

Quick Start

[Jump to bottom](#)

Jason Song edited this page 18 days ago · 55 revisions

- [一、准备工作](#)
- [二、安装步骤](#)
- [三、启动Apollo配置中心](#)
- [四、使用Apollo配置中心](#)

为了让大家更快的上手了解Apollo配置中心，我们这里准备了一个Quick Start，能够在几分钟内在本地环境部署、启动Apollo配置中心。

考虑到Docker的便捷性，我们还提供了Quick Start的Docker版本，如果你对Docker比较熟悉的话，可以参考[Apollo Quick Start Docker部署](#)通过Docker快速部署Apollo。

不过这里需要注意的是，Quick Start只针对本地测试使用，如果要部署到生产环境，还请另行参考[分布式部署指南](#)。

注：Quick Start需要有bash环境，Windows用户建议安装[Git Bash](#)，或者也可以直接通过IDE环境启动，详见[Apollo开发指南](#)。

一、准备工作

1.1 Java

- Apollo服务端：1.8+
- Apollo客户端：1.7+

由于Quick Start会在本地同时启动服务端和客户端，所以需要在本地安装Java 1.8+。

在配置好后，可以通过如下命令检查：

```
java -version
```

样例输出：

```
java version "1.8.0_74"  
Java(TM) SE Runtime Environment (build 1.8.0_74-b02)  
Java HotSpot(TM) 64-Bit Server VM (build 25.74-b02, mixed mode)
```

1.2 MySQL

- 版本要求：5.6.5+

Apollo的表结构对 timestamp 使用了多个default声明，所以需要5.6.5以上版本。

连接上MySQL后，可以通过如下命令检查：

```
SHOW VARIABLES WHERE Variable_name = 'version';
```

Variable_name	Value
version	5.7.11

1.3 下载Quick Start安装包

我们准备好了一个Quick Start安装包，大家只需要下载到本地，就可以直接使用，免去了编译、打包过程。

安装包共50M，如果访问github网速不给力的话，可以从百度网盘下载。

1. 从Github下载
 - checkout或下载[apollo-build-scripts项目](#)
 - **由于Quick Start项目比较大，所以放在了另外的repository，请注意项目地址**
 - <https://github.com/nobodyiam/apollo-build-scripts>
2. 从百度网盘下载
 - 通过[网盘链接](#)下载（提取码: 8eh3）
 - 下载到本地后，在本地解压apollo-quick-start.zip
3. 为啥安装包要58M这么大？
 - 因为这是一个可以自启动的jar包，里面包含了所有依赖jar包以及一个内置的tomcat容器

1.3.1 手动打包Quick Start安装包

Quick Start只针对本地测试使用，所以一般用户不需要自己下载源码打包，只需要下载已经打好的包即可。不过也有部分用户希望在修改代码后重新打包，那么可以参考如下步骤：

1. 修改apollo-configservice, apollo-adminservice和apollo-portal的pom.xml，注释掉spring-boot-maven-plugin和maven-assembly-plugin
2. 在根目录下执行 `mvn clean package -pl apollo-assembly -am -DskipTests=true`
3. 复制apollo-assembly/target下的jar包，rename为apollo-all-in-one.jar

二、安装步骤

2.1 创建数据库

Apollo服务端共需要两个数据库：ApolloPortalDB 和 ApolloConfigDB，我们把数据库、表的创建和样例数据都分别准备了sql文件，只需要导入数据库即可。

注意：如果你本地已经创建过Apollo数据库，请注意备份数据。我们准备的sql文件会清空Apollo相关的表。

2.1.1 创建ApolloPortalDB

通过各种MySQL客户端导入[sql/apolloportaldb.sql](#)即可。

下面以MySQL原生客户端为例：

```
source /your_local_path/sql/apolloportaldb.sql
```

导入成功后，可以通过执行以下sql语句来验证：

```
select `Id`, `AppId`, `Name` from ApolloPortalDB.App;
```

Id	AppId	Name
1	SampleApp	Sample App

2.1.2 创建ApolloConfigDB

通过各种MySQL客户端导入[sql/apolloconfigdb.sql](#)即可。

下面以MySQL原生客户端为例：

```
source /your_local_path/sql/apolloconfigdb.sql
```

导入成功后，可以通过执行以下sql语句来验证：

```
select `NamespaceId`, `Key`, `Value`, `Comment` from ApolloConfigDB.Item;
```

NamespaceId	Key	Value	Comment
1	timeout	100	sample timeout配置

2.2 配置数据库连接信息

Apollo服务端需要知道如何连接到你前面创建的数据库，所以需要编辑`demo.sh`，修改ApolloPortalDB和ApolloConfigDB相关的数据库连接串信息。

注意：填入的用户需要具备对ApolloPortalDB和ApolloConfigDB数据的读写权限。

```
#apollo config db info
apollo_config_db_url=jdbc:mysql://localhost:3306/ApolloConfigDB?characterEncoding=utf8
apollo_config_db_username=用户名
apollo_config_db_password=密码（如果没有密码，留空即可）

# apollo portal db info
apollo_portal_db_url=jdbc:mysql://localhost:3306/ApolloPortalDB?characterEncoding=utf8
apollo_portal_db_username=用户名
apollo_portal_db_password=密码（如果没有密码，留空即可）
```

注意：不要修改demo.sh的其它部分

三、启动Apollo配置中心

3.1 确保端口未被占用

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。

例如，在Linux/Mac下，可以通过如下命令检查：

```
lsof -i:8080
```

3.2 执行启动脚本

```
./demo.sh start
```

当看到如下输出后，就说明启动成功了！

```
==== starting service ====
Service logging file is ./service/apollo-service.log
Started [10768]
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 [10846]
```

Waiting **for** portal startup.....

Portal started. You can visit <http://localhost:8070> now!

3.3 异常排查

如果启动遇到了异常，可以分别查看service和portal目录下的log文件排查问题。

注：在启动apollo-configservice的过程中会在日志中输出eureka注册失败的信息，如
`com.sun.jersey.api.client.ClientHandlerException: java.net.ConnectException: Connection refused`。需要注意的是，这个是预期的情况，因为apollo-configservice需要向Meta Server（它自己）注册服务，但是因为在启动过程中，自己还没起来，所以会报这个错。后面会进行重试的动作，所以等自己服务起来后就会注册正常了。

3.4 注意

Quick Start只是用来帮助大家快速体验Apollo项目，具体实际使用时请参考：[分布式部署指南](#)。

另外需要注意的是Quick Start不支持增加环境，只有通过分布式部署才可以新增环境，同样请参考：[分布式部署指南](#)

四、使用Apollo配置中心

4.1 使用样例项目

4.1.1 查看样例配置

1. 打开<http://localhost:8070>

Quick Start集成了[Spring Security简单认证](#)，更多信息可以参考[Portal 实现用户登录功能](#)

Apollo配置中心

Username

Password

登录

2. 输入用户名apollo，密码admin后登录

Apollo 配置中心

搜索项目(项目ID、项目名)

帮助

apollo

我的项目

+

创建项目

SampleApp
Sample App

收藏的项目

您还没有收藏过任何项目,在项目主页可以收藏项目哟~

3. 点击SampleApp进入配置界面，可以看到当前有一个配置timeout=100

Apollo 配置中心

搜索项目(项目ID、项目名)

帮助

apollo

环境列表

DEV

项目信息

AppId: SampleApp
应用名: Sample App
部门: 样例部门1(TEST1)
负责人: apollo
Email: apollo@acme.com

私有

application properties

发布

回滚

发布历史

授权

创建灰度

表格

文本

更改历史

实例列表

过滤配置

同步配置

新增配置

发布状态	Key	Value	备注	最后修改人	最后修改时间	操作
已发布	timeout	100	sample timeout配置		2016-12-27 16:28:30	<div></div> <div></div>

如果提示 系统出错，请重试或联系系统负责人，请稍后几秒钟重试一下，因为通过Eureka注册的服务有一个刷新的延时。

4.1.2 运行客户端程序

我们准备了一个简单的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：

```
Apollo Config Demo. Please input key to get the value. Input quit to exit.  
>
```

输入 `timeout`，会看到如下信息：

```
> timeout  
> [SimpleApolloConfigDemo] Loading key : timeout with value: 100
```

如果运行客户端遇到问题，可以通过修改 `client/log4j2.xml` 中的level为DEBUG来查看更详细日志信息

```
<logger name="com.ctrip.framework.apollo" additivity="false" level="trace">  
  <AppenderRef ref="Async" level="DEBUG"/>  
</logger>
```

4.1.3 修改配置并发布

1. 在配置界面点击timeout这一项的编辑按钮


Apollo 配置中心

搜索项目(项目ID、项目名) 帮助 apollo

私有

application properties 发布 回滚 发布历史 授权 创建灰度

表格 文本 更改历史 实例列表 过滤配置 同步配置 新增配置

发布状态	Key	Value	备注	最后修改人	最后修改时间	操作
已发布	timeout	100	sample timeout配置		2016-12-27 16:26:30	 修改

2. 在弹出框中把值改成200并提交

应用ID/应用名

Go

修改配置项

Key

timeout

* Value

200

Comment

sample timeout配置

关闭

提交

3. 点击发布按钮，并填写发布信息

Apollo 配置中心

搜索项目(项目ID、项目名)

帮助

apollo

私有

application

properties

有修改 1

发布

发布配置

回滚

发布历史

权限

创建灰度

表格

文本

更改历史

实例列表 1

发布状态

Key

Value

备注

最后修改人

最后修改时间

操作

未发布

timeout

200

sample timeout配置

apollo

2016-12-27 16:42:34

环境列表

DEV

项目信息

AppId: SampleApp

应用名: Sample App

部门: 样例部门1(TEST1)

负责人: apollo

Email: apollo@acme.com

发布

Changes

Key	发布的值	未发布的值	修改人	修改时间
timeout	100	200	apollo	2016-12-27 16:53:43

* Release Name

Sample发布

Comment

发布内容blabla

取消

发布

4.1.4 客户端查看修改后的值

如果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：

```
[SimpleApolloConfigDemo] Changes for namespace application  
[SimpleApolloConfigDemo] Change - key: timeout, oldValue: 100, newValue: 200, changeType:
```

再次输入 `timeout` 查看对应的值，会看到如下信息：

```
> timeout  
> [SimpleApolloConfigDemo] Loading key : timeout with value: 200
```

4.2 使用新的项目

4.2.1 应用接入Apollo

这部分可以参考[Java应用接入指南](#)

4.2.2 运行客户端程序

由于使用了新的项目，所以客户端需要修改appId信息。

编辑 `client/META-INF/app.properties`，修改app.id为你新创建的app id。

`app.id`=你的appId

运行 `./demo.sh client` 启动Demo客户端即可。

► Pages 19

- 设计文档
 - [Apollo配置中心介绍](#)
 - [Apollo配置中心设计](#)
 - [Apollo核心概念之“Namespace”](#)
- 部署文档
 - [Quick Start](#)
 - [Docker方式部署Quick Start](#)
 - [分布式部署指南](#)
 - [Apollo源码解析（全）](#)
- 开发文档
 - [Apollo开发指南](#)

- Code Styles
 - [Eclipse Code Style](#)
 - [IntelliJ Code Style](#)
- [Portal实现用户登录功能](#)
- [邮件模板样例](#)
- 系统使用文档
 - [Apollo使用指南](#)
 - [Java客户端使用指南](#)
 - [.Net客户端使用指南](#)
 - [Go、Python、NodeJS、PHP等客户端使用指南](#)
 - [其它语言客户端接入指南](#)
 - [Apollo开放平台接入指南](#)
 - [Apollo使用场景和示例代码](#)
- FAQ
 - [常见问题回答](#)
 - [部署&开发遇到的常见问题](#)
- 其它
 - [版本历史](#)
 - [Apollo性能测试报告](#)

Clone this wiki locally

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

