

# 快速开始

为了无法计算的价值 | [] 阿里云

# 快速开始

# 开通服务

### 1. 创建阿里云账号

如果您还没有阿里云账号,请登陆**阿里云官网**,点击右上角"免费注册"创建阿里云账号,否则直接登陆阿里 云控制台(主账号登录)。

# 2. 开通 BatchCompute

### 2.1 定位 BatchCompute 产品

点菜单中"产品",在"弹性计算"中找到批量计算(BatchCompute)进入产品主页。

	(-)阿里云					Q.搜索		
	产品与服务 >							×
۲	容器镜像服务	Q 请输入关键词						
ଷ	文件存储 NAS	最近访问						
	云服务器 ECS	访问控制	*	批量计算	*	容器镜像服务	*	弹性计算
	云数据库 RDS 版	对象存储 OSS	*	文件存储 NAS	*	专有网络 VPC		数据库
								存储与CDN
A	负载均衡	弹性计算		数据库		存储与CDN		网络
a	对象存储 OSS	云服务器 ECS	*	云数据库 POLARDB		对象存储 OSS	*	分析
5	消息服务 MNS	负载均衡	*	云数据库 RDS 版	*	文件存储 NAS	*	云通信
	道良队列 RocketMO	弹性伸缩		云数据库 RDS PostgreSQL 版		表格存储		监控与管理
۲		容器服务		云数据库 MongoDB 版		归档存储		应用服务
4	访问控制	容器服务 Kubernetes 版		云数据库 Redis 版		CDN		互联网中间件
ØII	批量计算	容器镜像服务	*	云数据库 Memcache 版		PCDN		消息队列 MQ
		资源编排		云数据库 HybridDB for MySQL		全站加速		移动云
		批量计算	*	云数据库 HBase 版		云存储网关		视频服务
		函数计算		云数据库 OceanBase		智能云相册		大数据 (数加)
		弹性高性能计算		时序时空数据库 TSDB	\$	混合云备份		安全 (云盾)
		轻量应用服务器		分析型数据库 PostgreSQL版		混合云容灾		域名与网站 (万网)
		图形工作站		分析型数据库 MySQL 版		安全加速 SCDN		云市场
		弹性容器实例 ECI		数据传输服务 DTS		智能媒体管理		支持与服务
		Web应用托管服务(Web+)		数据管理 DMS		闪电立方		

## 2.2 开通 BatchCompute 服务。

点击"授权",则开通批量计算。



主账号只能通过此方法开通批量计算服务

# 3. 开通确认

开通成功后控制台概览如下:

	华北3(张家口) ▼	Q 搜索	
批量计算	概览		
极宽	可田米刑		
作业列表 重都列表 App利表 镜像列表	*2/17天主と 可用的技業主要(共和な) ecs.c5.large(2株/4GB) ecs.arl.medium(2株/4GB) ecs.arl.medium(2株/4GB) ecs.arl.medium(2株/4GB) ecs.arl.arge(2株/4GB) ecs.arl.arge(2株/4GB) ecs.arl.arge(2株/4GB) ecs.arl.arge(2株/4GB) ecs.arl.arge(2株/4GB) ecs.arl.arge(4株/4GB) ecs.arl.arge(4K/4G	可用約克价实例类型 ecs.s5.arge(2度/4/08) ecs.arf.medium(2度/4/08) ecs.arf.medium(2度/4/08) ecs.arf.medium(2度/4/08) ecs.arf.medium(2度/6/08) ecs.arf.medium(2E/6/68) ecs.arf.medium(2E/6/68) ecs.arf.medium(2E/6/68) ecs.arf.medium(2E/6/68) ecs.arf.medium(2E/6/68) ecs.arf.medium(2E/6/68) ecs.arf.medium(2E/6/68) ecs.arf.medium(2E/68	可用約資源类型 OnDemand (技術) Spot (泉作) 可用約気K盘类型 cloud_std(ticinety (高校五島) cloud_std(ticinety (高校五島) cloud_std(ticinety (高校五島) cloud_std(ticinety (高校五島) cloud_std(ticinety (高校五島)

# 4. RAM 信息确认

#### 4.1 登录到 RAM 访问控制

费用 工单 备案 ①	È业 支持与服务 []	□ <u>↓</u>
		基本资料 实名认证 安全设置
新手入门	用户指南	● 安全管控
		<ul> <li>访问控制 2</li> </ul>
公告		accesskeys
哲尤数据		● 会员积分
		▲ 推荐返利后台
		退出管理控制台

#### 4.2 角色确认

批量计算服务开通后在 RAM 角色管理中可以看到 AliyunBatchComputeDefaultRole 已经创建成功;若 AliyunBatchComputeDefaultRole 正常生成,则点击此处再次确认授权。

概览		1	RAM角	自色管理		
人员管理	^		什么是RA	M角色?		
用户组			RAM角色机	l.制是向您信任的实体(eg, RAM用户、某	1个应用或阿里云服务)进行授权的一种安全方法	法。根据不同应用场景,受信任的实体可能有如下一些例子:
用户			- 您云账户 - 其他云账/	下的一个RAM用户(可能是代表一个移动 户中的RAM用户(需要进行跨账户的资源	App的后端服务) (访问)	
设置			- ECS实例_ - 某些阿里;	上运行的应用程序代码(需要对云资源执: 云服务(需要对您账户中的资源进行操作	行操作) 才能提供服务)	
SSO 管理			- 企业的身f RAM角色颁	份提供商IdP,可以用于角色SSO 11发短时有效的访问令牌(STS令牌),使其	成为一种更安全的授予访问权限的方法。	
权限管理	~		特别说明:			
授权			KAM用包个	同于传统的教科书式角色(具含义是指一	-组权限集)。如果您需要使用教科书式用笆的1	功能,请参考RAM视展策略(Policy)。
权限策略管理			新建RAM角	输入角色名称或备注	Q	
RAM角色管理	1		RA	AM角色名称		酱注
OAuth应用管理		1		iyunBatchComputeDefaultRole 2		BatchCompute默认使用此角色来访问您在其他云产品中的资源
				iyunECIContainerGroupRole		弹性容器实例(ECI)默认使用此角色来访问您在其他云产品中的资源
				iyunMgwVpcDeployRole		MGW默认使用此角色来访问您在其他云产品中的资源。
				iyunNASManageENIRole		文件存储服务默认使用此角色来访问您在其他云产品中的资源
				iyunOSSTokenGeneratorRole		访问OSS的权限

注意:任何时候不要删除 AliyunBatchComputeDefaultRole 角色;若该角色被删除则主账号和子账号都 无法使用批量计算服务,需要重新执行1~4 的步骤方可。

# 5. 子账号开通 BatchCompute 服务。

若以主账号的形式提交作业则忽略本步骤;否则按本步骤创建子账号并对子账号授权。

#### 5.1 创建子账号

以阿里云主账号访问 RAM 控制台,在RAM控制台,单击人员管理 – 用户。

■ (-)阿	里云		0	1. 搜索	费用	工单	备案	企业	支持与服务	▶_	₫.	Ä	简体中文	:
RAM访问控制		RAM访	问控制 / 用户											
概览		用户	5											
人页管理 用户组	^	RAM 通常	1用户是一个身份实体,它通 : <b>的操作步骤如下</b> :	「常代表您的组织中需	需要访问云资调	的人员或	成应用程序	予。						×
用户		1.创 2.添	建用户,并为用户设置登录 加用户到用户组(需要先创	密码(用户登录控制 建用户组并完成对用	台场景)或创 户组的授权)	I Access	Key (应	用程序订	l用API场景)					
权限管理	^	。2 新建	用户登录名称	∨ 请输入	Q								(	C
授权			用户登录名称/显示名称		备注	ŧ				创建时间	1	操作	F	
权限策略管理 RAM角色管理				onaliyun.com	I.					2019年5 24日 17:50:11	5月	添加 添加	O到用户组 O权限 删除	*

#### (-)阿里云 Q RAM访问控制 / 用户 / 新建用户 RAM访问控制 ← 新建用户 概览 人员管理 \* 用户账号信息 登录名称 🕝 显示名称 🕝 用户组 1 2 gen\_test @ \_\_\_\_\_\_n.com gen\_test 用户 + 添加用户 设置 访问方式 🕜 SSO 管理 3 ☑ 控制台密码登录 用户使用账号密码进行阿里云控制台访问 权限管理 ✓ 编程访问 授权 控制台密码 ○ 自动生成默认密码 权限策略管理 4 ● 自定义登录密码 RAM角色管理 ..... OAuth应用管理 要求重置密码 ○ 用户在下次登录时必须重置密码 ◎ 无需重置 5 多因素认证 ○ 要求开启MFA认证 6 ● 不要求 7 返回

#### 按如下步骤创建子账号,账号名称、访问方式可以自由选择。

#### 5.2 对子账号授权

保存用户信息,并为子用户授予权限。

=	(-)阿里			Q 搜索			费用 ]	〔单 备案	企业	支持与服务	>	۵.	「日本」 前位	*中文
RAM	访问控制		RAM访问	<b>泊控制 / 用户 /</b> 新建用户										
概览			←新	建用户										
人员的	ê理 ●/0	^	<ol> <li>若升</li> </ol>	干通编程访问,请及时保存AccessKey 信息,页面并	¢闭后将无法	再次获取信息。								
用	⊃组		用户信息											
101			下载C	SV文件										
1000	6IB		× 🔽	用户登录名称	状态	登录密码	Access	Key ID	Acc	cessKeySecret			B	聚作
RAM	角色管理	Ŷ	2 🔁	gen_test@onaliyun.com	• 成功		LTINICI		riti				100	自复制
OAut	h应用管理			添加到用户组 3 添加权限										
			返回											

分别搜索并添加以下权限:

- AliyunECSFullAccess
- AliyunVPCFullAccess
- AliyunOSSFullAccess
- AliyunBatchComputeFullAccess
- AliyunContainerRegistryFullAccess
- AliyunCloudMonitorReadOnlyAccess
- AliyunBSSReadOnlyAccess

= (	こう阿里達	5	Q 搜索	Ŕ	费用	企业	更多	>_	<b>Ū</b> •	Ä	简体中文	0
RAM访问	可控制		添加权限									$\times$
概览 人员管理 用户组 用户	1	^	被授权主体 gen_test《 选择权限 <b>1</b>	onaliyun.com $\times$ )								
设置			系统权限策略 diyun	cont 条注	٢	Q	已选择	(5)			7	除
授权			AliyunContainerRegistryF	管理容器镜像服务(ContainerRegistry	)的权限		AI	yunEC	CFullA	ccess	×)	●联系
权限策 RAM角色	能管理	4	AliyunContainerRegistry	只读访问容器镜像服务(ContainerReg	jistry)的	又限	AI	yunOS	SFullA	ccess	×	我们
OAuth应.	用管理						AI	yunBat	tchCor	mputeF	ull ×	
		1	3 确定 取消				AI	iyunCo	ntaine	rRegist	ry ×	

确认授权成功,点击完成。

= (-)阿	里云	Q 搜索	费用	企业	更多	>_	۵.
RAM访问控制		添加权限					
概览 人员管理 用户组	^	授权结果:					
用户		<ul> <li>◇ AliyunECSFullAccess 授权成功</li> <li>◇ AliyunVPCFullAccess 授权成功</li> <li>◇ AliyunOSSFullAccess 授权成功</li> </ul>					
权限管理 授权 权限策略管理	^	<ul> <li>✓ AliyunContainerRegistryReadOnlyAccess 授权成功</li> <li>✓ AliyunBatchComputeFullAccess 授权成功</li> </ul>					
RAM角色管理		完成					

### 5.3 注销云账号登录

= (-)阿	里云	Q 搜索	费用 工单 备案 企业 支持	時与服务 🖸 🖨 🤤 简体中文
RAM访问控制		RAM访问控制 / 用户		۵
概览		用户		基本资料 实名认证 安全设置 ② 安全管控
入页管理 用户组	^	RAM用户是一个身份实体,它通常代表您的组织中需要 通常的操作步骤如下:	访问云资源的人员或应用程序。	<ul> <li>● 访问控制</li> <li>B accesskeys</li> </ul>
用户		<ol> <li>1.创建用户,并为用户设置登录密码(用户登录控制台)</li> <li>2.添加用户到用户组(需要先创建用户组并完成对用户</li> </ol>	场景)或创建AccessKey(应用程序调用 组的授权)	API场] ▼ 会员权益
设置 权限管理	~	新建用户 用户登录名称 > 请输入	Q	<ul> <li>会员积分</li> <li>推荐返利后台</li> </ul>
RAM角色管理		用户登录名称/显示名称	备注	2 退出管理控制台
OAuth应用管理 https://account.cons	ole.aliyun.c	gen_test@		2019年5月 31日 添加到用户组 16:29:45 添加权限 删除

# 控制台快速开始

介绍如何使用控制台来提交一个作业,目的是统计一个日志文件中 INFO、WARN、ERROR、DEBUG 出现的次数。

## 步骤预览

- 1. 作业准备
  - 上传数据文件到 OSS
  - 上传任务程序到 OSS
- 2. 使用控制台提交作业
- 3. 查看作业状态
- 4. 查看结果

#### 1. 作业准备

本作业是统计一个日志文件中 INFO、WARN、ERROR、DEBUG 出现的次数。

该作业包含3个任务: split、 count 和 merge:

- split 任务会把日志文件分成 3 份。
- 置 InstanceCount 为3,表示同时启动 3个 count 任务)。
- merge 任务会把 count 的结果统一合并起来。

DAG图例



#### 上传数据文件到OSS

下载本例子所需的数据: log-count-data.txt

将 log-count-data.txt 上传到:

oss://your-bucket/log-count/log-count-data.txt

- your-bucket 表示您自己创建的 bucket,本例假设 region 为: cn-shenzhen。
- 更多关于如何上传到 OSS, 请参考 OSS 文件上传 以及 常用 OSS 工具。

#### 上传任务程序到OSS

本例的作业程序是使用 python 编写的 , 下载本例所需的程序: log-count.tar.gz

本例不需要改动示例代码。直接将 log-count.tar.gz 上传到 oss , 如上传到 :

oss://your-bucket/log-count/log-count.tar.gz。

如何上传前面已经讲过。

- BatchCompute 只支持以 tar.gz 为后缀的压缩包, 请注意务必用以上方式(gzip)打包, 否则将会无法解析。
- 如果你要修改代码,可以解压后修改,然后要用下面的方法打包:

命令如下:

> cd log-count #进入目录

> tar -czf log-count.tar.gz \* #打包,将所有这个目录下的文件打包到 log-count.tar.gz

可以运行这条命令查看压缩包内容:

\$ tar -tvf log-count.tar.gz

可以看到以下列表:

conf.py count.py merge.py split.py

#### 2. 使用控制台提交作业

登录 BatchCompute 控制台。

单击 **作业列表** > **提交作业** 进行作业提交。请选择合适的 Region (该 region 需要和前面上传数据 的OSS的 bucket 的 region 一致 )。



如上图所示,首先填写作业名称、作业优先级等基本信息,接下来填写作业的详细描述,有两种方法:

- DAG编辑器: DAG编辑器可以用图形化的方式来描述作业 (Job) 包含的任务 (Task) 以及依赖关系。拖动编辑器左上角的"+"来添加 Task,拖动 Task上的箭头来描述依赖关系。单击 Task,可以为每个 Task 设置参数。

**JSON编辑器**:也可以直接使用 JSON 的方式描述作业 (Job) 包含的任务 (Task) 以及依赖 关系,关于作业的 JSON 描述及其参数说明,可参考作业 (Job) 相关的API使用文档。

我们这里直接采用下面已经准备好的JSON描述粘贴到 JSON编辑器中就可以完成作业描述

```
{
"DAG": {
"Dependencies": {
"split": [
"count"
],
"count": [
"merge"
],
"merge": []
},
"Tasks": {
"split": {
"InstanceCount": 1,
"LogMapping": {},
"AutoCluster": {
"Configs": {
"Networks": {
"VPC": {
"CidrBlock": "192.168.0.0/16"
}
}
},
"ResourceType": "OnDemand",
"InstanceType": "ecs.sn1ne.large",
"ImageId": "img-ubuntu-vpc"
},
"Parameters": {
"Command": {
"EnvVars": {},
"CommandLine": "python split.py",
"PackagePath": "oss://your-bucket/log-count/log-count.tar.gz"
},
"InputMappingConfig": {
"Lock": true
},
"StdoutRedirectPath": "oss://your-bucket/log-count/logs/",
"StderrRedirectPath": "oss://your-bucket/log-count/logs/"
},
"InputMapping": {
"oss://your-bucket/log-count/": "/home/input/"
},
"OutputMapping": {
"/home/output/": "oss://your-bucket/log-count/"
},
"MaxRetryCount": 0,
"Timeout": 21600,
"ClusterId": ""
```

```
},
"merge": {
"InstanceCount": 1,
"LogMapping": {},
"AutoCluster": {
"Configs": {
"Networks": {
"VPC": {
"CidrBlock": "192.168.0.0/16"
}
}
},
"ResourceType": "OnDemand",
"InstanceType": "ecs.sn1ne.large",
"ImageId": "img-ubuntu-vpc"
},
"Parameters": {
"Command": {
"EnvVars": {},
"CommandLine": "python merge.py",
"PackagePath": "oss://your-bucket/log-count/log-count.tar.gz"
},
"InputMappingConfig": {
"Lock": true
},
"StdoutRedirectPath": "oss://your-bucket/log-count/logs/",
"StderrRedirectPath": "oss://your-bucket/log-count/logs/"
},
"InputMapping": {
"oss://your-bucket/log-count/": "/home/input/"
},
"OutputMapping": {
"/home/output/": "oss://your-bucket/log-count/"
},
"MaxRetryCount": 0,
"Timeout": 21600,
"ClusterId": ""
},
"count": {
"InstanceCount": 3,
"LogMapping": {},
"AutoCluster": {
"Configs": {
"Networks": {
"VPC": {
"CidrBlock": "192.168.0.0/16"
}
}
},
"ResourceType": "OnDemand",
"InstanceType": "ecs.sn1ne.large",
"ImageId": "img-ubuntu-vpc"
},
"Parameters": {
"Command": {
"EnvVars": {},
```

```
"CommandLine": "python count.py",
"PackagePath": "oss://your-bucket/log-count/log-count.tar.gz"
},
"InputMappingConfig": {
"Lock": true
},
"StdoutRedirectPath": "oss://your-bucket/log-count/logs/",
"StderrRedirectPath": "oss://your-bucket/log-count/logs/"
},
"InputMapping": {
"oss://your-bucket/log-count/": "/home/input/"
},
"OutputMapping": {
"/home/output/": "oss://your-bucket/log-count/"
},
"MaxRetryCount": 0,
"Timeout": 21600,
"ClusterId": ""
}
}
},
"Description": "batchcompute job",
"Priority": 0,
"JobFailOnInstanceFail": true,
"Type": "DAG",
"Name": "log-count"
}
```

上述 JSON 描述的作业要正常运行, 您需要根据自己的实际情况, 对描述中一些配置项做 一些适配修改, 包括:

- 实例类型: "InstanceType": "ecs.sn1ne.large"。用户实际可用的实例类型,可以单击 **DAG编辑器**中的单个 Task,通过下拉选择框来选择。
- 程序包的OSS路径: "PackagePath": "oss://your-bucket/log-count/logcount.tar.gz",填写为实际的程序包路径。
- Stdout日志的OSS路径:oss://your-bucket/log-count/logs/,填写为实际的日志路径(需要在OSS上提前创建好)。
- Stderr日志的OSS路径: oss://your-bucket/log-count/logs/,填写为实际的日志路径(需要在OSS上提前创建好)。
- 输入数据的映射路径:"oss://your-bucket/log-count/": "/home/input/",填写为输入数 据的实际路径。

输出数据的映射路径:"/home/output/": "oss://your-bucket/log-count/",填写为输出数据的实际路径。

确定各个参数及路径填写正确后,点击左下角的提交作业并确认,就完成了作业提交。

#### 3. 查看作业状态

<	log-count	€ 返回作业列表			S	· 刷新 停止 作业详情 提交作业
作业详情	基本信息					
	作业ID : job-000	00000577A510500	00213C00000035	作业名称: log-count	状态:运行中	任务数量:3
	当实例失败时作的	业失败:true		类型:DAG	优先级:0	
	创建时间:2016-	07-05 18:55:16 (5	分钟以前)	开始时间:2016-07-05 1	8:58:07 结束时间:	
	备注:					
	任务运行情况	2				
	1个完成 2个正在	E等待 O个正在运行	· 总进度:	20%		
Ξ			spit	count	marge	
	任务名称	状态	进度	完成/总实例数	开始/结束时间	操作
	count	等待中	0%	0/3	/	查看
	merge	等待中	0%	0/1	/	章看同
	split	成功	100%	171	2016-07-05 18:58:07 / 2016-07-05 18:	58:18 查看

#### 单击作业列表中最新提交的 log-count 作业,可以查看详情:

单击任务名称 split,可以查看任务详情:

<	split t 返回任务列表			₿ 刷新	任务详情	提交	作业
任务详情	基本信息						
	作业ID:job-0000000577A51050000213C00000035	作业名称:log-count	ClusterId : AutoClu	uster (4核8GB,	镜像:m-28ga7w	bnb)	
	任务名称:split	状态:成功	实例数量:1			更	!\$ ♥
	实例运行情况						
	1个完成 0个正在运行 进度: 1009	16					
	٥						
=							

#### 单击绿色方块,可以查看实例的日志:

<	split 全 返回任s	查看日志(Instance: 0)			×	○ 刷新 任务详情	提交	医伸业
任务详情	***	名称	大小	日志时间 / 距今	操作			
	臺本18恩 作业ID:job-0000000	stdout.job-0000000577A51050000213C00 000035.split.0	0.031 kb	<b>2016-07-05 18:58:14</b> 3分钟以前	查看 下载	Clusterid : AutoCluster (4核8GB, 镜像:m-28ga	7wbnb)	
	任务名称:split					实例数量:1	ÿ	[\$ ¥
	实例运行情况							
	1个完成 0个正在运行	进度:100%						
	0							

#### 4. 查看结果

您可以登录 OSS 控制台 查看 your-bucket 这个 bucket 下面的这个文件:/log-count/merge\_result.json。 内容应该如下: {"INFO": 2460, "WARN": 2448, "DEBUG": 2509, "ERROR": 2583}

# 其他范例

# DAG作业快速开始

如果您还没开通批量计算服务,请先开通。

### 命令行工具安装和配置

命令行工具安装和配置

## 作业准备

本作业的目的是求和,将 input.txt 中的数字全部加起来,求和后写入 output.txt。

由于计算比较简单本作业只需1个任务。

本例将 OSS 的目录挂载为 VM 本地目录,使用文件方式操作。

#### 上传数据文件到OSS

先自行创建 input.txt。

```
input.txt的内容(确保一行一个数字):
```

2 40 51

将 input.txt 上传到您的 OSS bucket:

bcs o up input.txt oss://your-bucket/sum/inputs/

# 上传完成后check bcs o ls oss://your-bucket/sum/inputs/

- your-bucket 表示您自己创建的 bucket,本例子假设 region 为: cn-shenzhen。
- bcs o 命令提供几个 OSS 常用的功能,使用 bcs o -h 可以查看帮助,测试少量数据时使用很方便,但上传下载大量数据时不建议使用(没有实现多线程,上传下载慢。更多关于如何上传到 OSS,请参考 常用 OSS 工具。

#### 准备任务程序

sum.sh 内容:

#!/bin/bash
t=0
while read LINE
do
t=\$((\$t+\${LINE}))
done < /home/inputs/input.txt
echo \$t
echo \$t
echo \$t > /home/outputs/output.txt

注意在上一个步骤里我们把输入文件 input.txt 上传到了oss://your-bucket/sum/inputs/, 在以上程序中 input.txt 是从虚拟机的/home/inputs/目录中读取,这是通过批量计算中对 OSS 的挂载功能实现的,具体配置将在下一步骤"提交作业"中解释。

在这个示例程序里,我们通过 bash 脚本完成了求和的功能。您也可以在脚本里执行任何其他应用程序,如何 把应用程序部署到批量计算环境请参考 自定义镜像 和 使用 Docker 。

## 提交作业

在 sum.sh 所在目录运行下面的命令来提交作业:

bcs sub "sh sum.sh" -p sum.sh -r oss://your-bucket/sum/inputs/:/home/inputs/ -w oss://your-bucket/sum/outputs/:/home/outputs/

#### 这里使用 默认镜像和默认实例类型。

-r 表示只读挂载,将 OSS 目录oss://your-bucket/sum/inputs/只读挂载到 VM 本地目录 /home/inputs/,程序可以通过/home/inputs/路径来访问 input.txt 文件。

-w 表示可写挂载,将 OSS 目录oss://your-bucket/sum/outputs/挂载为 VM 本地目录 /home/outputs/,写入本地目录/home/outputs/下的文件 output.txt,会在程序运行完后,被自动 上传到 OSS 的对应目录。

#### 杳看作业运行状态及运行结果

bcs j # 获取作业列表, 每次获取作业列表后都会将列表缓存下来,一般第一个即是你刚才提交的作业 bcs j 1 # 查看缓存中第一个作业的详情 bcs log 1 # 查看缓存中第一个作业日志

可以使用以下命令查看结果:

bcs o cat oss://your-bucket/sum/outputs/output.txt

# 普通作业快速开始

本文档将介绍如何使用命令行工具来提交一个作业,目的是统计一个日志文件中 "INFO","WARN","ERROR","DEBUG"出现的次数。

### 命令行工具安装和配置

命令行工具安装和配置

作业准备

目的:统计一个日志文件中 "INFO"," WARN"," ERROR"," DEBUG" 出现的次数。

该作业包含3个任务: split, count 和 merge:

- split 任务会把日志文件分成 3 份。
- count 任务会统计每份日志文件中"INFO","WARN","ERROR","DEBUG"出现的次数 (count 任务需要配置 InstanceCount 为3,表示同时启动 3个 count 任务)。
- merge 任务会把 count 的结果合并起来。

DAG图例:



上传数据文件到OSS

下载本例子所需的数据: log-count-data.txt

将 log-count-data.txt 上传到: oss://your-bucket/log-count/log-count-data.txt

- your-bucket 表示您自己创建的 bucket,本例子假设 region为: cn-shenzhen。

bcs oss upload ./log-count-data.txt oss://your-bucket/log-count/log-count-data.txt

bcs oss cat oss://your-bucket/log-count/log-count-data.txt # 检查是否上传成功

- bcs o 命令提供几个 OSS 常用的功能,使用 bcs o -h 可以查看帮助,测试少量数据时使用很方便,但上传下载大量数据时不建议使用(没有实现多线程,上传下载慢。更多关于如何上传到 OSS,请参考 常用 OSS 工具。

#### 准备任务程序

本例的作业程序是使用 python 编写的 , 下载本例所需的程序: log-count.tar.gz

使用下面的目录解压:

mkdir log-count && tar -xvf log-count.tar.gz -C log-count

解压后的log-count/目录结构如下

log-count |-- conf.py # 配置 |-- split.py # split 任务程序 |-- count.py # count 任务程序 |-- merge.py # merge 任务程序

说明:不需要改动程序



#### 编写作业配置

在 log-count 的父目录下创建一个文件: job.cfg(此文件要与 log-count 目录平级), 内容如下:

```
[DEFAULT]
job_name=log-count
description=demo
pack=./log-count/
deps=split->count;count->merge
```

[split] cmd=python split.py

[count]

cmd=python count.py nodes=3

[merge] cmd=python merge.py

这里描述了一个多任务的作业,任务的执行顺序是 split>count>merge。

- 关于 cfg 格式的描述 , 请看 多任务支持 。

#### 提交命令

bcs sub --file job.cfg -r oss://your-bucket/log-count/:/home/input/ -w oss://your-bucket/log-count/:/home/output/

--r和-w表示只读挂载和可写映射,具体请查看OSS挂载。

- 同一个 oss 路径 , 可以挂载到不同的本地目录。但是不同的 oss 路径是不能挂载到同一个本地目录的 , 一定要注意。

- 这里需要注意的是,如果挂载的是目录,一定要以"/"结尾。

## 查看作业运行状态

bcs j # 获取作业列表,每次获取作业列表后都会将列表缓存下来,一般第一个即是你刚才提交的作业 bcs ch 1 # 查看缓存中作业的状态,这里的1是bcs j命令查询出的刚刚提交的作业序号 bcs log 1 # 查看缓存中序号为1的作业日志

#### 查看结果

Job 结束后,可以使用以下命令查看 OSS 的结果。

bcs oss cat oss://your-bucket/log-count/merge\_result.json

内容应该如下:

{"INFO": 2460, "WARN": 2448, "DEBUG": 2509, "ERROR": 2583}

# Java SDK 快速开始

# Java快速开始例子

本文档将介绍如何使用 Java 版 SDK 来提交一个作业,目的是统计一个日志文件中 "INFO","WARN","ERROR","DEBUG"出现的次数。

步骤

- 作业准备

- 上传数据文件到 OSS
- 使用示例代码
- 编译打包
- 上传到 OSS
- 使用 SDK 创建 (提交)作业
- 查看结果

# 1. 作业准备

本作业是统计一个日志文件中"INFO","WARN","ERROR","DEBUG"出现的次数。

该作业包含 3 个任务: split, count 和 merge:

- split 任务会把日志文件分成 3 份。
- count 任务会统计每份日志文件中"INFO","WARN","ERROR","DEBUG"出现的次数 (count 任务需要配置 InstanceCount 为 3, 表示同时启动 3 台机器运行个 count 程序)。
- merge 任务会把 count 任务的结果统一合并起来。

DAG图例:



#### (1) 上传数据文件到OSS

#### 下载本例所需的数据: log-count-data.txt

将 log-count-data.txt 上传到:

oss://your-bucket/log-count/log-count-data.txt

- your-bucket 表示您自己创建的 bucket,本例假设 region 为: cn-shenzhen.

- 如何上传到 OSS, 请参考 OSS 上传文档。

### (2) 使用示例代码

本示例将采用 Java 来编写作业任务,使用 maven 来编译,推荐使用 IDEA: http://www.jetbrains.com/idea/download/选择 Community 版本(免费).

示例程序下载: java-log-count.zip

这是一个 maven 工程。

- 注意:无需修改代码。

#### (3) 编译打包

运行命令编译打包:

mvn package

即可在 target 得到下面 3 个 jar 包:

batchcompute-job-log-count-1.0-SNAPSHOT-Split.jar batchcompute-job-log-count-1.0-SNAPSHOT-Count.jar batchcompute-job-log-count-1.0-SNAPSHOT-Merge.jar

再将 3 个 jar 包, 打成一个 tar.gz 压缩包, 命令如下:

> cd target #进入 target 目录

> tar -czf worker.tar.gz \*SNAPSHOT-\*.jar #打包

运行以下命令,查看包的内容是否正确:

> tar -tvf worker.tar.gz batchcompute-job-log-count-1.0-SNAPSHOT-Split.jar batchcompute-job-log-count-1.0-SNAPSHOT-Count.jar batchcompute-job-log-count-1.0-SNAPSHOT-Merge.jar

> - 注意: BatchCompute 只支持以 tar.gz 为后缀的压缩包, 请注意务必用以上方式 (gzip) 打包, 否则 将会无法解析。

### (4) 上传到OSS

本例将 worke.tar.gz 上传到 OSS 的 your-bucket 中:

```
oss://your-bucket/log-count/worker.tar.gz
```

- 如要运行本例子,您需要创建自己的 bucket,并且把 worker.tar.gz 文件上传至您自己创建的 bucket 路径下。

## 2. 使用SDK创建(提交)作业

#### (1) 新建一个maven工程

在 pom.xml 中增加以下 dependencies :

```
<dependencies>
<dependency>
<groupId>com.aliyun</groupId>
<artifactId>aliyun-java-sdk-batchcompute</artifactId>
<version>5.2.0</version>
</dependency>
<dependency>
<groupId>com.aliyun</groupId>
<artifactId>aliyun-java-sdk-core</artifactId>
<version>3.2.3</version>
</dependency>
</dependency>
</dependency>
```

- 请确定使用最新版本的 SDK: Java 版 SDK

#### (2) 新建一个java类: Demo.java

提交作业需要指定集群 ID 或者使用匿名集群参数。本例子使用匿名集群方式进行。匿名集群需要配置 2 个参数,其中:

- 可用的镜像 ID, 可以使用系统提供的 Image, 也可以自行制作镜像, 请看 使用镜像。
- 实例规格 (InstanceType,实例类型 ),请看目前支持类型。

在 OSS 中创建存储 StdoutRedirectPath(程序输出结果)和 StderrRedirectPath(错误日志)的文件路径,本例中创建的路径为

oss://your-bucket/log-count/logs/

- 如需运行本例,请按照上文所述的变量获取以及与上文对应的您的 OSS 路径对程序中注释中的变量进行修改。

Java SDK 提交程序模板如下,程序中具体参数含义请参照 SDK 接口说明。

Demo.java:

/\*

\* IMAGE\_ID: ECS 镜像,由上文所述获取 \* INSTANCE\_TYPE: 实例类型,由上文所述获取 \* REGION\_ID: 提交作业的地域,此项需与上文 OSS 存储 worker 的bucket 地域一致 \* ACCESS\_KEY\_ID: AccessKeyId 可以由上文所述获取 \* ACCESS\_KEY\_SECRET: AccessKeySecret 可以由上文所述获取 \* WORKER\_PATH:由上文所述打包上传的 worker 的 OSS 存储路径 \* LOG\_PATH: 错误反馈和 task 输出的存储路径, logs 文件需事先自行创建 \*/ import com.aliyuncs.batchcompute.main.v20151111.\*; import com.aliyuncs.batchcompute.model.v20151111.\*; import com.aliyuncs.batchcompute.pojo.v20151111.\*; import com.aliyuncs.exceptions.ClientException; import java.util.ArrayList; import java.util.List; public class Demo { static String IMAGE\_ID = "img-ubuntu";; //这里填写您的 ECS 镜像 ID static String INSTANCE\_TYPE = "ecs.sn1.medium"; //根据 region 填写合适的 InstanceType static String REGION\_ID = "cn-shenzhen"; //这里填写 region static String ACCESS\_KEY\_ID = ""; //"your-AccessKeyId"; 这里填写您的 AccessKeyId static String ACCESS\_KEY\_SECRET = ""; //"your-AccessKeySecret"; 这里填写您的 AccessKeySecret static String WORKER\_PATH = ""; //"oss://your-bucket/log-count/worker.tar.gz"; // 这里填写您上传的 worker.tar.gz 的 OSS 存储路径 static String LOG\_PATH = ""; // "oss://your-bucket/log-count/logs/"; // 这里填写您创建的错误反馈和 task 输出的 OSS 存储路径 static String MOUNT\_PATH = ""; // "oss://your-bucket/log-count/"; public static void main(String[] args){ /\*\* 构造 BatchCompute 客户端 \*/ BatchCompute client = new BatchComputeClient(REGION\_ID, ACCESS\_KEY\_ID, ACCESS\_KEY\_SECRET); try{ /\*\* 构造 Job 对象 \*/ JobDescription jobDescription = genJobDescription(); //创建 Job CreateJobResponse response = client.createJob(jobDescription); //创建成功后 , 返回 jobId String jobId = response.getJobId(); System.out.println("Job created success, got jobId: "+jobId); //查询 job 状态 GetJobResponse getJobResponse = client.getJob(jobId); Job job = getJobResponse.getJob(); System.out.println("Job state:"+job.getState());

} catch (ClientException e) { e.printStackTrace();
System.out.println("Job created failed, errorCode:"+ e.getErrCode()+", errorMessage:"+e.getErrMsg()); } }
private static JobDescription genJobDescription(){
JobDescription jobDescription = new JobDescription();
jobDescription.setName("java-log-count"); jobDescription.setPriority(0); jobDescription.setDescription("log-count demo"); jobDescription.setJobFailOnInstanceFail(true); jobDescription.setType("DAG");
DAG taskDag = new DAG();
/** 添加 split task */
TaskDescription splitTask = genTaskDescription(); splitTask.setTaskName("split"); splitTask.setInstanceCount(1); splitTask.getParameters().getCommand().setCommandLine("java -jar batchcompute-job-log-count-1.0-SNAPSHOT- Split.jar"); taskDag.addTask(splitTask);
/** 添加 count task */ TaskDescription countTask = genTaskDescription(); countTask.setTaskName("count"); countTask.setInstanceCount(3); countTask.getParameters().getCommand().setCommandLine("java -jar batchcompute-job-log-count-1.0- SNAPSHOT-Count.jar"); taskDag.addTask(countTask);
/** 添加 merge task */ TaskDescription mergeTask = genTaskDescription(); mergeTask.setTaskName("merge"); mergeTask.setInstanceCount(1); mergeTask.getParameters().getCommand().setCommandLine("java -jar batchcompute-job-log-count-1.0- SNAPSHOT-Merge.jar"); taskDag.addTask(mergeTask);
/** 添加 Task 依赖: split>count>merge */
List <string> taskNameTargets = new ArrayList(); taskNameTargets.add("merge"); taskDag.addDependencies("count", taskNameTargets);</string>
List <string> taskNameTargets2 = new ArrayList();</string>

taskNameTargets2.add("count"); taskDag.addDependencies("split", taskNameTargets2);

```
//dag
jobDescription.setDag(taskDag);
```

```
return jobDescription;
}
```

private static TaskDescription genTaskDescription(){

AutoCluster autoCluster = new AutoCluster(); autoCluster.setInstanceType(INSTANCE\_TYPE); autoCluster.setImageId(IMAGE\_ID); //autoCluster.setResourceType("OnDemand");

TaskDescription task = new TaskDescription(); //task.setTaskName("Find");

//如果使用 VPC , 需要配置 cidrBlock, 请确保 IP 段不冲突 Configs configs = new Configs(); Networks networks = new Networks(); VPC vpc = new VPC(); vpc.setCidrBlock("192.168.0.0/16"); networks.setVpc(vpc); configs.setNetworks(networks); autoCluster.setConfigs(configs);

//打包上传的作业的 OSS 全路径
Parameters p = new Parameters();
Command cmd = new Command();
//cmd.setCommandLine("");
//打包上传的作业的 OSS 全路径
cmd.setPackagePath(WORKER\_PATH);
p.setCommand(cmd);
//错误反馈存储路径
p.setStderrRedirectPath(LOG\_PATH);
//最终结果输出存储路
p.setStdoutRedirectPath(LOG\_PATH);

task.setParameters(p); task.addInputMapping(MOUNT\_PATH, "/home/input"); task.addOutputMapping("/home/output",MOUNT\_PATH);

```
task.setAutoCluster(autoCluster);
//task.setClusterId(clusterId);
task.setTimeout(30000); /* 30000 秒*/
task.setInstanceCount(1); /** 使用 1 个实例来运行 */
```

return task;

```
}
}
```

}

正常输出样例:

Job created success, got jobId: job-01010100010192397211 Job state:Waiting

# 3. 查看作业状态

您可以用 SDK 中的 获取作业信息 方法获取作业状态:

//查询 job 状态 GetJobResponse getJobResponse = client.getJob(jobId); Job job = getJobResponse.getJob(); System.out.println("Job state:"+job.getState());

Job 的 state 可能为: Waiting、Running、Finished、Failed、Stopped.

## 4. 查看结果

您可以登录 batchcompute 控制台 查看 job 状态。

Job 运行结束,您可以登录 OSS 控制台 查看your-bucket 这个 bucket 下面的这个文件:/log-count/merge\_result.json。

内容应该如下:

{"INFO": 2460, "WARN": 2448, "DEBUG": 2509, "ERROR": 2583}

您也可以使用 OSS 的 SDK 来获取结果。

# Python SDK 快速开始

本文档将介绍如何使用 Python 版 SDK 来提交一个作业,目的是统计一个日志文件中 "INFO"," WARN"," ERROR"," DEBUG"出现的次数。

### 步骤预览

- 作业准备
  - 上传数据文件到 OSS
  - 上传任务程序到 OSS
- 使用 SDK 创建(提交)作业
- 查看结果

# 1. 作业准备

本作业是统计一个日志文件中"INFO","WARN","ERROR","DEBUG"出现的次数。

该作业包含3个任务: split, count 和 merge:

- split 任务会把日志文件分成 3 份。
- count 任务会统计每份日志文件中"INFO","WARN","ERROR","DEBUG"出现的次数 (count 任务需要配置 InstanceCount 为 3 , 表示同时启动3台机器运行个 count 程序)。
- merge 任务会把 count 任务的结果统一合并起来。

DAG图例:



### (1) 上传数据文件到OSS

下载本例子所需的数据:log-count-data.txt

将 log-count-data.txt 上传到:

oss://your-bucket/log-count/log-count-data.txt

- your-bucket 表示您自己创建的 bucket,本例子假设 region为: cn-shenzhen。
- 如何上传到 OSS, 请参考 OSS 上传文档。

## (2) 上传任务程序到OSS

本例的作业程序是使用 python 编写的 , 下载本例子所需的程序: log-count.tar.gz

本例不需要改动示例代码。直接将 log-count.tar.gz 上传到 oss , 如上传到 :

oss://your-bucket/log-count/log-count.tar.gz。

如何上传前面已经讲过。

- BatchCompute 只支持以 tar.gz 为后缀的压缩包, 请注意务必用以上方式 (gzip ) 打包, 否则将会无法解析。

如果您要修改代码,可以解压后修改,然后要用下面的方法打包:

命令如下:

> cd log-count #进入目录

> tar -czf log-count.tar.gz \* #打包,将所有这个目录下的文件打包到 log-count.tar.gz

可以运行这条命令查看压缩包内容:

\$ tar -tvf log-count.tar.gz

可以看到以下列表:

conf.py count.py merge.py split.py

## 2. 使用SDK创建(提交)作业

python SDK 的相关下载与安装请参阅 这里。

v20151111 版本,提交作业需要指定集群 ID 或者使用匿名集群参数。本例子使用匿名集群方式进行,匿名集群需要配置 2 个参数,其中:

- 可用的镜像 ID, 可以使用系统提供的 Image , 也可以自行制作镜像, 请参考 使用镜像。
- 实例规格(InstanceType,实例类型),请参考目前支持类型。

在 OSS 中创建存储 StdoutRedirectPath(程序输出结果)和 StderrRedirectPath(错误日志)的文件路径,本例中创建的路径为

oss://your-bucket/log-count/logs/

- 如需运行本例,请按照上文所述的变量获取以及与上文对应的您的 OSS 路径对程序中注释中的变量进行修改。

Python SDK 提交程序模板如下,程序中具体参数含义请参照这里。

#encoding=utf-8 import sys

from batchcompute import Client, ClientError from batchcompute import CN\_SHENZHEN as REGION #这里的region根据实际情况填写 from batchcompute.resources import ( JobDescription, TaskDescription, DAG, AutoCluster ) ACCESS\_KEY\_ID=''# 填写您的 AK ACCESS\_KEY\_SECRET='' # 填写您的 AK IMAGE\_ID = 'img-ubuntu' #这里填写您的镜像 ID INSTANCE\_TYPE = 'ecs.sn1.medium' # 根据实际 region 支持的 InstanceType 填写 WORKER\_PATH = '' # 'oss://your-bucket/log-count/log-count.tar.gz' 这里填写您上传的 log-count.tar.gz 的 OSS 存储 路径 LOG\_PATH = '' # 'oss://your-bucket/log-count/logs/' 这里填写您创建的错误反馈和 task 输出的 OSS 存储路径 OSS\_MOUNT= " # 'oss://your-bucket/log-count/' 同时挂载到/home/inputs 和 /home/outputs client = Client(REGION, ACCESS\_KEY\_ID, ACCESS\_KEY\_SECRET) def main(): try: job\_desc = JobDescription() # Create auto cluster. cluster = AutoCluster() cluster.InstanceType = INSTANCE\_TYPE cluster.ResourceType = "OnDemand" cluster.ImageId = IMAGE\_ID # Create split task. split\_task = TaskDescription() split\_task.Parameters.Command.CommandLine = "python split.py" split task.Parameters.Command.PackagePath = WORKER PATH split task.Parameters.StdoutRedirectPath = LOG PATH split\_task.Parameters.StderrRedirectPath = LOG\_PATH split\_task.InstanceCount = 1 split task.AutoCluster = cluster split\_task.InputMapping[OSS\_MOUNT]='/home/input' split\_task.OutputMapping['/home/output'] = OSS\_MOUNT # Create map task. count task = TaskDescription(split task) count task.Parameters.Command.CommandLine = "python count.py" count task.InstanceCount = 3 count\_task.InputMapping[OSS\_MOUNT] = '/home/input' count\_task.OutputMapping['/home/output'] = OSS\_MOUNT # Create merge task merge\_task = TaskDescription(split\_task) merge\_task.Parameters.Command.CommandLine = "python merge.py" merge\_task.InstanceCount = 1 merge\_task.InputMapping[OSS\_MOUNT] = '/home/input' merge\_task.OutputMapping['/home/output'] = OSS\_MOUNT # Create task dag.  $task_dag = DAG()$ task\_dag.add\_task(task\_name="split", task=split\_task) task\_dag.add\_task(task\_name="count", task=count\_task) task\_dag.add\_task(task\_name="merge", task=merge\_task) task\_dag.Dependencies = {

'split': ['count'], 'count': ['merge'] }

# Create job description.
job\_desc.DAG = task\_dag
job\_desc.Priority = 99 # 0-1000
job\_desc.Name = "log-count"
job\_desc.Description = "PythonSDKDemo"
job\_desc.JobFailOnInstanceFail = True

```
job_id = client.create_job(job_desc).Id
print('job created: %s' % job_id)
```

```
except ClientError, e:
print (e.get_status_code(), e.get_code(), e.get_requestid(), e.get_msg())
```

```
if __name__ == '__main__':
sys.exit(main())
```

# 3. 查看作业状态

您可以用 SDK 中的 获取作业信息 方法获取作业状态:

jobInfo = client.get\_job(job\_id) print (jobInfo.State)

State 状态可能为: Waiting, Running, Finished, Failed, Stopped。

## 4. 查看结果

您可以登录 OSS 控制台 查看 your-bucket 下面的这个文件:/log-count/merge\_result.json。

内容应该如下:

{"INFO": 2460, "WARN": 2448, "DEBUG": 2509, "ERROR": 2583}

- 您也可以使用 OSS 的 SDK 来获取结果。