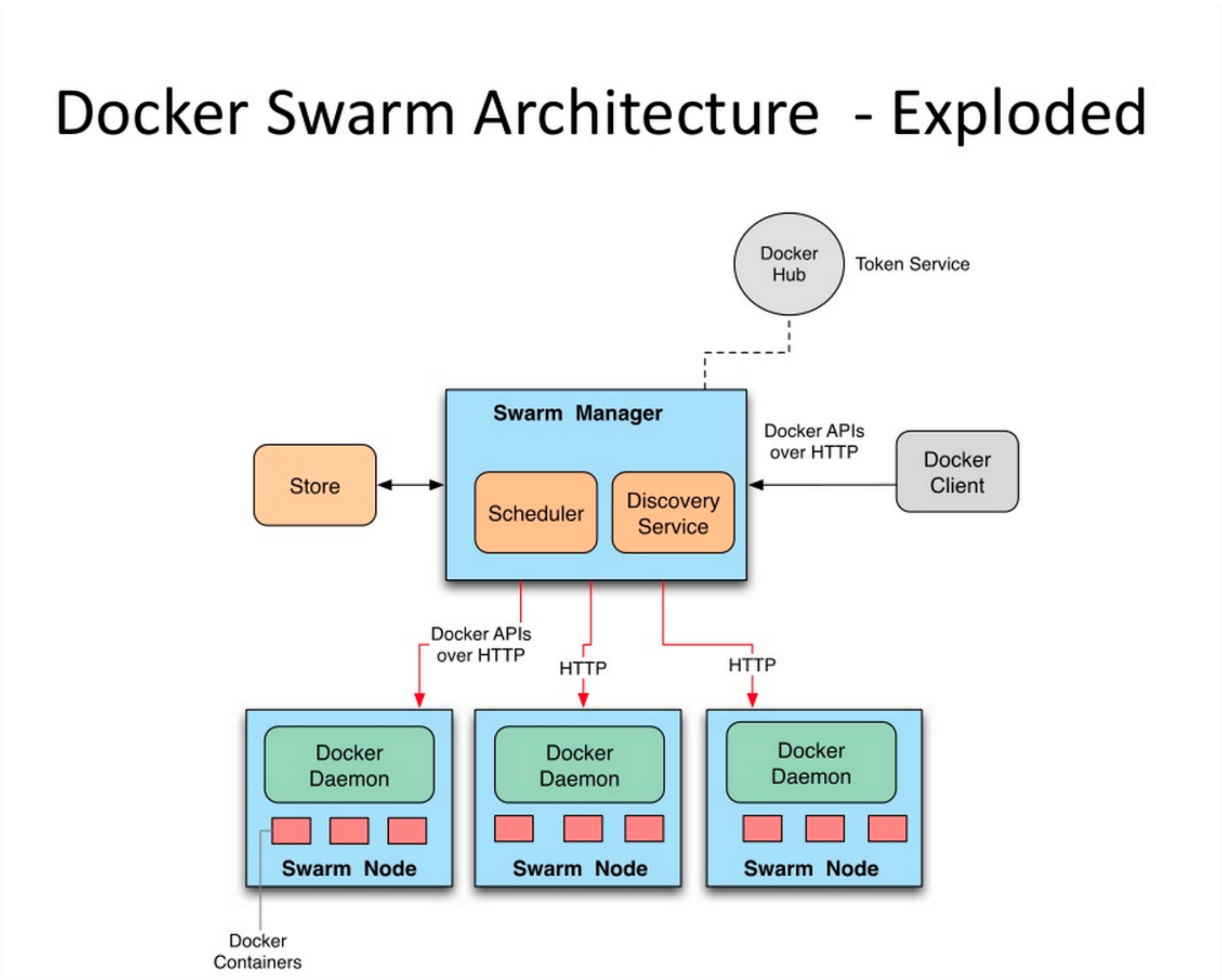
```
Xiaoy@MBP: /Users/Xiaoy/Repository/gdg_container_intro git:(master)
➜  docker-machine ssh manager
   ( '>')
  /) TC (\   Core is distributed with ABSOLUTELY NO WARRANTY.
 (/-_--_-\)           www.tinycorelinux.net

docker@manager:~$
```

命令说明

```
Commands:
  active                  Print which machine is active
  config                  Print the connection config for machine
  create                  Create a machine
  env                     Display the commands to set up the environment for the Docker client
  inspect                 Inspect information about a machine
  ip                      Get the IP address of a machine
  kill                    Kill a machine
  ls                      List machines
  provision               Re-provision existing machines
  regenerate-certs        Regenerate TLS Certificates for a machine
  restart                 Restart a machine
  rm                      Remove a machine
  ssh                     Log into or run a command on a machine with SSH.
  scp                     Copy files between machines
  mount                   Mount or unmount a directory from a machine with SSHFS.
  start                   Start a machine
  status                  Get the status of a machine
  stop                    Stop a machine
  upgrade                 Upgrade a machine to the latest version of Docker
  url                     Get the URL of a machine
  version                 Show the Docker Machine version or a machine docker version
  help                    Shows a list of commands or help for one command
```

SWARM

`Swarm` 是使用 `SwarmKit` 构建的 `Docker` 引擎内置（原生）的集群管理和编排工具。

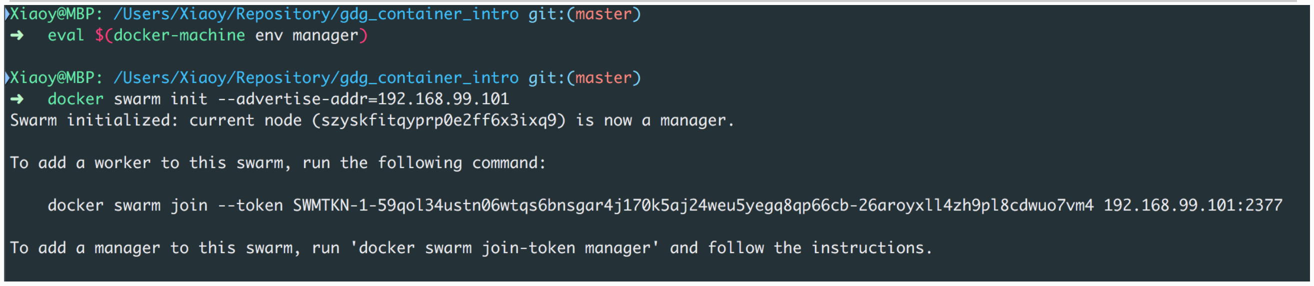


使用

初始化集群

在上节介绍 `docker-machine` 的时候，我们创建了 `manager` 节点，而初始化集群需要在管理节点内执行

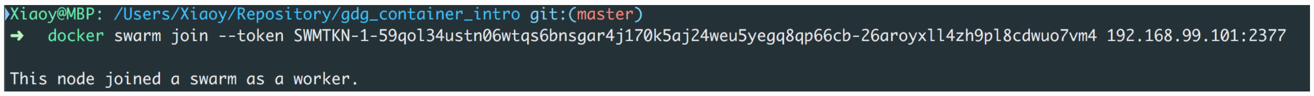
```
docker swarm init --advertise-addr=IP_ADDR
```



```
Xiaoy@MBP: /Users/Xiaoy/Repository/gdg_container_intro git:(master)
➜  eval $(docker-machine env manager)

Xiaoy@MBP: /Users/Xiaoy/Repository/gdg_container_intro git:(master)
➜  docker swarm init --advertise-addr=192.168.99.101
Swarm initialized: current node (szyskfitqyprp0e2ff6x3ixq9) is now a manager.

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

    docker swarm join --token SWMTKN-1-59qol34ustn06wtqs6bnsgar4j170k5aj24weu5yegq8qp66cb-26aroyxll4zh9pl8cdwuo7vm4 192.168.99.101:2377

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

现在来创建两个工作节点 `worker1`， `worker2` 并加入集群

```
docker-machine create -d virtualbox worker1

eval $(docker-machine env worker1)

docker swarm join --token SWMTKN-1-
59qol34ustn06wtqs6bnsgar4j170k5aj24weu5yegq8qp66cb-
26aroyxll4zh9pl8cdwuo7vm4 192.168.99.101:2377
```

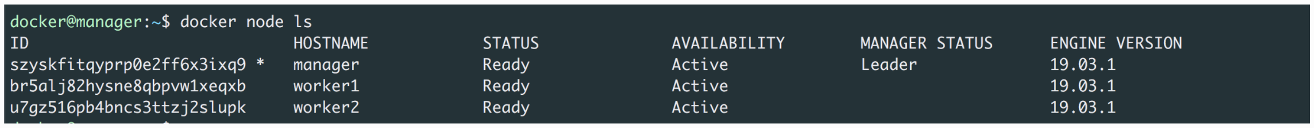


```
Xiaoy@MBP: /Users/Xiaoy/Repository/gdg_container_intro git:(master)
➜  docker swarm join --token SWMTKN-1-59qol34ustn06wtqs6bnsgar4j170k5aj24weu5yegq8qp66cb-26aroyxll4zh9pl8cdwuo7vm4 192.168.99.101:2377

This node joined a swarm as a worker.
```

同理 `worker2` 节点

进入 `manager` 节点执行

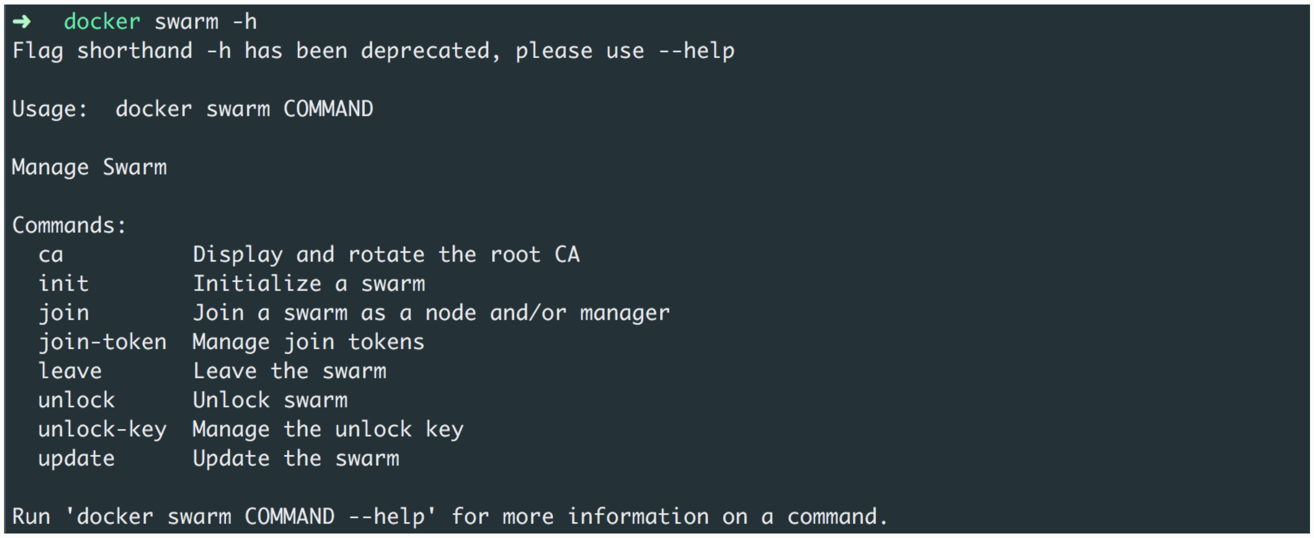
`docker node ls`



```
docker@manager:~$ docker node ls
ID                            HOSTNAME            STATUS              AVAILABILITY        MANAGER STATUS      ENGINE VERSION
szyskfitqyprp0e2ff6x3ixq9 *   manager             Ready               Active              Leader              19.03.1
br5alj82hysne8qbpvw1xeqxb     worker1             Ready               Active                                  19.03.1
u7gz516pb4bncs3ttzj2slupk     worker2             Ready               Active                                  19.03.1
```

由此，我们就得到了一个最小化的集群。

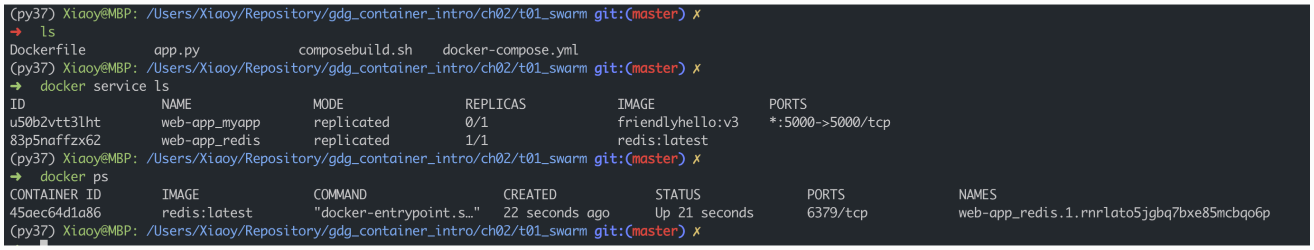
命令说明



```
➜  docker swarm -h
Flag shorthand -h has been deprecated, please use --help

Usage:  docker swarm COMMAND

Manage Swarm

Commands:
  ca          Display and rotate the root CA
  init        Initialize a swarm
  join        Join a swarm as a node and/or manager
  join-token  Manage join tokens
  leave       Leave the swarm
  unlock      Unlock swarm
  unlock-key  Manage the unlock key
  update      Update the swarm

Run 'docker swarm COMMAND --help' for more information on a command.
```

疑难解答

- 在 `docker stack deploy —c docker-compose.yml` 后，在 `docker ps` 中无法看到端口映射?



```
(py37) Xiaoy@MBP: /Users/Xiaoy/Repository/gdg_container_intro/ch02/t01_swarm git:(master) ✗
➜  ls
Dockerfile         app.py             composebuild.sh    docker-compose.yml
(py37) Xiaoy@MBP: /Users/Xiaoy/Repository/gdg_container_intro/ch02/t01_swarm git:(master) ✗
➜  docker service ls
ID                 NAME               MODE               REPLICAS           IMAGE              PORTS
u50b2vtt3lht       web-app_myapp      replicated         0/1                friendlyhello:v3   *:5000->5000/tcp
83p5naffzx62       web-app_redis      replicated         1/1                redis:latest
(py37) Xiaoy@MBP: /Users/Xiaoy/Repository/gdg_container_intro/ch02/t01_swarm git:(master) ✗
➜  docker ps
CONTAINER ID       IMAGE              COMMAND                  CREATED            STATUS             PORTS              NAMES
45aec64d1a86       redis:latest       "docker-entrypoint.s…"   22 seconds ago     Up 21 seconds      6379/tcp           web-app_redis.1.rnrlato5jgbq7bxe85mcbqo6p
(py37) Xiaoy@MBP: /Users/Xiaoy/Repository/gdg_container_intro/ch02/t01_swarm git:(master) ✗
```

关于docker swarm mode 部署后端口的问题，可以使用 `docker service ls` 来查看端口是否正确暴露，因为此时是通过service来暴露的，并不是直接在container上暴露，所以此时用 `docker ps` 是看不到的，但暴露的端口依旧可以访问，这样实现和k8s里的service实现是有些相似的。

- 执行 `docker-compose -f docker-compose.yml up -d` ,返回

```
Pulling myapp (friendlyhello:v2)...

ERROR: Get https://registry-1.docker.i... net/http: request canceled while waiting for connection (Client.Timeout exceeded while awaiting headers)
```

compose文件中如果已经build过，就用image直接指定这个image，注释掉build的指令。如果没有build过，就放开build指令，执行 `docker-compose` 的build它，当然也可以使用 `docker build` 来构建它。因为这一块在上一章节已经提到过，所以对于部分这次直接切入的同学可能会有疑惑。而到了docker stack时，已经不支持 `docker stack` 来build它了，需要统一使用docker build来构建镜像。