# Quickstart（体验）

下载apollo-quick-start-1.6.1

*数据库建两个表*

*修改demo.sh 连接数据库信息*

*./demo.sh start*

启动Apollo报错Config service failed to start in 120 seconds! Please check ./service/apollo-ser

修改demo.sh: declare -i max\_counter=120

Localhost  8070 web端、8080注册中心、8090

大概描述：Apollo 包含config 服务， port后台管理服务； 后台管理维护配置值 java导包客户端获取值 长连接 可以监听值变化 就是基于eureka的两个服务 一个负责后台管理的值维护 一个获取值更新到客户端

# 分布式部署（实际开发、上线）

上面的只是一个demo（打包在一块的，一键运行），了解一下，实际开发要自己一个个搭建；

apollo-configservice：提供配置获取接口，提供配置更新推送接口，接口服务对象为Apollo客户端

apollo-adminservice：提供配置管理接口，提供配置修改、发布等接口，接口服务对象为Portal，以及Eureka

apollo-portal：提供Web界面供用户管理配置

apollo-client：Apollo提供的客户端程序，为应用提供配置获取、实时更新等功能

下载源码 <https://github.com/ctripcorp/apollo>

接下来自己配置还原quickstart，了解一键运行实现的底层

1、打开根目录scripts下build.bat或者build.sh

里面跟demo.sh很相似 配置config db info和portal db info

表示分别连接ApolloConfigDB，这个数据库专门存储配置相关的，包括存储的键值对，以及ApolloPortalDB，这个存储web后台管理相关的

dev\_meta后面配置的是configservice部署以后的访问路径

1. 运行./build.sh 根据你配置的信息打包configservice、adminservice、portal（这三个就是java项目）
2. 进入configservice下面的target目录，找到apollo-configservice-1.6.2-github.zip包解压，进入解压后的scripts打卡startup.sh 里面的“SERVER\_PORT=”就指定了项目的端口，运行命令启动，该项目的服务路径就是第一步dev\_meta的值；同理启动adminservice，portal；可以发现这三个项目默认端口就是8080（configservice）,8070（portal），8090（adminservice）；这样就跟quickstart效果一样了； 报找不到目录 把包换个位置启动

几个疑问

Build.sh中

dev\_meta="http://localhost:8080"

ApolloPortalDB数据库serverconfig表key字段 apollo.portal.envs 的值value是dev（多个逗号分隔）

这两者关系：portal的web项目启动时读取 apollo.portal.envs的值，有几个环境就会在web页面显示几个，同时dev对应dev\_meta的注册中心,pro对应dev\_pro

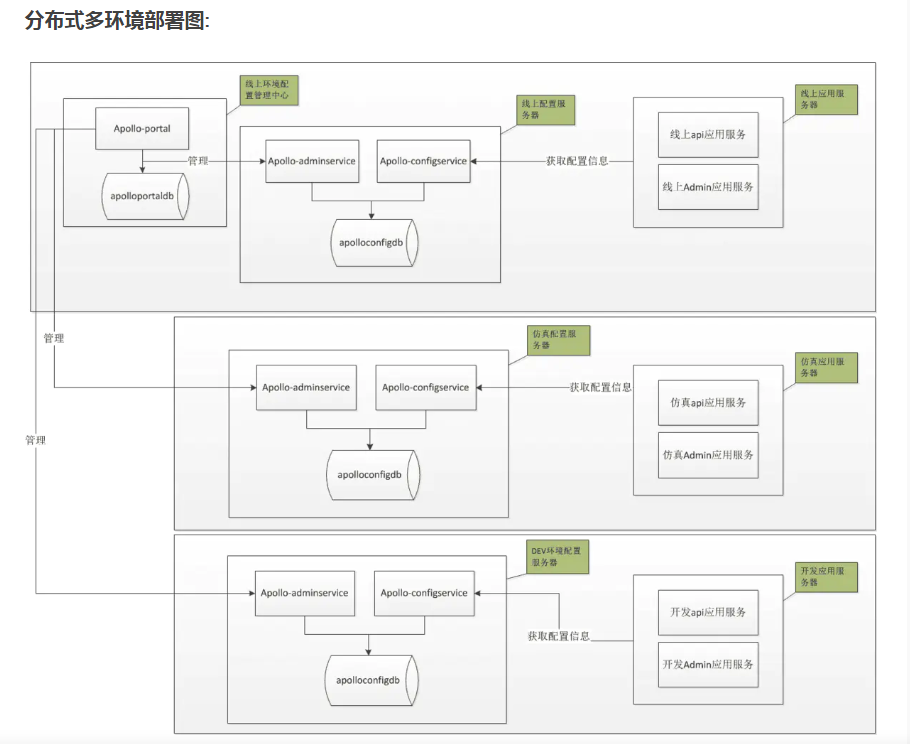
ApolloConfigDB数据库serverconfig表key字段eureka.service.url的值为http://localhost:8080/eureka/

Springboot客户端application.properties中配置的apollo.meta=http://127.0.0.1:8080

这两者关系:eureka.service.url值互相注册，是注册中心 apollo.meta配置哪个注册中心就会连上哪个 一般用于切环境 不同环境注册中心不一样；

Springboot客户端application.properties 和portal web端都是要连接注册中心configservice的

首先说明一下部署结构，再实验



· Portal（web界面：8070）部署一个就行，通过它来直接管理各个环境的配置

· Config Service、Admin Service和apolloconfigdb在每个环境都单独部署

· 应用需要配置指定的环境，默认为dev。

实验发现

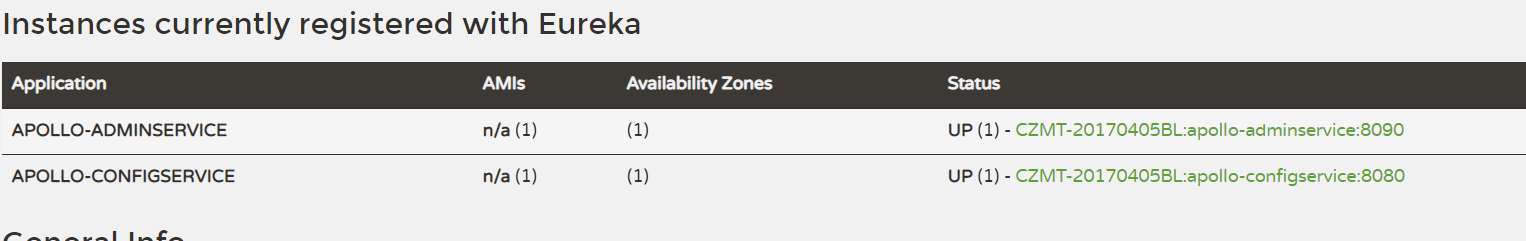
实验：

修改build.bat文件（也可以直接去修改打包之后的服务下的配置文件）

config db info连接ApolloConfigDB\_dev（本地使用不同数据库名称模仿不同服务器环境），portal db info连接ApolloPortalDB数据库；dev\_meta="http://localhost:8080"；运行build.bat打包configservice、adminservice、portal；数据库数据保持默认：eureka.service.url=http://localhost:8080/eureka/

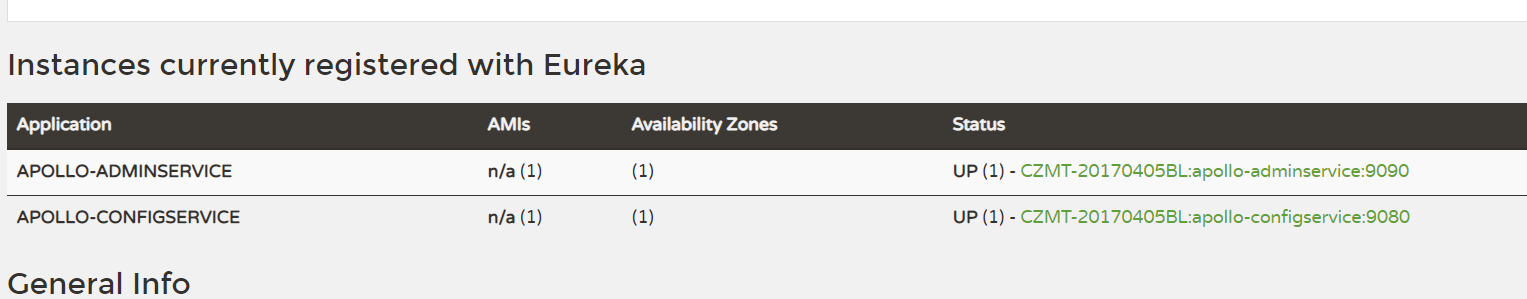
Target解压拷贝到别的目录（防止第二次打包覆盖第一次打包内容），依次启动configservice、adminservice、portal(startup.sh保持默认端口);

访问<http://localhost:8080/>



集群高可用：configservice、adminservice复制一份改startup.bat端口9080,9090启动 会发现注册到了localhost:8080，因为启动时读取ApolloConfigDB\_dev数据库的eureka.service.url=http://localhost:8080/eureka/;

多环境：停止上面两个，config db info连接ApolloConfigDB\_prod,eureka.service.url=http://localhost:9080/eureka/，端口还是9080,9090，启动访问localhost:9080;



这个时候环境已经ok，只需要告诉portal使用哪个环境；portaldb的serverconfig表的apollo.portal.envs加上DEV,pro 然后重启portal项目，重新登web页面



Java端每次获取不同环境的值只需要修改apollo.meta=http://127.0.0.1:9080，对应各自的注册中心

结论：Build.sh中数据库配置修改后会打进configservice、adminservice、portal这三个包

当configservice、adminservice启动时会连接打包时的数据库连接配置，读取到configdb数据库serverconfig表中eureka.service.url的值进行注册，也就是说会把自己注册进eureka.service.url所对应的值，所以改数据库连接配置对应不同环境不同数据， 不改数据启动多个configservice进行注册就是高可用;