矿卡自动驾驶项目基于Docker快速部署方案

中间件篇

概述

docker目前是行业内非常流行的虚拟化技术，可以屏蔽各个系统底层的差异，并且可以保证各个微服务运行环境相互隔离，最可贵的是，可以配合jenkins，docker-compose，docker-swarm，k8s等实现容器编排，最终实现服务的高性能，高并发和高可用。

本项目现阶段中间件均为单机standalone部署，采用docker-compose使用配置文件（docker-compose）方式进行部署最为方便快捷。可以最终做到在一台空白机器上只要安装好docker和docker-compose，将项目配置好的docker-compose.yml配置放到指定的目录下，即可快速启动各个中间件。

准备docker和docker-compose环境

安装docker

> *官网网站：https://docs.docker.com/install/linux/docker-ce/ubuntu/*

使用repository安装

```
#先卸载旧版本的docker。一些系统可能会自带docker
sudo apt-get remove docker docker-engine docker.io
containerd runc
# 更新索引
sudo apt-get update
#安装docker
sudo apt-get install \
    apt-transport-https \
    ca-certificates \
    curl \
    gnupg-agent \
    software-properties-common

# 添加Docker的官方GPG密钥
curl -fsSL https://download.docker.com/linux/ubuntu/gpg |
sudo apt-key add -
# 校验密钥是否添加成功，比如上一步添加的密钥是9DC8 5822 9FC7 DD38
854A E2D8 8D81 803C 0EBF CD88
sudo apt-key fingerprint 0EBFCD88
```

```
# 设置docker仓库，这里应该替换成阿里云的仓库地址
sudo add-apt-repository \
   "deb [arch=amd64]
https://download.docker.com/linux/ubuntu \
   $(lsb_release -cs) \
   stable"
```

直接安装

```
# 更新索引
sudo apt-get update
# 直接安装最新版本
sudo apt-get install docker-ce docker-ce-cli containerd.io
## 如果想安装指定版本，先查询所有的版本
sudo apt-cache madison docker-ce
## 指定版本安装
sudo apt-get install docker-ce=<VERSION_STRING> docker-ce-
cli=<VERSION_STRING> containerd.io
```

使用**deb**包安装

```
# 先下载deb安装包
（https://download.docker.com/linux/ubuntu/dists/）
sudo dpkg -i xxx.deb
```

docker安装完成以后测试

```
docker run hello-world
```

配置国内镜像加速

在`/etc/docker/daemon.json`中添加：

```
{
    "registry-mirrors": [
        "https://registry.docker-cn.com"
    ]
}
```

然后重启docker

```
systemctl daemon reload
systemctl restart docker
```

安装**docker-compose**

```
sudo curl -L
"https://github.com/docker/compose/releases/download/1.25.
4/docker-compose-$(uname -s)-$(uname -m)" -o
/usr/local/bin/docker-compose
sudo chmod +x /usr/local/bin/docker-compose
```

安装完成后测试docker-compose

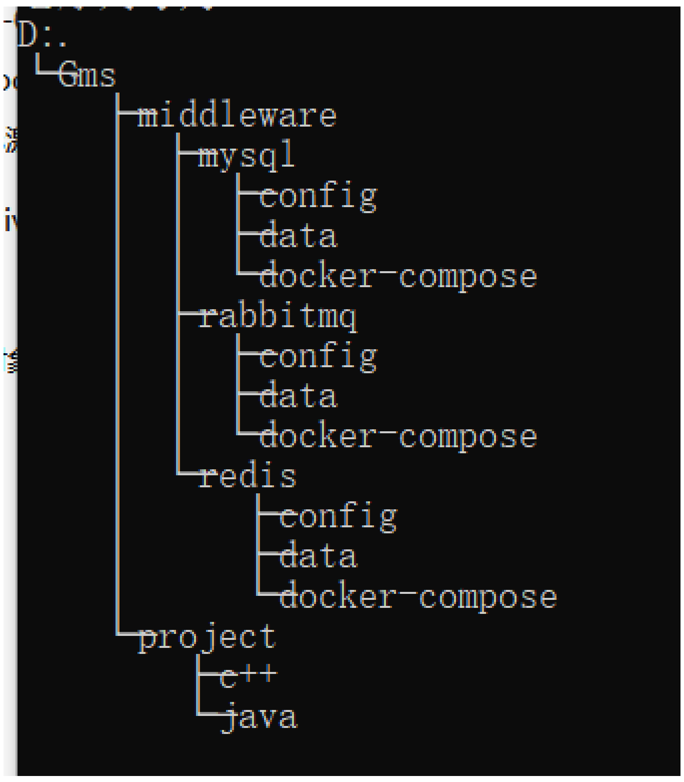
```
docker-compose --version
```

中间件分开部署

部署方案概述

所有的中间价均设置三个目录./config,./docker-compose,./data,分别用于存放配置文件，docker-compose脚本和数据

目录结构规划如下：



```
D:.
└─Gms
    ├─middleware
    │  ├─mysql
    │  │  ├─config
    │  │  ├─data
    │  │  └─docker-compose
    │  ├─rabbitmq
    │  │  ├─config
    │  │  ├─data
    │  │  └─docker-compose
    │  └─redis
    │      ├─config
    │      ├─data
    │      └─docker-compose
    └─project
        ├─c++
        └─java
```

部署MySQL

在mysql目录下的docker-compose中添加docker-compose.yml文件

```yaml
version: "2"
services:
  mysql:
    image: mysql
    ports:
      - "3306:3306"
    environment:
      MYSQL_ROOT_PASSWORD: 123456
    restart: always
    volumes:
      - /Gms/middleware/mysql/data:/var/lib/mysql
      - /Gms/middleware/mysql/config:/etc/mysql/
```

在目录下执行`docker-compose up -d`，即可启动mysql

部署Redis

在redis目录下的docker-compose中添加docker-compose.yml文件

```yaml
version: "2"
services:
  mysql:
    image: redis
    ports:
      - "6379:6379"
    environment:
      MYSQL_ROOT_PASSWORD: 123456
    restart: always
    volumes:
      - /Gms/middleware/redis/data:/var/lib/redis
      - /Gms/middleware/redis/config:/etc/redis/
```

在目录下执行`docker-compose up -d`，即可启动redis

部署RabbitMQ

在redis目录下的docker-compose中添加docker-compose.yml文件

```yaml
version: "2"
services:
  rabbitmq:
    image: rabbitmq
    ports:
      - "5672:5672"
      - "15672:15672"
    environment:
      MYSQL_ROOT_PASSWORD: 123456
    restart: always
    volumes:
      - /Gms/middleware/rabbitmq/data:/var/lib/rabbitmq
      - /Gms/middleware/rabbitmq/config:/etc/rabbitmq/
```

在目录下执行`docker-compose up -d`，即可启动rabbitmq

部署Kafka

在kafka目录下的docker-compose中添加docker-compose.yml文件

```yaml
version: "2"
services:
  zookeeper:
    image: wurstmeister/zookeeper
    ports:
      - "2188:2181"
  kafka:
    image: wurstmeister/kafka
    ports:
      - "9092:9092"
    environment:
      KAFKA_ADVERTISED_HOST_NAME: 192.168.2.100
      KAFKA_ZOOKEEPER_CONNECT: 192.168.2.100:2188
    volumes:
      - /usr/local/kafka:/kafka
      - /var/run/docker.sock:/var/run/docker.sock
  kafka-manager:
    image: sheepkiller/kafka-manager
    ports:
      - 10000:9000
    environment:
      ZK_HOSTS: 192.168.2.100:2188
```

在目录下执行`docker-compose up -d`，即可启动kafka

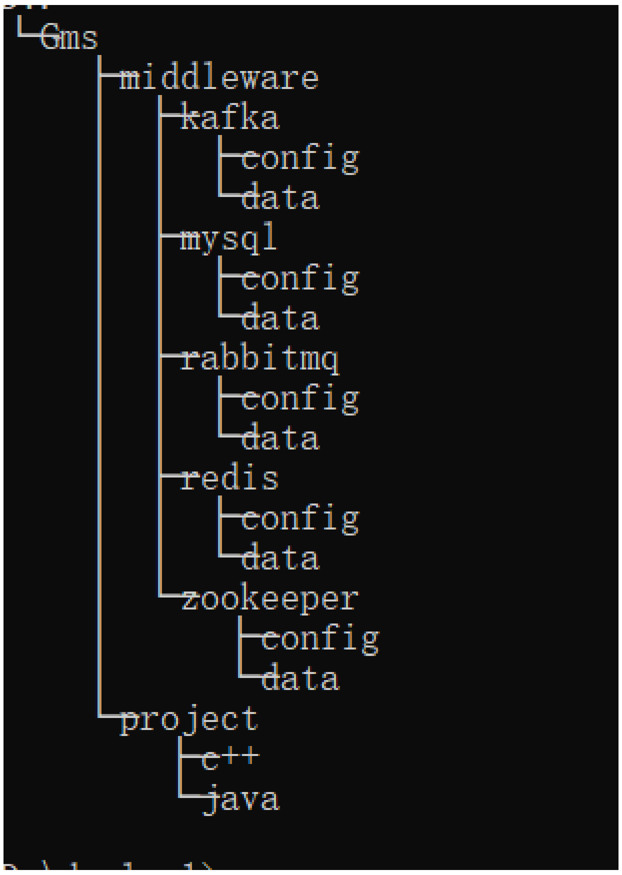
中间件集中部署方案

部署方案概述

所有的中间价均设置两个目录./config,./data,分别用于存放配置文件和数据

直接使用一个docker-compose.yml一键部署所有中间件，文件位置位于middleware下面

目录结构规划如下：



```
└Gms
    ├middleware
    │  ├kafka
    │  │  ├config
    │  │  └data
    │  ├mysql
    │  │  ├config
    │  │  └data
    │  ├rabbitmq
    │  │  ├config
    │  │  └data
    │  ├redis
    │  │  ├config
    │  │  └data
    │  └zookeeper
    │      ├config
    │      └data
    └project
        ├c++
        └java
```

部署中间件

middleware目录中添加docker-compose.yml文件

```yaml
version: "2"
services:
  mysql:
    image: mysql
    ports:
      - "3306:3306"
    environment:
      MYSQL_ROOT_PASSWORD: 123456
    restart: always
    volumes:
      - /Gms/middleware/mysql/data:/var/lib/mysql
      - /Gms/middleware/mysql/config:/etc/mysql/
  redis:
    image: redis
    ports:
      - "6379:6379"
    environment:
      MYSQL_ROOT_PASSWORD: 123456
    restart: always
    volumes:
      - /Gms/middleware/redis/data:/var/lib/redis
      - /Gms/middleware/redis/config:/etc/redis/
  rabbitmq:
    image: rabbitmq
    ports:
      - "5672:5672"
      - "15672:15672"
    environment:
      MYSQL_ROOT_PASSWORD: 123456
    restart: always
    volumes:
      - /Gms/middleware/rabbitmq/data:/var/lib/rabbitmq
      - /Gms/middleware/rabbitmq/config:/etc/rabbitmq/
  zookeeper:
    image: wurstmeister/zookeeper
    ports:
      - "2188:2181"
    restart: always
    volumes:
      - /Gms/middleware/zookeeper/data:/var/lib/zookeeper
      - /Gms/middleware/zookeeper/config:/etc/zookeeper/
  kafka:
    image: wurstmeister/kafka
    ports:
      - "9092:9092"
    environment:
      KAFKA_ADVERTISED_HOST_NAME: 192.168.2.100 #此处ip可能需要修改
      KAFKA_ZOOKEEPER_CONNECT: 192.168.2.100:2188 #此处ip可能需要修改
    volumes:
      - /Gms/middleware/kafka/data:/var/lib/kafka
      - /Gms/middleware/kafka/config:/etc/kafka/
      - /var/run/docker.sock:/var/run/docker.sock
  kafka-manager:
    image: sheepkiller/kafka-manager
    ports:
      - 9000:9000
    environment:
      ZK_HOSTS: 192.168.2.100:2188 #此处ip可能需要修改
```

启动中间件

在middleware目录下执行`docker-compose up -d`命令即可启动各个中间件