

阿里云 应用配置管理

快速入门

文档版本：20181116

法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档，您的阅读或使用行为将被视为对本声明全部内容的认可。

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。
2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。
3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。
4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的“现状”、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。
5. 阿里云网站上所有内容，包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含“阿里云”、“Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。
6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

通用约定

格式	说明	样例
	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	 禁止： 重置操作将丢失用户配置数据。
	该类警示信息可能导致系统重大变更甚至故障，或者导致人身伤害等结果。	 警告： 重启操作将导致业务中断，恢复业务所需时间约10分钟。
	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	 说明： 您也可以通过按 Ctrl + A 选中全部文件。
>	多级菜单递进。	设置 > 网络 > 设置网络类型
粗体	表示按键、菜单、页面名称等UI元素。	单击 确定。
<code>courier</code> 字体	命令。	执行 <code>cd /d C:/windows</code> 命令，进入Windows系统文件夹。
斜体	表示参数、变量。	<code>bae log list --instanceid Instance_ID</code>
[]或者[a b]	表示可选项，至多选择一个。	<code>ipconfig [-all -t]</code>
{ }或者{a b}	表示必选项，至多选择一个。	<code>swich {stand slave}</code>

目录

法律声明.....	I
通用约定.....	I
1 开通 ACM 服务.....	1
2 创建并动态调整配置.....	2
3 为不同环境设置不同配置.....	9

1 开通 ACM 服务

您需要先开通 ACM 服务才能使用 ACM。本文介绍了如何开通 ACM 服务。

前提条件

您已注册阿里云账号并完成实名认证。

操作步骤

1. 打开 [ACM 产品主页](https://www.aliyun.com/product/acm) (<https://www.aliyun.com/product/acm>)。
2. 在页面右上角单击登录。
3. 在登录页面上输入您的阿里云账号和密码，并单击登录。
登录成功后，您将跳转至 ACM 产品主页。
4. 在产品主页上单击立即开通，然后在云产品开通页页面上勾选我已阅读并同意《应用配置管理服务协议》，并单击立即开通。



2 创建并动态调整配置

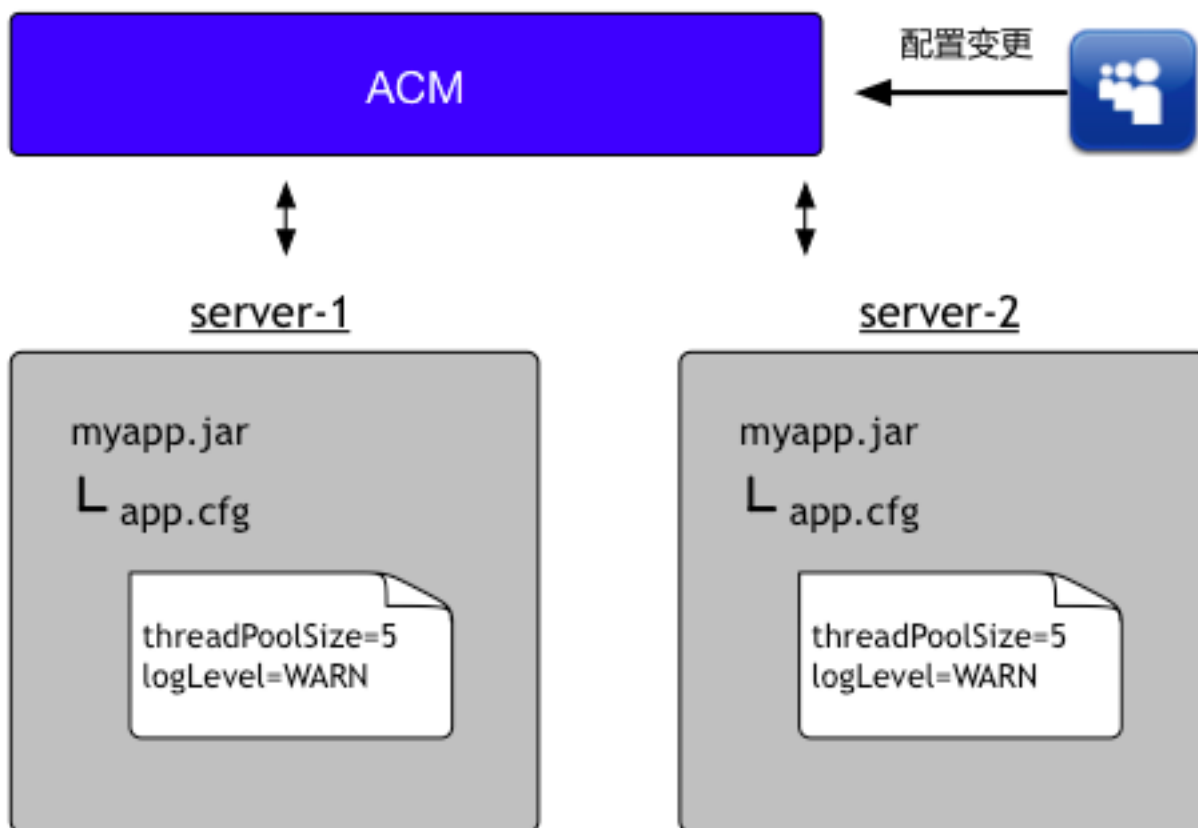
如果一个应用部署在多台服务器上，一旦需要更改配置，就不得不在所有服务器上进行同样的更改，显然效率很低。借助 ACM，您可以为应用创建一个配置，并在程序中使用 ACM 的原生 API 监听此配置的变更。当您在 ACM 控制台更改此配置后，所有部署了该应用的服务器都会收到变更的配置内容，应用状态也随之刷新。

前提条件

- 您已[开通 ACM 服务](#)。
- 服务器上已安装 [JDK](#) 并设置环境变量 `JAVA_HOME`。

背景信息

业务应用 `myapp.jar` 部署在生产环境的两台服务器上。该应用包含一个配置文件 `app.cfg`，配置文件里包含线程池大小（`threadPoolSize`）和日志级别（`logLevel`）这两个配置项。现在需要同时调整该应用在三台机器上的配置，并动态刷新应用的状态。此业务场景如下图所示：



配置内容为:

```
## app.cfg ##
threadPoolSize=5
```

```
logLevel=WARN
```

在本示例任务中，我们首先在 ACM 上为应用 myapp 创建了一个配置，并在程序中使用 ACM 的原生 API 监听这个配置的变更。当我们在 ACM 控制台上更改配置后，所有部署了该应用的服务器都会收到变更的配置内容，应用状态也随之刷新。

第 1 步：在 ACM 中创建配置

1. 登录 [ACM 控制台](#)。
2. 在左侧导航栏中选择配置列表，然后在页面右上角单击 + 图标。



3. 在新建配置页面上输入以下内容，并单击发布。

- DataID : com.acm.myapp.app.cfg
- Group : myapp
- 配置内容：

```
threadPoolSize=5  
logLevel=WARN
```

如图所示：

新建配置

* Data ID:

com.acm.myapp.app.cfg

收起

* Group:

myapp

标签:

请输入标签

归属应用:

描述:

* 目标地域:

☒ public

数据加密

☐ 关 ?

配置格式:

☐ TEXT ☐ JSON ☐ XML ☐ YAML ☐ HTML ☒ properties

* 配置内容: ?

1 threadPoolSize=5

2 logLevel=WARN

第 2 步：使用 API 监听配置变更

1. 运行以下命令来创建 Maven 工程，或者下载样例工程 [myapp.zip](#)。



说明：

关于如何安装和使用 Maven，请参考 [Maven 官方文档](#)。

```
mvn archetype:generate -DgroupId=com.acm.sample -DartifactId=myapp -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false
```

创建的工程结构如下：

```
myapp
|-- pom.xml
|-- src
|   |-- main
```



```

|-- java
|   |-- com
|   |   |-- acm
|   |   |   |-- sample
|   |   |   |   |-- App.java
|-- test
|   |-- java
|   |   |-- com
|   |   |   |-- mycompany
|   |   |   |   |-- app
|   |   |   |   |   |-- AppTest.java

```

2. 在 pom.xml 中添加 ACM Client 原生 API 依赖。

```

<dependencies>
  <dependency>
    <groupId>com.alibaba.edas.acm</groupId>
    <artifactId>acm-sdk</artifactId>
    <version>1.0.8</version>
  </dependency>
  <!-- 有日志实现，下面可去掉 -->
  <dependency>
    <groupId>ch.qos.logback</groupId>
    <artifactId>logback-classic</artifactId>
    <version>1.1.7</version>
  </dependency>
</dependencies>

```

3. 在 pom.xml 中添加 maven-assembly-plugin 打包插件

```

<plugin>
  <artifactId>maven-assembly-plugin</artifactId>
  <version>2.4</version>
  <configuration>
    <finalName>myapp</finalName>
    <descriptorRefs>
      <descriptorRef>jar-with-dependencies</descriptorRef>
    </descriptorRefs>
    <appendAssemblyId>false</appendAssemblyId>
    <archive>
      <manifest>
        <mainClass>com.acm.sample.App</mainClass>
      </manifest>
    </archive>
  </configuration>
  <executions>
    <execution>
      <id>make-assembly</id>
      <phase>package</phase>
      <goals>
        <goal>single</goal>
      </goals>
    </execution>
  </executions>
</plugin>

```

4. 使用 API 监听配置变更。



说明：

以下代码中的用户变量，例如 `$endpoint`、`$namespace`、`$accesskey` 等，可在 ACM 控制台的命名空间页面查看，如下图所示。



```
//-- App.java
package com.acm.sample;

import java.io.IOException;
import java.io.StringReader;
import java.util.Properties;
import com.alibaba.edas.acm.listener.ConfigChangeListener;
import com.alibaba.edas.acm.ConfigService;
import com.alibaba.edas.acm.exception.ConfigException;

public class App {

    private static Properties appCfg = new Properties();

    public static void initAndWatchConfig() {
        final String dataId = "com.acm.myapp.app.cfg";
        final String group = "myapp";
        final long timeoutInMills = 3000;

        // 从控制台命名空间管理中拷贝对应值
        Properties properties = new Properties();
        properties.put("endpoint", "$endpoint");
        properties.put("namespace", "$namespace");
        properties.put("accessKey", "$accessKey");
        properties.put("secretKey", "$secretKey");

        // 如果是加密配置，则添加下面两行进行自动解密
        // properties.put("openKMSFilter", true);
        // properties.put("regionId", "$regionId");

        ConfigService.init(properties);

        // 直接获取配置内容
        try {
```

```
        String configInfo = ConfigService.getConfig(dataId,
group, timeoutInMills);
        appCfg.load(new StringReader(configInfo));
    } catch (ConfigException e1) {
        e1.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
}

// 监听配置变化，获取最新推送值
ConfigService.addListener(dataId, group, new ConfigChangeListener() {
    public void receiveConfigInfo(String configInfo) {
        try {
            appCfg.load(new StringReader(configInfo));
        } catch (Exception e) {
            // process exception
        }
        refreshApp();
    }
});
}

public static void refreshApp() {
    System.out.println("current thread pool size: " + appCfg.
getProperty("threadPoolSize"));
    System.out.println("current log level: " + appCfg.
getProperty("logLevel"));
    System.out.println("");
}

public static void main(String[] args) {
    initAndWatchConfig();

    // 让主线程不退出
    while (true) {
        try {
            Thread.sleep(1000);
        } catch (InterruptedException e) {
        }
    }
}
}
```

第 3 步：部署并启动应用

1. 将应用打成 Jar 包，并拷贝到两台服务器上。在项目根目录下执行以下打包命令

```
mvn clean package
```

2. 在 Shell 中部署并启动应用。

```
${JAVA_HOME}/java -cp myapp.jar com.acm.sample.App
```



说明：

为了运行 Java 程序，服务器上必须安装 [JDK](#) 并设置环境变量 **JAVA_HOME**。

第 4 步：在 ACM 控制台查询并更改配置

1. 在 ACM 控制台的配置列表页面上，搜索第 1 步#在 ACM 中创建配置中创建的配置。
2. 单击操作栏中的编辑。
3. 在编辑配置页面上将配置内容更改为以下内容，并单击发布。

```
threadPoolSize=15  
logLevel=DEBUG
```

4. 在内容比较对话框中，确认配置更改无误，并单击确认发布。



验证结果

发布配置之后，可以看到在两台部署的机器上，应用同时收到配置变更信息，并打印了以下信息。

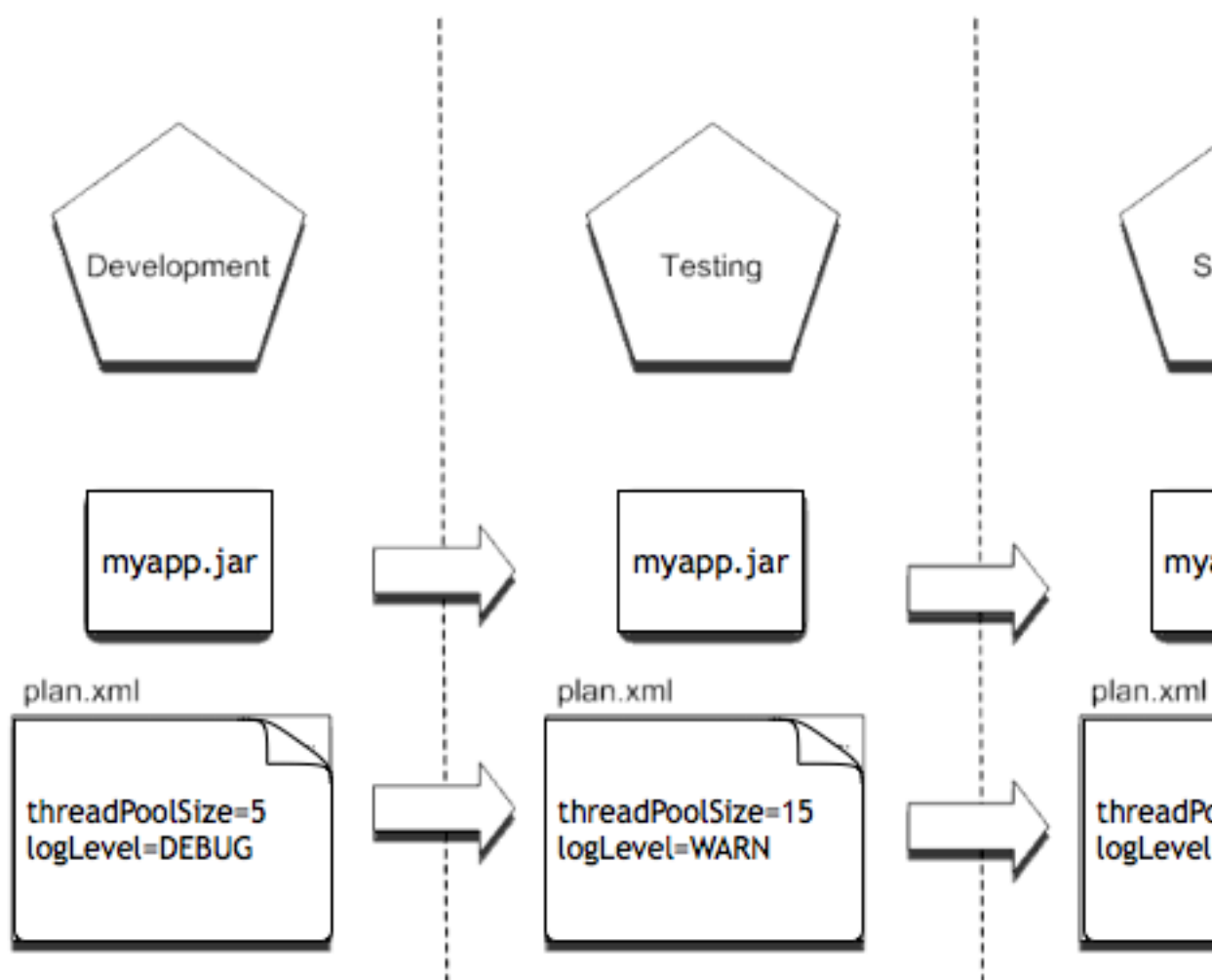
```
current thread pool size: 15  
current log level: DEBUG
```

3 为不同环境设置不同配置

本文以一个例子介绍了如何使用 ACM 的命名空间 (Namespace) 功能，为应用在测试、预生产和生产环境下的同一个配置设置不同的值。

背景信息

在本示例任务中，我们将使用 ACM 的命名空间 (Namespace) 功能，为应用在测试、预生产和生产环境下的同一个配置设置不同的值。期望实现的效果如下：



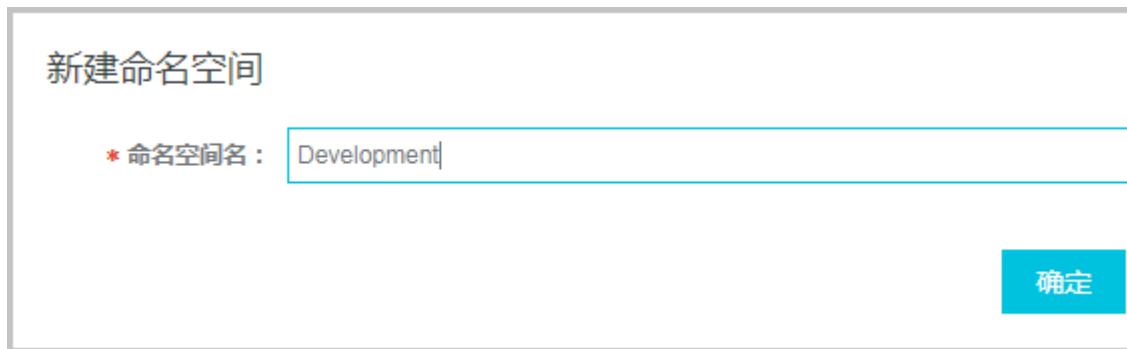
第 1 步：在 ACM 上创建命名空间

下面以创建命名空间 Development 为例。

1. 登录 [ACM 控制台](#)。
2. 在左侧导航栏选择命名空间，单击右上角的新建命名空间。

3. 在新建命名空间对话框中输入命名空间名称

Development。

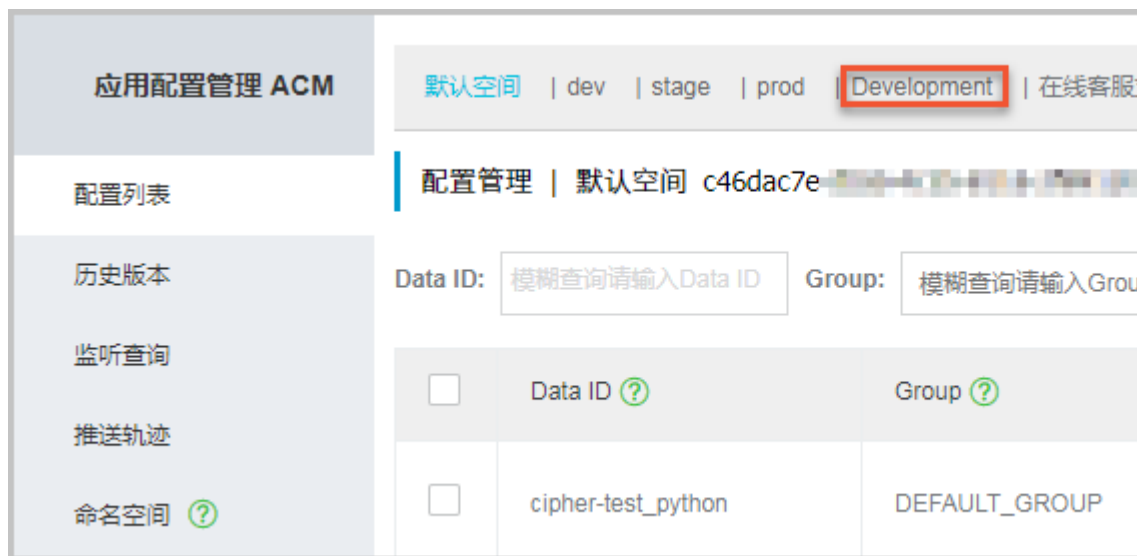


4. 重复 1-3 步，继续创建 Testing 和 Staging 命名空间。

第 2 步：在命名空间下创建配置

1. 在配置列表页面，选择命名空间

Development。



	Data ID ?	Group ?
☐	cipher-test_python	DEFAULT_GROUP

2. 按照第 1 步#在 ACM 中创建配置的说明创建同名配置。

总结

在实际业务场景中，经常需要针对不同环境为同一个配置项设置不同的值。如本示例任务所示，ACM 的命名空间 (Namespace) 功能可以帮助您轻松完成多环境配置管理。