



链滴

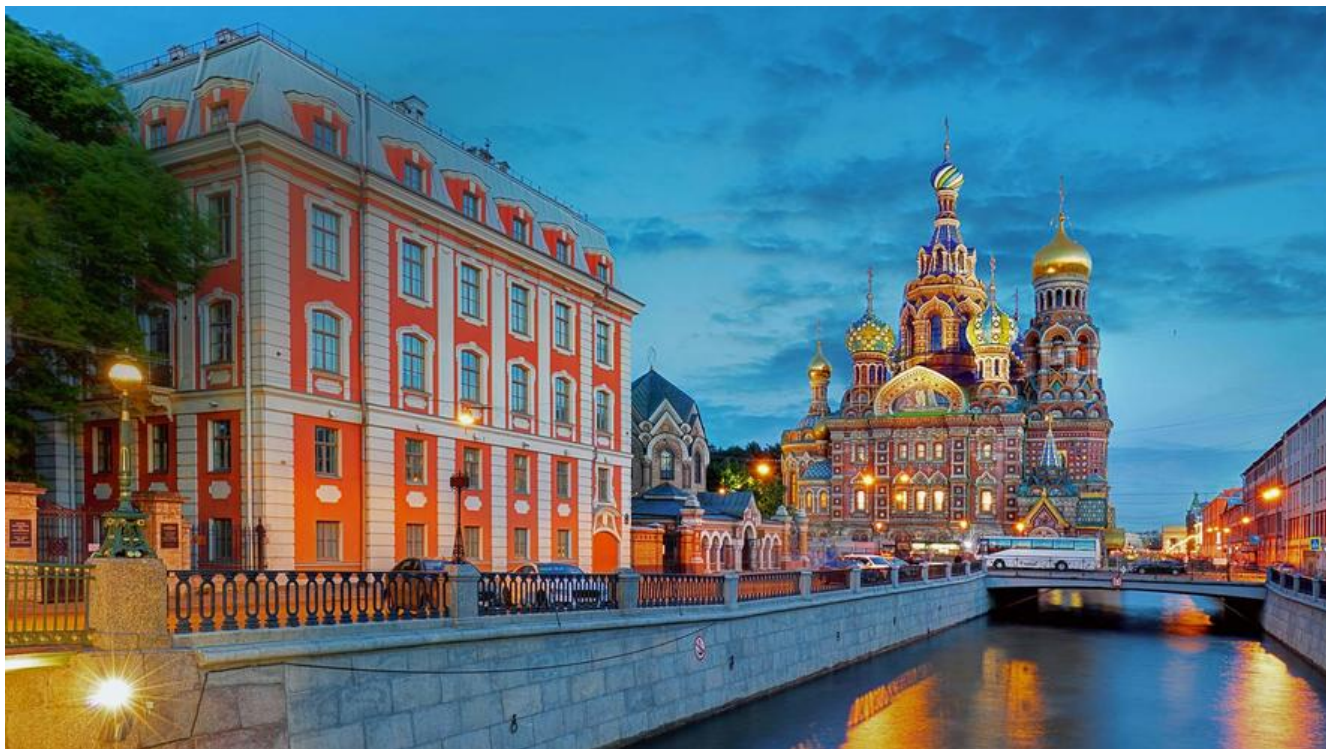
linux 下安装部署 apollo, 本地 springboot 整合 apollo 获取参数 (亲测成功)

作者: [LIUSHUYUAN](#)

原文链接: <https://ld246.com/article/1596617995688>

来源网站: [链滴](#)

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



1. linux部署apollo

1.1. 环境准备

jdk : 1.8+ mysql 5.6.5+

1.2. 下载上传到服务器

两种方式：

1.2.1. 下载源码自己编译（需要修改源码的可以选择）

<https://github.com/ctripcorp/apollo>

1.2.2. 下载官方编译好的

<https://github.com/ctripcorp/apollo/releases>

我选择的是第二种，本地下载这三个文件：

How to upgrade from v1.6.1 to v1.6.2

There is no schema change between v1.6.1 and v1.6.2
So simply deploy v1.6.2 executables with the following sequences:

1. apollo-configservice
2. apollo-adminservice
3. apollo-portal

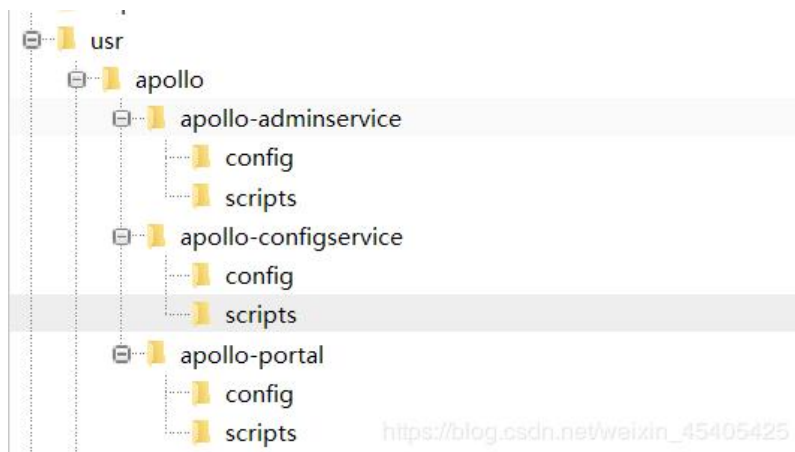
Assets 8

 apollo-adminservice-1.6.2-github.zip	49.6 MB
 apollo-adminservice-1.6.2-github.zip.sha1	41 Bytes
 apollo-configservice-1.6.2-github.zip	52.7 MB
 apollo-configservice-1.6.2-github.zip.sha1	41 Bytes
 apollo-portal-1.6.2-github.zip	37.3 MB
 apollo-portal-1.6.2-github.zip.sha1	41 Bytes
 Source code (zip)	
 Source code (tar.gz)	

https://blog.csdn.net/weixin_45405425

新建一个apollo目录，在apollo下新建apollo-adminservice, apollo-configservice, apollo-portal 个目录，将三个压缩包分别上传到对应的文件夹下面，分别解压后删除压缩包，解压后的目录结构：



解压命令：unzip xxx.zip



1.3. 下载sql文件，生成数据库

地址：<https://github.com/nobodyiam/apollo-build-scripts/tree/master/sql>

下载两个数据库文件并加载到自己的数据库上：

-  apolloconfigdb
-  apolloportaldb

1.4. 修改配置

1.4.1. 分别修改三个服务下的数据连接配置文件 /config/application-github.properties

```
# DataSource
spring.datasource.url = jdbc:mysql://139.198.114:3306/ApolloConfigDB?characterEncoding=utf8
spring.datasource.username = root
spring.datasource.password = Rc
```

修改自己的数据库端口和密码

原文链接：[linux 下安装部署 apollo, 本地 springboot 整合 apollo 获取参数 \(亲测成功\)](#)

1.4.2. 分别修改三个服务下的启动端口号配置文件 /scripts/startup.sh (可选)

三个服务默认端口号：

apollo-adminservice: 8090;

apollo-configservice: 8080;

apollo-portal: 8070;

我修改为：

apollo-adminservice: 8001;

apollo-configservice: 8002;

apollo-portal: 8003;

```
#!/bin/bash
SERVICE_NAME=apollo-adminservice
## Adjust log dir if necessary
LOG_DIR=/opt/logs/100003172
## Adjust server port if necessary
SERVER_PORT=${SERVER_PORT:=8001}
```

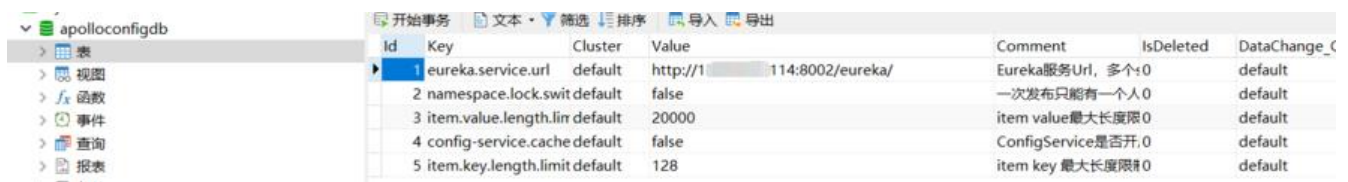
1.4.3. 修改apollo-portal服务的下的meta配置: apollo-portal/config/apollo-env.properties

这里的地址是apollo-configservice的服务地址，分别是不同环境下的服务地址，这里我只配置了（发-dev）环境下的地址。（服务器ip+端口）

```
1 local.meta=http://114.8002
2 dev.meta=http://114.8002
3 fat.meta=http://fill-in-fat-meta-server:8080
4 uat.meta=http://fill-in-uat-meta-server:8080
5 lpt.meta=${lpt_meta}
6 pro.meta=http://fill-in-pro-meta-server:8080
7
```

1.4.4. 修改数据库中的meta地址

修改apolloconfigdb数据库中serverconfig表中的eureka.service.url：其中的地址为apollo-configservice的服务地址：



Id	Key	Cluster	Value	Comment	IsDeleted	DataChange
1	eureka.service.url	default	http://114.8002/eureka/	Eureka服务Url, 多个:0	0	default
2	namespace.lock.swit	default	false	一次发布只能有一个人0	0	default
3	item.value.length.lir	default	20000	item value最大长度0	0	default
4	config-service.cache	default	false	ConfigService是否开0	0	default
5	item.key.length.limit	default	128	item key 最大长度限制0	0	default

1.5. 在apollo目录下编写启动start.sh和关闭shutdown.sh脚本

start.sh:

```
#!/bin/bash
/usr/apollo/apollo-configservice/scripts/startup.sh
/usr/apollo/apollo-adminservice/scripts/startup.sh
```

/usr/apollo/apollo-portal/scripts/startup.sh

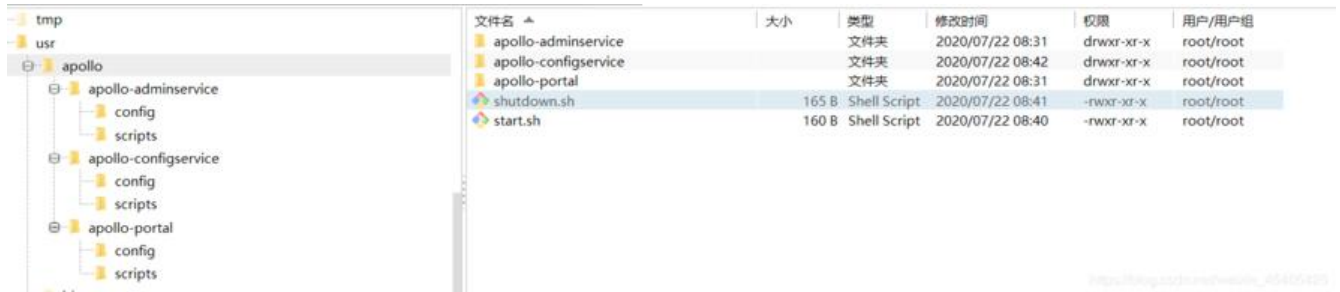
shutdown.sh

#!/bin/bash

/usr/apollo/apollo-adminservice/scripts/shutdown.sh

/usr/apollo/apollo-configservice/scripts/shutdown.sh

/usr/apollo/apollo-portal/scripts/shutdown.sh



1.6.启动服务访问apollo

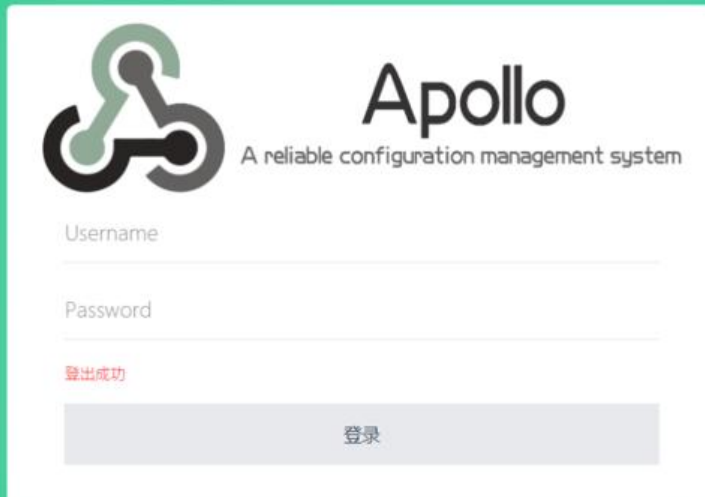
运行start.sh, 启动三个服务后: 输入如下地址

<http://自己的ip地址:8003/>

这是portal的服务地址 (注意自己修改的端口号)

```
[root@lsvr apollo]# ./start.sh
2020年 07月 22日 星期三 09:33:38 CST ==== Starting ====
Started [6129]
Waiting for server startup.....
2020年 07月 22日 星期三 09:34:31 CST Server started in 50 seconds!
2020年 07月 22日 星期三 09:34:31 CST ==== Starting ====
Started [7294]
Waiting for server startup.....
2020年 07月 22日 星期三 09:35:28 CST Server started in 55 seconds!
2020年 07月 22日 星期三 09:35:28 CST ==== Starting ====
Started [8595]
Waiting for server startup.....
2020年 07月 22日 星期三 09:36:05 CST Server started in 35 seconds!
```

账号: apollo, 密码: admin;



https://blog.csdn.net/weixin_45405425

2.springBoot整合apollo并获取参数

2.1.项目引入apollo依赖

```
<dependency>
  <groupId>com.ctrip.framework.apollo</groupId>
  <artifactId>apollo-client</artifactId>
  <version>1.4.0</version>
</dependency>
```

2.2.apollo发布配置

新建项目：



https://blog.csdn.net/weixin_45405425

创建项目

* 部门: 样例部门1(TEST1)

* AppId: apollo-test
(应用唯一标识)

* 应用名称: apollo测试
(建议格式 xx-yy-zz 例apollo-server)

* 应用负责人: apollo | apollo

项目管理员: apollo | apollo
(应用负责人默认具有项目管理员权限, 项目管理员可以创建Namespace和集群、分配用户权限)

[提交](#)

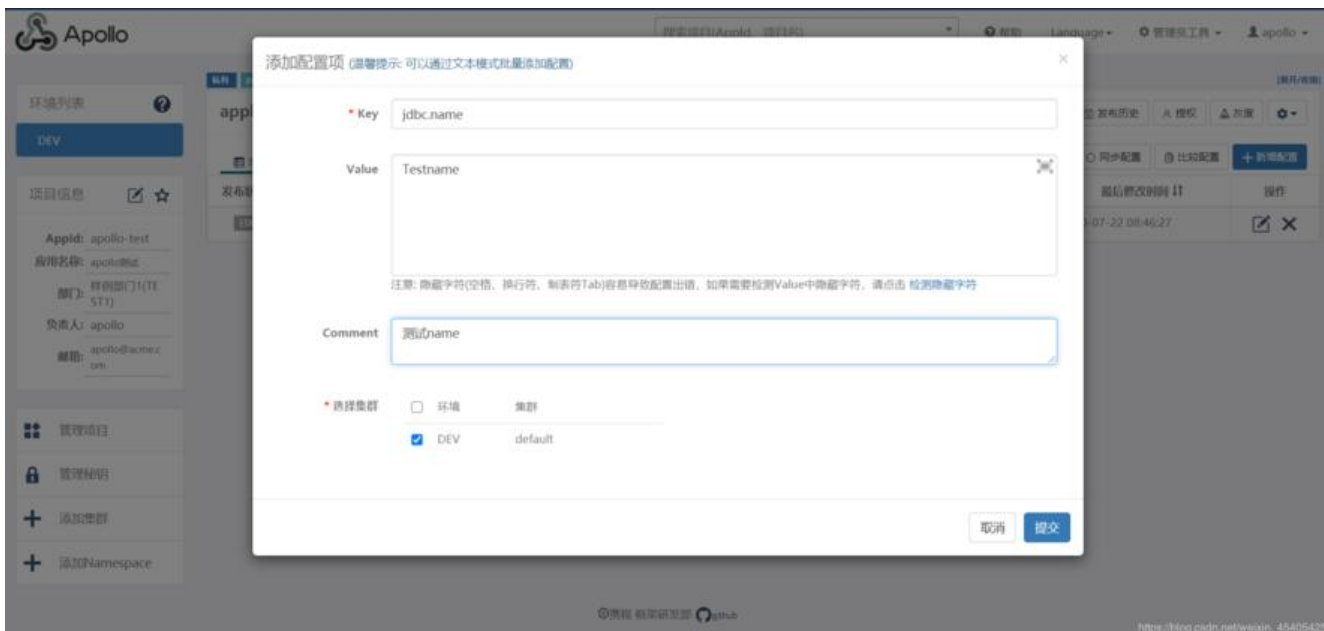
点击新增配置:

The screenshot shows the Apollo configuration center interface. The main content area displays the configuration for an application named 'application'. A table lists the configurations:

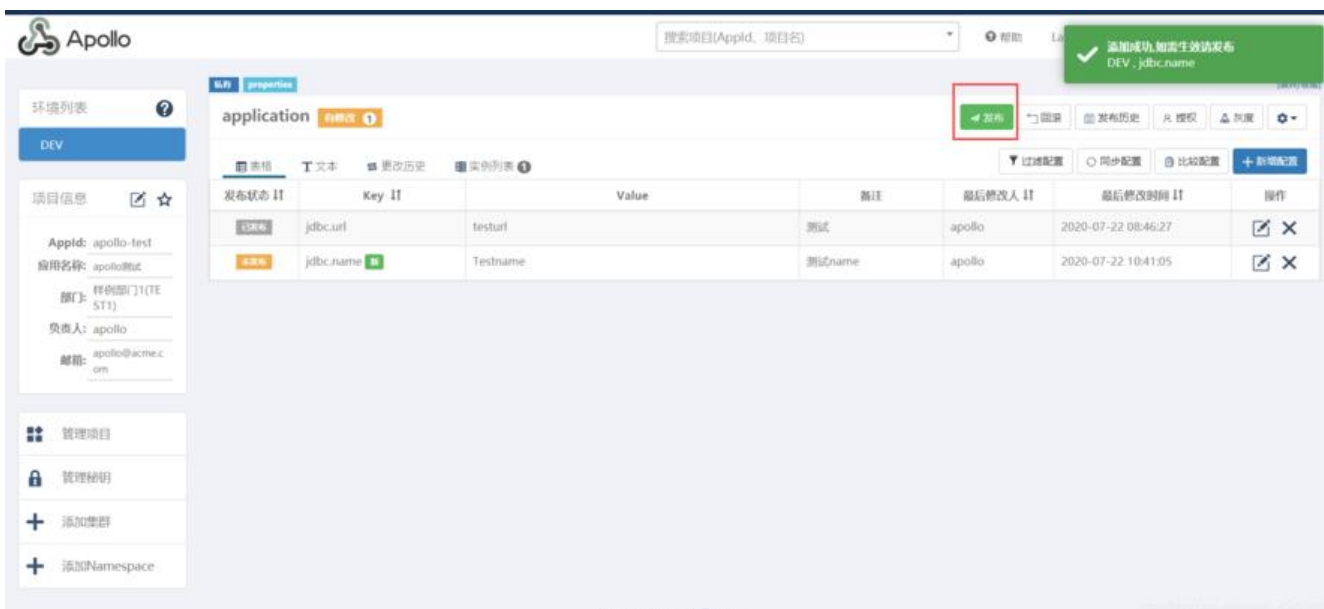
发布状态	Key	Value	备注	最后修改人	最后修改时间	操作
回滚	jdbc.url	testurl	测试	apollo	2020-07-22 08:46:27	编辑 删除

A red box highlights the '+ 新增配置' button in the top right corner of the configuration table.

填写配置点击提交:



点击发布：



发布 (只有发布过的配置才会被客户端获取到, 此次发布只会作用于当前环境:DEV)

Changes	Key	发布的值	未发布的值	修改人	修改时间
	jdbc.name 新		Testname	apollo	2020-07-22 10:41:05

* Release Name

Comment

取消

2.2添加eureka注册地址（网络策略）

分布式部署的时候，本地访问服务器上的Apollo，apollo-configservice和apollo-adminservice需把自己的IP和端口注册到Meta Server（apollo-configservice本身）。

参考官官方给的解决方案：

<https://github.com/ctripcorp/apollo/wiki/%E5%88%86%E5%B8%83%E5%BC%8F%E9%83%A%E7%BD%B2%E6%8C%87%E5%8D%97>

1.4 网络策略

分布式部署的时候，apollo-configservice 和 apollo-adminservice 需要把自己的IP和端口注册到Meta Server (apollo-configservice本身)。

Apollo客户端和Portal会从Meta Server获取服务的地址 (IP+端口)，然后通过服务地址直接访问。

需要注意的是，apollo-configservice 和 apollo-adminservice 是基于内网可信网络设计的，所以出于安全考虑，请不要将apollo-configservice 和 apollo-adminservice 直接暴露在公网。

所以如果实际部署的机器有多块网卡 (如docker)，或者存在某些网卡的IP是Apollo客户端和Portal无法访问的 (如网络安全限制)，那么我们就需要在 apollo-configservice 和 apollo-adminservice 中做相关限制以避免Eureka将这些网卡的IP注册到Meta Server。

具体文档可以参考Ignore Network Interfaces章节。具体而言，就是分别编辑apollo-configservice/src/main/resources/application.yml和apollo-adminservice/src/main/resources/application.yml，然后把需要忽略的网卡加进去。

如下面这个例子就是对于 apollo-configservice，把docker0和veth.*的网卡在注册到Eureka时忽略掉。

```
spring:
  application:
    name: apollo-configservice
  profiles:
    active: ${apollo_profile}
  cloud:
    inetutils:
      ignoredInterfaces:
        - docker0
        - veth.*
```

注意，对于application.yml修改时要小心，千万不要把其它信息改错了，如spring.application.name等。

另外一种方式是直接指定要注册的IP，可以修改startup.sh，通过JVM System Property在运行时传入，如 -Deureka.instance.ip-address=\${指定的IP}，也可以通过OS Environment Variable，如 EUREKA_INSTANCE_IP_ADDRESS=\${指定的IP}，或者也可以修改apollo-adminservice或apollo-configservice 的bootstrap.yml文件，加入以下配置

```
eureka:
  instance:
    ip-address: ${指定的IP}
```

最后一种方式是直接指定要注册的IP+PORT，可以修改startup.sh，通过JVM System Property在运行时传入，如 -Deureka.instance.homePageUrl=http://\${指定的IP}:\${指定的Port}，也可以通过OS Environment Variable，如 EUREKA_INSTANCE_HOME_PAGE_URL=http://\${指定的IP}:\${指定的Port}，或者也可以修改apollo-adminservice或apollo-configservice 的bootstrap.yml文件，加入以下配置

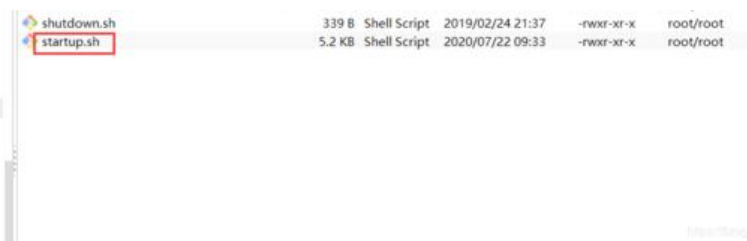
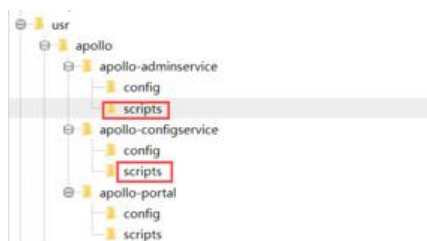
```
eureka:
  instance:
    homePageUrl: http://${指定的IP}:${指定的Port}
    preferIpAddress: false
```

做完上述修改并重启后，可以查看Eureka页面 (http://\${config-service-url:port}) 检查注册上来的IP信息是否正确。

如果Apollo部署在公有云上，本地开发环境无法连接，但又需要做开发测试的话，客户端可以升级到0.11.0版本及以上，然后配置跳过Apollo Meta Server服务发现

https://blog.codn.net/wordpress_46405625

解决步骤:



https://blog.codn.net/wordpress_46405625

原文链接: [linux 下安装部署 apollo, 本地 springboot 整合 apollo 获取参数 \(亲测成功\)](#)

在服务器下的admin和config服务的startup.sh分别添加:

-Deureka.instance.ip-address=服务器ip地址

添加在export JAVA_OPTS的下面:

```
#!/bin/bash
SERVICE_NAME=apollo-adminservice
## Adjust log dir if necessary
LOG_DIR=/opt/logs/100003172
## Adjust server port if necessary
SERVER_PORT=${SERVER_PORT:-8001}

## Create log directory if not existed because JDK 8+ won't do that
mkdir -p $LOG_DIR

## Adjust memory settings if necessary
export JAVA_OPTS="-Xms2560m -Xmx2560m -Xss256k -XX:MetaspaceSize=128m -XX:MaxMetaspaceSize=384m -XX:NewSize=1536m -XX:MaxNewSize=1536m -XX:SurvivorRatio=8"

## Only uncomment the following when you are using server jvm
export JAVA_OPTS="$JAVA_OPTS -server -XX:-ReduceInitialCardMarks"

##### The following is the same for configservice_adminservice_port! #####
export JAVA_OPTS="$JAVA_OPTS
-Deureka.instance.ip-address=135.149.114.114 -XX:ParallelGCThreads=4 -XX:MaxTenuringThreshold=9 -XX:+DisableExplicitGC -XX:+ScavengeBeforeFullGC -XX:SoftRefLRUPolicyMSPerMB=0 -XX
:+HeapDumpOnOutOfMemoryError -XX:-OmitStackTraceInFastThrow -Duser.timezone=Asia/Shanghai -Dclient.encoding.override=UTF-8 -Dfile.encoding=UTF-8 -Djava.security.egd=file:/dev/./
# DataSource URL USERNAME PASSWORD
if [ "$DS_URL" != "" ]
then
export JAVA_OPTS="$JAVA_OPTS -Dspring.datasource.url=$DS_URL -Dspring.datasource.username=$DS_USERNAME -Dspring.datasource.password=$DS_PASSWORD"
fi
export JAVA_OPTS="$JAVA_OPTS -Dserver.port=$SERVER_PORT -Dlogging.file=$LOG_DIR/$SERVICE_NAME.log -XX:HeapDumpPath=$LOG_DIR/HeapDumpOnOutOfMemoryError/"
PATH_TO_JAR=$SERVICE_NAME.jar
SERVER_URL="http://localhost:$SERVER_PORT"

function checkPidAlive {
for i in `ls -t $SERVICE_NAME*.pid 2>/dev/null`

```

添加之后, 重启三个服务, 用之前写好的shutdown.sh关闭, start.sh启动。

2.3.application.yml

```
spring:
  application:
    name: apollo-test
```

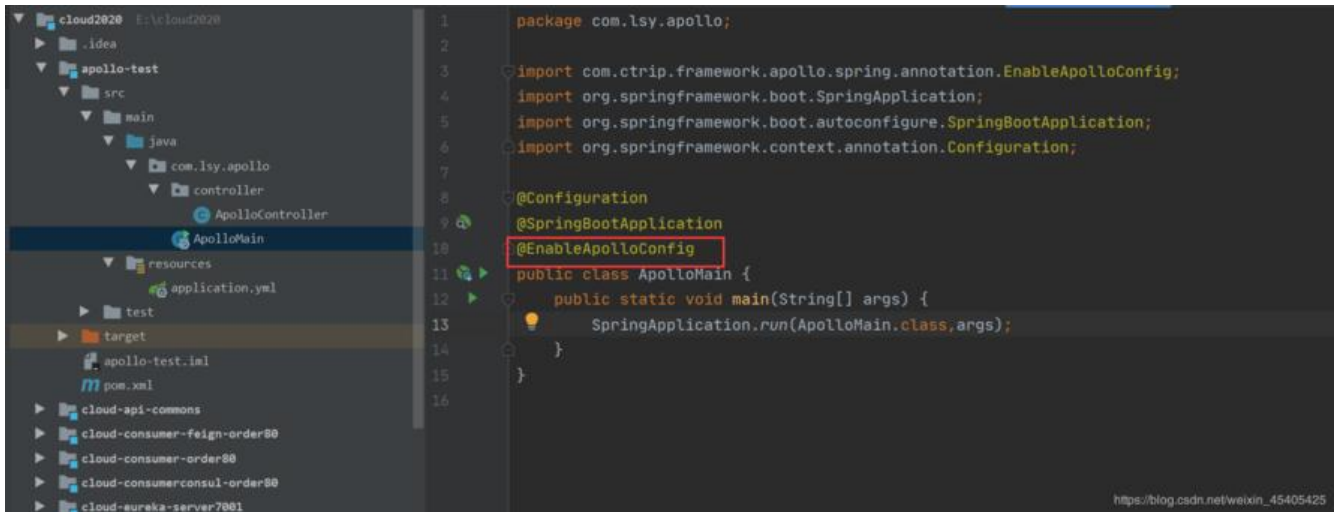
```
server:
  port: 80
```

```
#apollo相关配置
# apollo 相关配置
```

```
app:
  id: apollo-test # 与 Apollo 配置中心中的 AppId 一致
```

```
apollo:
  meta: http://自己服务器ip地址:8002 # Apollo 中的 Eureka 注册中心地址
  #cluster: # 指定 Apollo 集群, 默认为 default, 相同集群实例使用对应集群的配置
  #cacheDir: # 配置缓存目录, 网络不可用时任然可提供配置服务
  bootstrap:
    enable: true # 启用 apollo
```

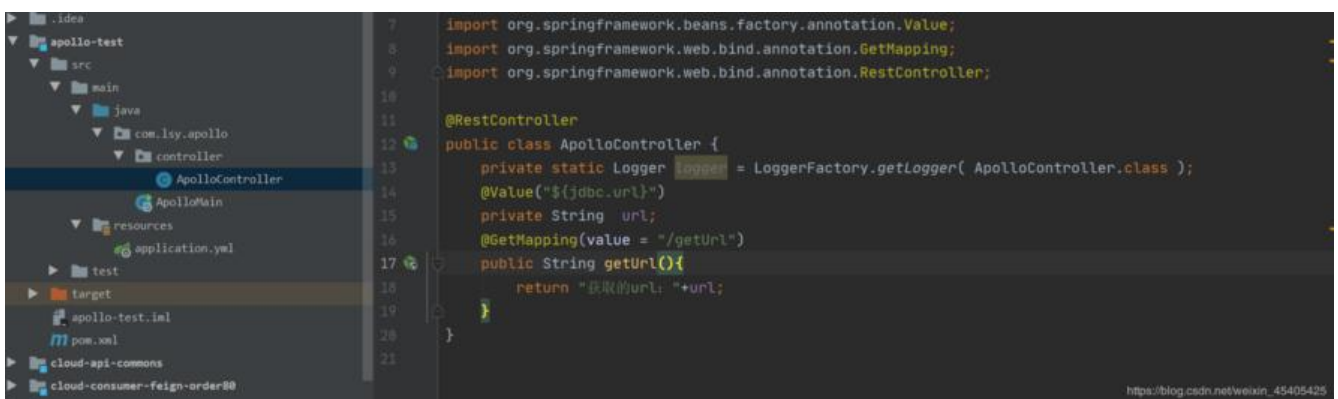
2.4.启动类添加@EnableApolloConfig注解:



```
1 package com.lsy.apollo;
2
3 import com.ctrip.framework.apollo.spring.annotation.EnableApolloConfig;
4 import org.springframework.boot.SpringApplication;
5 import org.springframework.boot.autoconfigure.SpringBootApplication;
6 import org.springframework.context.annotation.Configuration;
7
8 @Configuration
9 @SpringBootApplication
10 @EnableApolloConfig
11 public class ApolloMain {
12     public static void main(String[] args) {
13         SpringApplication.run(ApolloMain.class,args);
14     }
15 }
16
```

https://blog.csdn.net/weixin_45405425

2.5.controller获取:



```
7 import org.springframework.beans.factory.annotation.Value;
8 import org.springframework.web.bind.annotation.GetMapping;
9 import org.springframework.web.bind.annotation.RestController;
10
11 @RestController
12 public class ApolloController {
13     private static Logger logger = LoggerFactory.getLogger( ApolloController.class );
14     @Value("${jdbc.url}")
15     private String url;
16     @GetMapping(value = "/getUrl")
17     public String getUrl(){
18         return "获取的url: "+url;
19     }
20 }
21
```

https://blog.csdn.net/weixin_45405425

访问得到aplo配置:



获取的url: testurl

https://blog.csdn.net/weixin_45405425