

狂神说Docker（下半场）

版权声明：关于狂神说Java

不为任何机构站台，编程是爱好，恭喜你发现宝藏男孩一枚~希望你们关注我是因为喜欢我！

所有的课程都是免费的，任何利用我课程收费的都是骗子，请大家注意！

B站唯一账号：狂神说Java 唯一公众号：狂神说

学习前，三连关注分享支持，是最基本的尊重，拒绝白嫖！



所有学习，官网！

Docker Compose

简介

Docker

DockerFile build run 手动操作，单个容器！

微服务。100个微服务！依赖关系。

Docker Compose 来轻松高效的管理容器。定义运行多个容器。

官方介绍

定义、运行多个容器。

YAML file 配置文件。

single command。命令有哪些？

Compose is a tool for defining and running multi-container Docker applications. With Compose, you use a YAML file to configure your application's services. Then, with a single command, you create and start all the services from your configuration. To learn more about all the features of Compose, see [the list of features](#).

所有的环境都可以使用 Compose。

Compose works in all environments: production, staging, development, testing, as well as CI workflows. You can learn more about each case in [Common Use Cases](#).

三步骤：

Using Compose is basically a three-step process:

1. Define your app's environment with a `Dockerfile` so it can be reproduced anywhere.
 - Dockerfile 保证我们的项目在任何地方可以运行。
2. Define the services that make up your app in `docker-compose.yml` so they can be run together in an isolated environment.
 - services 什么是服务。
 - docker-compose.yml 这个文件怎么写！
3. Run `docker-compose up` and Compose starts and runs your entire app.
 - 启动项目

作用：批量容器编排。

我自己理解

Compose 是Docker官方的开源项目。需要安装！

`Dockerfile` 让程序在任何地方运行。 web服务。 redis、mysql、nginx ... 多个容器。 run

Compose

```
1 version: '2.0'
2 services:
3   web:
4     build: .
5     ports:
6       - "5000:5000"
7     volumes:
8       - ./code
9       - logvolume01:/var/log
10    links:
11      - redis
12    redis:
13      image: redis
14  volumes:
15    logvolume01: {}
```

docker-compose up 100 个服务。

Compose：重要的概念。

- 服务services， 容器。应用。（web、redis、mysql....）
- 项目project。一组关联的容器。 博客。web、mysql。

安装

1、下载

```

1 sudo curl -L
  "https://github.com/docker/compose/releases/download/1.26.2/docker-
  compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
2
3 # 这个可能快点!
4 curl -L
  https://get.daocloud.io/docker/compose/releases/download/1.25.5/docker-
  compose-`uname -s`-`uname -m` > /usr/local/bin/docker-compose

```

点击左侧的箭头按钮。

```

[root@kuangshen home]# curl -L https://get.daocloud.io/docker/compose/releases/download/1.25.5/docker-compose-`uname -s`-`uname -m` > /usr/local/bin/docker-compose
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload   Total   Spent    Left   Speed
100 423    100 423    0     0   403      0  0:00:01  0:00:01 --:--:--  403
100 16.7M  100 16.7M    0     0  8188k      0  0:00:02  0:00:02 --:--:-- 21.0M
[root@kuangshen home]# ^C
[root@kuangshen home]# cd /usr/local/bin
[root@kuangshen bin]# ll
total 54928
-rwxr-xr-x 1 root root 17586312 Jul 24 20:39 docker-compose
-rwxr-xr-x 1 root root 4739112 May 13 21:59 redis-benchmark
-rwxr-xr-x 1 root root 9617624 May 13 21:59 redis-check-aof
-rwxr-xr-x 1 root root 9617624 May 13 21:59 redis-check-rdb
-rwxr-xr-x 1 root root 5049928 May 13 21:59 redis-cli
lrwxrwxrwx 1 root root 12 May 13 21:59 redis-sentinel -> redis-server
-rwxr-xr-x 1 root root 9617624 May 13 21:59 redis-server
[root@kuangshen bin]#

```

2、授权

```

1 sudo chmod +x /usr/local/bin/docker-compose

```

```

[root@kuangshen home]# ^C
[root@kuangshen home]# cd /usr/local/bin
[root@kuangshen bin]# ll
total 54928
-rwxr-xr-x 1 root root 17586312 Jul 24 20:39 docker-compose
-rwxr-xr-x 1 root root 4739112 May 13 21:59 redis-benchmark
-rwxr-xr-x 1 root root 9617624 May 13 21:59 redis-check-aof
-rwxr-xr-x 1 root root 9617624 May 13 21:59 redis-check-rdb
-rwxr-xr-x 1 root root 5049928 May 13 21:59 redis-cli
lrwxrwxrwx 1 root root 12 May 13 21:59 redis-sentinel -> redis-server
-rwxr-xr-x 1 root root 9617624 May 13 21:59 redis-server
[root@kuangshen bin]# sudo chmod +x docker-compose
[root@kuangshen bin]# docker-compose version
docker-compose version 1.25.5, build 8a1c60f6
docker-py version: 4.1.0
CPython version: 3.7.5
OpenSSL version: OpenSSL 1.1.0l 10 Sep 2019
[root@kuangshen bin]#

```

多看官网.....

体验

地址: <https://docs.docker.com/compose/gettingstarted/>

python 应用。计数器。redis!

- 1、应用 app.py
- 2、Dockerfile 应用打包为镜像
- 3、Docker-compose yaml文件 (定义整个服务, 需要的环境。web、redis) 完整的上线服务!
- 4、启动 compose 项目 (docker-compose up)

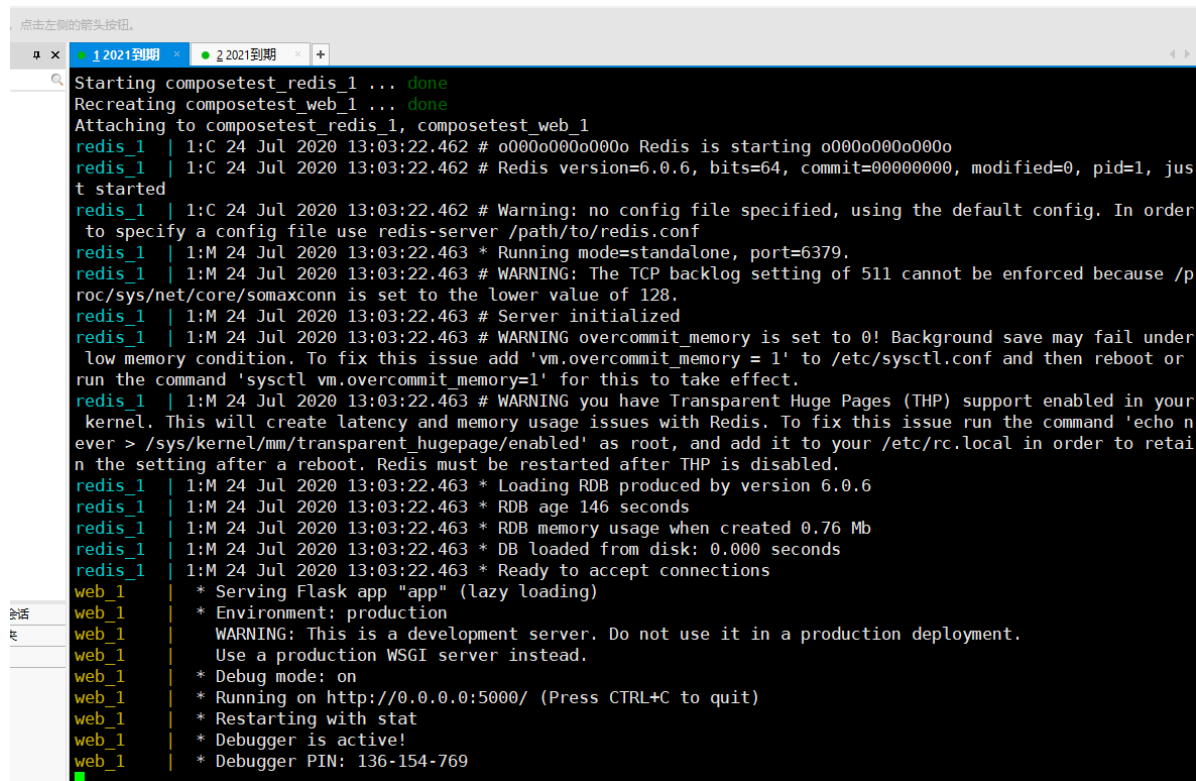
流程：

- 1、创建网络
- 2、执行 Docker-compose yaml
- 3、启动服务。

Docker-compose yaml

Creating composetest_web_1 ... done

Creating composetest_redis_1 ... done



```
Starting composetest_redis_1 ... done
Recreating composetest_web_1 ... done
Attaching to composetest_redis_1, composetest_web_1
redis_1 | 1:C 24 Jul 2020 13:03:22.462 # o000o000o000o Redis is starting o000o000o000o
redis_1 | 1:C 24 Jul 2020 13:03:22.462 # Redis version=6.0.6, bits=64, commit=00000000, modified=0, pid=1, just started
redis_1 | 1:C 24 Jul 2020 13:03:22.462 # Warning: no config file specified, using the default config. In order to specify a config file use redis-server /path/to/redis.conf
redis_1 | 1:M 24 Jul 2020 13:03:22.463 * Running mode=standalone, port=6379.
redis_1 | 1:M 24 Jul 2020 13:03:22.463 # WARNING: The TCP backlog setting of 511 cannot be enforced because /proc/sys/net/core/somaxconn is set to the lower value of 128.
redis_1 | 1:M 24 Jul 2020 13:03:22.463 # Server initialized
redis_1 | 1:M 24 Jul 2020 13:03:22.463 # WARNING overcommit_memory is set to 0! Background save may fail under low memory condition. To fix this issue add 'vm.overcommit_memory = 1' to /etc/sysctl.conf and then reboot or run the command 'sysctl vm.overcommit_memory=1' for this to take effect.
redis_1 | 1:M 24 Jul 2020 13:03:22.463 # WARNING you have Transparent Huge Pages (THP) support enabled in your kernel. This will create latency and memory usage issues with Redis. To fix this issue run the command 'echo never > /sys/kernel/mm/transparent_hugepage/enabled' as root, and add it to your /etc/rc.local in order to retain the setting after a reboot. Redis must be restarted after THP is disabled.
redis_1 | 1:M 24 Jul 2020 13:03:22.463 * Loading RDB produced by version 6.0.6
redis_1 | 1:M 24 Jul 2020 13:03:22.463 * RDB age 146 seconds
redis_1 | 1:M 24 Jul 2020 13:03:22.463 * RDB memory usage when created 0.76 Mb
redis_1 | 1:M 24 Jul 2020 13:03:22.463 * DB loaded from disk: 0.000 seconds
redis_1 | 1:M 24 Jul 2020 13:03:22.463 * Ready to accept connections
web_1 | * Serving Flask app "app" (lazy loading)
web_1 | * Environment: production
web_1 | WARNING: This is a development server. Do not use it in a production deployment.
web_1 | Use a production WSGI server instead.
web_1 | * Debug mode: on
web_1 | * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
web_1 | * Restarting with stat
web_1 | * Debugger is active!
web_1 | * Debugger PIN: 136-154-769
```

- 1、文件名 composetest
- 2、服务

```
1 version: '3'
2 services:
3   web:
4     build: .
5     ports:
6       - "5000:5000"
7   redis:
8     image: "redis:alpine"
```

自动的默认规则？

```

Nothing to do
[root@kuangshen ~]# clear
[root@kuangshen ~]# docker ps
CONTAINER ID   NAMES      IMAGE          COMMAND                  CREATED        STATUS        PORTS
1ac14aab1a54   redis:alpine  "docker-entrypoint.s..."  3 minutes ago  Up 15 seconds  6379/t
cp            composetest_redis_1
[root@kuangshen ~]# docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS
4851571e3436   composetest_web  "python app.py"          31 seconds ago  Up 30 seconds  0.0.0.
0:5000->5000/tcp composetest_web_1
1ac14aab1a54   redis:alpine  "docker-entrypoint.s..."  6 minutes ago  Up 30 seconds  6379/t
cp            composetest_redis_1
[root@kuangshen ~]# curl localhost:5000
Hello World! I have been seen 1 times.
[root@kuangshen ~]# curl localhost:5000
Hello World! I have been seen 2 times.
[root@kuangshen ~]# curl localhost:5000
Hello World! I have been seen 3 times.
[root@kuangshen ~]# curl localhost:5000
Hello World! I have been seen 4 times.
[root@kuangshen ~]# curl localhost:5000
Hello World! I have been seen 5 times.
[root@kuangshen ~]# curl localhost:5000
Hello World! I have been seen 6 times.
[root@kuangshen ~]#

```

服务启动成功!

服务正常

docker images

```

[root@kuangshen ~]# docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
composetest_web  latest    dcd9421baf78   8 minutes ago  81.3MB
<none>         <none>    2e50cd447752   15 minutes ago  73.1MB
redis          alpine    c7b388ce3d39   2 days ago    32.1MB
python         3.6-alpine d6b500d78779   3 weeks ago    70.1MB
python         3.7-alpine 6ca3e0b1ab69   3 weeks ago    73.1MB
hello-world    latest    bf756fb1ae65   6 months ago   13.3kB

```

- 1 [root@kuangshen ~]# docker service ls
- 2 Error response from daemon: This node is not a swarm manager. Use "docker swarm init" or "docker swarm join" to connect this node to swarm and try again.

默认的服务名 文件名_服务名_num

多个服务器。集群。A B _num 副本数量

服务redis服务 => 4个副本。

集群状态。服务都不可能只有一个运行实例。弹性、10 HA 高并发。

kubectl service 负载均衡。

3、网络规则

```

docker: 'network' is not a docker command.
See 'docker --help'
[root@kuangshen ~]# docker network ls
NETWORK ID     NAME                DRIVER              SCOPE
7f2a3e7b5dab   bridge              bridge               local
caff1f59b938   composetest_default bridge               local
c2965e19a00f   host                host                 local
ffffebb0a079b   none                null                 local

```

10个服务 => 项目 (项目中的内容都在同个网络下。域名访问)

```

    "Network": "",
  },
  "ConfigOnly": false,
  "Containers": {
    "1ac14aab1a545e735cf515f1aa5b0ac9aa8e2e66429e8e322c62ed87b0a370b1": {
      "Name": "composetest_redis_1",
      "EndpointID": "8df4a09d9f71ede71895c6a56daa6bfea071d35570df834a8ede386836365ce9",
      "MacAddress": "02:42:ac:1b:00:02",
      "IPv4Address": "172.27.0.2/16",
      "IPv6Address": ""
    },
    "4851571e3436171794527e16793986cb848418186051cbb8daa78e973ce872b": {
      "Name": "composetest_web_1",
      "EndpointID": "c3d027525e9947f83ed202de32a0bde7deb8278a99596fc7ba935438158c97fd",
      "MacAddress": "02:42:ac:1b:00:03",
      "IPv4Address": "172.27.0.3/16",
      "IPv6Address": ""
    }
  },
  "Options": {},

```

如果在同一个网络下，我们可以直接通过域名访问。

HA!

停止：docker-compose down ctrl+c

```

web_1      | 172.27.0.1 - - [24/Jul/2020 13:04:52] "GET / HTTP/1.1" 200 151
web_1      | 172.27.0.1 - - [24/Jul/2020 13:04:53] "GET / HTTP/1.1" 200 151
web_1      | 172.27.0.1 - - [24/Jul/2020 13:04:54] "GET / HTTP/1.1" 200 151
^CGracefully stopping... (press Ctrl+C again to force)
Stopping composetest_web_1 ... done
Stopping composetest_redis_1 ... done
[root@kuangshen composetest]#

```

docker-compose

以前都是单个 docker run 启动容器。

docker-compose。通过 docker-compose 编写 yaml 配置文件、可以通过 compose 一键启动所有服务，停止。！

Docker小结：

- 1、Docker 镜像。run => 容器
- 2、DockerFile 构建镜像（服务打包）
- 3、docker-compose 启动项目（编排、多个微服务/环境）
- 4、Docker 网络

yaml 规则

docker-compose.yaml 核心。！

<https://docs.docker.com/compose/compose-file/#compose-file-structure-and-examples>

```

1 # 3层！
2

```

```

3 version: ' ' # 版本
4 services: # 服务
5     服务1: web
6         # 服务配置
7         images
8         build
9         network
10        .....
11    服务2: redis
12        ....
13    服务3: redis
14 # 其他配置 网络/卷、全局规则
15 volumes:
16 networks:
17 configs:

```

depends_on

Express dependency between services. Service dependencies cause the following behaviors:

- `docker-compose up` starts services in dependency order. In the following example, `db` and `redis` are started before `web`.
- `docker-compose up SERVICE` automatically includes `SERVICE`'s dependencies. In the example below, `docker-compose up web` also creates and starts `db` and `redis`.
- `docker-compose stop` stops services in dependency order. In the following example, `web` is stopped before `db` and `redis`.

Simple example:

启动顺序
web -> redis

```

version: "3.8"
services:
  web:
    build: .
    depends_on:
      - db
      - redis
  redis:
    image: redis
  db:
    image: postgres

```

学习，要掌握规律！

只要多写，多看。compose.yaml 配置。！

1、官网文档

<https://docs.docker.com/compose/compose-file/#specifying-durations>

2、开源项目 compose.yaml

redis、mysql、mq！

开源项目

博客

下载程序、安装数据库、配置.....

compose 应用。=> 一键启动!

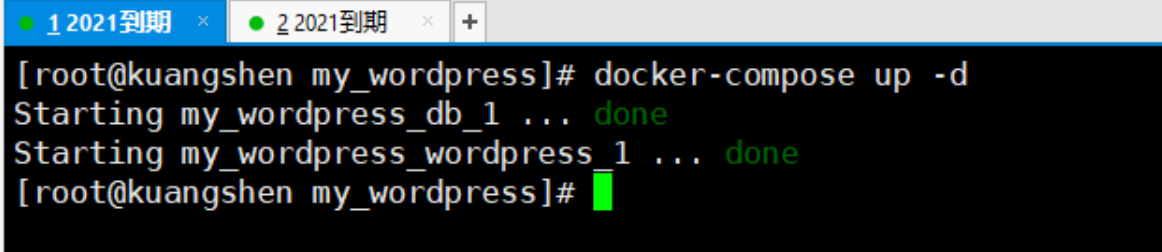
- 1、下载项目 (docker-compose.yaml)
- 2、如果需要文件。Dockerfile
- 3、文件准备齐全 (直接一键启动项目!)

前台启动

docker -d

docker-compose up -d

侧的前头按钮。



```
[root@kuangshen my_wordpress]# docker-compose up -d
Starting my_wordpress_db_1 ... done
Starting my_wordpress_wordpress_1 ... done
[root@kuangshen my_wordpress]#
```

一切都很简单!

实战

- 1、编写项目微服务
- 2、dockerfile 构建镜像
- 3、docker-compose.yaml 编排项目
- 4、丢到服务器 docker-compose up

小结:

未来项目只要有 docker-compose 文件。按照这个规则, 启动编排容器。!

公司: docker-compose。直接启动。

网上开源项目: docker-compose 一键搞定。

假设项目要重新部署打包

```
1 | docker-compose up --build # 重新构建!
```


总结:

工程、服务、容器

项目 compose: 三层

- 工程 Porject
- 服务 服务
- 容器 运行实例! docker k8s 容器.

Docker Compose 搞定!

Docker Swarm

集群

购买服务器

4台服务器 2G!

阿里云

搜索文档、控制台、API、解决方案和资源

费用 工单 备案 企业 支持 官网 购物车 帮助

云服务器 ECS

我的资源

中国 (香港)

云服务器	即将过期	已过期	运行中	已停止	近期创建
1	0 续费	0 续费	1	0	0

实例ID	IP地址	付费方式	实例状态
i-j6c1wvscj5y05fyq1yiq	8.210.51.92(公网)	包年包月	运行中
launch-advisor-20200620	172.31.117.17(私网)	2025年6月21日 00:00:00到期	

导出全部地域数据

创建实例

资源概览

常用功能

续费管理 用户设置 权益配额

帮助与文档 产品价格 产品详情

您的帐号价值连城

强烈建议您不使用主账号执行日常任务，免费开通RAM子账号。

关闭 查看详情

重要提醒

填写阿里云易用性问卷，说出您的心声，有机会收获ET公仔礼盒

立即前往 >>

近期重要功能

- 【新功能】阿里云块存储服务全面提供...
- 2020年6月16日
- 【新功能】效率神器--发送远程命令：通...

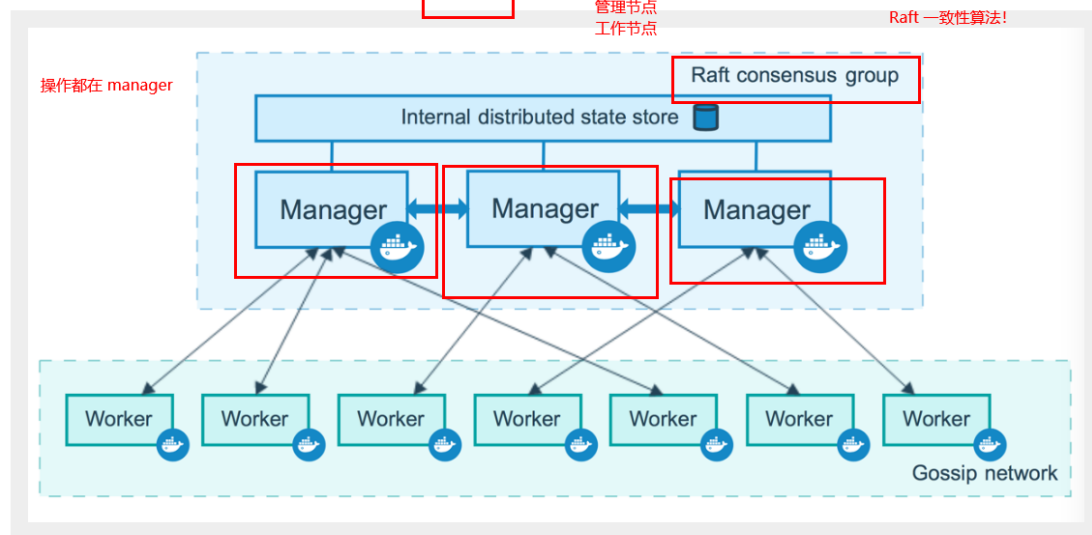
mode.

There are two types of nodes: **managers** and **workers**.

节点

管理节点
工作节点

Raft 一致性算法!



If you haven't already, read through the [swarm mode overview](#) and [key concepts](#).

搭建集群

左侧的箭头按钮。

```
[root@iZ2vcg4bxglqi65j2wdgm3Z ~]# docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
6ddf7e46cff4        bridge             bridge              local
cd4a3e83d894        host               host                local
820d7c2ce5a3        none               null                local
[root@iZ2vcg4bxglqi65j2wdgm3Z ~]#

820d7c2ce5a3        none               null                local
[root@iZ2vcg4bxglqi65j2wdgm3Z ~]# docker swarm --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.
[root@iZ2vcg4bxglqi65j2wdgm3Z ~]#
```

```
Usage: docker swarm init [OPTIONS]

Initialize a swarm

Options:
  --advertise-addr string      Advertised address (format: <ip|interface>[:port])
  --autolock                  Enable manager autolocking (requiring an unlock
                              key to start a stopped manager)
  --availability string       Availability of the node
                              ("active"|"pause"|"drain") (default "active")
  --cert-expiry duration      Validity period for node certificates
                              (ns|us|ms|s|m|h) (default 2160h0m0s)
  --data-path-addr string     Address or interface to use for data path
                              traffic (format: <ip|interface>)
  --data-path-port uint32     Port number to use for data path traffic (1024
                              - 49151). If no value is set or is set to 0,
                              the default port (4789) is used.
  --default-addr-pool ipNetSlice default address pool in CIDR format (default [])
  --default-addr-pool-mask-length uint32 default address pool subnet mask length (default 2
  --dispatcher-heartbeat duration Dispatcher heartbeat period (ns|us|ms|s|m|h)
                              (default 5s)
  --external-ca external-ca  Specifications of one or more certificate
                              signing endpoints
  --force-new-cluster        Force create a new cluster from current state
  --listen-addr node-addr    Listen address (format: <ip|interface>[:port])
                              (default 0.0.0.0:2377)
  --max-snapshots uint       Number of additional Raft snapshots to retain
  --snapshot-interval uint   Number of log entries between Raft snapshots
                              (default 10000)
  --task-history-limit int    Task history retention limit (default 5)
```

私网、公网！

172.24.82.149 用自己的！

```
[root@iZ2vcg4bxglqi65j2wdgm3Z ~]# docker swarm init --advertise-addr 172.24.82.149
Swarm initialized: current node (t3jgzjpkhks7jj788rjtxuu3y) is now a manager.

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

    docker swarm join --token SWMTKN-1-4a1qo02iwnaoqbt9vuagbucslkk9p451u3apf7ed7ybm50a3x-dkmm5q0oocys8lc91222xerwx 172.24.82.149:2377

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

初始化节点 `docker swarm init`

docker swarm join 加入一个节点！

```
1 # 获取令牌
2 docker swarm join-token manager
3 docker swarm join-token worker
```

```
[root@iZ2vcg4bxglqi65j2wdgm0Z ~]# docker swarm join --token SWMTKN-1-4a1qo02iwnaoqbt9vuagbucslkk9p451u3apf7ed7ybm50a3x-dkmm5q0oocys8lc91222xerwx 172.24.82.149:2377
This node joined a swarm as a worker.
[root@iZ2vcg4bxglqi65j2wdgm0Z ~]#
```

把后面的节点都搭建进去！

```
sh/47.108.197.19:22
添加当前会话，点击左侧的箭头按钮。
1 docker-1 2 docker-2 3 docker-3 4 docker-4
[root@iZ2vcg4bxglqi65j2wdgm3Z ~]# docker node ls
ID                                     HOSTNAME                                STATUS    AVAILABILITY    MANAGER STATUS    E
ENGINE VERSION
6oddah9pb1wqt5uv4hs7yokxx            iZ2vcg4bxglqi65j2wdgm0Z               Ready     Active           1
9.03.12
qzc0k9nuv4yqibtjvwgj8zk1t           iZ2vcg4bxglqi65j2wdgm1Z               Ready     Active           1
9.03.12
ayw6tzy97c2kk9non04jggj7p           iZ2vcg4bxglqi65j2wdgm2Z               Ready     Active           Reachable        1
9.03.12
t3jgzjpkhks7jj788rjtxuu3y *         iZ2vcg4bxglqi65j2wdgm3Z               Ready     Active           Leader            1
9.03.12
[root@iZ2vcg4bxglqi65j2wdgm3Z ~]#
```

100台！

- 1、生成主节点 init
- 2、加入（管理者、worker）

目标：双主双从！

Raft协议

双主双从：假设一个节点挂了！其他节点是否可以用！

Raft协议：保证大多数节点存活才可以用。只要>1，集群至少大于3台！

实验：

- 1、将docker1机器停止。宕机！双主，另外一个主节点也不能使用了！

```
[root@iZ2vcg4bxglqi65j2wdgm2Z ~]# docker node ls
Error response from daemon: rpc error: code = Unknown desc = The swarm does not have a leader. It's possible
that too few managers are online. Make sure more than half of the managers are online.
[root@iZ2vcg4bxglqi65j2wdgm2Z ~]#
```

- 2、可以将其他节点离开

```
1 docker-1 2 docker-2 3 docker-3 4 docker-4
[root@iZ2vcg4bxglqi65j2wdgm2Z ~]# docker node ls
ID                                     HOSTNAME                                STATUS    AVAILABILITY    MANAGER STATUS
US ENGINE VERSION
6oddah9pb1wqt5uv4hs7yokxx            iZ2vcg4bxglqi65j2wdgm0Z               Ready     Active
19.03.12
qzc0k9nuv4yqibtjvwgj8zk1t           iZ2vcg4bxglqi65j2wdgm1Z               Down      Active
19.03.12
ayw6tzy97c2kk9non04jggj7p *         iZ2vcg4bxglqi65j2wdgm2Z               Ready     Active           Leader
19.03.12
t3jgzjpkhks7jj788rjtxuu3y           iZ2vcg4bxglqi65j2wdgm3Z               Ready     Active           Reachable
19.03.12
[root@iZ2vcg4bxglqi65j2wdgm2Z ~]#
```

- 3、work就是工作的、管理节点操作！3台机器设置为了管理节点。

十分简单：集群，可用！3个主节点。>1台管理节点存活！

Raft协议：保证大多数节点存活，才可以使用，高可用！

体会

弹性、扩缩容！集群！

以后告别 docker run！

docker-compose up！ 启动一个项目。单机！

集群：swarm **docker service**

容器 => 服务！

容器 => 服务！ => 副本！

redis 服务 => 10个副本！（同时开启10个redis容器）

体验：创建服务、动态扩展服务、动态更新服务。

```
2
左侧的箭头按钮。
x 1 docker-1 x 2 docker-2 x 3 docker-3 x 4 docker-4 x +
[root@iZ2vcg4bxglqi65j2wdgm3Z ~]# docker service --help

Usage:  docker service COMMAND

Manage services

Commands:
  create      Create a new service
  inspect     Display detailed information on one or more services
  logs        Fetch the logs of a service or task
  ls          List services
  ps          List the tasks of one or more services
  rm          Remove one or more services
  rollback    Revert changes to a service's configuration
  scale       Scale one or multiple replicated services
  update      Update a service

Run 'docker service COMMAND --help' for more information on a command.
[root@iZ2vcg4bxglqi65j2wdgm3Z ~]#
```

灰度发布：金丝雀发布！

```
左侧的箭头按钮。
x 1 docker-1 x 2 docker-2 x 3 docker-3 x 4 docker-4 x +
[root@iZ2vcg4bxglqi65j2wdgm3Z ~]# docker service create -p 8888:80 --name my-nginx nginx
34wxynfryorresp2fng0ldus3
overall progress: 1 out of 1 tasks
1/1: running [=====>]
verify: Service converged
[root@iZ2vcg4bxglqi65j2wdgm3Z ~]#
```

- 1 docker run 容器启动！不具有扩缩容器
- 2 docker service 服务！ 具有扩缩容器，滚动更新！

查看服务 REPLICAS

```
[root@iZ2vcg4bxglqi65j2wdgm3Z ~]# docker service ps my-nginx
ID            NAME          IMAGE          NODE                DESIRED STATE  CURRENT STA
TE            ERROR
012hvlvq5bre  my-nginx.1    nginx:latest   iZ2vcg4bxglqi65j2wdgm1Z  Running        Running abo
ut a minute ago
[root@iZ2vcg4bxglqi65j2wdgm3Z ~]# docker service ls
ID            NAME          MODE          REPLICAS          IMAGE          PORTS
34wxynfryorr my-nginx      replicated    1/1                nginx:latest   *:8888->80/tcp
[root@iZ2vcg4bxglqi65j2wdgm3Z ~]#
```

动态扩缩容

```
[root@iZ2vcg4bxglqi65j2wdgm3Z ~]# docker service update --relipcas 3 my-nginx
unknown flag: --relipcas
See 'docker service update --help'.
[root@iZ2vcg4bxglqi65j2wdgm3Z ~]# docker service update --replicas 3 my-nginx
my-nginx
overall progress: 3 out of 3 tasks
1/3: running [=====>]
2/3: running [=====>]
3/3: running [=====>]
verify: Service converged
[root@iZ2vcg4bxglqi65j2wdgm3Z ~]# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS
NAMES
[root@iZ2vcg4bxglqi65j2wdgm3Z ~]# docker service update --replicas 10 my-nginx
```

服务，集群中任意的节点都可以访问。服务可以有多个副本动态扩缩容实现高可用！

弹性、扩缩容！

10台！ 10000台！ 卖给别人！ 虚拟化！

服务的高可用，任何企业，云！

```
[root@iZ2vcg4bxglqi65j2wdgm3Z ~]# docker service ls
ID                NAME                MODE                REPLICAS            IMAGE                PORTS
34wxynfryorr     my-nginx            replicated          1/1                 nginx:latest
[root@iZ2vcg4bxglqi65j2wdgm3Z ~]# docker service scale my-nginx=5
my-nginx 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@iZ2vcg4bxglqi65j2wdgm3Z ~]# docker service scale my-nginx=2
my-nginx scaled to 2
overall progress: 2 out of 2 tasks
1/2: running [=====>]
2/2: running [=====>]
verify: Waiting 1 seconds to verify that tasks are stable...
```

移除！

docker swarm 其实并不难

只要会搭建集群、会启动服务、动态管理容器就可以了！

概念总结

swarm

集群的管理和编号。 docker可以初始化一个 swarm 集群，其他节点可以加入。（管理、工作者）

Node

就是一个docker节点。多个节点就组成了一个网络集群。（管理、工作者）

Service

任务，可以在管理节点或者工作节点来运行。核心。！ 用户访问！

Task

容器内的命令，细节任务！

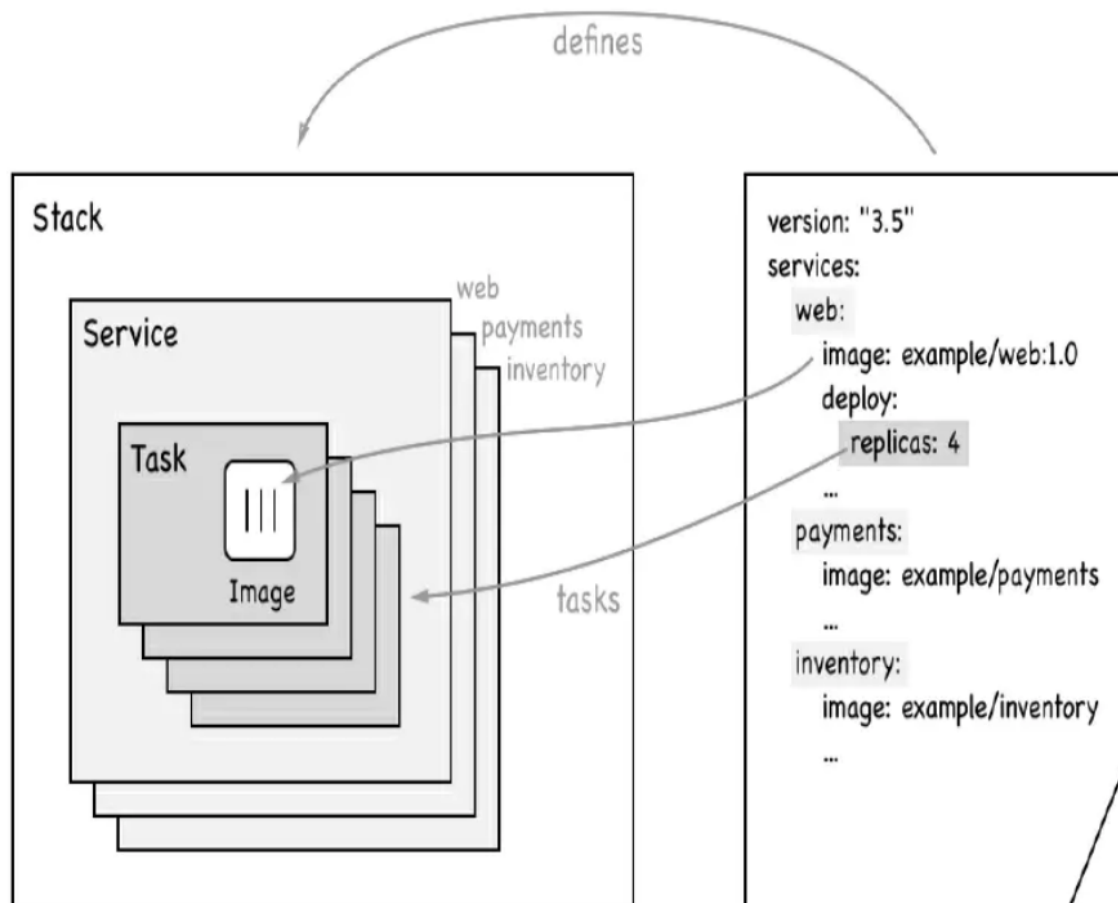
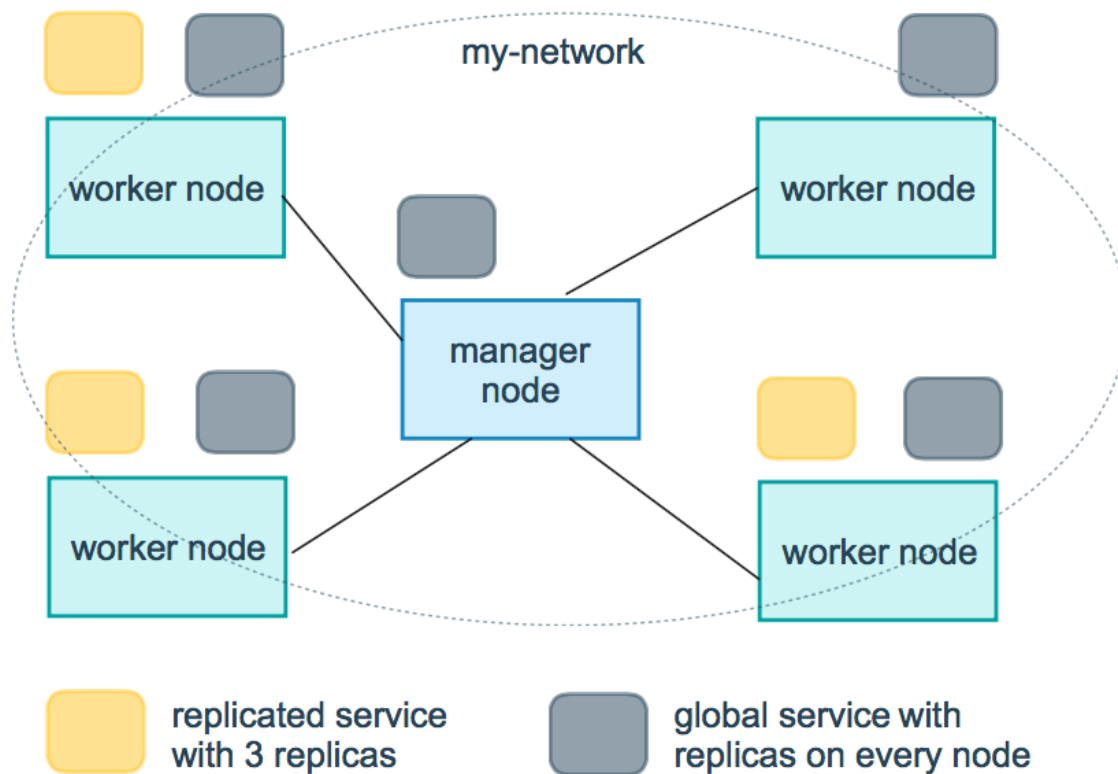


Diagram showing the relationship between stack, services and tasks

逻辑是不变的。

命令 -> 管理 -> api -> 调度 -> 工作节点 (创建Task容器维护创建!)

服务副本与全局服务



调整service以什么方式运行

```

1  --mode string
2  Service mode (replicated or global) (default "replicated")
3
4  docker service create --mode replicated --name mytom tomcat:7 默认的
5
6  docker service create --mode global --name haha alpine ping baidu.com
7  #场景? 日志收集
8  每一个节点有自己的日志收集器, 过滤。把所有日志最终再传给日志中心
9  服务监控, 状态性能。

```

拓展: 网络模式: "PublishMode": "ingress"

Swarm:

Overlay:

ingress : 特殊的 Overlay 网络! 负载均衡的功能! IPVS VIP!

虽然docker在4台机器上, 实际网络是同一个! ingress 网络, 是一个特殊的 Overlay 网络

```

        IPv4Address : 10.0.0.2/24 ,
        "IPv6Address": ""
    },
    "Options": {
        "com.docker.network.driver.overlay.vxlanid_list": "4096"
    },
    "Labels": {},
    "Peers": [
        {
            "Name": "41268e507c00",
            "IP": "172.24.82.150"
        },
        {
            "Name": "cf680f0f8f9c",
            "IP": "172.24.82.148"
        },
        {
            "Name": "995045ae0a90",
            "IP": "172.24.82.149"
        },
        {
            "Name": "aa096e80e487",
            "IP": "172.24.82.147"
        }
    ]
}

```

整体!

k8s!

Docker Stack

```

[root@iZ2vcg4bxglqi65j2wdgm3Z ~]# docker stack

Usage:  docker stack [OPTIONS] COMMAND

Manage Docker stacks

Options:
  --orchestrator string  Orchestrator to use (swarm|kubernetes|all)

Commands:
  deploy      Deploy a new stack or update an existing stack
  ls          List stacks
  ps          List the tasks in the stack
  rm          Remove one or more stacks
  services    List the services in the stack

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

```

docker-compose 单机部署项目!

Docker Stack部署, 集群部署!

```

1  # 单机
2  docker-compose up -d wordpress.yml
3  # 集群
4  docker stack deploy wordpress.yml
5

```

```

6 # docker-compose 文件
7 version: '3.4'
8 services:
9   mongo:
10     image: mongo
11     restart: always
12     networks:
13       - mongo_network
14     deploy:
15       restart_policy:
16         condition: on-failure
17       replicas: 2
18   mongo-express:
19     image: mongo-express
20     restart: always
21     networks:
22       - mongo_network
23     ports:
24       - target: 8081
25         published: 80
26         protocol: tcp
27         mode: ingress
28     environment:
29       ME_CONFIG_MONGODB_SERVER: mongo
30       ME_CONFIG_MONGODB_PORT: 27017
31     deploy:
32       restart_policy:
33         condition: on-failure
34       replicas: 1
35   networks:
36     mongo_network:
37       external: true

```

Docker Secret

安全! 配置密码, 证书!

```

services    List the services in the stack

Run 'docker stack COMMAND --help' for more information on a command.
[root@iZ2vcg4bxglqi65j2wdgm3Z ~]# docker secret

Usage:  docker secret COMMAND

Manage Docker secrets

Commands:
  create    Create a secret from a file or STDIN as content
  inspect   Display detailed information on one or more secrets
  ls        List secrets
  rm        Remove one or more secrets

Run 'docker secret COMMAND --help' for more information on a command.
[root@iZ2vcg4bxglqi65j2wdgm3Z ~]#

```

Docker Config

配置

```
[root@iZ2vcg4bxglqi65j2wdgm3Z ~]# docker config

Usage:  docker config COMMAND

Manage Docker configs

Commands:
  create   Create a config from a file or STDIN
  inspect  Display detailed information on one or more configs
  ls       List configs
  rm       Remove one or more configs

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

Docker下半场!

Docker Compose Swarm!

了解，学习方式：

网上找案例跑起来试试！查看命令帮助文档 --help， 官网！

拓展到K8S

云原生时代

Go语言！必须掌握！ Java、Go！

并发语言！

B语言 C 语言的创始人。Unix创始人 V8

Go（又称 **Golang**）是 [Google](#) 的 Robert Griesemer，Rob Pike 及 Ken Thompson 开发的一种**静态强类型、编译型语言**。Go 语言语法与 [C](#) 相近，但功能上有：内存安全，[GC](#)（垃圾回收），[结构形态](#)及 CSP-style [并发计算](#)。

go 指针！

罗伯特·格瑞史莫（Robert Griesemer），罗勃·派克（Rob Pike）及[肯·汤普逊](#)（Ken Thompson）于2007年9月开始设计Go，稍后Ian Lance Taylor、Russ Cox加入项目。Go是基于[Inferno](#)操作系统所开发的。Go于2009年11月正式宣布推出，成为开放源代码项目，并在[Linux](#)及[Mac OS X](#)平台上进行了实现，后来追加了Windows系统下的实现。在2016年，Go被软件评价公司TIOBE 选为“TIOBE 2016 年最佳语言”。目前，Go每半年发布一个二级版本（即从a.x升级到a.y）

java => Go

zijie => Go

