利用docker+个人测试机搭建个人开发环境

> 作者： 潘宣宇 11104333

前提申请个人测试机

> 相关路径： paas 平台 --> CMDB --> 计算资源 --> 个人测试机 --> 主机申请

docker

docker 安装

1. 安装docker

```
yum install docker -y
```

2. 启动docker 并设置开机启动服务

```
systemctl start docker
systemctl enable docker
```

3. 查看doker版本

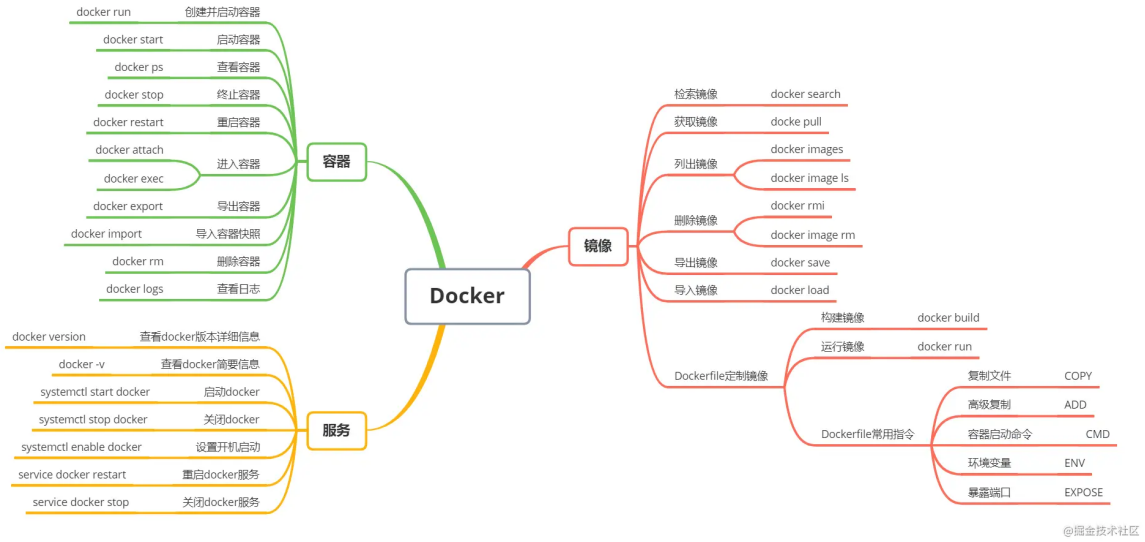
```
docker -v
```

4. 查看docker进程

```
ps -ef | grep docker
```

docker 常用命令

> 参考链接： 一张脑图整理Docker常用命令



docker-compose 安装

> 这个可以按需安装

1. docker-compose 安装

```
sudo curl -L
https://download.fastgit.org/docker/compose/releases/download/1.27.4/docker-
compose-\`uname -s\`-\`uname -m\` > /usr/local/bin/docker-compose
```

2. 赋予执行权限

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

docker 中各开发组件的安装

安装mysql

> 安装单体的mysql，最基本的安装

1. 拉取mysql镜像

```
docker pull mysql
```

镜像查询搜索可以查看 docker hub ，也可以输入命令 `docker serch mysql` 查询

2. 创建mysql数据存储相关文件

```
mkdir -p /root/mysql/datadir
mkdir -p /root/mysql/conf
mkdir -p /root/mysql/log
```

3. 启动mysql容器

```
docker run --name mysql -p 3306:3306 -v /root/mysql/datadir:/var/lib/mysql -v
/root/mysql/conf:/etc/mysql/conf.d -v /root/mysql/logs:/var/log/mysql -e
MYSQL_ROOT_PASSWORD=123456 -d d1165f221234
```

`-v /root/mysql/datadir:/var/lib/mysql` 这些都是设置保存到虚拟机本机的文件配置

`MYSQL_ROOT_PASSWORD` root帐号 密码
`-d` 后面根的为 镜像id

可以通过 `docker ps` 查看是否启动成功，以及相关容器id

4. 进入mysql容器

```
docker exec -it mysql bash
```

这里使用 `docker exec -it` + `容器名称` 或者 `容器id` + `bash` 或者 `/bin/bash` 都可以进入

```
[root@vm_11104333-persona1-10-101-36-22.v-sz-1.vivo.lan:/root]
# docker exec -it mysql bash
root@128267553df4:/#
```

5. 进入mysql

```
root@128267553df4:/# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 13
Server version: 8.0.26 MySQL Community Server - GPL

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

6. 退出mysql 与退出容器 均使用 exit 命令即可。

安装redis集群

> 参考链接： https://juejin.cn/post/6992872034065727525
>
> 主要思想搭建6个redis容器，然后分别启动，并建立集群，这里启用使用shell命令实现一次启动多个。

1. 拉取redis镜像

```
docker pull redis
```

2. 创建虚拟网卡，主要是用于redis-cluster能于外界进行网络通信，一般常用桥接模式。

```
docker network create myredis
```

可以通过 `docker network ls` 命令查看相关docker 网络信息

3. 写配置文件，下面为shell命令。

```
for port in $(seq 6379 6384);
do
mkdir -p /root/redis/node-${port}/conf
touch /root/redis/node-${port}/conf/redis.conf
cat  << EOF > /root/redis/node-${port}/conf/redis.conf
port ${port}
requirepass 123456
bind 0.0.0.0
protected-mode no
daemonize no
appendonly yes
cluster-enabled yes
cluster-config-file nodes.conf
cluster-node-timeout 5000
cluster-announce-ip  10.101.36.22
cluster-announce-port ${port}
cluster-announce-bus-port 1${port}
EOF
done
```

`port` ：节点端口；
`requirepass` ：设置密码，访问时需要验证
`protected-mode` ：保护模式，默认值 yes，即开启。开启保护模式以后，需配置 `bind ip` 或者设置访问密码；关闭保护模式，外部网络可以直接访问；
`daemonize` ：是否以守护线程的方式启动（后台启动），默认 no；
`appendonly` ：是否开启 AOF 持久化模式，默认 no；
`cluster-enabled` ：是否开启集群模式，默认 no；
`cluster-config-file` ：集群节点信息文件；

`cluster-node-timeout`：集群节点连接超时时间；
`cluster-announce-ip`：集群节点 IP，这里直接填 **服务器的IP** 地址即可
`cluster-announce-port`：集群节点映射端口；
`cluster-announce-bus-port`：集群节点总线端口。

4. 启动redis

```
for port in $(seq 6379 6384); \
do \
   docker run -it -d -p ${port}:${port} -p 1${port}:1${port} \
  --privileged=true -v
/root/redis/node-${port}/conf/redis.conf:/usr/local/etc/redis/redis.conf \
  --privileged=true -v /home/redis/node-${port}/data:/data \
  --restart always --name redis-${port} --net myredis \
  --sysctl net.core.somaxconn=1024 redis redis-server
/usr/local/etc/redis/redis.conf
done
```

-it：交互
-d：后台运行，容器启动完成后打印容器
--privileged：是否让docker 应用容器 获取宿主机root权限（特殊权限-）
-p: 端口映射
-v：文件挂载
--sysctl 参数来设置系统参数，通过这些参数来调整系统性能
--restart always：在容器退出时总是重启容器
--name： 给容器取名
--net myredis： 使用我们创建的虚拟网卡 （想详细了解，可以去看看Docker 网络方面知识）

5. 进入一个redis容器并创建集群

```
docker exec -it redis-6379 bash
```

创建集群

```
redis-cli  -a 123456 --cluster create 10.101.36.22:6379 10.101.36.22:6380
10.101.36.22:6381 10.101.36.22:6382 10.101.36.22:6383 10.101.36.22:6384   --
cluster-replicas 1
```

zookeeper集群搭建

> 这里使用 docker-compose 进行多容器的管理，不再像redis集群搭建一样用shell命令去创建多个并逐个启用的方式。
>
> 编写 docker-comose yaml 文件，这个文件可以理解为就是docker 服务配置的一个文件，但是他可以配置多个服务以及关联性等，有个很坑的点是yaml中没有tab 格式话只有空格。可能会遇到这个文件编写的错误，建议用vscode编写然后全选，可以查看是否有制表符或者空格对齐问题。

1. 创建docker-compose的yaml文件， 建议找个合适的文件目录保存这个文件。

   > vi/vim 参考链接: vi/vim命令大全

   ```
   vi docker-compose-zookeeper.yaml
   ```

   复制下文内容保存

   ```
   version: '3'

   services:
     zoo1:
       image: zookeeper:3.4.11
       restart: always
       hostname: zoo1
       ports:
         - 2181:2181
       networks:
         - zookeeper
       environment:
         ZOO_MY_ID: 1
         ZOO_SERVERS: server.1=zoo1:2888:3888 server.2=zoo2:2888:3888
   server.3=zoo3:2888:3888

     zoo2:
       image: zookeeper:3.4.11
       restart: always
       hostname: zoo2
       ports:
         - 2182:2181
       networks:
         - zookeeper
       environment:
         ZOO_MY_ID: 2
         ZOO_SERVERS: server.1=zoo1:2888:3888 server.2=zoo2:2888:3888
   server.3=zoo3:2888:3888
   ```

```
  zoo3:
    image: zookeeper:3.4.11
    restart: always
    hostname: zoo3
    ports:
      - 2183:2181
    networks:
      - zookeeper
    environment:
      ZOO_MY_ID: 3
      ZOO_SERVERS: server.1=zoo1:2888:3888 server.2=zoo2:2888:3888
server.3=zoo3:2888:3888
networks:
  zookeeper:
    driver: bridge
```

2. 启用相关容器

```
docker-compose -f docker-compose-zookeeper.yaml up -d
```

> 这里需要注意的是： 镜像image 以及网络 network 不需要提前拉取和建立，执行docker-compose文件时都会默认进行拉取镜像和创建网络。
>
> - **其他**
> - 使用docker-compose 启用的服务停止 docker-compose -f docker-compose-zookeeper.yaml rm -sf

rabbit 集群搭建

> 采用与zookeeper一样的方式，内容就不多写了，直接上docker-compose yaml 文件

1. 创建 docker-compose-rabbitmq.yaml 文件

```
version: '3'
services:
  rabbitmq1:
    image: rabbitmq:management
    container_name: rabbitmq1
    restart: always
    hostname: rabbitmq1
```

```
    ports:
      - "5672:5672"
      - "15672:15672"
    volumes:
      - /opt/docker_volume/rabbitmq/rabbitmq1/data:/var/lib/rabbitmq
    environment:
      - RABBITMQ_DEFAULT_USER=root
      - RABBITMQ_DEFAULT_PASS=root
      - RABBITMQ_ERLANG_COOKIE=CURIOAPPLICATION
      - RABBITMQ_NODENAME:rabbitmq1
    networks:
      - rabbitmq
  rabbitmq2:
    image: rabbitmq:management
    container_name: rabbitmq2
    restart: always
    hostname: rabbitmq2
    ports:
      - "5673:5672"
    volumes:
      - /opt/docker_volume/rabbitmq/rabbitmq2/data:/var/lib/rabbitmq
    environment:
      - RABBITMQ_ERLANG_COOKIE=CURIOAPPLICATION
      - RABBITMQ_NODENAME:rabbitmq2
      - RABBITMQ_CLUSTERED=true
      - RABBITMQ_CLUSTER_WITH=rabbit@rabbitmq1
      - RABBITMQ_RAM_NODE=true
    networks:
      - rabbitmq
  rabbitmq3:
    image: rabbitmq:management
    container_name: rabbitmq3
    restart: always
    hostname: rabbitmq3
    ports:
      - "5674:5672"
    volumes:
      - /opt/docker_volume/rabbitmq/rabbitmq3/data:/var/lib/rabbitmq
    environment:
      - RABBITMQ_ERLANG_COOKIE=CURIOAPPLICATION
      - RABBITMQ_NODENAME:rabbitmq3
      - RABBITMQ_CLUSTERED=true
      - RABBITMQ_CLUSTER_WITH=rabbit@rabbitmq1
      - RABBITMQ_RAM_NODE=true
    networks:
      - rabbitmq
networks:
  rabbitmq:
    driver: bridge
```

2. 启用容器

```
docker-compose -f docker-compose-rabbitmq.yaml up -d
```