主讲老师： Fox

> 有道笔记地址： https://note.youdao.com/s/Ct26vIFy

课前须知：学习本节课需要先掌握docker的使用

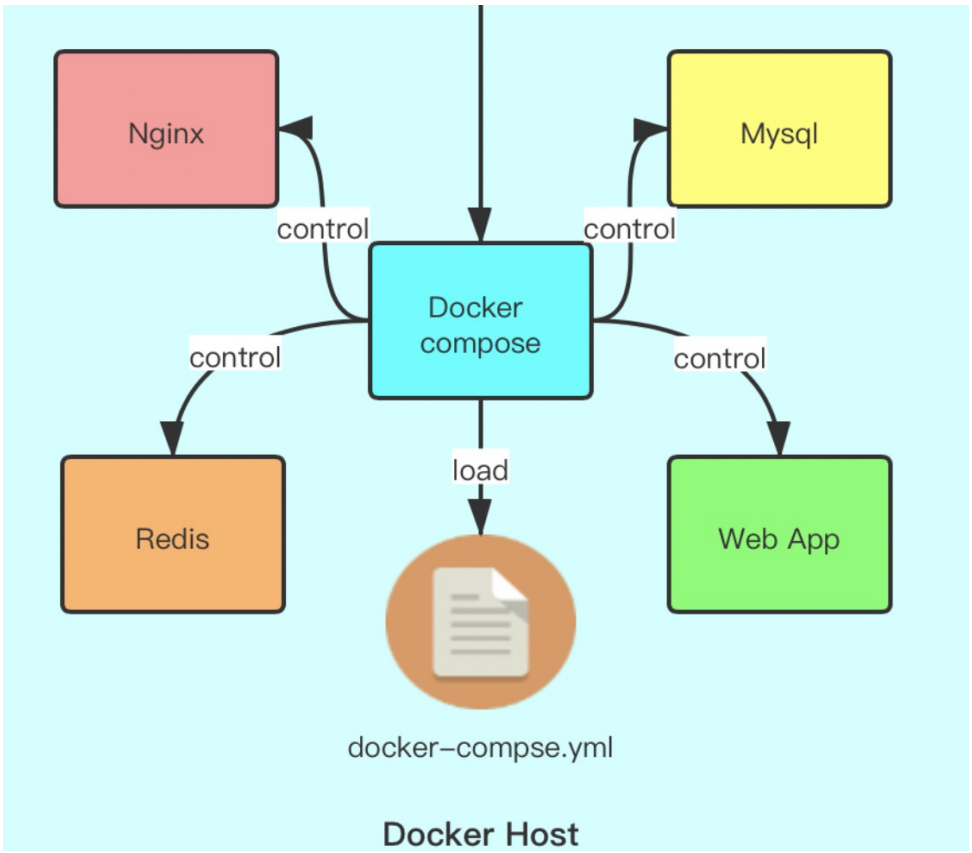
> docker version: 24.0.2

1. Docker Compose使用

Docker帮助我们解决服务的打包安装的问题，随着而来的问题就是服务过多的带来如下问题：

1. 多次使用 Dockerfile、Build、Image 命令或者 DockerHub 拉取 Image;
2. 需要创建多个Container，多次编写启动命令;
3. Container互相依赖的如何进行管理和编排。

Compose 是一个用于定义和运行多容器的Docker应用的工具。使用Compose，你可以在一个配置文件（yaml格式）中配置你应用的服务，然后使用一个命令，即可创建并启动配置中引用的所有服务。



Compose 使用的三个步骤：

1. 使用 Dockerfile 定义应用程序的环境
2. 使用 docker-compose.yml 定义构成应用程序的服务，这样它们可以在隔离环境中一起运行
3. 最后，执行 docker compose up 命令来启动并运行整个应用程序

1.1 Docker Compose管理容器的结构

Docker Compose将所管理的容器分为三层，**分别是工程（project），服务（service）以及容器（container）**。 Docker Compose运行目录下的所有文件（docker-compose.yml、 extends文件或

环境变量文件等）组成一个工程（默认为 docker-compose.yml所在目录的目录名称）。一个工程可包含多个服务，每个服务中定义了容器运行的镜像、参数和依赖，一个服务可包括多个容器实例。
同一个docker compose内部的容器之间可以用服务名相互访问，**服务名就相当于hostname，可以直接 ping 服务名，得到的就是服务对应容器的ip，如果服务做了扩容，一个服务对应了多个容器，则 ping 服务名 会轮询访问服务对应的每台容器ip ，docker底层用了LVS等技术帮我们实现这个负载均衡。**

1.2 Docker Compose.yml常用指令

image

指定镜像名称或者镜像id，如果该镜像在本地不存在，Compose会尝试pull下来。
示例：

```
image: java
```

build
指定Dockerfile文件的路径。可以是一个路径，例如：

```
build: ./dir
```

也可以是一个对象，用以指定Dockerfile和参数，例如：

```
build:
  context: ./dir
  dockerfile: Dockerfile-alternate
  args:
    buildno: 1
```

command

覆盖容器启动后默认执行的命令。
示例：

```
command: bundle exec thin -p 3000
```

也可以是一个list，类似于Dockerfile总的CMD指令，格式如下：

```
command: [bundle, exec, thin, -p, 3000]
```

links

显示链接到其他服务中的容器。可以指定服务名称和链接的别名使用`SERVICE:ALIAS`的形式，或者只指定服务名称，示例：

```
web:
  links:
    - db
    - db:database
    - redis
```

external_links

表示链接到docker-compose.yml外部的容器，甚至并非Compose管理的容器，特别是对于那些提供共享容器或共同服务。格式跟links类似，示例：

```
external_links:
  - redis_1
  - project_db_1:mysql
  - project_db_1:postgresql
```

ports

暴露端口信息。使用宿主端口:容器端口的格式，或者仅仅指定容器的端口（此时宿主机将会随机指定端口），类似于`docker run -p`，示例：

```
ports:
  - "3000"
  - "3000-3005"
  - "8000:8000"
  - "9090-9091:8080-8081"
  - "49100:22"
  - "127.0.0.1:8001:8001"
  - "127.0.0.1:5000-5010:5000-5010"
```

expose

暴露端口，只将端口暴露给连接的服务，而不暴露给宿主机，示例：

```
expose:
 - "3000"
 - "8000"
```

volumes

卷挂载路径设置。可以设置宿主机路径 （`HOST:CONTAINER`） 或加上访问模式 （`HOST:CONTAINER:ro`）。示例：

```
volumes:
  # Just specify a path and let the Engine create a volume
  - /var/lib/mysql

  # Specify an absolute path mapping
  - /opt/data:/var/lib/mysql

  # Path on the host, relative to the Compose file
  - ./cache:/tmp/cache

  # User-relative path
  - ~/configs:/etc/configs/:ro

  # Named volume
  - datavolume:/var/lib/mysql
```

volumes_from

从另一个服务或者容器挂载卷。可以指定只读或者可读写，如果访问模式没有指定，则默认是可读写。示例：

```
volumes_from:
 - service_name
 - service_name:ro
 - container:container_name
```

```
  - container:container_name:rw
```

environment

设置环境变量。可以使用数组或者字典两种方式。只有一个key的环境变量可以在运行Compose的机器上找到对应的值，这有助于加密的或者特殊主机的值。示例：

```
environment:
  RACK_ENV: development
  SHOW: 'true'
  SESSION_SECRET:

environment:
  - RACK_ENV=development
  - SHOW=true
  - SESSION_SECRET
```

env_file

从文件中获取环境变量，可以为单独的文件路径或列表。如果通过 `docker-compose -f FILE` 指定了模板文件，则 `env_file` 中路径会基于模板文件路径。如果有变量名称与 `environment` 指令冲突，则以 `envirment` 为准。示例：

```
env_file: .env

env_file:
  - ./common.env
  - ./apps/web.env
  - /opt/secrets.env
```

extends

继承另一个服务，基于已有的服务进行扩展。

net

设置网络模式。示例：

```
net: "bridge"
net: "host"
```

```
net: "none"
net: "container:[service name or container name/id]"
```

dns

配置dns服务器。可以是一个值，也可以是一个列表。示例：

```
dns: 8.8.8.8
dns:
  - 8.8.8.8
  - 9.9.9.9
```

dns_search

配置DNS的搜索域，可以是一个值，也可以是一个列表，示例：

```
dns_search: example.com
dns_search:
  - dc1.example.com
  - dc2.example.com
```

其他

docker-compose.yml 还有很多其他命令，这里仅挑选常用命令进行讲解，其它不作赘述。如果感兴趣的，可以参考docker-compose.yml文件官方文档：https://docs.docker.com/compose/compose-file/

1.3 利用Docker Compose部署nacos环境

参考：https://nacos.io/zh-cn/docs/quick-start-docker.html

1）创建一个空目录docker-mall

2）在docker-mall目录下新建一个编排文件docker-compose-env.yml，内容如下：

```
version: '3'
services:
  mysql:
    image: mysql:8.0
    container_name: mysql
    ports:
```

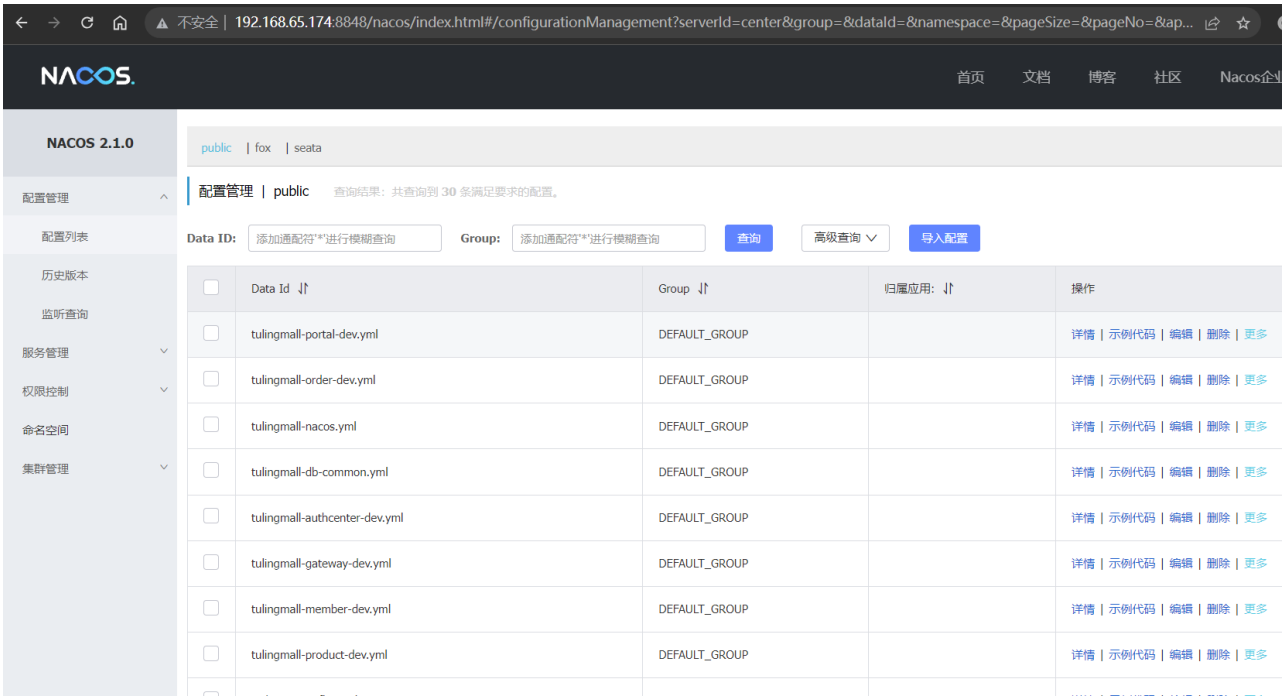
```
      - 3306:3306
    restart: always
    volumes:
      - ./mysql:/var/lib/mysql
    environment:
      - MYSQL_ROOT_PASSWORD=root
      - MYSQL_DATABASE=nacos
      - MYSQL_USER=nacos
      - MYSQL_PASSWORD=nacos
    command: --default-authentication-plugin=mysql_native_password
  nacos:
    image: nacos/nacos-server:v2.1.0
    container_name: nacos
    volumes:
      - ./nacos/logs/:/home/nacos/logs
      - ./nacos/init.d/:/home/nacos/init.d
    ports:
      - "8848:8848"
      - "9848:9848"
      - "9555:9555"
    environment:
      - PREFER_HOST_MODE=hostname
      - MODE=standalone
      - SPRING_DATASOURCE_PLATFORM=mysql
      - MYSQL_SERVICE_HOST=mysql
      - MYSQL_SERVICE_DB_NAME=nacos
      - MYSQL_SERVICE_PORT=3306
      - MYSQL_SERVICE_USER=nacos
      - MYSQL_SERVICE_PASSWORD=nacos
      - MYSQL_SERVICE_DB_PARAM=characterEncoding=utf8&connectTimeout=1000&socketTimeout=3000&autoReconnect=true&useSSL=false
    depends_on:
      - mysql
    restart: always
```

3）启动compose，在docker-mall目录执行如下命令：

```
docker compose -f docker-compose-env.yml up -d
```

执行成功后会启动nacos容器，可以通过http://192.168.65.174:8848/nacos/index.html访问nacos控制台

4) 访问nacos控制台，并导入电商项目微服务相关配置



2. 使用Docker Compose编排电商微服务项目

2.1 构建电商微服务docker镜像

可以利用idea插件Alibaba Cloud Toolkit上传服务到远程linux服务器，并构建docker镜像。

在IntelliJ IDEA中安装和配置Cloud Toolkit 使用IntelliJ IDEA部署应用到Linux服务器

以tulingmall-member为例

1）在linux服务器上创建tulingmall/tulingmall-member目录

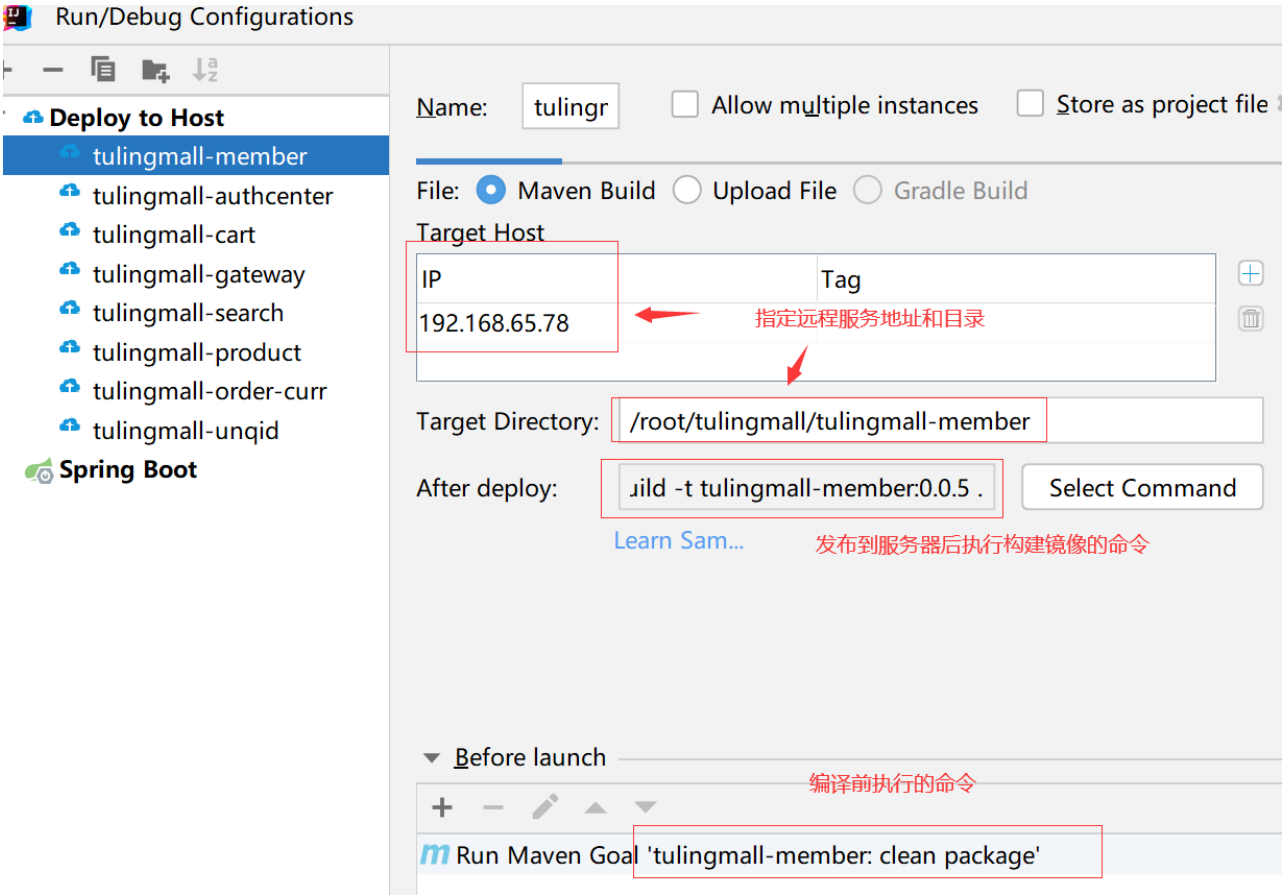
2）创建一个Dockerfile文件，用于生成docker镜像

```
vim Dockerfile

# 该镜像需要依赖的基础镜像
FROM java:8
# 将当前目录下的jar包复制到docker容器的/目录下
ADD tulingmall-member-0.0.5.jar /tulingmall-member-0.0.5.jar
# 声明服务运行在8877端口
EXPOSE 8877
```

```
# 指定docker容器启动时运行jar包
ENTRYPOINT ["java", "-jar","/tulingmall-member-0.0.5.jar"]
```

3）借助Alibaba Cloud Toolkit插件编译tulingmall-member服务，上传到远程服务器tulingmall-member目录下，并生成docker镜像



上传到了服务器指定目录下

```
[root@192-168-65-78 tulingmall-member]# ll
总用量 97696
-rw-r--r--. 1 root root       327 6月    5 12:57 Dockerfile
-rw-r--r--. 1 root root 100035549 6月    7 17:02 tulingmall-member-0.0.5.jar
```

生成了tulingmall-member的镜像

```
[root@192-168-65-78 tulingmall-member]# docker images
REPOSITORY                                TAG       IMAGE ID       CREATED          SIZE
192.168.65.78:5000/tulingmall-product     latest    50ccd6bde335   12 days ago      763MB
tulingmall-product                        0.0.5     50ccd6bde335   12 days ago      763MB
tulingmall-order-curr                     0.0.5     1d519d953646   12 days ago      786MB
tulingmall-gateway                        0.0.5     c0c4f38f77a3   12 days ago      712MB
tulingmall-search                         0.0.5     bea59ce3b528   12 days ago      740MB
192.168.65.78:5000/tulingmall-member      latest    bda576aeff7c   12 days ago      743MB
tulingmall-member                         0.0.5     539502315144   12 days ago      743MB
tulingmall-cart                           0.0.5     69d27ec38470   2 weeks ago      741MB
```

同上面的步骤一样，依次生成tulingmember-product,tulingmall-order-curr,tulingmall-gateway等微服务的镜像

2.2 编排电商微服务

```
[root@192-168-65-78 tulingmall]# ll
总用量 20
-rw-r--r--. 1 root root 4579 6月   7 22:23 docker-compose-app.yml
-rw-r--r--. 1 root root  599 6月   7 22:02 docker-compose-member.yml
-rw-r--r--. 1 root root 1215 6月   7 22:21 docker-compose-order.yml
-rw-r--r--. 1 root root  618 6月   7 22:23 docker-compose-product.yml
drwxr-xr-x. 2 root root   63 6月   5 16:14 tulingmall-authcenter
drwxr-xr-x. 2 root root   57 6月   6 11:38 tulingmall-cart
drwxr-xr-x. 2 root root   60 6月   5 16:45 tulingmall-gateway
drwxr-xr-x. 2 root root   59 6月   5 12:57 tulingmall-member
drwxr-xr-x. 2 root root   63 6月   5 22:10 tulingmall-order-curr
drwxr-xr-x. 2 root root   60 6月   5 16:44 tulingmall-product
drwxr-xr-x. 2 root root   59 6月   7 21:40 tulingmall-search
drwxr-xr-x. 2 root root   58 6月   5 16:43 tulingmall-unqid
```

以tulingmall-member服务为例

1) 在tulingmall目录下新建微服务编排文件docker-compose-member.yml，内容如下：

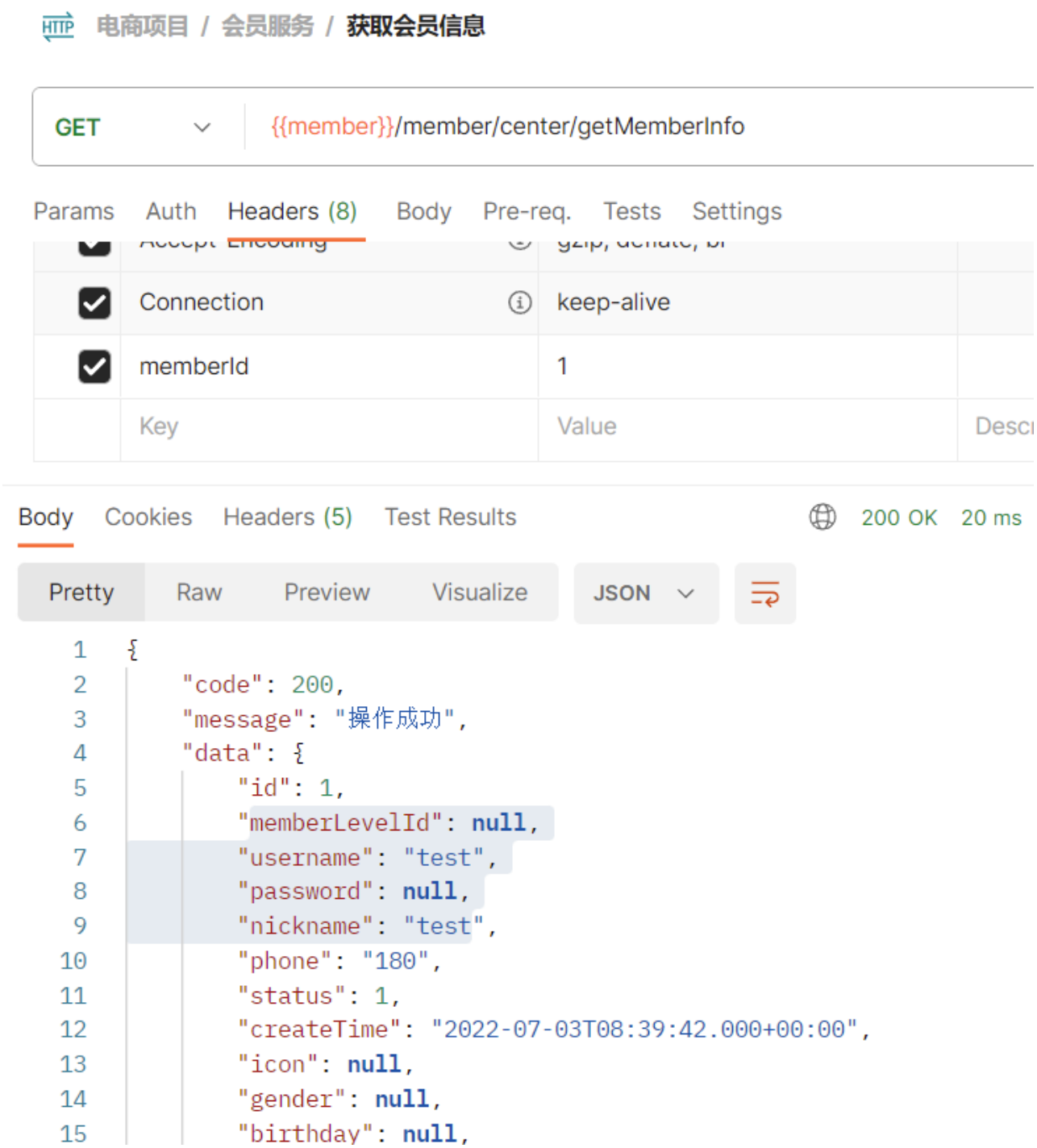
```yaml
version: "3"
services:
  tulingmall-member:
    image: tulingmall-member:0.0.5
    build: ./tulingmall-member
    container_name: tulingmall-member
    ports:
      - 8877:8877
    volumes:
      - /etc/localtime:/etc/localtime:ro
      - /skywalking-agent:/skywalking-agent
    environment:
      - SPRING_CLOUD_NACOS_CONFIG_SERVER_ADDR=192.168.65.174:8848
      - JAVA_TOOL_OPTIONS=-Xmx1g -Xms1g -XX:MaxMetaspaceSize=512m -javaagent:/skywalking-agent/skywalking-agent.jar -DSW_AGENT_NAME=tulingmall-member -DSW_AGENT_COLLECTOR_BACKEND_SERVICES=192.168.65.204:11800
    cap_add:
      - SYS_PTRACE
```

2) 启动compose，在tulingmall目录执行如下命令：

```
docker compose -f docker-compose-member.yml up -d
```

```
[root@192-168-65-78 tulingmall]# docker compose -f docker-compose-member.yml up -d
[+] Building 0.0s (0/0)
[+] Running 2/2
 ✓ Network tulingmall_default    Created                                    1.4s
 ✓ Container tulingmall-member   Started                                    6.8s
```

3) 访问下tulingmall-member的服务接口是否正常



所有微服务的配置文件docker-compose-app.yml

```
version: "3"
services:
```

```
  tulingmall-authcenter:
    image: tulingmall-authcenter:0.0.5  #指定镜像名称
    build: ./tulingmall-authcenter  #指定Dockfile所在路径
    container_name: tulingmall-authcenter  #指定启动容器名称
    ports:
      - 9999:9999
    volumes:
      - /etc/localtime:/etc/localtime:ro  #同步宿主机与容器时间，ro代表readonly只读
      - /skywalking-agent:/skywalking-agent
    environment:
      - SPRING_CLOUD_NACOS_CONFIG_SERVER_ADDR=192.168.65.174:8848
      - JAVA_TOOL_OPTIONS=-Xmx1g -Xms1g -XX:MaxMetaspaceSize=512m -javaagent:/skywalking-agent/skywalking-agent.jar -DSW_AGENT_NAME=tulingmall-authcenter -DSW_AGENT_COLLECTOR_BACKEND_SERVICES=192.168.65.204:11800
    cap_add:
      - SYS_PTRACE #这个参数是让docker能支持在容器里能执行jdk自带的类似jinfo，jmap这些命令，如果不需要在容器里执行这些命令可以不加
  tulingmall-gateway:
    image: tulingmall-gateway:0.0.5
    build: ./tulingmall-gateway
    container_name: tulingmall-gateway
    ports:
      - 8888:8888
    volumes:
      - /etc/localtime:/etc/localtime:ro
      - /skywalking-agent:/skywalking-agent
    environment:
      - SPRING_CLOUD_NACOS_CONFIG_SERVER_ADDR=192.168.65.174:8848
      - JAVA_TOOL_OPTIONS=-Xmx1g -Xms1g -XX:MaxMetaspaceSize=512m -javaagent:/skywalking-agent/skywalking-agent.jar -DSW_AGENT_NAME=tulingmall-gateway -DSW_AGENT_COLLECTOR_BACKEND_SERVICES=192.168.65.204:11800
    depends_on:
      - tulingmall-authcenter #gateway在authcenter启动之后再启动
    cap_add:
      - SYS_PTRACE
  tulingmall-member:
    image: tulingmall-member:0.0.5
    build: ./tulingmall-member
    container_name: tulingmall-member
    ports:
      - 8877:8877
```

```
    volumes:
      - /etc/localtime:/etc/localtime:ro
      - /skywalking-agent:/skywalking-agent
    environment:
      - SPRING_CLOUD_NACOS_CONFIG_SERVER_ADDR=192.168.65.174:8848
      - JAVA_TOOL_OPTIONS=-Xmx1g -Xms1g -XX:MaxMetaspaceSize=512m -javaagent:/skywalking-agent/skywalking-agent.jar -DSW_AGENT_NAME=tulingmall-member -DSW_AGENT_COLLECTOR_BACKEND_SERVICES=192.168.65.204:11800
    cap_add:
      - SYS_PTRACE
  tulingmall-product:
    image: tulingmall-product:0.0.5
    build: ./tulingmall-product
    container_name: tulingmall-product
    ports:
      - 8866:8866
    volumes:
      - /etc/localtime:/etc/localtime:ro
      - /skywalking-agent:/skywalking-agent
    environment:
      - SPRING_CLOUD_NACOS_CONFIG_SERVER_ADDR=192.168.65.174:8848
      - JAVA_TOOL_OPTIONS=-Xmx1g -Xms1g -XX:MaxMetaspaceSize=512m -javaagent:/skywalking-agent/skywalking-agent.jar -DSW_AGENT_NAME=tulingmall-product -DSW_AGENT_COLLECTOR_BACKEND_SERVICES=192.168.65.204:11800
    cap_add:
      - SYS_PTRACE
  tulingmall-order-curr:
    image: tulingmall-order-curr:0.0.5
    build: ./tulingmall-order-curr
    container_name: tulingmall-order-curr
    ports:
      - 8844:8844
    volumes:
      - /etc/localtime:/etc/localtime:ro
      - /skywalking-agent:/skywalking-agent
    environment:
      - SPRING_CLOUD_NACOS_CONFIG_SERVER_ADDR=192.168.65.174:8848
      - JAVA_TOOL_OPTIONS=-Xmx1g -Xms1g -XX:MaxMetaspaceSize=512m -javaagent:/skywalking-agent/skywalking-agent.jar -DSW_AGENT_NAME=tulingmall-order-curr -DSW_AGENT_COLLECTOR_BACKEND_SERVICES=192.168.65.204:11800
    cap_add:
```

```
      - SYS_PTRACE
  tulingmall-cart:
    image: tulingmall-cart:0.0.5
    build: ./tulingmall-cart
    container_name: tulingmall-cart
    ports:
      - 8855:8855
    volumes:
      - /etc/localtime:/etc/localtime:ro
      - /skywalking-agent:/skywalking-agent
    environment:
      - SPRING_CLOUD_NACOS_CONFIG_SERVER_ADDR=192.168.65.174:8848
      - JAVA_TOOL_OPTIONS=-Xmx1g -Xms1g -XX:MaxMetaspaceSize=512m -
javaagent:/skywalking-agent/skywalking-agent.jar -DSW_AGENT_NAME=tulingmall-cart -
DSW_AGENT_COLLECTOR_BACKEND_SERVICES=192.168.65.204:11800
    cap_add:
      - SYS_PTRACE
  tulingmall-unqid:
    image: tulingmall-unqid:0.0.5
    build: ./tulingmall-unqid
    container_name: tulingmall-unqid
    ports:
      - 8833:8833
    volumes:
      - /etc/localtime:/etc/localtime:ro
      - /skywalking-agent:/skywalking-agent
    environment:
      - SPRING_CLOUD_NACOS_CONFIG_SERVER_ADDR=192.168.65.174:8848
      - JAVA_TOOL_OPTIONS=-Xmx1g -Xms1g -XX:MaxMetaspaceSize=512m -
javaagent:/skywalking-agent/skywalking-agent.jar -DSW_AGENT_NAME=tulingmall-unqid -
DSW_AGENT_COLLECTOR_BACKEND_SERVICES=192.168.65.204:11800
    cap_add:
      - SYS_PTRACE
  tulingmall-search:
    image: tulingmall-search:0.0.5
    build: ./tulingmall-search
    container_name: tulingmall-search
    ports:
      - 8054:8054
    volumes:
      - /etc/localtime:/etc/localtime:ro
```

```
111       - /skywalking-agent:/skywalking-agent
112     environment:
113       - SPRING_CLOUD_NACOS_CONFIG_SERVER_ADDR=192.168.65.174:8848
114       - JAVA_TOOL_OPTIONS=-Xmx1g -Xms1g -XX:MaxMetaspaceSize=512m -
    javaagent:/skywalking-agent/skywalking-agent.jar -DSW_AGENT_NAME=tulingmall-search -
    DSW_AGENT_COLLECTOR_BACKEND_SERVICES=192.168.65.204:11800
115     cap_add:
116       - SYS_PTRACE
```