应用配置管理 ACM

产品简介

产品简介

产品概述

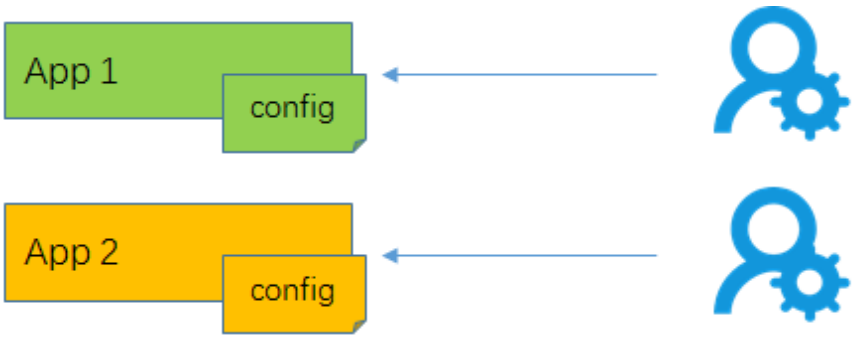
应用配置管理（Application Configuration Management，简称 ACM），其前身为淘宝内部配置中心 Diamond，是一款在分布式架构环境中对应用配置进行集中管理和推送的应用配置中心产品。基于该应用配置中心产品，您可以在微服务、DevOps、大数据等场景下极大地减轻配置管理的工作量，增强配置管理的服务能力。

在应用生命周期管理中，开发人员通常会将应用中需要变更的一些配置项或者元数据从代码中分离出来，放在单独的配置文件中管理，称为应用配置。应用配置是管理应用变更的常见手段之一。发布应用后，运维人员或最终用户可以通过调整配置来适配环境，调整应用程序的运行行为。

ACM 是分布式系统的配置中心。通过提供配置变更、配置推送、历史版本管理、灰度发布、配置变更审计等配置管理工具，ACM 帮助您集中管理所有应用环境中的配置，降低分布式系统中管理配置的成本，并降低因错误的配置变更带来可用性下降甚至发生故障的风险。

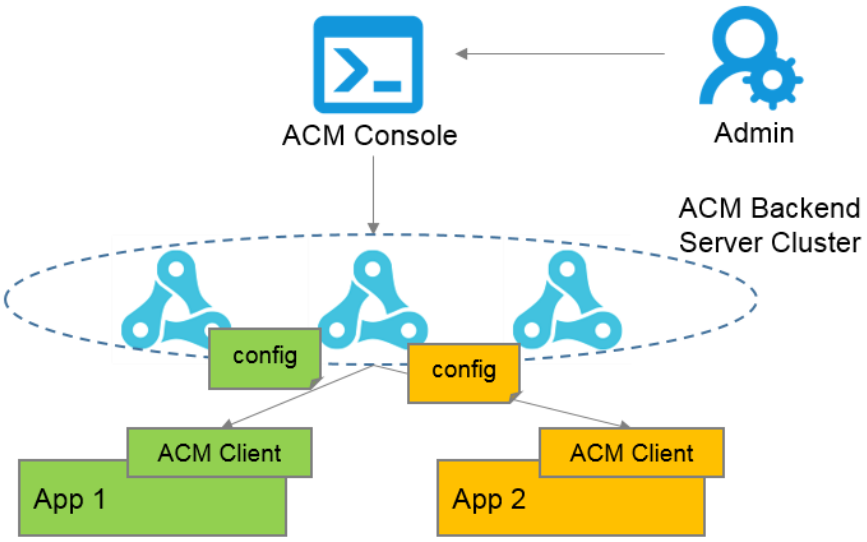
传统架构中的配置管理

在传统架构中，如果配置信息有变更，通常需要登陆服务器手动修改配置来使配置生效，如下图。



ACM 配置管理

在 ACM 的配置管理场景下，您只需要在 ACM 控制台上更改配置，配置信息会自动被推送到各个服务器中并在秒级延迟内生效。完整的 ACM 产品包括三个主要部分：客户端、服务端和用于配置管理的控制台。



ACM 价值

通过 ACM 管理配置可以为 IT 运维带来以下好处：

- 更新的配置秒级自动下发到各个机器，极大降低配置手动分发的工作量；
- 通过接入 ACM 配置监听接口，各应用端的配置可立即生效，应用基本无需重启；
- 所有配置监听、更改和版本自动记录在案，增强了审计、版本管理、诊断等各方面的能力。

功能概述

本文总结了 ACM 的功能。

核心功能/特性

- 配置的“增删改查”：产品核心功能
- 配置的导入导出：方便用户在多环境间同步配置
- 批量操作：提供批量管理功能
- 聚合数据的“增删改查”：支持修改一部分数据
- 配置描述：方便他人理解配置用途
- 配置标签：配置归类有助于提高配置管理效率
- 在线编辑多种文件格式：支持语法高亮、预发校验，提升在线编辑效率，减少编辑错误
- 多语言支持：支持 Java/CPP/Python/Shell/HTTP 的 OpenAPI

高级特性

- 变更 Diff：变更内容一览无余，可降低误操作风险

- 灰度发布：控制变更风险
- 变更历史：让一切操作有据可查
- 一键回滚：降低变更风险
- 推送轨迹：提高排查问题效率
- 命名空间：用于隔离日常、预发、生产等环境

稳定性

- 多级缓存：主备库存储容灾，Server 全量缓存，SDK 缓存
- 流量控制：防止错用对集群稳定性造成影响
- 容量管理：防止错用对集群稳定性造成影响
- 同城容灾：机房级容灾能力

性能/容量

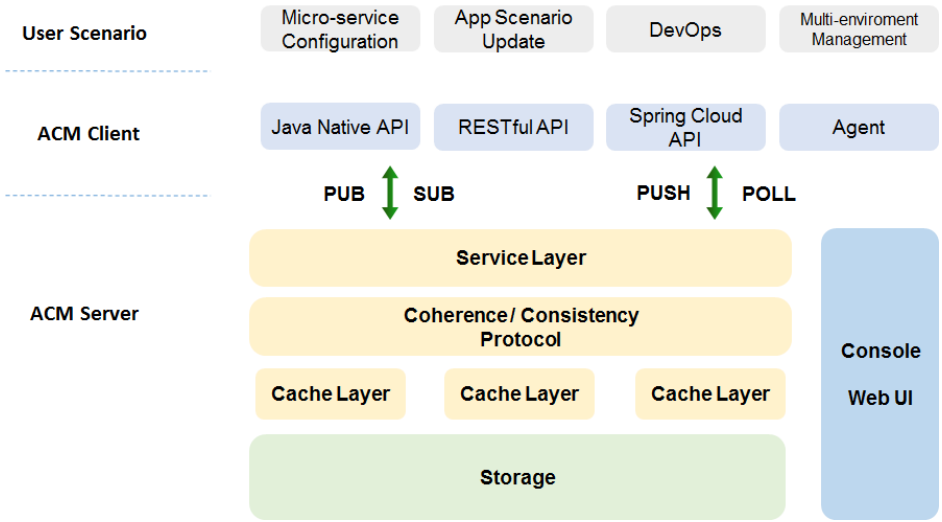
- 可管理数以百万计的配置：有效应对业务爆发式增长
- 支持数以百万计的连接：有效应对业务爆发式增长
- 十万级并发秒级推送：有效应对业务爆发式增长

安全

- HTTPS：提高安全等级
- 鉴权：防止数据受其他用户影响
- 数据加密：提高敏感数据的安全性

技术架构

ACM 的架构图和相应描述如下。



ACM 服务端

即 ACM 分布式服务节点，每一个节点包括以下部分：

- 服务协议层：用于进行协议转换，鉴权验证等。
- 一致性管理层：用于配置的一致性管理和配置推送。
- 配置缓存层：通过分布式缓存提高配置查询和推送效率。
    - 存储层：后台是一个分布式存储，用于存放配置，并具备高性能和高扩展性。
    - 控制台：ACM 控制台，用于配置管理。

ACM 客户端

客户端基于 RESTful API 设计，因此可实现跨语言访问。ACM 提供 Java Native API 以及基于 Spring Cloud Config 的配置读取接口，方便您进行便捷开发。

在某些场景下，ACM 提供 Agent 来动态替换宿主机器上的配置文件。此时 ACM 配置项和配置文件对应关系需手动指定。

产品对比

本文针对几款主流的应用配置类产品进行对比分析。

类似产品介绍

ZooKeeper

ZooKeeper 是一个分布式应用程序协调服务，是 Google Chubby 的开源实现。它是一个为分布式应用提供一致性服务的软件，提供的功能包括配置维护、域名服务、分布式同步、组服务等。在 Hadoop 集群等场景下，ZooKeeper 同时充当应用配置管理的角色。但是由于它是 CP(Consistency，Partition Tolerance) 类应用，因此在可用性和性能上都会受到一定影响。

ETCD

和 ZooKeeper 类似，etcd 是一个高可用的键值存储系统，主要用于配置共享和服务发现。etcd 是由 CoreOS 开发并维护的，灵感来自于 ZooKeeper 和 Doozer。它使用 Go 语言编写，并通过 Raft 一致性算法处理日志复制以保证强一致性。etcd 和 ZooKeeper 类似，同样可以用来做应用管理配置。但是由于它是强一致的管理类应用，因此其可用性和性能在某些场景会受到一定影响。

Spring Cloud Config Server

和 ACM 类似，Spring Cloud Config Server 为服务端和客户端提供了分布式系统的外部配置支持。配置服务器为各应用的所有环境提供了一个中心化的外部配置。与 ACM 不同的是，Spring Cloud 配置服务器默认采用 Git 来存储配置信息，其配置存储、版本管理、发布 等功能都基于 Git 或其他外围系统来实现。除此之外，在配置功能方面，ACM 和 Spring Cloud Config 也有很大不同。

产品对比

以下表格详细描述了 ACM 和 ETCD， ZooKeeper，和 Spring Cloud Config 在应用配置管理方面的区别。

产品	ACM	Spring Cloud Config Server	ZooKeeper	ETCD
配置修改	直接在 ACM 控制台上进行配置修改	一般在 Git 仓库上进行配置修改	通过调用 ZK API 修改	通过调用 etcd API 修改
配置自动推送	修改过的配置自动推送到监听的客户端	客户端只能在启动的时候加载	修改过的配置自动推送到监听的客户端	修改过的配置自动推送到监听的客户端
API接口	基于 RESTful API，同时支持 Java Native 接口，Spring Cloud 接口，和其他语言类接口	基于 RESTful API 和 Spring Cloud 规范，同时也支持其他语言客户端	支持 Java 原生接口	基于 RESTful API 的接口
版本管理	在 ACM 上自动记录各个修改的版本信息	通过 Git 间接管理版本	不带任何版本控制	不带任何版本控制
配置推送追踪	可查询所有客户端配置推送状态和轨迹	无法查询配置推送历史	无法查询配置推送历史	无法查询配置推送历史

注意：ACM 不是 CP 类应用，因此和 ETCD，ZK 定位不同，ACM 并不适合用于严格的事务类配置服务，如分布式锁等。

使用场景

ACM 产品从 2008 年开始就被大量应用于阿里巴巴内部应用的配置管理，是阿里巴巴内部被众多应用最广泛依赖的中间件产品之一。ACM 在众多核心场景有广泛的使用案例。本文例举了几个场景进行详细介绍。

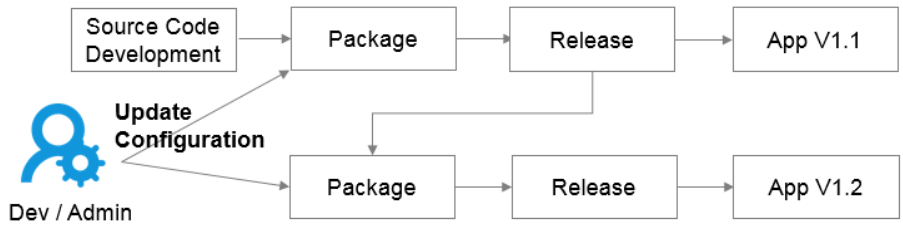
- 微服务应用架构

- 分布式架构下的服务治理
- 应用业务场景动态推送
- 大数据实时计算算法调整
- 企业级互联网架构的异地多活

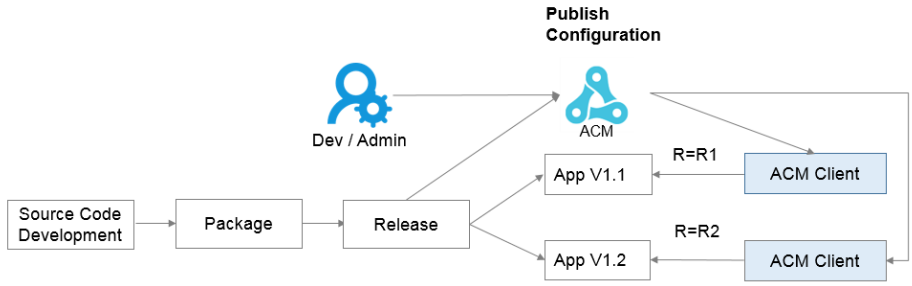
微服务应用架构下的配置管理

在微服务架构下，随着应用和机器数目的增多，配置（如 DB_URL 访问信息、服务连接池、服务内部缓存大小等）管理的工作会变得很繁琐。单个应用下多个机器的配置分发，应用对应用的配置依赖，都将成为挑战。

在传统架构的应用发布过程中，修改一个应用配置就需要将整个应用重新打包发布，整个过程非常繁琐，且容易出错，如下图所示。



在基于 ACM 的微服务场景下，应用的重要配置信息被发布到 ACM 中。新的配置发布并不依赖配置打包。在新版本的配置发布后，所有应用立即生效，如下图所示。



采用 ACM 作为配置中心为微服务带来以下好处：

- 所有配置中心化，在应用众多的情况下配置管理变得更加方便。
- 所有配置不依赖版本发布，使得配置变更更加灵活。
- ACM 天生支持灰度发布和回滚，使得配置的变更发布在微服务架构下变得更加安全。

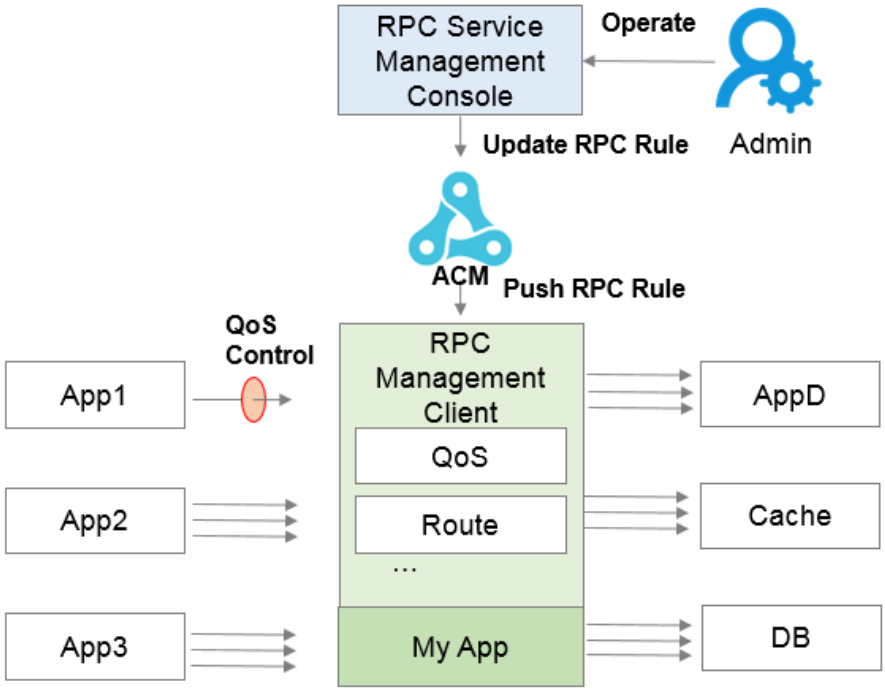
分布式架构下的服务治理下的服务治理

在各类分布式架构下，如何基于某类 RPC 框架（RESTful，HSF，Dubbo 等）做好服务治理是非常关键的。其中，服务治理的服务路由、服务限流、服务降级、服务鉴权等，均可通过配置中心实现。

以限流降级为例，在阿里巴巴双11，每一次服务限流降级相关操作都要求秒级响应，此需求就通过 ACM 来实现。

其中，每个 RPC 的服务端都会通过 ACM 注册监听对应服务的流控管理信息。当应用需要流控时，管理员在服

务治理控制台做出限流操作，服务治理系统就会通过 ACM 将流控信息推送到对应的目标应用服务端，让应用服务端对应配置采取相应的限流动作。



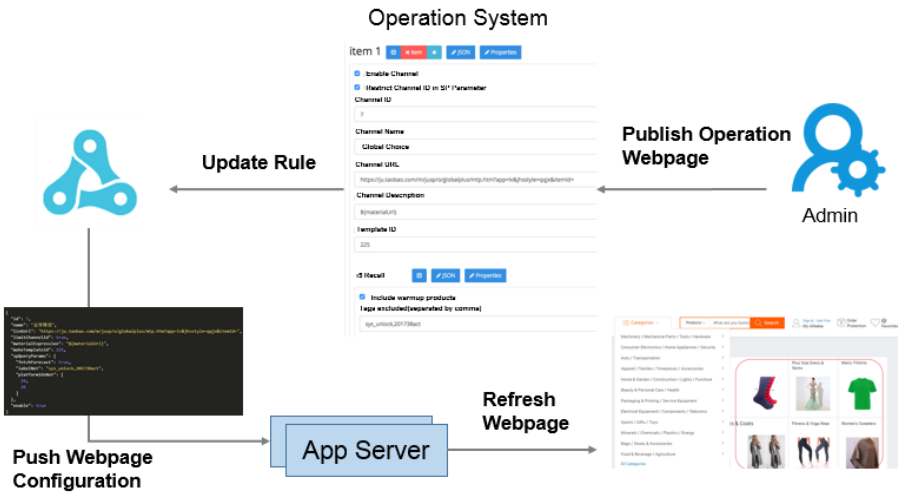
采用 ACM 为分布式架构下的服务治理带来以下好处：

- 良好的性能，通过采用配置推送的方式来监听服务治理信息，对性能几乎无影响。
- 相关的服务治理信息可以秒级推送到，响应时间迅速。
- 当限流降级错推以后还可以通过秒级配置回滚来恢复状态。

应用业务场景动态推送

如何让前端页面快速响应营运活动，降低开发成本，提高运营效率也是 ACM 的一大场景。

以某电商运营为例。通过在前端 Javascript 埋入 ACM 的配置内容（如三方库版本号，静态资源 URL 等），在运营活动发布时，运营人员可以通过运营工具修改 ACM 配置规则，直接让前端 Javascript 的展示生效。



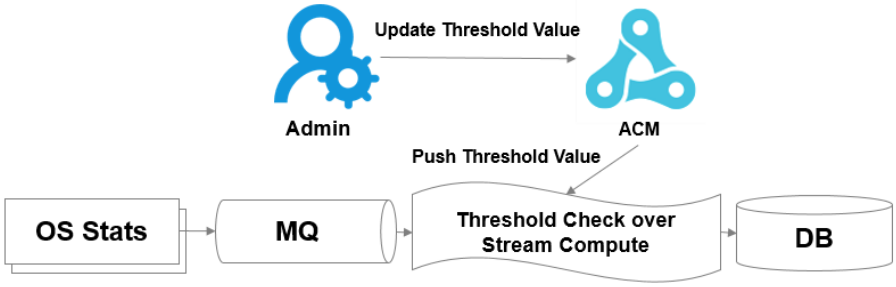
采用 ACM 为应用的业务场景推送带来以下好处：

- 通过配置将业务静态代码和业务场景解耦，显著提高运营相关的应用发布流程。

大数据实时计算算法调整

在大数据的实时计算领域，通常需要动态调整计算参数来取得最精确的实时计算结果。

以阿里巴巴内部某 APM 类监控系统为例。该监控系统需要动态调节业务的阈值，来控制实时计算系统，计算业务的报警。而且在阈值修改过程中，要求整个过程实时性好，应用不能停机。该监控系统的计算阈值采用 ACM 的规则来推送。



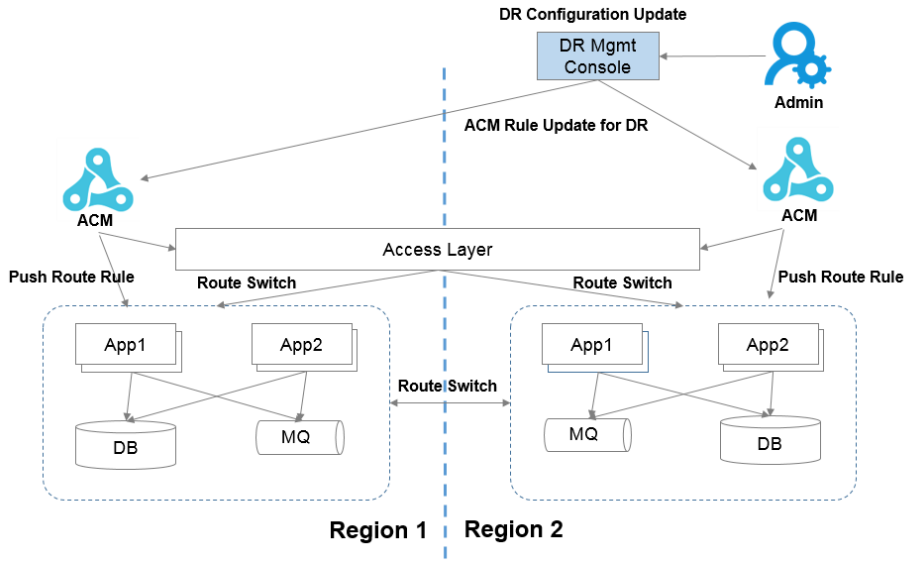
采用 ACM 为应用在大数据实时计算场景带来以下好处：

- 应用计算参数动态配置，动态生效，生效时间块，性能影响低。

企业级互联网架构下的异地多活场景

异地多活是企业级互联网架构中的一种高级容灾架构。相对于传统容灾架构，有业务恢复时间短，容量需求低，运维快捷简单等特点。目前多活架构在阿里巴巴，饿了么等公司被广泛采用。

在阿里巴巴内部，容灾多活架构的核心算法，ID 分片和对应的的路由规则均采用 ACM 来动态推送。其中，相应的客户端和服务端，如 RPC，MQ，DB 都植入了路由路径。当容灾演练或者真实灾难发生时，管理员只需要动态的推送规则，相应的规则会影响到所有架构组件。示意图如下。



采用 ACM 为异地多活场景中的应用带来以下好处：

- 基础架构和容灾逻辑解耦，具体的路由逻辑由容灾规则切换决定。
- 生效快，理论上容灾的切换规则可以秒级推送到十万级别机器。

名词解释

配置

在系统开发过程中通常会将一些需要变更的参数、变量等从代码中分离出来独立管理，以独立的配置文件的形式存在。目的是让静态的系统工件或者交付物（如 WAR，JAR 包等）更好地和实际的物理运行环境进行适配。配置管理一般包含在系统部署的过程中，由系统管理员或者运维人员完成这个步骤。配置变更是调整系统运行时的行为的有效手段之一。

动态配置和静态配置

系统配置可以是静态或者动态配置。配置的版本与软件本身的版本强绑定的称之为静态配置。反之，在同一个版本的系统部署且运行期间配置可以连续发生多次变更的则称之为动态配置。例如build-version: 1.0.0 配置与软件版本绑定，称之为静态配置；而线程池大小的配置则可以在系统运行期间连续多次变更，称之为动态配置。

配置管理

在数据中心中，系统中所有配置的编辑、存储、分发、变更管理、历史版本管理、变更审计等所有与配置相关的活动统称为配置管理。

配置推送

配置管理中，常需要配置管理系统将配置的变更分发到相关的系统，从分发到配置生效的过程称为配置推送。

推送轨迹

从配置变更，配置推送到配置生效过程的整个轨迹称为推送轨迹。通过查看某个配置的推送轨迹，我们可以获知一个配置变更在哪些应用上，哪些机器上，在哪个时间点生效，产生了哪些影响等。

配置监听

配置监听，是指 ACM 允许系统通过使用 SDK 等方式向 ACM 系统注册监听器（Listener），从而监听并消费该配置的变更。

配置项

是指一个具体的可配置的参数与其值域，通常是 param-key=param-value 的形式存在。例如我们常配置系统的日志输出级别（logLevel=INFO|WARN|ERROR） 就是一个配置项。

配置集

一组相关或者不相关的配置项的集合称为配置集。通常系统中的一个配置文件就是一个配置集，其内包含了系统各个方面配置。例如一个配置集可能包含了系统如数据源、线程池、日志级别等配置项。

配置集 ID（Data ID）

在 ACM 中代表配置集的 ID, 是配置组织的维度之一。一般通过 Data ID 来组织划分系统的配置集。一个系统或者应用可以包含多个配置集，每个配置集可以用有意义的名称来标识这个配置集。Data ID 通常采用类 Java 包命名方式（如com.taobao.tc.refund.log.level）的命名规则保证全局唯一性，此命名规则非强制。

配置分组（Group）

ACM 中配置集的分组，是配置组织的维度之一。通常使用一个有意义的字符串来分组配置集，例如 Buy，Trade 等，用以区分相同 Data ID 的配置集。ACM 创建配置时，如果用户未填 Group 名字，则默认用 DEFAULT_GROUP 代替。Group 的常用场景是同一个配置类型用于不同应用或者组件，如 database_url 配置，MQ_topic 配置等。

命名空间（Namespace）

ACM 中的命名空间，用于进行租户粒度的配置隔离。不同的命名空间下，可以存在相同的 Group，Data ID 的配置。Namespace 的常用场景之一是不同环境的配置的区分隔离，如开发测试环境和生产环境的配置隔离等。

配置快照

ACM 客户端 SDK 会在本地生成配置的快照。当客户端无法连接到 ACM Server 时，可以利用快照提示系统的整体容灾能力。配置快照类似于 Git 中的本地 commit 的概念，也类似缓存，会在适当的时机更新，但是没有缓存过期（expire）的概念。

版本说明

V4.4.0

发布时间：20180518

主要功能发布：

- 全新支持 ECS 实例 RAM 角色：无需设置任何敏感信息（如AK/SK），即可让应用在 ECS 上访问 ACM 配置信息，解决云上应用安全合规问题。[相关文档]

其他优化：

- 修复了与 EDAS 的兼容性问题。

V4.3.0

发布时间：20180408

主要功能发布：

- 支持加密配置：通过集成 KMS（密钥管理服务），用户可以在 ACM 上加密自己的配置，让配置更安全。
- 支持 RAM（资源访问控制服务）策略：用户可通过 RAM 授权子账号、服务账号和其他主账号读取自己的配置。
- 支持容量管理策略：基于精细化的容量管理策略，用户现在可以直接通过 API 或 SDK 写入自己的配置。
- 支持导入、导出和克隆配置：用户可以方便地将自己的配置从某个地域和命名空间批量导出到指定地域和命名空间。

其他优化：

- 修复了与 EDAS 的兼容性问题。

V4.2.0

发布时间：20180201

主要功能发布：

- 支持配置打标签、配置描述，方便用户按照自己的维度管理配置和了解配置的更多信息。
- 增强了配置编辑能力，支持以 YAML、HTML 等格式在线编辑，支持配置格式的保存。
- 提供 OpenAPI，支持 Shell 语言，方便用户实施多语言扩展，或者在 DevOps 场景获取配置。
- 支持获取一个租户下的配置列表。
- 支持中英文。
- 支持单机版下载。

其他优化：

- 优化了命名空间组件。
- 解决了与 EDAS 冲突问题。
- SDK 瘦身发到公共仓库，减少了 JAR 冲突问题。

V4.1.0

发布时间：20171210

主要功能发布：

- 客户端支持 nodejs、c++。
- 支持国际化、中英文。
- 支持命名空间的删除和详情查看。
- 支持 Group 自动联想。
- 与 EDAS 打通了租户数据。

其他优化：

- 优化了环境组件，解决了异步偶尔无法加载用户命名空间的问题。
- 配置管理页面显示命名空间 ID，方便用户查找。
- 命名空间管理默认展示空间 ID 列，方便用户查看命名空间与 ID 的关系。
- 解决了更新配置冲掉归属应用字段的问题。
- 增加了 nodejs、cpp 示例代码。

V4.0.0

发布时间：20171010

主要功能发布：

- 支持配置的发布、修改、删除、灰度发布。
- 支持配置的历史版本查询。
- 支持配置的推送轨迹查询。
- 支持配置的监听查询。

- 客户端支持 Java Native SDK、Java Spring Cloud SDK。