阿里云
分布式任务调度 SchedulerX

快速入门

文档版本：20200619

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. 阿里云文档中所有内容，包括但不限于图片、架构设计、页面布局、文字描述，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。
	该类警示信息可能会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，恢复业务时间约十分钟。
	用于警示信息、补充说明等，是用户必须了解的内容。	 **注意：** 权重设置为0，该服务器不会再接受新请求。
	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。	单击**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。	在**结果确认**页面，单击**确定**。
Courier字体	命令。	执行cd /d C:/window命令，进入Windows系统文件夹。
斜体	表示参数、变量。	bae log list --instanceid Instance_ID
[]或者[a|b]	表示可选项，至多选择一个。	ipconfig [-all|-t]
{}或者{a|b}	表示必选项，至多选择一个。	switch {active|stand}

目录

1 准备工作

1.1 开通SchedulerX（免费）

在开始使用SchedulerX前，需要先开通。SchedulerX目前在公测期，免费。

背景信息

分布式任务调度目前处于公测期，免费使用。

操作步骤

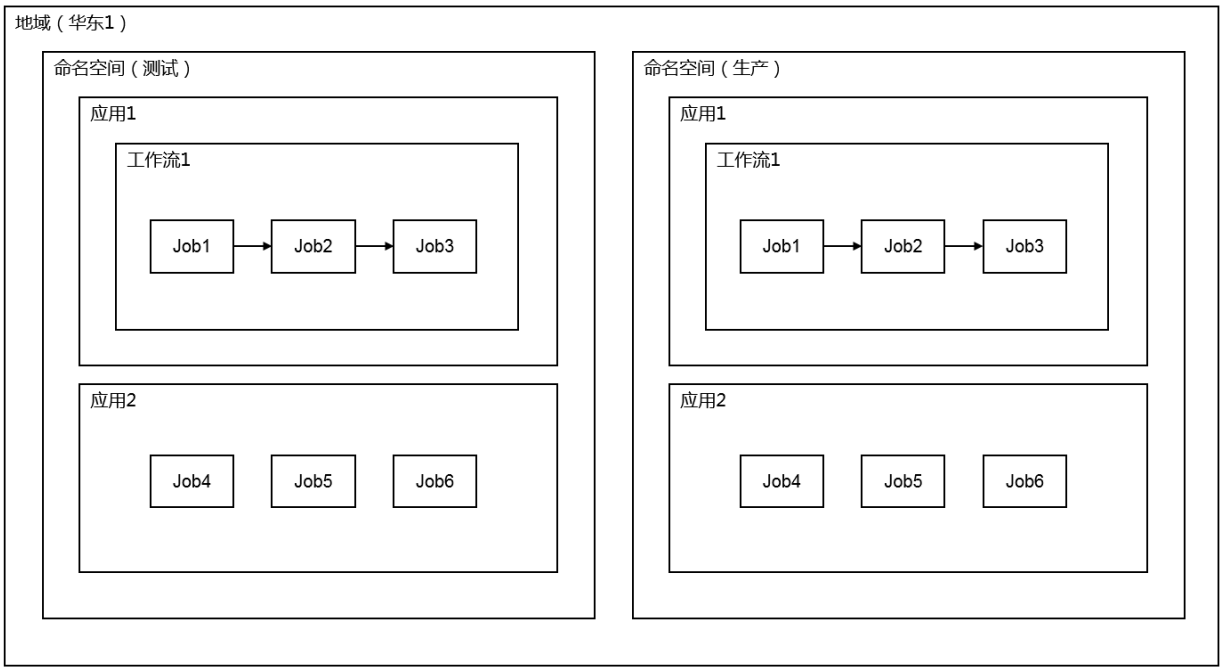
1. 登录分布式任务调度平台。
2. 首次使用并登录分布式任务调度平台，在弹出的对话框中单击**开通**。
3. 在**分布式任务管理（按量后付费）**页面单击**立即购买**。

   付费类型为**后付费类型**，**任务计算量**为**按任务计算量付费**，不可修改。
4. 在**确认订单**页面**服务协议**区域单击**我已阅读并同意分布式任务管理（按量后付费）服务协议**，单击**去支付**。

1.2 创建资源

在使用SchedulerX前，您需要先创建相关资源，包括命名空间、调度任务分组、调度任务和调度工作流。

背景信息



资源	说明	使用场景
命名空间	在具体地域（Region）中，命名空间用于实现资源和服务的隔离。	当您对资源有较高的安全要求时，需要创建命名空间。
应用	在具体的命名空间下，和应用绑定，关联一组机器。	通过groupId绑定应用。
任务	在具体的应用下，任务和一段代码逻辑绑定，用来实现任务调度。	任务是SchedulerX调度的最小单位，用来实现周期性的任务调度。
工作流	在具体应用下，工作流用来实现任务的依赖编排。	工作流是SchedulerX对任务进行依赖编排的封装，支持上下游数据传递。

创建命名空间（可选）

1. 在左侧导航栏单击**命名空间**。
2. 在顶部菜单栏选择**地域**，然后单击**创建命名空间**。
3. 在**创建命名空间**页面输入命名空间的**名称**和**描述**，然后单击**确定**。

   命名空间创建成功，会提示**创建成功**。返回**命名空间**页面，列表中包含刚创建的命名空间信息。

创建应用

1. 在左侧导航栏单击**应用管理**。
2. 在**应用管理**页面的**所属命名空间**列表中选择命名空间，然后单击**创建应用**。

3. 在**创建应用**对话框的**应用名**右侧的文本框中输入应用名，在**应用ID**右侧的文本框中输入应用分组名称，设置高级配置参数（可选），然后单击**确定**。

   应用ID要求在当前命名空间内唯一，否则将创建失败。

高级配置参数说明：

参数	解释	默认值
load5	不能超过客户端机器CPU可用核数	0

参数	解释	默认值
内存使用率	表示近5分钟进程内存平均使用率不能大于该阈值，否则判断客户端机器繁忙。	90%
磁盘使用率	表示磁盘使用率不能大于该值，否则判断客户端机器不健康，状态繁忙。	95%
不触发繁忙机器	机器繁忙时是否继续触发客户端执行。	不触发
无可用机器告警	没有可用机器的时候是否发送告警。	关
任务最大数量	一个分组最多多少个Job。	1000

任务分组创建完成后，自动返回**应用管理**页面，在分组列表中查看分组是否已存在。

2 客户端快速接入 SchedulerX

2.1 Java应用接入SchedulerX

您可以为您的Java应用快速接入SchedulerX，实现分布式任务调度能力。

前提条件

- 创建命名空间（可选）
- 创建应用

Java应用客户端接入SchedulerX

1. 在应用程序的pom.xml文件中添加SchedulerxWorker依赖。

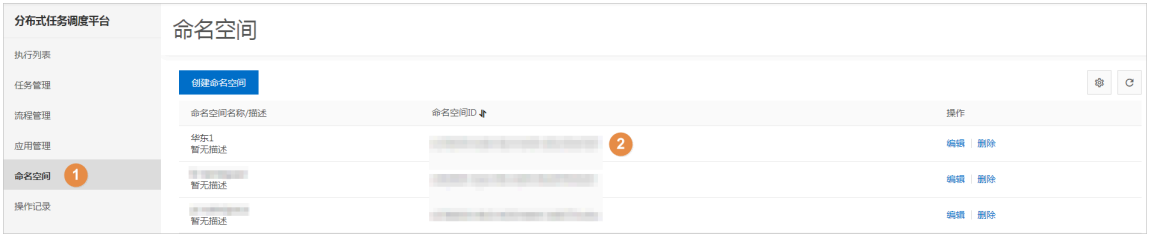
   请参见#unique_8，schedulerx2.version使用最新客户端版本。

   ```
   <dependency>
     <groupId>com.aliyun.schedulerx</groupId>
     <artifactId>schedulerx2-worker</artifactId>
     <version>${schedulerx2.version}</version>
   ```

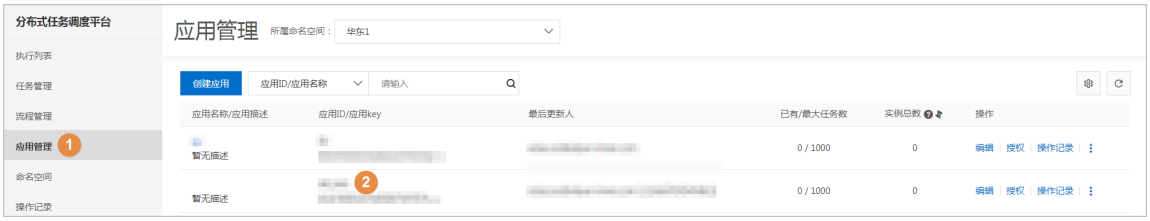
```
</dependency>
```

2. 在main函数中初始化SchedulerxWorker。

   - 初始化SchedulerxWorker时，会用到您部署应用的地域（Region）和对应的Endpoint。详情请参见Endpoint列表。
   - Namespace为命名空间ID，可以在控制台**命名空间**页面获取。



   - GroupId为应用ID，可以在控制台**应用管理**页面获取。



   - AliyunAccessKey和AliyunSecretKey为阿里云账号的AccessKeyID和AccessKeySecret，可以在用户信息管理控制台的**安全信息管理**页面获取。

```
private static void initSchedulerxWorker() throws Exception {
    SchedulerxWorker schedulerxWorker = new SchedulerxWorker();
    schedulerxWorker.setEndpoint("xxxx");
    schedulerxWorker.setNamespace("xxxx");
    schedulerxWorker.setGroupId("xxxx");
    schedulerxWorker.setAliyunAccessKey("xxxx");
    schedulerxWorker.setAliyunSecretKey("xxxx");
    schedulerxWorker.init();
}
```

说明：

- 一个应用如果包含多个业务，或者想把定时任务进行归类，可以建立多个分组，例如应用animals新建了两个分组animals.dogs和 animals.cats。此时不用申请两批实例分别接入这两个分组，在应用客户端中将这两个分组配置到groupId=后面即可，例如groupId=animals.dogs,animals.cats。
- 在初始化SchedulerxWorker客户端时，如果还有其它配置需求，可以参考Schedulerx Worker 配置参数说明添加配置。

3. 在应用中创建类JobProcessor，实现任务调度。

   本文仅介绍如何实现一个最简单的定时打印“**Hello SchedulerX2.0**”的JobProcessor类。

```
package com.aliyun.schedulerx.test.job;

import com.alibaba.schedulerx.worker.domain.JobContext;
import com.alibaba.schedulerx.worker.processor.JavaProcessor;
import com.alibaba.schedulerx.worker.processor.ProcessResult;

@Component
public class MyHelloJob extends JavaProcessor {

    @Override
    public ProcessResult process(JobContext context) throws Exception {
        System.out.println("hello schedulerx2.0");
        return new ProcessResult(true);
    }
}
```

结果验证

1. 客户端接入完成，将该应用发布到阿里云。
2. 在左侧导航栏单击**应用管理**。
3. 在**应用管理**页面查看**实例总数**。
   - 如果**实例总数**为0，则说明应用接入失败。请检查、修改本地应用。
   - 如果**实例总数**不为0，显示接入的实例个数，则说明应用接入成功。在**操作**列单击**查看实例**，即可在**连接实例**对话框中查看实例列表。

后续步骤

应用接入SchedulerX完成后，即可在分布式任务调度平台创建调度任务。详情请参见#unique_10/unique_10_Connect_42_section_ww9_0re_gta。

2.2 Spring应用接入SchedulerX

您可以为您的Spring应用快速接入SchedulerX，实现分布式任务调度能力。

前提条件

- 创建命名空间（可选）

- 创建应用

Spring应用客户端接入SchedulerX

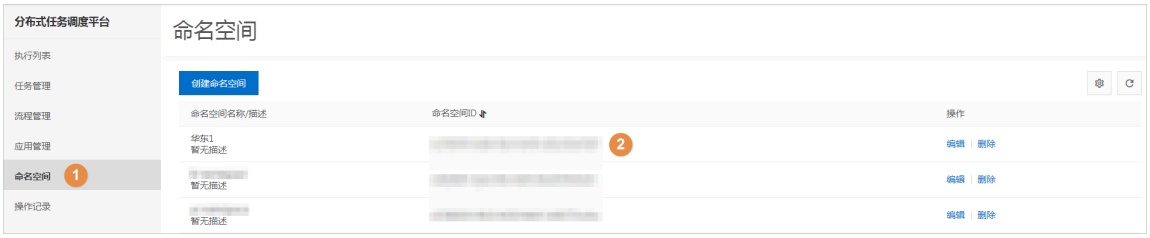
1. 在应用程序的pom.xml文件中添加SchedulerxWorker依赖。

   请参见#unique_8，schedulerx2.version使用最新客户端版本。

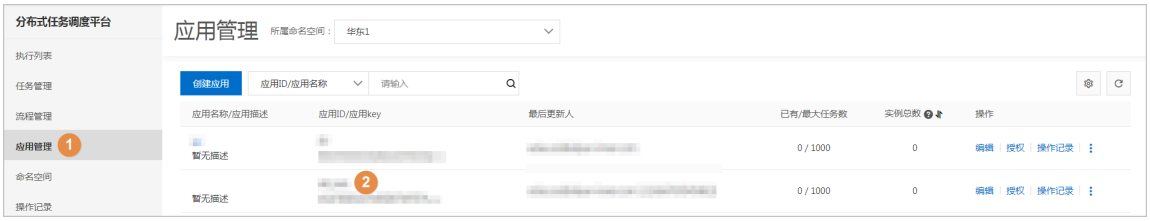
```
<dependency>
   <groupId>com.aliyun.schedulerx</groupId>
   <artifactId>schedulerx2-worker</artifactId>
   <version>${schedulerx2.version}</version>
</dependency>
```

2. 在xml配置文件中初始化SchedulerxWorker（注入Bean）。

   - 初始化SchedulerxWorker时，会用到您部署应用的地域（Region）和对应的Endpoint。详情请参见Endpoint列表。
   - namespace为命名空间ID，可以在控制台**命名空间**页面获取。



   - groupId为应用ID，可以在控制台**应用管理**页面获取。



   - aliyunAccessKey和aliyunSecretKey为阿里云账号的AccessKeyID和AccessKeySecret，可以在用户信息管理控制台的**安全信息管理**页面获取。

```
<bean id="schedulerxWorker" class="com.alibaba.schedulerx.worker.SchedulerxWorker">
   <property name="endpoint">
     <value>${endpoint}</value>
   </property>
   <property name="namespace">
     <value>${namespace}</value>
   </property>
   <property name="groupId">
     <value>${groupId}</value>
   </property>
   <property name="aliyunAccessKey">
     <value>${aliyunAccessKey}</value>
   </property>
   <property name="aliyunSecretKey">
     <value>${aliyunSecretKey}</value>
```

```
    </property>
</bean>
```

说明：

- 一个应用如果包含多个业务，或者想把定时任务进行归类，可以建立多个分组，例如应用animals建了两个分组animals.dogs和animals.cats。此时不用申请两批实例分别接入这两个分组，在应用客户端中将这两个分组配置到groupId=后面即可，例如groupId=animals.dogs,animals.cats。
- 在初始化SchedulerxWorker客户端时，如果还有其它配置需求，可以参考Schedulerx Worker 配置参数说明添加配置。

3. 在应用中创建类JobProcessor，实现任务调度。

   本文仅介绍如何实现一个最简单的定时打印“**Hello SchedulerX2.0**”的JobProcessor类。

```
package com.aliyun.schedulerx.test.job;

import com.alibaba.schedulerx.worker.domain.JobContext;
import com.alibaba.schedulerx.worker.processor.JavaProcessor;
import com.alibaba.schedulerx.worker.processor.ProcessResult;

@Component
public class MyHelloJob extends JavaProcessor {

    @Override
    public ProcessResult process(JobContext context) throws Exception {
        System.out.println("hello schedulerx2.0");
        return new ProcessResult(true);
    }
}
```

结果验证

1. 客户端接入完成，将该应用发布到阿里云。
2. 在左侧导航栏单击**应用管理**。
3. 在**应用管理**页面查看**实例总数**。
   - 如果**实例总数**为0，则说明应用接入失败。请检查、修改本地应用。
   - 如果**实例总数**不为0，显示接入的实例个数，则说明应用接入成功。在**操作**列单击**查看实例**，即可在**连接实例**对话框中查看实例列表。

后续步骤

应用接入SchedulerX完成后，即可在分布式任务调度平台创建调度任务。详情请参见#unique_10/unique_10_Connect_42_section_ww9_0re_gta。

2.3 Spring Boot应用接入SchedulerX

您可以为您的Spring Boot应用快速接入SchedulerX，实现分布式任务调度能力。

前提条件

- 创建命名空间（可选）
- 创建应用

Spring Boot应用客户端接入SchedulerX

1. 在应用程序的pom.xml文件中添加SchedulerxWorker依赖。

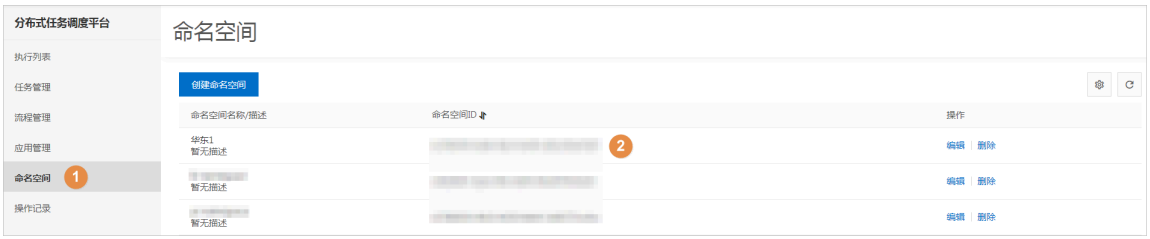
   请参见#unique_8，schedulerx2.version使用最新客户端版本。

```
<dependency>
 <groupId>com.aliyun.schedulerx</groupId>
 <artifactId>schedulerx2-spring-boot-starter</artifactId>
 <version>${schedulerx2.version}</version>
```

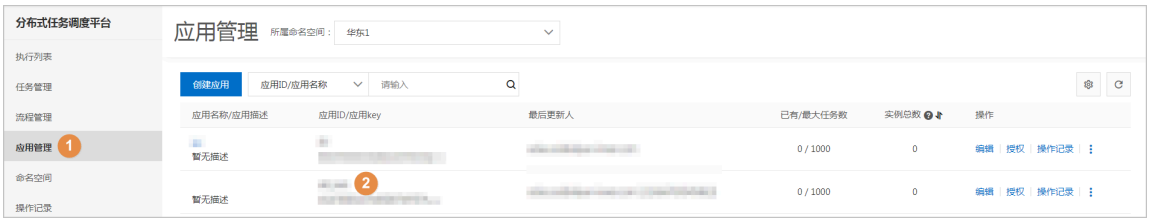
```
</dependency>
```

2. 在application.properties文件中设置相关参数，初始化SchedulerxWorker。

   - 初始化SchedulerxWorker时，会用到您部署应用的地域（Region）和对应的Endpoint。详情请参见Endpoint列表。
   - namespace为命名空间ID，可以在控制台**命名空间**页面获取。



   - groupId为应用ID，可以在控制台**应用管理**页面获取。



   - aliyunAccessKey和aliyunSecretKey为阿里云账号的AccessKeyID和AccessKeySecret，可以在用户信息管理控制台的**安全信息管理**页面获取。

```
spring.schedulerx2.endpoint=${endpoint}
spring.schedulerx2.namespace=${namespace}
spring.schedulerx2.groupId=${groupId}
spring.schedulerx2.aliyunAccessKey=${aliyunAccessKey}
spring.schedulerx2.aliyunSecretKey=${aliyunSecretKey}
```

> **说明：**
> 一个应用如果包含多个业务，或者想把定时任务进行归类，可以建立多个分组，例如应用animals新建了两个分组animals.dogs和animals.cats。此时不用申请两批实例分别接入这两个分组，在应用客户端中将这两个分组配置到groupId=后面即可，例如groupId=animals.dogs,animals.cats。

3. 在应用中创建类JobProcessor，实现任务调度。

   本文仅介绍如何实现一个最简单的定时打印“**Hello SchedulerX2.0**”的JobProcessor类。

```
package com.aliyun.schedulerx.test.job;

import com.alibaba.schedulerx.worker.domain.JobContext;
import com.alibaba.schedulerx.worker.processor.JavaProcessor;
import com.alibaba.schedulerx.worker.processor.ProcessResult;

@Component
public class MyHelloJob extends JavaProcessor {
```

```
    @Override
    public ProcessResult process(JobContext context) throws Exception {
        System.out.println("hello schedulerx2.0");
        return new ProcessResult(true);
    }
}
```

结果验证

1. 客户端接入完成，将该应用发布到阿里云。
2. 在左侧导航栏单击**应用管理**。
3. 在**应用管理**页面查看**实例总数**。
   - 如果**实例总数**为0，则说明应用接入失败。请检查、修改本地应用。
   - 如果**实例总数**不为0，显示接入的实例个数，则说明应用接入成功。在**操作**列单击**查看实例**，即可在**连接实例**对话框中查看实例列表。

后续步骤

应用接入SchedulerX完成后，即可在分布式任务调度平台创建调度任务。详情请参见#unique_10/unique_10_Connect_42_section_ww9_0re_gta。

2.4 Agent接入（调度任务）

如果您无需为应用接入调度任务，仅想创建一个独立的脚本调度任务，也可以使用SchedulerX提供的Agent快速创建脚本任务。

前提条件

- 创建命名空间（可选）
- 创建应用

背景信息

脚本任务目前支持Shell、Python和Go三种语言。

运行环境要求为JRE 1.8及以上版本。

操作步骤

1. 下载schedulerxAgent-1.1.0。

2. 解压下载的压缩包。
3. 进入schedulerxAgent/conf目录，编辑agent.properties文件，添加**endpoint**、**namespace**（命名空间 ID）、**groupId**（应用 ID）和阿里云账号的**aliyunAccessKey**（AccessKeyID） 和**aliyunSecretKey**（AccessKeySecret）。

```
endpoint=
namespace=
groupId=
aliyunAccessKey=
aliyunSecretKey=
```

地域（Region）和Endpoint的关系请参见Endpoint列表。

4. 进入schedulerxAgent/bin目录，执行start-1g.sh命令启动SchedulerX。

> **说明:**
> start-1g.sh仅为示例，您需要根据任务负载及机器配置情况执行对应的命令，如start-2g.sh、start-4g.sh或start-8g.sh。

如果您想停止任务调度，可执行stop.sh命令。

2.5 在本地接入公网测试环境

本文介绍如何将本地接入云上公网测试环境来进行本地开发、调试和验证。

前提条件

- 开通SchedulerX（免费）
- 创建应用

> **说明:**
> 在创建应用时，请选择**公网**地域。

背景信息

云上公网环境不保证稳定性和数据安全性，只是方便本地开发调试使用。请测试验证完成后，删除任务并在线上正式环境创建相同的任务。

操作步骤

1. 在应用pom.xml文件中添加SchedulerxWorker依赖。

   针对不同应用，在初始化SchedulerxWorker时会有所不同。

   - 普通Java或Spring应用

```
<dependency>
```

```
    <groupId>com.aliyun.schedulerx</groupId>
    <artifactId>schedulerx2-worker</artifactId>
    <version>${schedulerx2.version}</version>
</dependency>
```

- Spring Boot应用

```
<dependency>
 <groupId>com.aliyun.schedulerx</groupId>
 <artifactId>schedulerx2-spring-boot-starter</artifactId>
 <version>${schedulerx2.version}</version>
</dependency>
```

2. 初始化SchedulerxWorker。

针对不同应用，在初始化SchedulerxWorker的时候会有所不同。

- 普通Java应用

在main函数中初始化SchedulerxWorker。

```
private static void initSchedulerxWorker() throws Exception {
    // 测试环境为acm.aliyun.com
    String endpoint = "xxx";
    // 为测试环境命名空间里面的namespace
    String namespace = "xxx-xxx";
    // 替换之前创建的分组的Group ID
    String groupId = "schedulerx.defaultGroup";
    SchedulerxWorker schedulerxWorker = new SchedulerxWorker();
    schedulerxWorker.setGroupId(groupId);
    schedulerxWorker.setNamespace(namespace);
    schedulerxWorker.setEndpoint(endpoint);
    schedulerxWorker.setAliyunAccessKey("xxxx");
    schedulerxWorker.setAliyunSecretKey("xxxx");
    schedulerxWorker.init();
    LOGGER.info("SchedulerxWorker init success. groupId={}", groupId);
}
```

说明:

- 一个应用如果包含多个业务，或者想把定时任务进行归类，可以建立多个分组，例如应用animals新建了两个分组animals.dogs和 animals.cats。此时不用申请两批实例分别接入这两个分组，在应用客户端中将这两个分组配置到groupId=后面即可，例如groupId=animals.dogs,animals.cats。
- 在初始化SchedulerxWorker客户端时，如果还有其它配置需求，可以参考Schedulerx Worker 配置参数说明添加配置。

- Spring应用

在xml配置文件中注入SchedulerxWorker Bean。

```
<bean id="schedulerxWorker" class="com.alibaba.schedulerx.worker.SchedulerxWorker">
    // 测试环境为acm.aliyun.com
    <property name="endpoint">
```

```
      <value>${endpoint}</value>
    </property>
    <property name="namespace">
      <value>${namespace}</value>
    </property>
    <property name="groupId">
      <value>${groupId}</value>
    </property>
    <property name="aliyunAccessKey">
      <value>${aliyunAccessKey}</value>
    </property>
    <property name="aliyunSecretKey">
      <value>${aliyunSecretKey}</value>
    </property>
</bean>
```

- Spring Boot应用

  在application.properties文件中添加如下配置：

```
// 测试环境为acm.aliyun.com
spring.schedulerx2.endpoint=acm.aliyun.com
// 测试环境为8080
spring.schedulerx2.endpointPort=8080
// 为测试环境命名空间里面的namespace
spring.schedulerx2.namespace=${namespace}
// 填写之前创建的分组的Group ID
spring.schedulerx2.groupId=${groupId}
spring.schedulerx2.aliyunAccessKey=${aliyunAccessKey}
spring.schedulerx2.aliyunSecretKey=${aliyunSecretKey}
```

> **说明：**
>
> 一个应用如果包含多个业务，或者想把定时任务进行归类，可以建立多个分组，例如应用animals新建了两个分组animals.dogs和animals.cats。此时不用申请两批实例分别接入这两个分组，在应用客户端中将这两个分组配置到groupId=后面即可，例如groupId=animals.dogs,animals.cats。

3. 在应用中创建类JobProcessor，实现任务调度。

   本文仅介绍如何实现一个最简单的定时打印**“Hello SchedulerX2.0”**的JobProcessor类。

```
package com.aliyun.schedulerx.test.job;

import com.alibaba.schedulerx.worker.domain.JobContext;
import com.alibaba.schedulerx.worker.processor.JavaProcessor;
import com.alibaba.schedulerx.worker.processor.ProcessResult;

@Component
public class MyHelloJob extends JavaProcessor {

    @Override
    public ProcessResult process(JobContext context) throws Exception {
        System.out.println("hello schedulerx2.0");
        return new ProcessResult(true);
    }
```

```
}
```

4. 运行本地应用。

结果验证

1. 客户端接入完成，将该应用发布到阿里云。
2. 在左侧导航栏单击**应用管理**。
3. 在**应用管理**页面查看**实例总数**。
   - 如果**实例总数**为0，则说明应用接入失败。请检查、修改本地应用。
   - 如果**实例总数**不为0，显示接入的实例个数，则说明应用接入成功。在**操作**列单击**查看实例**，即可在**连接实例**对话框中查看实例列表。

2.6 容器服务Kubernetes版接入SchedulerX

前提条件

- #unique_16
- 开通SchedulerX（免费）
- 了解Kubernetes基本概念和操作，例如kubeconfig

背景信息

使用SchedulerX需要理解以下3个概念：

- 分组
  - 客户端的组织单位
  - 任务的组织单位
- 任务：调度单位，需要创建任务，配置所属分组。
- 客户端：任务执行节点，需要添加SchedulerX客户端，实现对应Java任务处理接口，配置所属分组启动名为SchedulerxWorker的Agent。

三者的关系为：任务只能调度到对应分组的客户端。例如创建了分组group-sample，在该分组下创建任务job-sample，同时配置所属分组为group-sample来启动客户端agent1、agent2和agent3，则调度任务job-sample就会被调度到客户端agent1、agent2和agent3上面运行。

关于SchedulerX的更多信息，请参见分布式任务调度 SchedulerX。

在容器服务Kubernetes版中安装SchedulerX组件

1. 登录容器服务控制台。
2. 在左侧导航栏选择**市场** > **应用目录**。
3. 在**应用目录**页面搜索并单击**ack-schedulerx**。
4. 在**ack-schedulerx**页面右侧**集群**列表中选择集群，然后单击**创建**。
   - 如果是在ACK购买的K8s集群，选择集群后，直接创建。

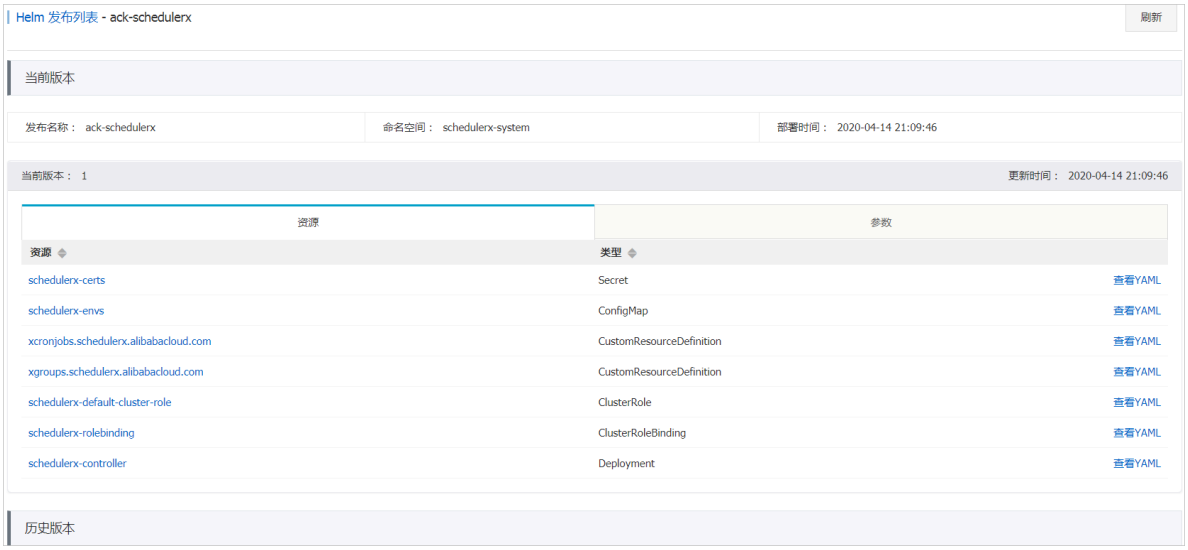
   - 如果是自建纳管到ACK的K8s集群，单击**参数**页签，设置参数**controller.aliyun_accessKey_secret_name**。

说明:

命名空间为**schedulerx-system**，不可修改。

安装SchedulerX组件大约需要2分钟，请耐心等待。

创建成功后，会自动跳转到**Helm 发布列表 - ack-schedulerx**页面，检查安装结果。



创建分组

1. 创建xgroup.yaml，设置相关参数。

   xgroup.yaml示例如下：

```
apiVersion: schedulerx.alibabacloud.com/v1alpha1
kind: XGroup
metadata:
  name: xgroup-sample
spec:
  appName: ackApp
```

   xgroup.yaml配置参数说明：

   - GVK（Group、Version 和 Kind）信息
     - apiVersion：格式为<group>/<version>，例如schedulerx.alibabacloud.com/v1alpha1。
     - kind: XGroup
   - spec信息

参数名	类型	默认值	是否必填	说明
appName	string	无	必填	应用名，用户自定义，用于后续管理。

2. 执行kubectl apply -f xgroup.yaml命令，创建任务分组。

> **说明：**
>
> - 分组创建后，不允许更新。如果需要更新，请删除分组后重新创建。
> - 分组创建后，如果新建了任务或添加了客户端，即该分组下包含任务或客户端，则无法删除分组，需要先移除任务和客户端。

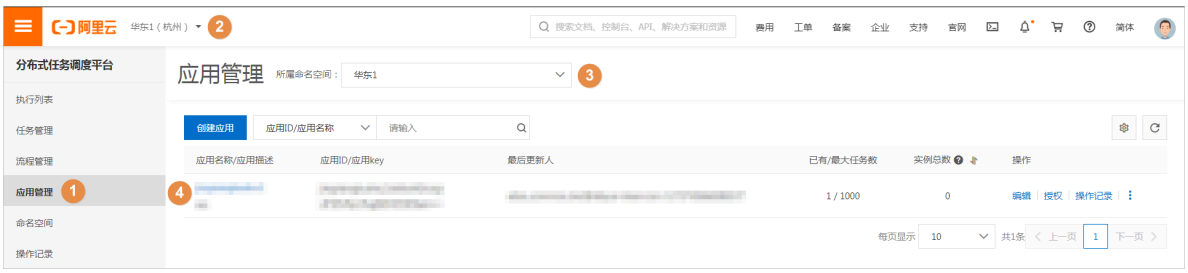
3. 执行kctl get xgroup xgroup-sample -o yaml命令，查看xgroup资源。

打印结果如下：

```
apiVersion: schedulerx.alibabacloud.com/v1alpha1
kind: XGroup
metadata:
  annotations:
    kubectl.kubernetes.io/last-applied-configuration: |
      {"apiVersion":"schedulerx.alibabacloud.com/v1alpha1","kind":"XGroup","metadata
":{"annotations":{},"name":"xgroup-sample","namespace":"default"},"spec":{"
appName":"ackApp"}}
  creationTimestamp: "2019-09-19T04:21:12Z"
  finalizers:
  - GroupCleanup
  generation: 1
  name: xgroup-sample
  namespace: default
  resourceVersion: "143176160"
  selfLink: /apis/schedulerx.alibabacloud.com/v1alpha1/namespaces/default/
xgroups/xgroup-sample
  uid: e9a1xxxx-xxxx-11e9-xxxx-be9f1a43xxxx
spec:
  appName: ackApp
status:
  appGroupId: 283
  conditions:
  - lastTransitionTime: "2019-09-19T04:21:12Z"
    lastUpdateTime: "2019-09-19T04:21:12Z"
    reason: CreateGroupSuccess
    status: "True"
    type: Ready
```

status的conditions下**status**为**True**，且**type**为**Ready** ，则表示分组创建成功。

也可以登录分布式任务调度平台，查看分组创建情况。



创建任务

1. 创建xcronjob.yaml，设置相关参数。

   xcronjob.yaml示例如下：

```
apiVersion: schedulerx.alibabacloud.com/v1alpha1
kind: XCronJob
metadata:
  name: xcronjob-sample
spec:
  group: xgroup-sample
  jobType: java
  jobProcessor: processor.SimpleJobProcessor
  executeMode: standalone
  timeExpression: 0 0 2 * * ?
```

   xcronjob.yaml配置参数说明：

   - GVK（Group、Version 和 Kind）信息
     - apiVersion：格式为<group>/<version>，例如schedulerx.alibabacloud.com/v1alpha1。
     - kind: XCronJob
   - spec信息

参数名	类型	默认值	是否必填	说明
group	string	无	是	该任务所属分组名。
jobType	string	java	否	任务类型，指实现任务的编程语言，当前支持Java、Python、Shell和Go。详情请参见#unique_17/unique_17_Connect_42_sect
jobProcessor	string	无	否（有条件）	任务实现全限定类名，如果jobType == java，该字段必填。
content	string	无	否（有条件）	任务实现代码，如果jobType == java，该字段必填。

参数名	类型	默认值	是否必填	说明
executeMode	string	standalone	否	任务执行模式，当前支持单击运行、广播运行、并行计算、内存网格、网格计算和分片运行。详情请参见#unique_17/unique_17_Connect_42_sect
description	string	无	否	任务描述
timeType	int	1	否	任务调度表达式类型，当前支持cron(1)、fix_rate(3)、seco详情请参见#unique_18、#unique_19
timeExpression	string	无	是	任务调度表达式，例如： - cron：`0 0 2 * * ?`要确保频率大于分钟级。 - fixed_rate：`30`(>0)，单位是s，每30s运行一次。 - second_delay：`2`(1-60)，单位是s, 上次运行结束后延迟2s再运行一次。
parameters	string	无	否	任务参数，可以在任务运行时从上下文获取。
maxConcurrency	int	1	否	最大同时运行任务实例数，默认为1。超过该并发度的调度实例会被忽略。

参数名	类型	默认值	是否必填	说明
retryMaxAttempts	int	0	否	失败重试次数，默认为0，不重置。
retryInterval	int	30	否	失败重试间隔，单位s，默认为30s。

通过示例可以看到指定的group是刚刚创建的xgroup-sample，默认使用Cron调度表达式，Java任务类型，处理的接口类名为processor.SimpleJobProcessor。

2. 执行kubectl apply -f xcronjob.yaml命令，创建任务。
3. 执行kctl get xcronjob xcronjob-sample -o yaml命令，查看xcronjob资源。

打印结果如下：

```
apiVersion: schedulerx.alibabacloud.com/v1alpha1
kind: XCronJob
metadata:
  creationTimestamp: "2019-09-19T06:33:13Z"
  finalizers:
  - JobCleanup
  generation: 1
  name: xcronjob-sample
  namespace: default
  ownerReferences:
  - apiVersion: schedulerx.alibabacloud.com/v1alpha1
    blockOwnerDeletion: true
    controller: true
    kind: XGroup
    name: xgroup-sample
    uid: e9a1xxxx-xxxx-11e9-xxxx-be9f1a43xxxx
  resourceVersion: "143570391"
  selfLink: /apis/schedulerx.alibabacloud.com/v1alpha1/namespaces/default/
xcronjobs/xcronjob-sample
  uid: 5b5exxxx-daa7-xxxx-a76d-4af3xxxx44xx
spec:
  executeMode: standalone
  group: xgroup-sample
  jobProcessor: processor.SimpleJobProcessor
  jobType: java
  timeExpression: 0 0 2 * * ?
status:
  conditions:
  - lastTransitionTime: "2019-09-19T06:33:13Z"
    lastUpdateTime: "2019-09-19T06:33:14Z"
    reason: JobUpdateSuccess
    status: "True"
    type: Ready
```

```
jobId: 1304
```

status的conditions下**status**为**True**，且**type**为**Ready**，则表示任务创建成功。

也可以登录分布式任务调度平台，查看任务创建情况。

创建客户端

1. 客户端接入SchedulerX。

   根据应用类型不同，客户端的接入方式也不同。

   - Java应用：客户端接入详情请参见Java应用接入SchedulerX。
   - Spring应用：客户端接入详情请参见Spring应用接入SchedulerX。
   - Spring Boot应用：客户端接入详情请参见Spring Boot应用接入SchedulerX。

2. 将客户端打包，并通过Dockerfile创建镜像，并上传到镜像仓库。

   Dockfile示例如下：

   ```
   FROM openjdk:8-jdk-alpine
   COPY ./target/schedulerx-k8s-demo-1.0-SNAPSHOT-spring-boot.jar app.jar
   ENTRYPOINT ["java","-jar","/app.jar"]
   ```

   SchedulerX提供了已经上传到镜像仓库的镜像Demo（`registry.cn-shanghai.aliyuncs.com/schedulerx/demo:latest`），方便您体验、试用。

3. 创建xagentpool.yaml，设置相关参数。

   xagentpool.yaml示例如下：

   ```
   apiVersion: schedulerx.alibabacloud.com/v1alpha1
   kind: XAgentPool
   metadata:
     name: xagentpool-sample
   spec:
     group: xgroup-sample
     replicas: 2
     template:
       containers:
         - name: standalone
   ```

```
    image: registry.cn-shanghai.aliyuncs.com/schedulerx/demo:latest
```

xagentpool.yaml配置参数说明：

- GVK（Group、Version 和 Kind）信息
  - apiVersion：格式为<group>/<version>，例如schedulerx.alibabacloud.com/v1alpha1。
  - kind: XGroup
- spec信息

参数名	类型	默认值	是否必填	说明
group	string	无	是	该任务所属分组名。
replicas	int	无	是	执行器个数。
workloadType	string	Deployment	否	执行器工作负载，默认是Deployment，可选值还有DaemonSet。
template	PodSpec	无	是	任务执行器Pod模板。

通过示例可以看到指定的group是刚刚创建的xgroup-sample，运行两个执行器，执行器镜像为image，该image即客户端镜像。

4. 执行kubectl apply -f xagentpool.yaml命令，创建客户端。
5. 执行kctl get xagentpool xagentpool-sample -o yaml命令，查看xagentpool资源。

打印结果如下：

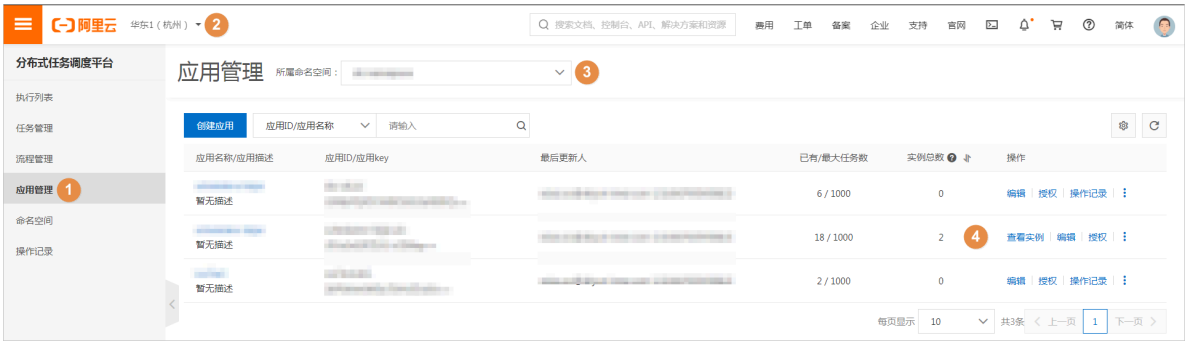
```
apiVersion: schedulerx.alibabacloud.com/v1alpha1
kind: XAgentPool
metadata:
  annotations:
    kubectl.kubernetes.io/last-applied-configuration: |
      {"apiVersion":"schedulerx.alibabacloud.com/v1alpha1","kind":"XAgentPool","
metadata":{"annotations":{},"name":"xagentpool-sample","namespace":"default"},"
spec":{"group":"xgroup-sample","replicas":2,"template":{"containers":[{"image":"
registry.cn-shanghai.aliyuncs.com/schedulerx/demo:latest","name":"standalone"}]}}}
  creationTimestamp: "2019-09-25T10:11:39Z"
  generation: 1
  name: xagentpool-sample
  namespace: default
  ownerReferences:
  - apiVersion: schedulerx.alibabacloud.com/v1alpha1
    blockOwnerDeletion: true
    controller: true
    kind: XGroup
```

```
    name: xgroup-sample
    uid: c920xxxx-df7c-xxxx-a76d-xxxx350bxxxx
  resourceVersion: "170986882"
  selfLink: /apis/schedulerx.alibabacloud.com/v1alpha1/namespaces/default/
xagentpools/xagentpool-sample
  uid: dd83xxxx-df7c-xxxx-a156-xxxx1a43xxxx
spec:
  group: xgroup-sample
  replicas: 2
  template:
    containers:
    - image: registry.cn-shanghai.aliyuncs.com/schedulerx/demo:latest
      name: standalone
      resources: {}
status:
  conditions:
  - lastTransitionTime: "2019-09-25T10:11:40Z"
    lastUpdateTime: "2019-09-25T10:11:40Z"
    reason: update deployment sucess
    status: "True"
    type: Ready
```

status的conditions下**status**为**True**，且**type**为**Ready**，则表示任务创建成功。

实际上，每个agentPool的创建都会在相同命名空间下创建名为[agentPoolName]-deployment的Deployment或者[agentPoolName]-daemonset的DeamonSet，可以自行查看

也可以登录分布式任务调度平台，查看客户端创建情况。



单击**查看实例**，可以查看客户端实例的详细信息。

删除SchedulerX组件

当不再需要SchedulerX组件时，可以删除chedulerX组件。

说明：

- 在删除SchedulerX组件之前，请确保集群内的所有XGroup、XCronJob和XAgentPool类型资源都已经删除完毕，否则无法删除CRDs，并且会导致下次安装出现异常。
- XGroup删除之后，对应的SchedulerX应用分组不会自动删除，需要登录分布式任务调度平台，手动删除。

1. 登录容器服务控制台。
2. 在左侧导航栏选择**应用** > **发布**。
3. 在**发布**页面**ack-schedulerx**的**操作**列单击**删除**。
4. 在**删除应用**对话框中单击**确定**。

2.7 Endpoint列表

初始化SchedulerxWorker时，会用到您部署应用的地域（Region）和对应的Endpoint。

地域（Region）	Endpoint	用途
华东1（杭州）	addr-hz-internal.edas.aliyun.com	线上生产环境
华东2（上海）	addr-sh-internal.edas.aliyun.com	线上生产环境
华北2（北京）	addr-bj-internal.edas.aliyun.com	线上生产环境
华北3（张家口）	addr-cn-zhangjiakou-internal.edas.aliyun.com	线上生产环境
华南1（深圳）	addr-sz-internal.edas.aliyun.com	线上生产环境
美国（弗吉尼亚）	addr-us-east-1-internal.acm.aliyun.com	线上生产环境
公网	acm.aliyun.com	本地接入测试环境，不能用于生产。