开源项目：Apollo

简介

产生原因

- 随着程序功能的日益复杂，程序的配置日益增多：各种功能的开关、参数的配置、服务器的地址等。
- 对程序配置的期望值也越来越高：配置修改后实时生效，分环境、分集群管理配置，完善的权限、审核机制等
- 在这样的大环境下，传统的通过配置文件、数据库等方式已经越来越无法满足开发人员对配置管理的需求。

简介

Apollo（阿波罗）是携程框架部门研发的配置管理平台，能够集中化管理应用不同环境、不同集群的配置，配置修改后能够实时推送到应用端，并且具备规范的权限、流程治理等特性。
服务端基于Spring Boot和Spring Cloud开发，打包后可以直接运行，不需要额外安装Tomcat等应用容器。
Java客户端不依赖任何框架，能够运行于所有Java运行时环境，同时对Spring环境也有较好的支持。.Net客户端不依赖任何框架，能够运行于所有.Net运行时环境。

开源项目：Apollo

简介

产生原因

- 随着程序功能的日益复杂，程序的配置日益增多：各种功能的开关、参数的配置、服务器的地址等。
- 对程序配置的期望值也越来越高：配置修改后实时生效，分环境、分集群管理配置，完善的权限、审核机制等
- 在这样的大环境下，传统的通过配置文件、数据库等方式已经越来越无法满足开发人员对配置管理的需求。

简介

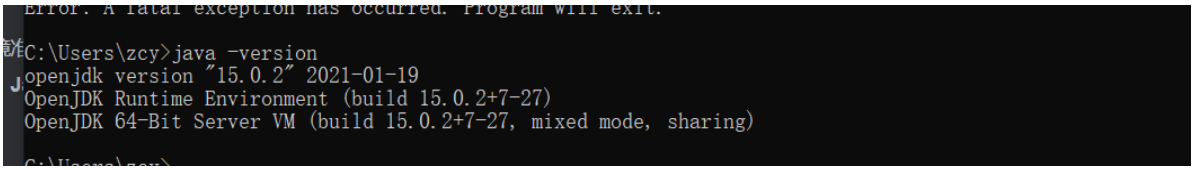
Apollo（阿波罗）是携程框架部门研发的配置管理平台，能够集中化管理应用不同环境、不同集群的配置，配置修改后能够实时推送到应用端，并且具备规范的权限、流程治理等特性。
服务端基于Spring Boot和Spring Cloud开发，打包后可以直接运行，不需要额外安装Tomcat等应用容器。
Java客户端不依赖任何框架，能够运行于所有Java运行时环境，同时对Spring环境也有较好的支持。.Net客户端不依赖任何框架，能够运行于所有.Net运行时环境。

部署

环境准备

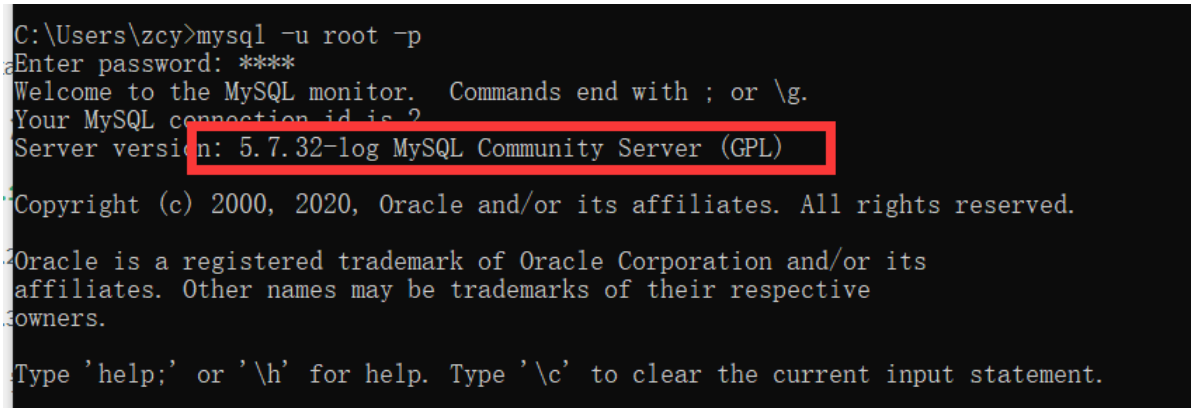
Java

Java1.8以上。



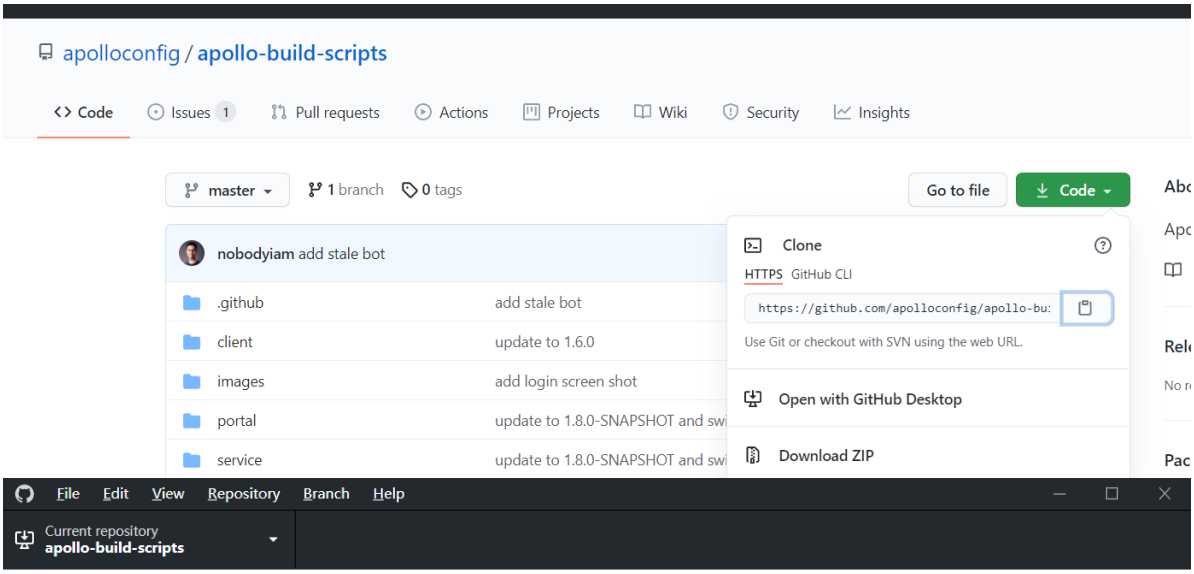
mySQL

5.6以上



Apollo-quick-start 下载

仓库下载

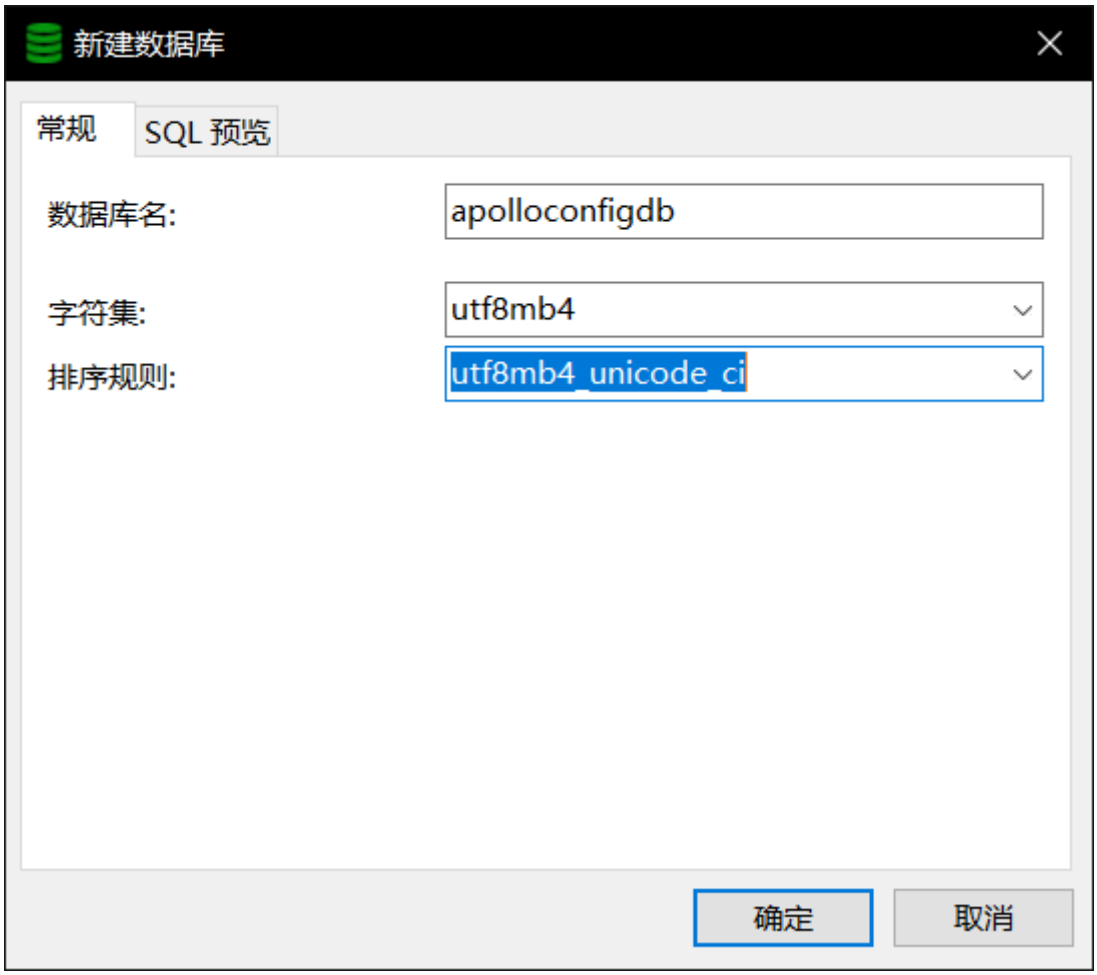

apolloconfig / apollo-build-scripts
Code
Issues 1
Pull requests
Actions
Projects
Wiki
Security
Insights
master
1 branch
0 tags
Go to file
Code
nobodyiam add stale bot
.github
add stale bot
client
update to 1.6.0
images
add login screen shot
portal
service
Clone
HTTPS GitHub CLI
https://github.com/apolloconfig/apollo-bu:
Use Git or checkout with SVN using the web URL.
Open with GitHub Desktop
Download ZIP

File
Edit
View
Repository
Branch
Help
Current repository
apollo-build-scripts

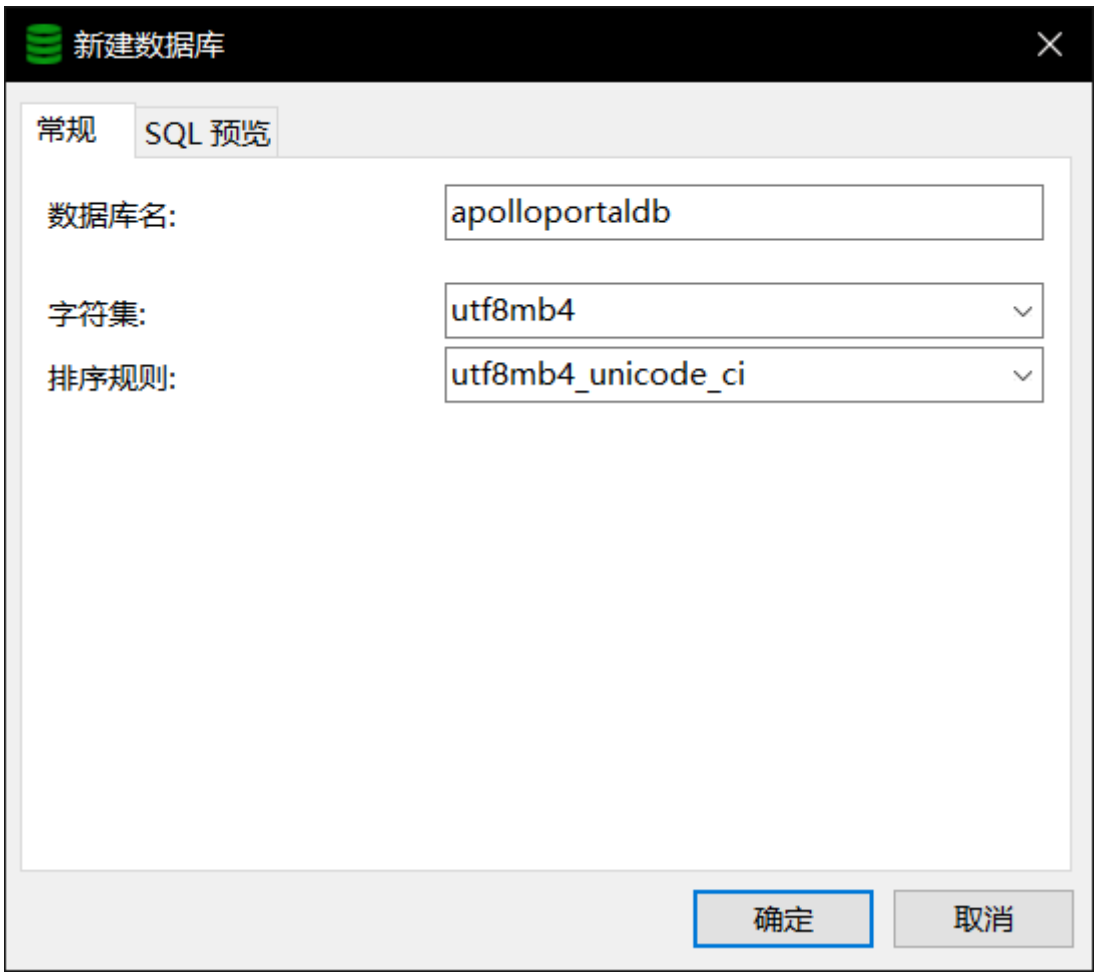

Cloning apollo-build-scripts
Cloning into 'E:\MyJavaProject\apolloquickstart\apollo-build-scripts'...

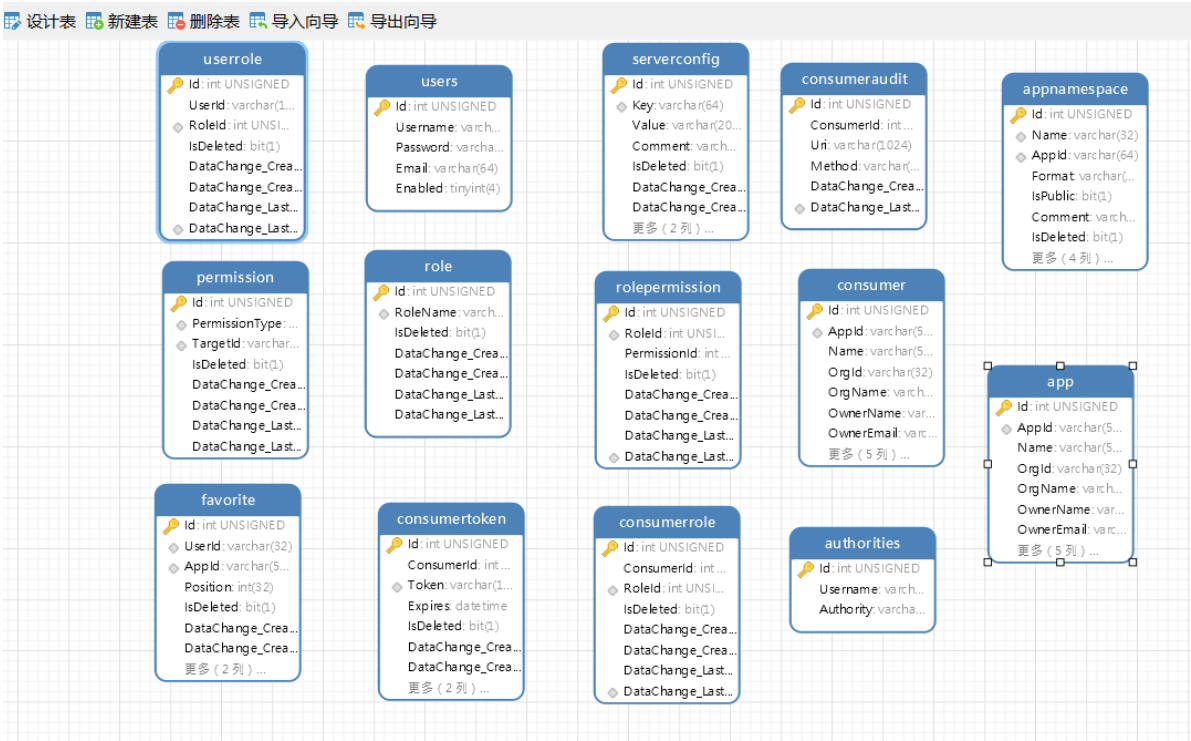
创建数据库

创建ApolloPortalDB



创建ApolloConfigDB





启动Apollo配置中心

确定端口未占用

Quick Start脚本会在本地启动3个服务，分别使用8070, 8080, 8090端口，请确保这3个端口当前没有被使用。

```
C:\Users\zcy>
C:\Users\zcy>netstat -aon|findstr "8080"

C:\Users\zcy>netstat -aon|findstr "8070"

C:\Users\zcy>netstat -aon|findstr "8090"
```

3个端口均为占用

执行./sh文件，发现不能执行sh文件

处理方法：

- 在windows下想要执行shell脚本，需要使用到"Git Bash"，所以我们需要先安装Git。
- 配置Git环境变量

但是最终启动失败

```
zcy@DESKTOP-OF11RIE MINGW64 /e/MyJavaProject/apolloquickstart/apollo-build-scrip
ts (master)
$ ./demo.sh start
Windows new JAVA_HOME is: /c/Users/zcy/JDKS~1/OPENJD~1.2

==== starting service ====
Service logging file is ./service/apollo-service.log
Started [1498]
Waiting for config service startup.....................
Config service failed to start in 120 seconds! Please check ./service/apollo-service.log for more information.

zcy@DESKTOP-OF11RIE MINGW64 /e/MyJavaProject/apolloquickstart/apollo-build-scripts (master)
$

zcy@DESKTOP-OF11RIE MINGW64 /e/MyJavaProject/apolloquickstart/apollo-build-scripts (master)
$
```

查看Apollo-service.log 日志后，发现

```
2021-07-23 21:43:28.235  INFO 4100 --- [main] o.hibernate.jpa.internal.util.LogHelper  : HHH000204: Processing PersistenceUnitInfo [name: default]
2021-07-23 21:43:28.364  INFO 4100 --- [main] org.hibernate.Version                    : HHH000412: Hibernate ORM core version 5.4.27.Final
2021-07-23 21:43:28.648  INFO 4100 --- [main] o.hibernate.annotations.common.Version   : HCANN000001: Hibernate Commons Annotations {5.1.2.Final}
2021-07-23 21:43:28.856  INFO 4100 --- [main] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Starting...
2021-07-23 21:43:29.885 ERROR 4100 --- [main] com.zaxxer.hikari.pool.HikariPool        : HikariPool-1 - Exception during pool initialization.

java.sql.SQLException: Access denied for user 'root'@'localhost' (using password: NO)
        at com.mysql.cj.jdbc.exceptions.SQLError.createSQLException(SQLError.java:129)
        at com.mysql.cj.jdbc.exceptions.SQLError.createSQLException(SQLError.java:97)
        at com.mysql.cj.jdbc.exceptions.SQLExceptionsMapping.translateException(SQLExceptionsMapping.java:122)
        at com.mysql.cj.jdbc.ConnectionImpl.createNewIO(ConnectionImpl.java:835)
        at com.mysql.cj.jdbc.ConnectionImpl.<init>(ConnectionImpl.java:455)
        at com.mysql.cj.jdbc.ConnectionImpl.getInstance(ConnectionImpl.java:240)
        at com.mysql.cj.jdbc.NonRegisteringDriver.connect(NonRegisteringDriver.java:199)
        at com.zaxxer.hikari.util.DriverDataSource.getConnection(DriverDataSource.java:138)
        at com.zaxxer.hikari.pool.PoolBase.newConnection(PoolBase.java:358)
        at com.zaxxer.hikari.pool.PoolBase.newPoolEntry(PoolBase.java:206)
        at com.zaxxer.hikari.pool.HikariPool.createPoolEntry(HikariPool.java:477)
        at com.zaxxer.hikari.pool.HikariPool.checkFailFast(HikariPool.java:560)
        at com.zaxxer.hikari.pool.HikariPool.<init>(HikariPool.java:115)
        at com.zaxxer.hikari.HikariDataSource.getConnection(HikariDataSource.java:112)
        at org.hibernate.engine.jdbc.connections.internal.DatasourceConnectionProviderImpl.getConnection(DatasourceConnectionProviderImpl.java:122)
        at org.hibernate.engine.jdbc.env.internal.JdbcEnvironmentInitiator$ConnectionProviderJdbcConnectionAccess.obtainConnection(JdbcEnvironmentInitiator.java:180)
```

MySQL相关配置不正确：

修改后如下：

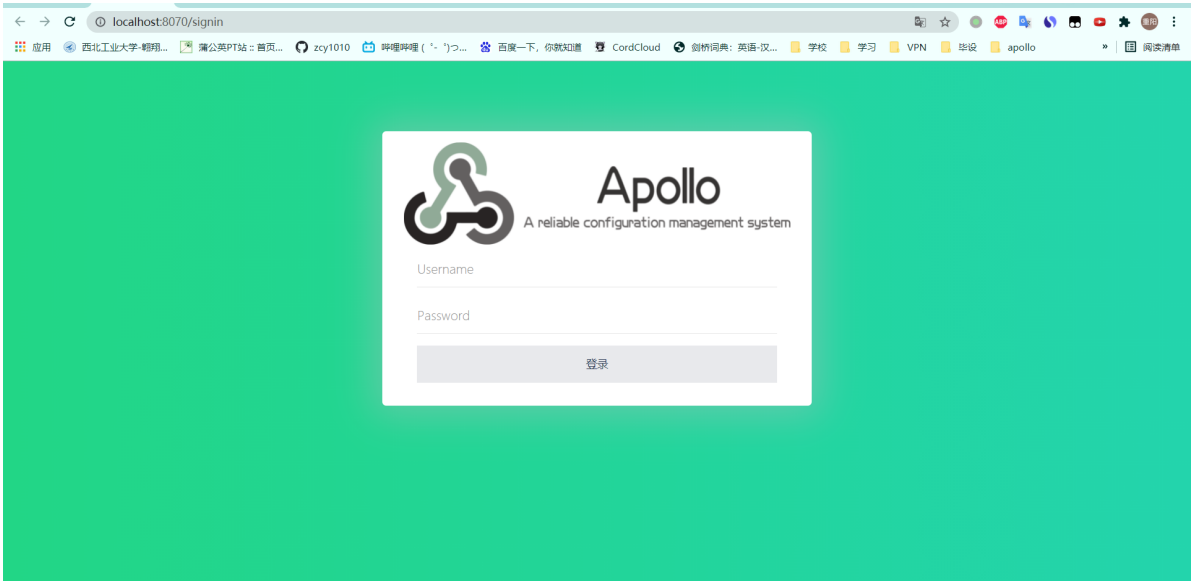
```
: > MyJavaProject > apolloquickstart > apollo-build-scripts > demo.sh
  1   #!/bin/bash
  2
  3   # apollo config db info
  4   apollo_config_db_url="jdbc:mysql://localhost:3306/ApolloConfigDB?characterEncoding=utf8
  5   apollo_config_db_username=root
  6   apollo_config_db_password=root
  7
  8   # apollo portal db info
  9   apollo_portal_db_url="jdbc:mysql://localhost:3306/ApolloPortalDB?characterEncoding=utf8
 10   apollo_portal_db_username=root
 11   apollo_portal_db_password=root
 12
```

启动成功：

```
zcy@DESKTOP-OF11RIE MINGW64 /e/MyJavaProject/apolloquickstart/apollo-build-scripts (master)
$ ./demo.sh start
Windows new JAVA_HOME is: /c/Users/zcy/JDKS~1/OPENJD~1.2
==== starting service ====
Service logging file is ./service/apollo-service.log
Started [1640]
Waiting for config service startup.....
Config service started. You may visit http://localhost:8080 for service status now!
Waiting for admin service startup
Admin service started
==== starting portal ====
Portal logging file is ./portal/apollo-portal.log
Started [1688]
Waiting for portal startup...
Portal started. You can visit http://localhost:8070 now!
```

成功启动：

1. 输入用户名apollo，密码admin后登录



启动客户端程序

准备了一个简单的Demo客户端来演示从Apollo配置中心获取配置。

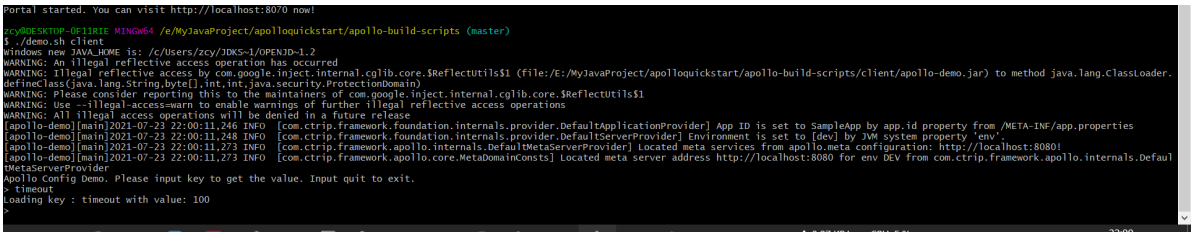
程序很简单，就是用户输入一个key的名字，程序会输出这个key对应的值。

如果没找到这个key，则输出undefined。

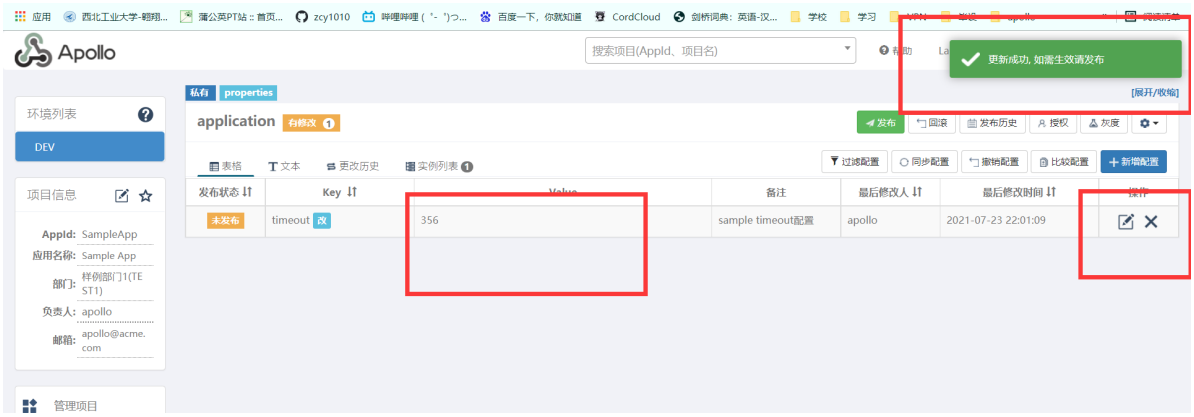
同时，客户端还会监听配置变化事件，一旦有变化就会输出变化的配置信息。

运行`./demo.sh client`启动Demo客户端，忽略前面的调试信息，可以看到如下提示：

输入`timeout`，会看到如下信息：



在配置界面点击timeout这一项的编辑按钮



点击发布按钮，并填写发布信息

客户端一直在运行的话，在配置发布后就会监听到配置变化，并输出修改的配置信息：

```
[apollo-demo][main]2021-07-23 22:00:11,273 INFO  [com.ctrip.framework.apollo
tMetaServerProvider
Apollo Config Demo. Please input key to get the value. Input quit to exit.
> timeout
Loading key : timeout with value: 100
> Changes for namespace application
Change - key: timeout, oldValue: 100, newValue: 356, changeType: MODIFIED
```

再次输入`timeout`查看对应的值，会看到如下信息：

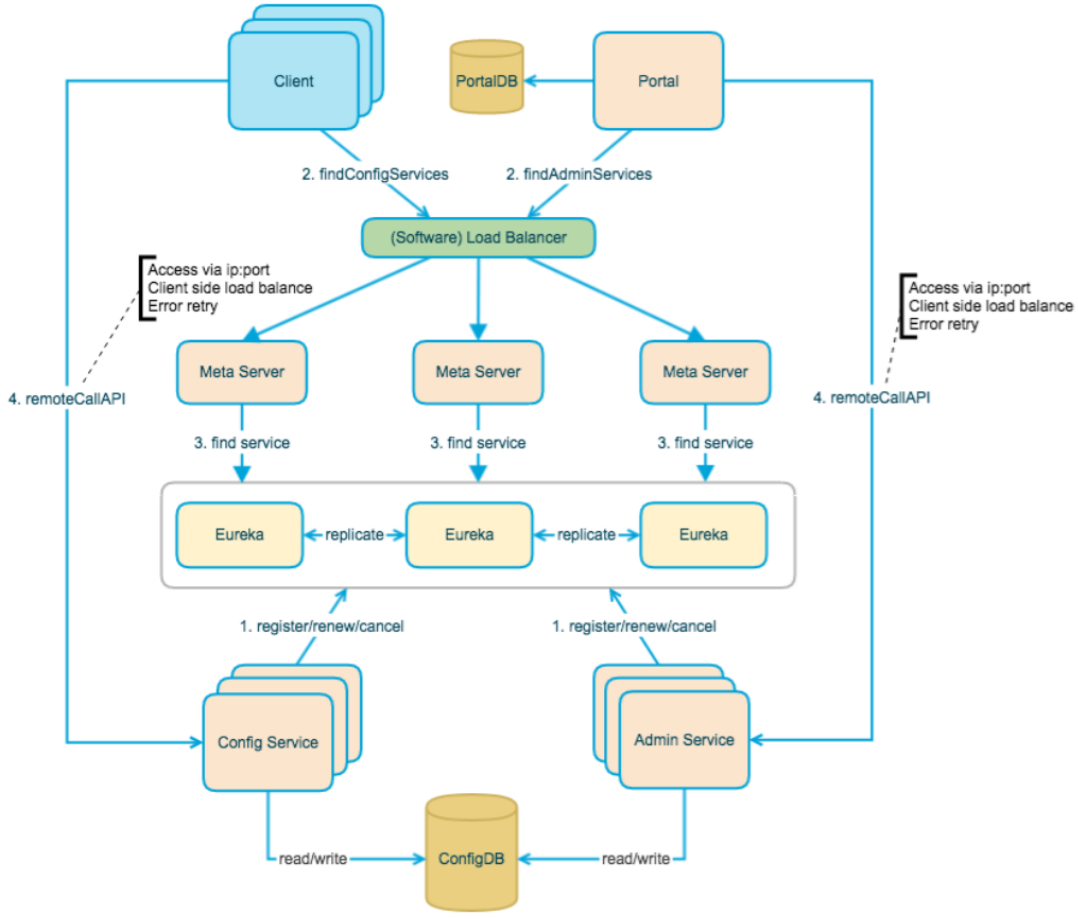
```
[apollo-demo][main]2021-07-23 22:00:11,273 INFO  [com.ctrip.framework.apollo.core.MetaDomainConsts] L
tMetaServerProvider
Apollo Config Demo. Please input key to get the value. Input quit to exit.
> timeout
Loading key : timeout with value: 100
> Changes for namespace application
Change - key: timeout, oldValue: 100, newValue: 356, changeType: MODIFIED

> timeout
Loading key : timeout with value: 356
>
```

接入新的apollo

//TODO

Apollo 框架介绍

下图是Apollo的作者宋顺给出的架构图：



四个核心模块及其主要功能

1. **ConfigService**
    - 提供配置获取接口
    - 提供配置推送接口
    - 服务于Apollo客户端
2. **AdminService**
    - 提供配置管理接口
    - 提供配置修改发布接口
    - 服务于管理界面Portal
3. **Client**
    - 为应用获取配置，支持实时更新
    - 通过MetaServer获取ConfigService的服务列表
    - 使用客户端软负载SLB方式调用ConfigService
4. **Portal**
    - 配置管理界面
    - 通过MetaServer获取AdminService的服务列表
    - 使用客户端软负载SLB方式调用AdminService

三个辅助服务发现模块

1. **Eureka**
    - 用于服务发现和注册
    - Config/AdminService注册实例并定期报心跳
    - 和ConfigService住在一起部署
2. **MetaServer**
    - Portal通过域名访问MetaServer获取AdminService的地址列表
    - Client通过域名访问MetaServer获取ConfigService的地址列表
    - 相当于一个Eureka Proxy
    - 逻辑角色，和ConfigService住在一起部署
3. **NginxLB**
    - 和域名系统配合，协助Portal访问MetaServer获取AdminService地址列表
    - 和域名系统配合，协助Client访问MetaServer获取ConfigService地址列表
    - 和域名系统配合，协助用户访问Portal进行配置管理

更详细的介绍可见https://mp.weixin.qq.com/s/-hUaQPzfsl9Lm3IqQW3VDQ