链滴

Docker-Compose 基础与实战，看这一篇就够了

作者：jianzh5
原文链接：https://ld246.com/article/1574852028068
来源网站：链滴

what & why

Compose 项目是 Docker 官方的开源项目，负责实现对 Docker 容器集群的快速编排。使用前面介的Dockerfile我们很容易定义一个单独的应用容器。然而在日常开发工作中，经常会碰到需要多个容相互配合来完成某项任务的情况。例如要实现一个 Web 项目，除了 Web 服务容器本身，往往还需再加上后端的数据库服务容器；再比如在分布式应用一般包含若干个服务，每个服务一般都会部署多实例。如果每个服务都要手动启停，那么效率之低、维护量之大可想而知。这时候就需要一个工具能管理一组相关联的的应用容器，这就是Docker Compose。

Compose有2个重要的概念

- **项目（Project）**：由一组关联的应用容器组成的一个完整业务单元，在 docker-compose.yml 文中定义。
- **服务（Service）**：一个应用的容器，实际上可以包括若干运行相同镜像的容器实例。

docker compose 安装与卸载

安装

二进制包在线安装

```
curl -L https://github.com/docker/compose/releases/download/1.25.0/docker-compose-`uname -s`-`uname -m` -o /usr/local/bin/docker-compose

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

这个方法现在基本行不通，下载太慢了，不推荐使用。

二进制包离线安装

从https://github.com/docker/compose/releases/download/1.25.0/docker-compose-Linux-x86_64下载对应的安装包，比如我下载了Linux-x86_64的。

将下载好的安装包剪切到/usr/local/bin/docker-compose目录下

mv /app/download/docker-compose-Linux-x86_64 /usr/local/bin/docker-compose

添加执行权限

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

pip安装

- 先安装好pip工具

```
#安装依赖
yum -y install epel-release
#安装PIP
yum -y install python-pip
#升级PIP
pip install --upgrade pip
```

- 验证pip 版本

```
[root@tymonitor bin]# pip --version
pip 8.1.2 from /usr/lib/python2.7/site-packages (python 2.7)
```

- 安装docker compose

pip install -U docker-compose==1.25.0

如果安装过程中出现如下所示的错误，请先执行yum install python-devel后再执行安装命令。

安装成功

- 验证docker compose版本

```
[root@tymonitor bin]# docker-compose --version
docker-compose version 1.25.0, build b42d419
```

安装补全插件

curl -L https://raw.githubusercontent.com/docker/compose/1.25.0/contrib/completion/bash/docker-compose > /etc/bash_completion.d/docker-compose

卸载

二进制卸载

rm /usr/local/bin/docker-compose

pip卸载

pip uninstall docker-compose

docker compose 重要命令

命令选项

- -f, --file FILE 指定使用的 Compose 模板文件，默认为 docker-compose.yml，可以多次指定。
- -p, --project-name NAME 指定项目名称，默认将使用所在目录名称作为项目名。
- --x-networking 使用 Docker 的可拔插网络后端特性
- --x-network-driver DRIVER 指定网络后端的驱动，默认为 bridge
- --verbose 输出更多调试信息。
- -v, --version 打印版本并退出。

常用&重要命令

- config

验证 Compose 文件格式是否正确，若正确则显示配置，若格式错误显示错误原因。

如：docker-compose -f skywalking.yml config

此命令不会执行真正的操作，而是显示 docker-compose 程序解析到的配置文件内容：

```
[root@tymonitor skywalking]# docker-compose -f skywalking.yml config
services:
  elasticsearch:
    container_name: elasticsearch
    image: elasticsearch:6.8.5
    ports:
    - 9200:9200/tcp
    - 9300:9300/tcp
    restart: always
    volumes:
    - /app/skywalking/elasticsearch/data:/usr/share/elasticsearch/data:rw
    - /app/skywalking/elasticsearch/conf/elasticsearch.yml:/usr/share/elasticsearch/config/elasticsearch.yml:rw
    - /app/skywalking/elasticsearch/logs:/user/share/elasticsearch/logs:rw
  es-head:
    container_name: es-head
    image: tobias74/elasticsearch-head:6
    ports:
    - 9100:9100/tcp
    restart: always
version: '3.0'
```

- images

列出 Compose 文件中包含的镜像。如docker-compose -f skywalking.yml images

```
[root@tymonitor skywalking]# docker-compose -f skywalking.yml images
  Container               Repository            Tag       Image Id       Size
------------------------------------------------------------------------------
elasticsearch     elasticsearch                 6.8.5    4f90d9a6692f   865.8 MB
es-head           tobias74/elasticsearch-head   6        301c944ca40a   812.4 MB
skywalking-oap    apache/skywalking-oap-server  6.5.0    cd2ac06bceec   190.1 MB
skywalking-ui     apache/skywalking-ui          6.5.0    5d0366fabea8   123 MB
```

- ps

列出项目中目前的所有容器。如docker-compose -f skywalking.yml ps

```
[root@tymonitor skywalking]# docker-compose -f skywalking.yml ps
     Name                   Command               State                          Ports
-----------------------------------------------------------------------------------------------------------------
elasticsearch    /usr/local/bin/docker-entr ...   Up      0.0.0.0:9200->9200/tcp, 0.0.0.0:9300->9300/tcp
es-head          /bin/sh -c grunt server          Up      0.0.0.0:9100->9100/tcp
skywalking-oap   bash docker-entrypoint.sh        Up      0.0.0.0:11800->11800/tcp, 1234/tcp, 0.0.0.0:12800->12800/tcp
skywalking-ui    bash docker-entrypoint.sh        Up      0.0.0.0:8080->8080/tcp
```

- build

构建（重新构建）项目中的服务容器。如：docker-compose -f skywalking.yml build，一般搭配自义镜像，比如编写的Dockfile，功能类似于docker build .

- up

该命令十分强大（重点掌握），它将尝试自动完成包括构建镜像，（重新）创建服务，启动服务，并联服务相关容器的一系列操作。如docker-compose -f skywalking.yml up。默认情况，docker-compose up 启动的容器都在前台，控制台将会同时打印所有容器的输出信息，可以很方便进行调试。

```
[root@tymonitor skywalking]# docker-compose -f skywalking.yml up
Creating network "skywalking_default" with the default driver
Pulling es-head (tobias74/elasticsearch-head:6)...
6: Pulling from tobias74/elasticsearch-head
d660b1f15b9b: Pull complete
d660b1f15b9b: Downloading [============>                                      ]  13.97MB/54.25MB
46dde23c37b3: Downloading [==========================================>        ]  14.76MB/17.54MB
6ebaeb074589: Downloading [=================>                                 ]  14.97MB/43.3MB
e7428f935583: Waiting
eda527043444: Waiting
f3088daa8887: Waiting
80e2dc61b67d: Waiting
26d4b3db26a2: Waiting
2850af7cc432: Waiting
89b8c1907caa: Waiting
28f5c6564baf: Waiting
a1b3170171aa: Waiting
3a3364bb5b79: Waiting
```

如果使用docker-compose up -d将会在后台启动并运行所有的容器。一般推荐生产环境下使用该选。

默认情况，如果服务容器已经存在，docker-compose up 将会尝试停止容器，然后重新创建（保持用 volumes-from 挂载的卷），以保证新启动的服务匹配 docker-compose.yml 文件的最新内容。果用户不希望容器被停止并重新创建，可以使用 docker-compose up --no-recreate。这样将只会动处于停止状态的容器，而忽略已经运行的服务。如果用户只想重新部署某个服务，可以使用 docker compose up --no-deps -d <SERVICE_NAME> 来重新创建服务并后台停止旧服务，启动新服务，不会影响到其所依赖的服务。此命令有如下选项：

①：-d 在后台运行服务容器。

②：--no-color 不使用颜色来区分不同的服务的控制台输出。

③：--no-deps 不启动服务所链接的容器。

④：--force-recreate 强制重新创建容器，不能与 --no-recreate 同时使用。

⑤：--no-recreate 如果容器已经存在了，则不重新创建，不能与 --force-recreate 同时使用。

⑥：--no-build 不自动构建缺失的服务镜像。

⑦：-t, --timeout TIMEOUT 停止容器时候的超时（默认为 10 秒）。

- down

此命令停止用up命令所启动的容器并移除网络，如docker-compose -f skywalking.yml down

- stop

格式为 docker-compose stop [options] [SERVICE...]

停止已经处于运行状态的容器，但不删除它。通过 docker-compose start 可以再次启动这些容器，果不指定service则默认停止所有的容器。如docker-compose -f skywalking.yml stop elasticsearch

选项：

-t, --timeout TIMEOUT 停止容器时候的超时（默认为 10 秒）。

- start

启动已经存在的服务容器。用法跟上面的stop刚好相反,如docker-compose -f skywalking.yml start lasticsearch

- restart

重启项目中的服务。用法跟上面的stop,start一样

- logs

格式为docker-compose logs [options] [SERVICE...]

查看服务容器的输出。默认情况下，docker-compose 将对不同的服务输出使用不同的颜色来区分可以通过 --no-color 来关闭颜色。该命令在调试问题的时候十分有用。如docker-compose -f skywaking.yml logs 查看整体的日志，docker-compose -f skywalking.yml logs elasticsearch 查看单独器的日志

docker compose 模板文件

模板文件是使用 Compose 的核心，涉及到的指令关键字也比较多。本文主要列出几个常见&重要的令，其他指令大家可以自行百度。

默认的模板文件名称为 docker-compose.yml，格式为 YAML 格式。

```
version: '3'
services:
  elasticsearch:
    image: elasticsearch:6.8.5
    container_name: elasticsearch
    restart: always
    volumes:
      - /app/skywalking/elasticsearch/data:/usr/share/elasticsearch/data:rw
      - /app/skywalking/elasticsearch/conf/elasticsearch.yml:/usr/share/elasticsearch/config/ela
ticsearch.yml
      - /app/skywalking/elasticsearch/conf/jvm.options:/usr/share/elasticsearch/config/jvm.opt
ons
      - /app/skywalking/elasticsearch/logs:/usr/share/elasticsearch/logs:rw
    environment:
      - TZ=Asia/Shanghai
      - xpack.monitoring.enabled=false
      - xpack.watcher.enabled=false
    ports:
      - "9200:9200"
      - "9300:9300"
```

注意每个服务都必须通过 image 指令指定镜像或 build 指令（需要 Dockerfile）等来自动构建生成像。如果使用 build 指令，在 Dockerfile 中设置的选项(例如：CMD, EXPOSE, VOLUME, ENV 等)会自动被获取，无需在 docker-compose.yml 中重复设置。

常用&重要命令

- images

指定为镜像名称或镜像 ID。如果镜像在本地不存在，Compose 将会尝试拉取这个镜像。

```
image: apache/skywalking-oap-server:6.5.0
image: apache/skywalking-ui:6.5.0
```

- ports

暴露端口信息。

使用宿主端口：容器端口 (HOST:CONTAINER) 格式，或者仅仅指定容器的端口（宿主将会随机选择口）都可以，端口字符串都使用引号包括起来的字符串格式。

```
ports:
   - "3000"
   - "8080:8080"
   - "127.0.0.1:8001:8001"
```

- volumes

数据卷所挂载路径设置。可以设置为宿主机路径(HOST:CONTAINER)或者数据卷名称(VOLUME:CONAINER)，并且可以设置访问模式 （HOST:CONTAINER:ro）。

```
volumes:
     - /app/skywalking/elasticsearch/data:/usr/share/elasticsearch/data:rw
     - conf/elasticsearch.yml:/usr/share/elasticsearch/config/elasticsearch.yml

version: "3"
services:
  my_src:
    image: mysql:8.0
    volumes:
      - mysql_data:/var/lib/mysql
volumes:
  mysql_data:
```

- ulimits

指定容器的 ulimits 限制值。

例如，指定最大进程数为 65535，指定文件句柄数为 20000（软限制，应用可以随时修改，不能超硬限制） 和 40000（系统硬限制，只能 root 用户提高）。

```
ulimits:
   nproc: 65535
   nofile:
     soft: 20000
     hard: 40000
```

- depends_on

解决容器的依赖、启动先后的问题。以下例子中会先启动 redis mysql 再启动 web

```
version: '3'
services:
  web:
    build: .
    depends_on:
```

```
    - db
    - redis
  redis:
    image: redis
  db:
    image: mysql
```

- environment

设置环境变量。你可以使用数组或字典两种格式。

```
environment:
    SW_STORAGE: elasticsearch
    SW_STORAGE_ES_CLUSTER_NODES: elasticsearch:9200

environment:
    - SW_STORAGE= elasticsearch
    - SW_STORAGE_ES_CLUSTER_NODES=elasticsearch:9200
```

- restart

指定容器退出后的重启策略为始终重启。该命令对保持服务始终运行十分有效，在生产环境中推荐配为 always 或者 unless-stopped。

```
restart: always
```

docker-compose 实战

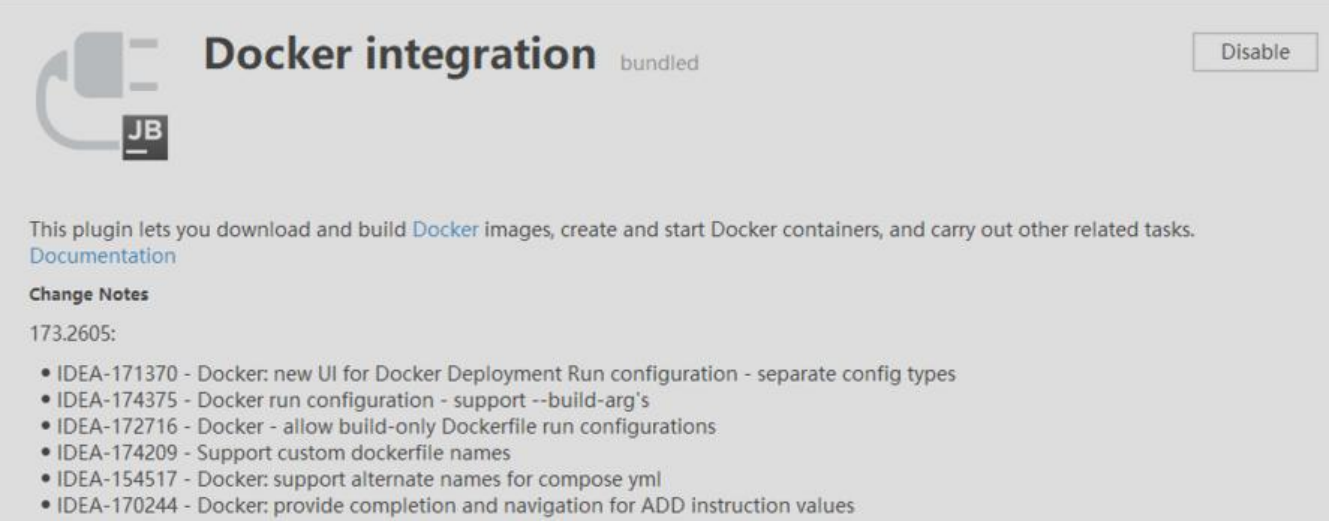
首先我需要推荐两件事：

- 配置docker加速镜像，配置国内加速镜像

创建或修改/etc/docker/daemon.json

```
sudo mkdir -p /etc/docker
sudo tee /etc/docker/daemon.json <<-'EOF'
{
   "registry-mirrors": [
      "https://hub-mirror.c.163.com",
      "https://mirror.ccs.tencentyun.com",
      "https://reg-mirror.qiniu.co"
   ]
}
EOF
sudo systemctl daemon-reload
sudo systemctl restart docker
```

- 给你的ide工具装上docker插件,智能提示，方便编写

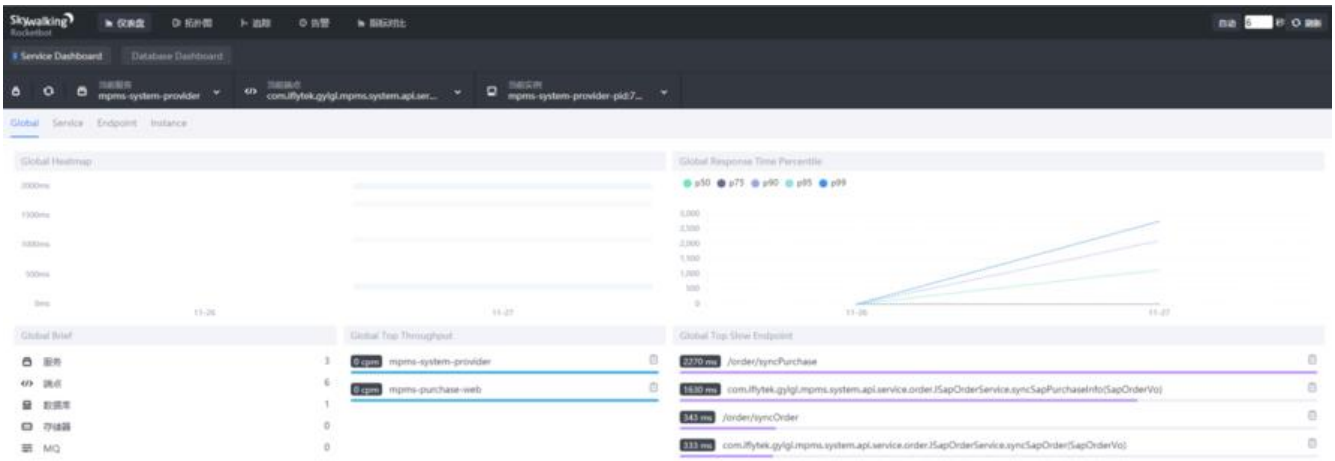


本次实战我们以docker-compose部署skywalking为例。编写skywalking.yml，内容如下。

```
version: '3.3'
services:
  elasticsearch:
    image: docker.elastic.co/elasticsearch/elasticsearch:6.8.5
    container_name: elasticsearch
    restart: always
    ports:
      - 9200:9200
      - 9300:9300
    environment:
      discovery.type: single-node
    ulimits:
      memlock:
        soft: -1
        hard: -1
  oap:
    image: skywalking/oap
    container_name: oap
    depends_on:
      - elasticsearch
    links:
      - elasticsearch
    restart: always
    ports:
      - 11800:11800
      - 12800:12800
    environment:
      SW_STORAGE: elasticsearch
      SW_STORAGE_ES_CLUSTER_NODES: elasticsearch:9200
  ui:
    image: skywalking/ui
    container_name: ui
    depends_on:
      - oap
    links:
      - oap
```

```
    restart: always
    ports:
      - 8080:8080
    environment:
      SW_OAP_ADDRESS: oap:12800
```

部署完成后将其上传至服务器，执行docker-compose -f /app/skywalking.yml up -d即可。



原文链接：Docker-Compose 基础与实战，看这一篇就够了