Apollo是携程部门研发的分布式配置中心，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，适用于微服务配置管理场景。

* 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/Spring Boot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。

## Apollo配置中心介绍

### 1、What is Apollo

#### 1.1 背景

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……

对程序配置的期望值也越来越高：配置修改后实时生效，灰度发布，分环境、分集群管理配置，完善的权限、审核机制……

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。

Apollo配置中心应运而生！

#### 1.2 Apollo简介

Apollo支持4个维度管理key-Value格式的配置：

* application（应用）
* environment（环境）
* cluster（集群）
* namespace（命名空间）

#### 1.3 配置基本概念

既然Apollo定位于配置中心，那么先简单介绍一下什么是配置。

配置有以下几个属性：

**（1）配置独立于程序的只读变量**

* 配置首先是独立于程序的，同一份程序在不同的配置下会有不同的行为。
* 其次，配置对于程序是只读的，程序通过读配置来改变自己的行为，但是程序不应该去改变配置。
* 常见的配置有：DB Connection Str、Thread Pool Size、Buffer Size、Request Timeout、Feature Switch、Server Urls等。

**（2）配置伴随整个应用的整个生命周期**

配置贯穿于应用的整个生命周期，应用在启动时通过读取配置来初始化，在运行时根据配置调整行为。

**（3）配置可以有多中加载方式**

配置也有多种加载方式，常见的由程序内部hard code，配置文件，环境变量，启动参数，基于数据库等。

**（4）配置需要治理**

* 权限控制：由于配置能改变程序的行为，不正确的配置甚至能引起灾难，所以对配置的修改必须有比较完善的权限控制。
* 不同环境、集群配置管理：同一份程序在不同的环境（开发、测试、生产）、不同的集群（如不同的数据中心）经常需要有完善的环境、集群配置管理。
* 框架类组件配置管理
  + 还有一类比较特殊的配置-框架类组件配置，比如CAT客户端的配置。
  + 虽然这类框架类组件是由其他团队开发、维护，但是运行时是在业务实际应用内的，所有本质上可以认为框架类组件也是应用的一部分。
  + 这一类组件对应的配置也需要有比较完善的管理方式。

### 2、Why Apollo

Apollo设计之初就立志于成为一个有治理能力的配置发布平台，目前提供了以下特性：

**（1）统一管理不同环境、不同集群的配置**

* Apollo提供了一个统一界面集中式管理不同环境

**（2）配置修改实时生效（热发布）**

**（3）版本发布管理**

**（4）灰度发布**

**（5）权限管理、发布审核、操作审计**

**（6）客户端配置信息监控**

**（7）提供Java和.NET原生客户端**

**（8）提供开发平台API**

**（9）部署简单**