

腾讯云批量计算

命令行工具

产品文档



腾讯云

【版权声明】

©2013-2017 腾讯云版权所有

本文档著作权归腾讯云单独所有，未经腾讯云事先书面许可，任何主体不得以任何形式复制、修改、抄袭、传播全部或部分本文档内容。

【商标声明】



及其它腾讯云服务相关的商标均为腾讯云计算（北京）有限责任公司及其关联公司所有。本文档涉及的第三方主体的商标，依法由权利人所有。

【服务声明】

本文档意在向客户介绍腾讯云全部或部分产品、服务的当时的整体概况，部分产品、服务的内容可能有所调整。您所购买的腾讯云产品、服务的种类、服务标准等应由您与腾讯云之间的商业合同约定，除非双方另有约定，否则，腾讯云对本文档内容不做任何明示或模式的承诺或保证。

文档目录

文档声明.....	2
命令行工具.....	4
前置准备	4
简单开始	8
执行远程代码包.....	12
远程存储映射.....	15

命令行工具

前置准备

安装命令行工具前请确保您的系统已经安装了 Python 环境，更多内容请参考[如何安装python环境和pip工具](#)。

1. 安装腾讯云命令行工具 CLI

I. 确认已经通过内测申请

否则在使用命令行工具的 Batch 相关命令时会提示没有访问权限。

II. 安装命令行工具 CLI：

- 没安装 CLI：通过 pip 可以快速安装腾讯云命令行工具 CLI，详细安装请参加 [安装命令行工具](#)

```
$ sudo pip install qcloudcli
```

- 已安装 CLI：通过 pip 可以快速升级，命令见下

```
$ sudo pip install --upgrade qcloudcli
```

III. 检验命令行工具 CLI 是否安装成功，并包含 Batch 相关能力：

```
$ qcloudcli batch help
```

You should use the qcloudcli as follow format:

```
qcloudcli <module> <action> [options and parameters]
```

The action name you input is error! The module [batch] support the valid action as follows:

```
DescribeAvailableCvmInstanceTypes |DescribeTask
```

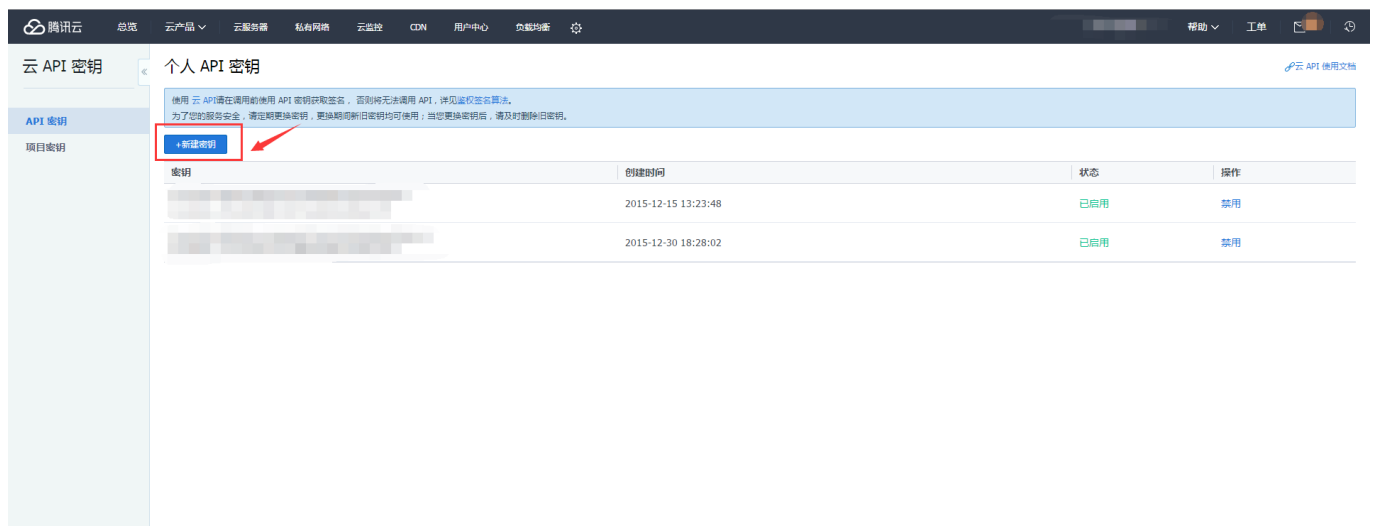
```
DescribeJob |SubmitJob
```

DescribeJobs | TerminateTaskInstance

2. 获取 SecretID 和 SecretKey。

I. 登录腾讯云 [API 密钥控制台](#)。

II. 新建密钥或使用现有云 API 密钥。点击云 API 密钥 ID 进入详情页，获取 SecretID 及 SecretKey。



3. 准备 COS 目录

I. 创建 Bucket 和子文件夹



创建一个 Bucket，命名随意，在 Bucket 里再创建 3 个文件夹以便后续使用，文件夹名字见上图。

II. 记好 COS 访问的地址



先通过上图指示查看 COS Bucket 的访问域名，然后通过域名 + 文件夹名称拼凑出 3 个文件夹的访问域名，刚才官方帐号创建的 3 个文件夹的访问地址分别是

- `cos://batchdemo-1251783334.cosgz.myqcloud.com/logs/`
- `cos://batchdemo-1251783334.cosgz.myqcloud.com/input/`
- `cos://batchdemo-1251783334.cosgz.myqcloud.com/output/`

请务必根据你自己的 Bucket 访问域名拼凑出文件夹访问地址

4. 下载 Batch Demo 文件

[下载地址](#)，解压后目录结构如下，后续请按照下列顺序逐个体验 Batch 的使用和能力

- [1_SimpleStart](#)
- [2_RemoteCodePkg](#)
- [3_StoreMapping](#)

Demo 以 Python + Batch 内测版本命令行工具 的形式提供，Batch 的能力和可配置项较多，通过 Python 脚本可以更便捷的操作。

5. Demo 自定义信息通用部分说明

以『1_SimpleStart』的自定义信息部分为例，

```
# custom (Change to your info)
imageId = "img-m4q71qnf"
Application = {
    "DeliveryForm": "LOCAL",
    "Command": " python -c \"fib=lambda n:1 if n<=2 else fib(n-1)+fib(n-2); print(fib(20))\" "
}
StdoutRedirectPath = "your cos path"
StderrRedirectPath = "your cos path"
```

- imageId：内测期间需要使用包含 Cloud-init 服务的镜像，官方在云市场提供了 CentOS 6.5 版本的可直接使用镜像，镜像 ID img-m4q71qnf，自定义镜像需要基于此镜像来制作，[镜像云市场地址](#)
- StdoutRedirectPath、StderrRedirectPath：请填写第三步里准备的 COS 目录里的 logs 文件夹完整访问地址，比如示例中则是 cos://batchdemo-1251783334.cosgz.myqcloud.com/logs/，请替换成你自己的访问地址
- Application：启动命令行，暂时不用修改

简单开始

1. 前置准备

请根据 [前置准备](#) 里的说明完成准备，并了解如何配置自定义信息里的通用部分。

2. 查看和修改 Demo

使用编辑器打开 1_SimpleStart.py 文件

```
# custom (Change to your info)
imageId = "img-m4q71qnf"
Application = {
    "DeliveryForm": "LOCAL",
    "Command": " python -c \"fib=lambda n:1 if n<=2 else fib(n-1)+fib(n-2); print(fib(20))\" "
}
StdoutRedirectPath = "your cos path"
StderrRedirectPath = "your cos path"
```

自定义部分 custom，除了 Application 以外其余都在前置准备里已经说明，下面重点介绍 Application 的含义

- DeliveryForm：应用程序的交付方式，包括软件打包、容器镜像、CVM 内部直接运行三种，这里 LOCAL 代表的是 CVM 内部直接运行
- Command：任务启动命令，这里执行的是一段 Python 脚本，以 1 为起点，将斐波拉契数列的前 20 个数字求和并输出到 StdOutput 里

1_SimpleStart.py 较为简单，仅需要根据前置准备修改通用部分即可直接执行，所以这里无需再修改

3. 提交作业

Demo 里已经通过 Python 脚本 + Batch

内测版本命令行工具的形式封装了提交作业流程，所以只需要按照下面的示例执行 Python 脚本即可

```
$ python 1_SimpleStart.py
{
  "Response": {
    "RequestId": "d069ce2f-abfc-451f-81fd-9327dbf5cf39",
    "JobId": "job-clump52n"
  }
}
```

如果返回 JobId 字段则代表提交成功，没有则检查返回值排查错误，也可以加入 [用户反馈](#) 里的沟通群向管理员咨询。

4. 查看状态

```
$ qcloudcli batch DescribeJob --Version 2017-03-12 --JobId job-xxx
```

通过 DescribeJob 查看执行状态，--JobId 请替换成实际 JobId，常见状态见下

- STARTING 启动中
- RUNNING 执行中
- SUCCEED 执行成功
- FAILED 执行失败

完整返回信息如下

```
{
  "Response": {
    "JobState": "STARTING",
    "Zone": "ap-guangzhou-2",
    "JobName": "SimpleStart",
    "Priority": 1,
    "RequestId": "8e5ef77c-fa41-4b9f-8c80-107f2546e8ed",
    "TaskSet": [
```

```
{  
  "TaskName": "Task1",  
  "TaskState": "STARTING"  
},  
  "JobId": "job-n4ohivif",  
  "DependenceSet": []  
}
```

5. 查看结果

结果保存在前置准备里配置的 StdoutRedirectPath 和 StderrRedirectPath 目录里，结果见后

文件列表

基础配置

域名管理

[+ 上传文件](#)[创建文件夹](#)[< 返回上一级](#) | logs

文件名	大小	自定义
stderr.job-clump52n.Task1.0.log	161B	--
stderr.job-jrsv8uyp.Task1.0.log	161B	--
stdout.job-clump52n.Task1.0.l...	0B	--
stdout.job-jrsv8uyp.Task1.0.log	0B	--

成功时请查看标准输出 stdout.job-xxx.xxxx.0.log，内容如下

6765

失败时请查看标准错误 stderr.job-xxx.xxxx.0.log，可能的内容如下

```
/bin/sh: -c: line 0: syntax error near unexpected token `('
```

```
/bin/sh: -c: line 0: `python -c \"fib=lambda n:1 if n<=2 else fib(n-1)+fib(n-2); print(fib(20))\" '
```

执行远程代码包

Batch 支持 HTTP 的方式从 .tgz 格式文件里获取代码包，用户可以将代码打包后上传到 COS 里，相比 LOCAL 模式可以更方便的组织代码

1. 前置准备

请根据 [前置准备](#) 里的说明完成准备，并了解如何配置自定义信息里的通用部分。

2. 查看和修改 Demo

使用编辑器打开 2_RemoteCodePkg.py 文件

```
# custom (Change to your info)
imageId = "img-m4q71qnf"
Application = {
    "DeliveryForm": "PACKAGE",
    "Command": "python ./codepkg/fib.py",
    "PackagePath": "http://batchdemo-1251783334.cosgz.myqcloud.com/codepkg/codepkg.tgz"
}
StdoutRedirectPath = "your cos path"
StderrRedirectPath = "your cos path"
```

自定义部分 custom，除了 Application 以外其余都在前置准备里已经说明，下面重点介绍 Application 的含义

- DeliveryForm：应用程序的交付方式，包括软件打包、容器镜像、CVM 内部直接运行三种，这里 PACKAGE 代表的是软件打包的方式
- PackagePath：软件包的地址，HTTP 方式提供，必须是 .tgz 格式，Batch 会将这个软件包下载到被调度的 CVM 某个目录下，然后在该目录执行 Command
- Command：任务启动命令，这里直接调用了软件包里的一个 Python 脚本文件，你可以下载软件包并查看里面的文件结构和内容

fib.py 的内容如下

```
fib = lambda n:1 if n<=2 else fib(n-1)+fib(n-2)
print("Remote Code Package : %d"%(fib(20)))
```

2_RemoteCodePkg.py 较为简单，仅需要根据前置准备修改通用部分即可直接执行，所以这里无需再修改

3. 提交作业

Demo 里已经通过 Python脚本 + Batch

内测版本命令行工具的形式封装了提交作业流程，所以只需要按照下面的示例执行 Python 脚本即可

```
$ python 2_RemoteCodePkg.py
{
  "Response": {
    "RequestId": "d069ce2f-abfc-451f-81fd-9327dbf5cf39",
    "JobId": "job-clump52n"
  }
}
```

如果返回 JobId 字段则代表提交成功，没有则检查返回值排查错误，也可以加入 [用户反馈](#) 里的沟通群向管理员咨询。

4. 查看状态

参考 [简单开始](#) 同名章节

5. 查看结果

参考 [简单开始](#) 同名章节

2_RemoteCodePkg.py 的执行结果如下

Remote Code Package : 6765

远程存储映射

远程映射是 Batch 对存储使用相关的辅助功能，能够将 COS、CFS 等远程存储映射到本地的文件夹上

1. 前置准备

请根据 [前置准备](#) 里的说明完成准备，并了解如何配置自定义信息里的通用部分。

2. 上传输入数据文件

number.txt 的内容如下

```
1
2
3
4
5
6
7
8
9
```

将文件上传到前置准备里创建的 input 文件夹里

[< 返回](#) | batchdemo

文件列表

基础配置

域名管理

+ 上传文件

创建文件夹

[< 返回上一级](#) | input

文件名	大小	自定义访问权限 ?
number.txt	17B	--

3. 查看和修改 Demo

使用编辑器打开 3_StoreMapping.py 文件

```
# custom (Change to your info)
imageId = "img-m4q71qnf"
Application = {
  "DeliveryForm": "PACKAGE",
  "Command": "python ./codepkg/countnum.py",
  "PackagePath": "http://batchdemo-1251783334.cosgz.myqcloud.com/codepkg/codepkg.tgz"
}
StdoutRedirectPath = "your cos path"
StderrRedirectPath = "your cos path"
InputMapping = {
  "SourcePath": "your input remote path",
  "DestinationPath": "/data/input/"
}
OutputMapping = {
  "SourcePath": "/data/output/",
  "DestinationPath": "your output remote path"
}
```

与 2_RemoteCodePkg.py 相比，自定义部分中修改如下

- Application 的 Command 改为执行 countnum.py
- InputMapping：输入映射，SourcePath 远程存储地址（修改为前置准备里 input 文件夹的地址），DestinationPath 本地目录（暂不修改）
- OutputMapping 输出映射，SourcePath 本地目录（暂不修改），DestinationPath 远程存储地址（修改为前置准备里 output 文件夹的地址）

countnum.py 的内容如下

```
import os

inputfile = "/data/input/number.txt"
outputfile = "/data/output/result.txt"

def readFile(filename):
    total = 0
    fopen = open(filename, 'r')
    for eachLine in fopen:
        total += int(eachLine)
    fopen.close()
    print "total = ",total
    fwrite = open(outputfile, 'w')
    fwrite.write(str(total))
    fwrite.close()

print("Local input file is ",inputfile)
readFile(inputfile)
```

打开文件 input/number.txt，并把每一行的数字相加，然后把结果写到 output/Result.txt 里

4. 提交作业

Demo 里已经通过 Python脚本 + Batch

内测版本命令行工具的形式封装了提交作业流程，所以只需要按照下面的示例执行 Python 脚本即可

```
$ python 3_StoreMapping.py
{
  "Response": {
    "RequestId": "d069ce2f-abfc-451f-81fd-9327dbf5cf39",
    "JobId": "job-clump52n"
  }
}
```

如果返回 JobId 字段则代表提交成功，没有则检查返回值排查错误，也可以加入 [用户反馈](#) 里的沟通群向管理员咨询。

5. 查看状态

参考 [简单开始](#) 同名章节。

6. 查看结果

Batch 会将输出数据从本地目录拷贝到远程存储目录，3_StoreMapping.py 的执行结果保存在了 result.txt 里，这个文件将自动同步到 COS 上

[返回](#) | batchdemo

文件列表

基础配置

域名管理

+ 上传文件

创建文件夹

[返回上一级](#) | output

文件名	大小	自定义访问权限 ?
result.txt	2B	--

45