



链滴

# springboot|springboot 集成 apollo 做配置中心

作者: [xiaodaojava](#)

原文链接: <https://ld246.com/article/1574586067509>

来源网站: 链滴

许可协议: [署名-相同方式共享 4.0 国际 \(CC BY-SA 4.0\)](#)



## javaDEMO

本网站记录了最全的各种JavaDEMO,保证下载,复制就是可用的,包括基础的, 集合的, spring的, Mybatis的等等各种,助力你从菜鸟到大牛,记得收藏哦~~

<https://www.javastudy.cloud>

## apollo配置中心总述

apollo 是携程开源一款配置管理中心,能够集中化管理应用不同环境、不同集群的配置,配置修改后够实时推送到应用端. github地址和文档参考如下:<https://github.com/ctripcorp/apollo>

其主要组成部分有以下:

MetaServer: 类似于zookeeper, mq的namesrv, 基于Eureka提供注册的功能

client: 客户端, 就是我们的应用,需要获取配置的应用

configServer: 提供配置的读取和推送的功能. client通过metaServer获取到ConfigServer的地址,然后通过configServer或获取相关的配置

Admin Service: 提供配置的修改和发布的功能,portal通过metaServer获取到AdminServer的地址,然后通过adminServer对配置进行修改和发布

portal: 可以理解为后台管理页面,如上所述,先通过metaServer获取到AdminServer地址,然后通过adminServer对配置修改和发布

## 启动apollo各个服务端

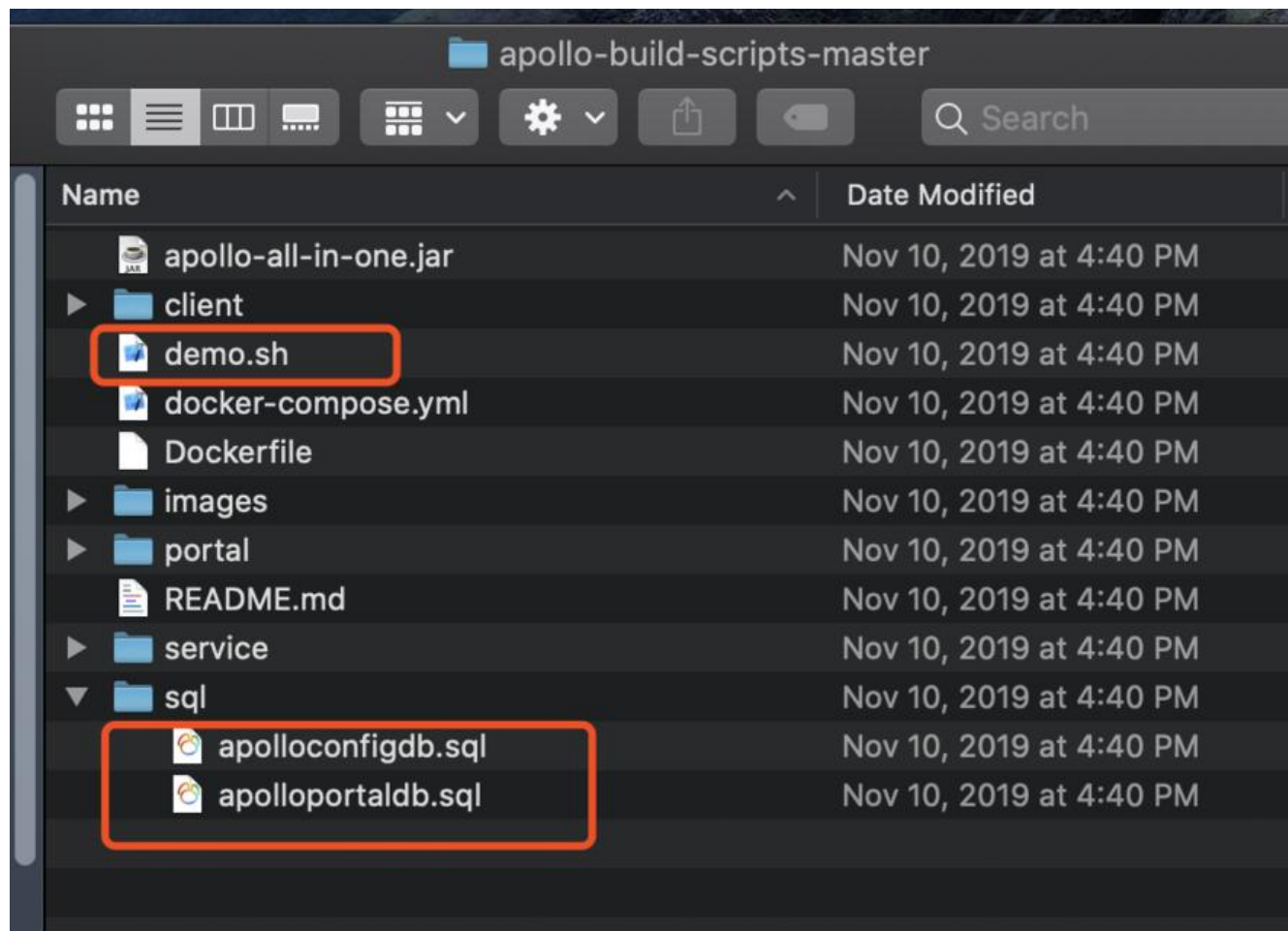
官方文档提及到,可以通过docker-compose来部署,我建议学习测试可以直接通过jar包来启动,到生产环境可使用k8s容器编排来部署,后续会有k8s相关的DEMO出来

用jar方式启动参考文档:<https://github.com/ctripcorp/apollo/wiki/Quick-Start>

# 下载可启动的Jar包

官文下载地址:

<https://codeload.github.com/nobodyiam/apollo-build-scripts/zip/master>



下载完如上图所示,有三个地方需要注意,需要先数据库中导入这两个sql文件,然后在demo.sh中修改数据库相关配置

## 配置数据库

导入sql的方法,可以用mysql命令行 source , 更推荐复制到可视化界面中,如navicat , datagrip然后再行,执行完之后的数据库如下图所示:

- ▼ ApolloConfigDB
  - ▶ App
  - ▶ AppNamespace
  - ▶ Audit
  - ▶ Cluster
  - ▶ Commit
  - ▶ GrayReleaseRule
  - ▶ Instance
  - ▶ InstanceConfig
  - ▶ Item
  - ▶ Namespace
  - ▶ NamespaceLock
  - ▶ Release
  - ▶ ReleaseHistory
  - ▶ ReleaseMessage
  - ▶ ServerConfig
- ▼ ApolloPortalDB
  - ▶ App
  - ▶ AppNamespace
  - ▶ Authorities
  - ▶ Consumer
  - ▶ ConsumerAudit
  - ▶ ConsumerRole
  - ▶ ConsumerToken
  - ▶ Favorite
  - ▶ Permission
  - ▶ Role
  - ▶ RolePermission
  - ▶ ServerConfig
  - ▶ UserRole
  - ▶ Users

修改demo.sh文件对应的数据库配置,注意,为方便启动, 这里改为主机地址,不用localhost

Users > lixiang > Downloads > apollo-build-scripts-master > demo.sh

```
1  #!/bin/bash
2
3  # apollo config db info
4  apollo_config_db_url=jdbc:mysql://192.168.1.128:33309/ApolloConfigDB?characterEncoding=utf8
5  apollo_config_db_username=root
6  apollo_config_db_password=javastudy
7
8  # apollo portal db info
9  apollo_portal_db_url=jdbc:mysql://192.168.1.128:33309/ApolloPortalDB?characterEncoding=utf8
10 apollo_portal_db_username=root
11 apollo_portal_db_password=javastudy
```

同样, 下面的server url 也要更改成主机ip地址

```
25  # meta server url
26  config_server_url=http://192.168.1.128:8080
27  admin_server_url=http://192.168.1.128:8090
28  eureka_service_url=$config_server_url/eureka/
29  portal_url=http://192.168.1.128:8070
```

修改数据库ApolloConfigDB中ServerConfig表中eureka.service.url数据,修改为自己的ip



| Id | Key                | Cluster | Value                           | Comment                     |
|----|--------------------|---------|---------------------------------|-----------------------------|
| 1  | eureka.service.url | default | http://192.168.1.128:8080/eu... | Eureka服务Url, 多个service以逗号分隔 |

## 启动服务端程序

考虑到大多数小伙伴都是用windows,不能执行sh文件, 所以这里用docker里面的centos:8镜像来启动

1. 启动镜像, 并绑定8080(ConfigServer的端口),8070(Portal的端口),8090(Admin的端口)

```
# 启动容器
docker run -d -it --name apollo-server -p 8080:8080 -p 8070:8070 -p 8090:8090 centos:8 bash
# 创建文件夹
docker exec -it apollo-server mkdir -p /var/www
# 把apollo相关的文件拷贝到docker容器中
docker cp apollo-build-scripts-master/ apollo-server:/var/www/apollo
# 进入到容器中
docker exec -it apollo-server bash
# 进入到刚才拷贝的路径
cd /var/www/
# 改变文件所有者为root
chown root apollo -R
# 安装jdk
yum install -y java-1.8.0-openjdk-headless.x86_64
# 运行 demo.sh 启动 apollo 服务端
cd apollo
sh demo.sh start
```

## 可能出现的错误



如果遇到AdminService健康检查半天通不过,如下:

Admin service failed to start in 120 seconds! Please check ./service/apollo-service.log for more information

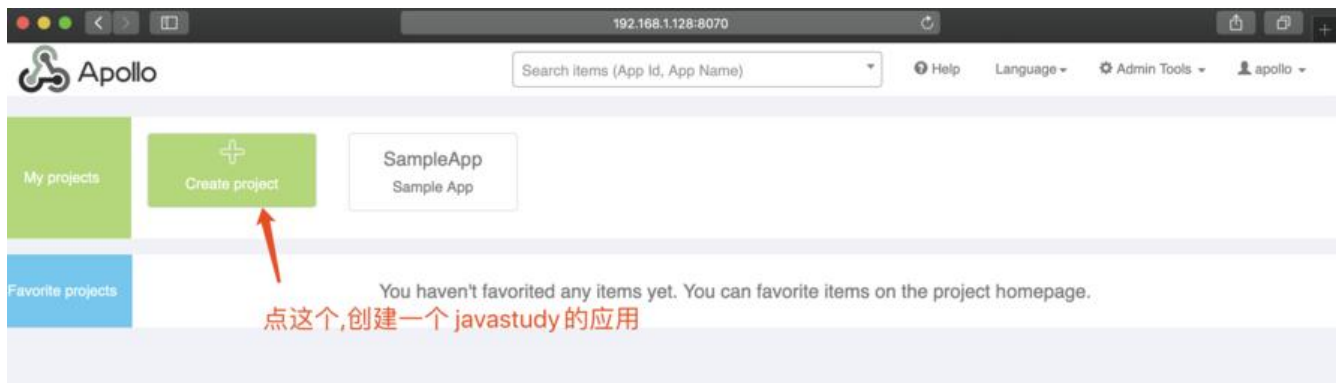
可编辑demo.sh 77行的 checkServerAlive方法如下所示:

```
76
77 function checkServerAlive {
78     declare -i counter=0
79     declare -i max_counter=48 # 24*5=120s
80     declare -i total_time=0
81
82     SERVER_URL="$1"
83     return 0
84 }
85
```

## 配置客户端程序获取配置

### 测试配置

输入apollo的portal的地址: <http://192.168.1.128:8070/signin> 用户名密码是默认的 apollo/admin



然后创建一个应用,点submit

\* Department

\* App Id   
(Application's unique identifiers)

\* App Name   
(Suggested format xx-yy-zz e.g. apollo-server)

\* App Owner

App Administrators   
(The application owner has project administrator permission by default.  
Project administrators can create namespace, cluster, and assign user permissions)

这三个先用默认的  
先都写javastudy

然后输入测试的配置

192.168.1.128:8070/config.html?#/appid=javastudy

Search items (App Id, App Name)

Help Language Admin Tools apollo

Environments  
DEV

Project Info  
App Id: javastudy  
App Name: javastudy  
Department: 样例部门1(TEST1)  
App Owner: apollo | apollo  
Email: apollo@apollo.com

Manage Project  
+ Add Cluster  
+ Add Namespace

Add Configuration (Reminder: Configuration can be added in batch via text mode)

\* Key   
输入一个测试 key

Value   
输入一个测试 value

Note: Hidden characters (Spaces, Newline, Tab) easily cause configuration errors. If you want to check hidden characters in Value, please click Check Hidden Characters

Comment

\* Select Cluster  
☐ Environment ☒ Cluster  
☒ DEV default

然后点提交

然后点发布

Private properties [Expand/Collapse]

application Modified 1

Release Rollback Release History Authorize

Grayscale

Table Text Change History Instance List 0 Filter Synchronize Compare + Add Configuration

Tips: This namespace has never been released. Apollo client will not be able to fetch the configuration and will record 404 log information. Please release it in time.

| Release Status | Key ↑↓       | Value      | Comment | Last Modifier ↑↓ | Last Modified Time ↑↓ | Operation   |
|----------------|--------------|------------|---------|------------------|-----------------------|---|
| Unreleased     | test_key New | test_value |         | apollo           | 2019-11-24 16:44:38   | <input type="button" value="Edit"/> <input type="button" value="Delete"/> |

点这个发布

原文链接: [springboot|springboot 集成 apollo 做配置中心](#)

## 确认发布

Release (Only the released configurations can be fetched by clients, and this release will only be applied to the current environment: DEV) ×

| Changes | Key                       | Released values | Unreleased values | Modifier | Modified Time       |
|---------|---------------------------|-----------------|-------------------|----------|---------------------|
|         | test_key <span>New</span> |                 | test_value        | apollo   | 2019-11-24 16:44:38 |

\* Release Name

20191124164627-release

Comment

Add an optional extended description...

确认发布

Cancel

Release

## 测试获取配置

客户端的程序是基于<https://start.spring.io/>生成的最简的springboot程序

## 添加依赖

只需要添加这一个依赖就可以了

```
compile group: 'com.ctrip.framework.apollo', name: 'apollo-client', version: '1.5.1'
```

## 添加配置

在application.properties中添加

```
app.id=javastudy
apollo.meta=http://192.168.1.128:8080
```

## 单元测试代码

```
import com.ctrip.framework.apollo.Config;
import com.ctrip.framework.apollo.ConfigService;
import org.junit.jupiter.api.Test;
import org.springframework.boot.test.context.SpringBootTest;
```

```
@SpringBootTest
class DemoApplicationTests {
```

```
    @Test
    void contextLoads() {
```



```
Config appConfig = ConfigService.getAppConfig();
String property = appConfig.getProperty("test_key", "default_value");
System.out.println(property);
}

}
```

运行单元测试,即可看到相应的结果,由于mac下宿主机ping不通docker容器, 这里我就不给结果截图了, mac的同学可以把这个测试写到测试的controller中, 然后打成jar包, 把Jar包放到docker容器中,然后主机去访问这个docker里面的java程序就可以拿到相应的配置了.

## DEMO总评

使用Apollo的一个好处在于他可以做分布式配置管理,可以动态修改配置,可以与springboot深度集成. 此DEMO中虽然没有体现出来,大家可以多开几个docker:centos 机器为不同的环境来学习测试,加油!