批量计算

命令行工具

产品文档

腾讯云

文档目录

命令行工具
前置准备

最近更新时间：2024-01-13 11:19:29

安装腾讯云命令行工具 TCCLI 前请确保您的系统已经安装了 Python 环境，详情请参见 前提条件。

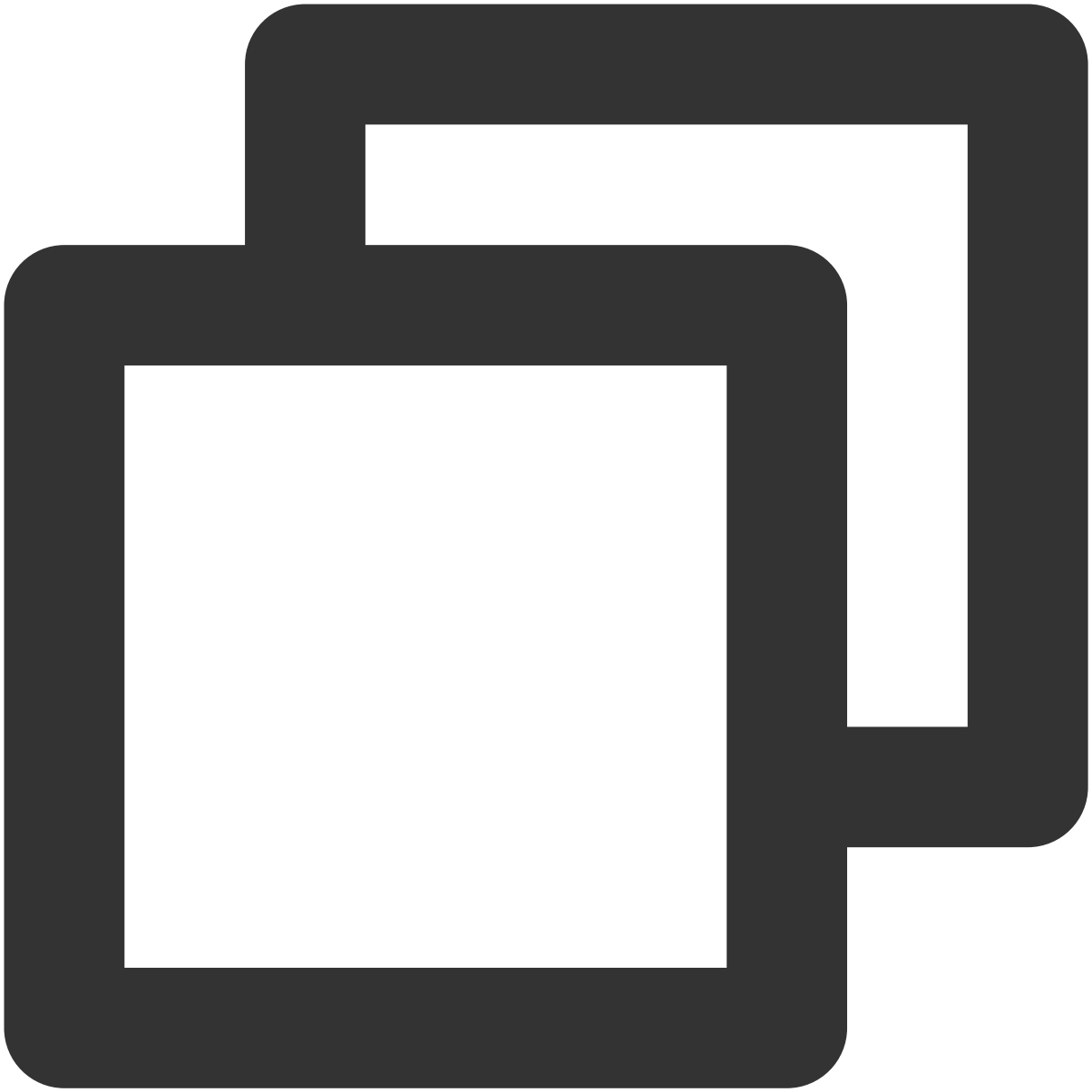
步骤1：安装 TCCLI

安装 TCCLI

请结合您的实际情况，执行对应命令。

未安装 TCCLI

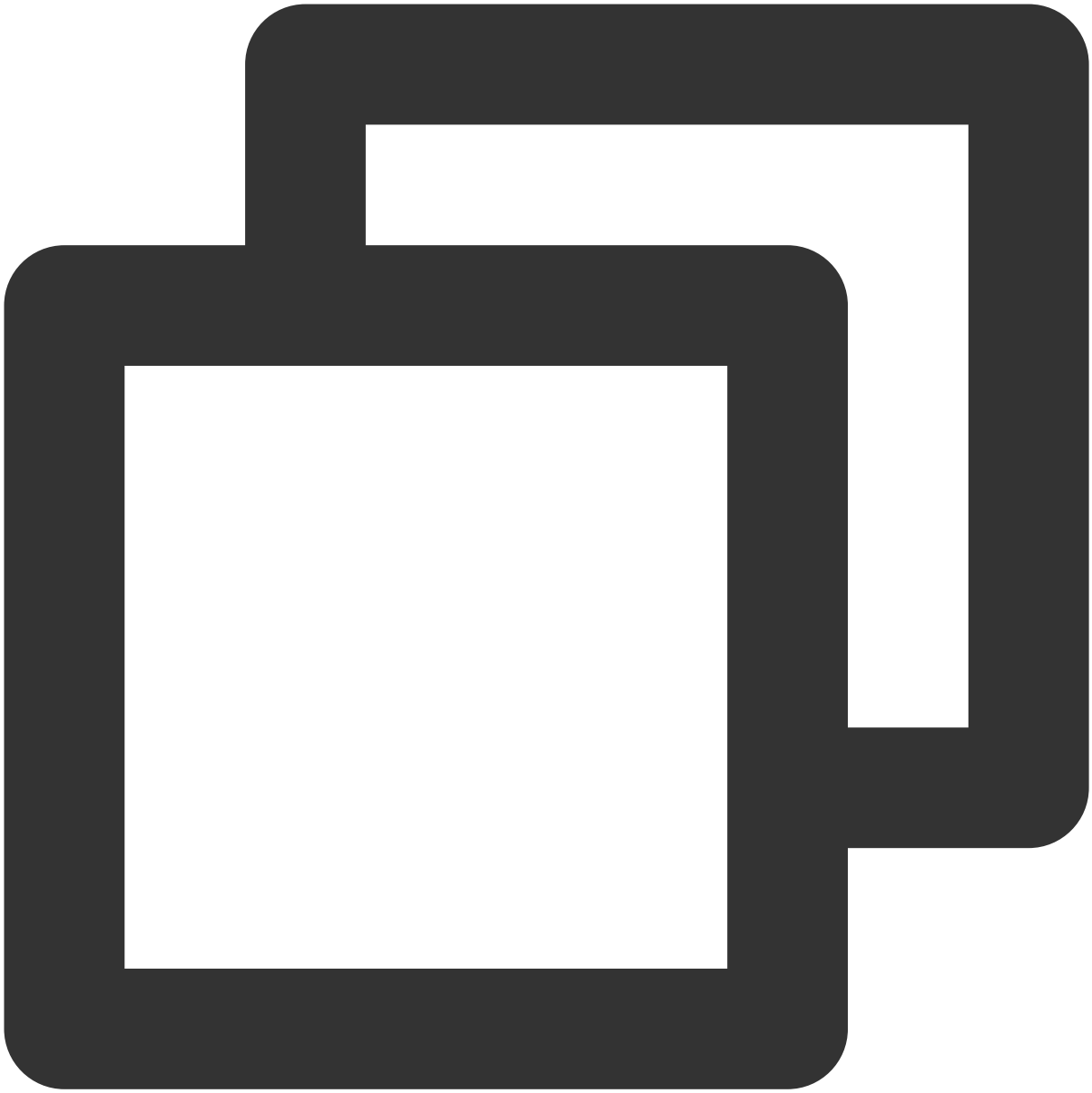
执行以下命令，通过 pip 可以快速安装 TCCLI，详情请参见 安装命令行工具。



```
$ sudo pip install tccli
```

已安装 TCCLI

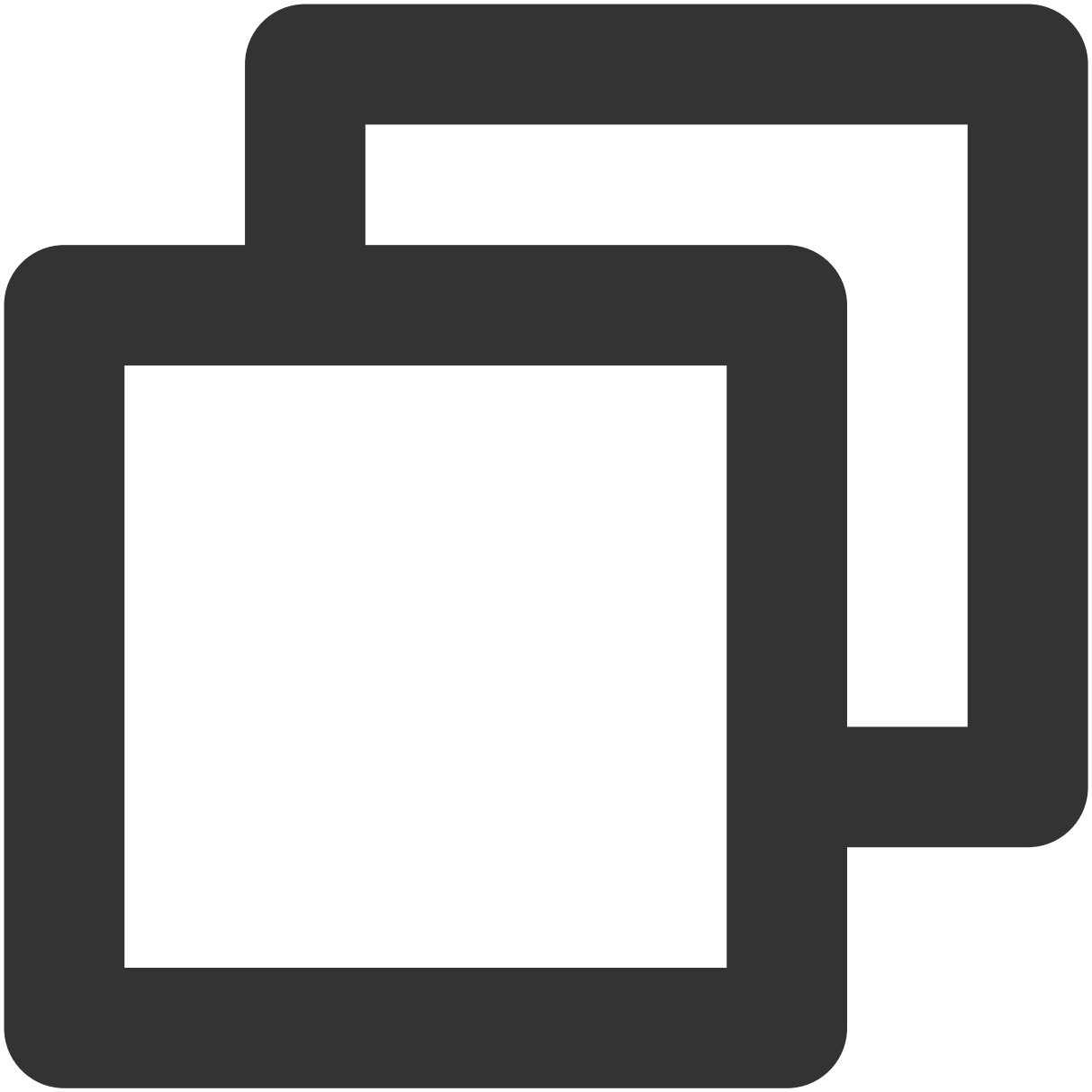
执行以下命令，通过 pip 可以快速升级。



```
$ sudo pip install --upgrade tccli
```

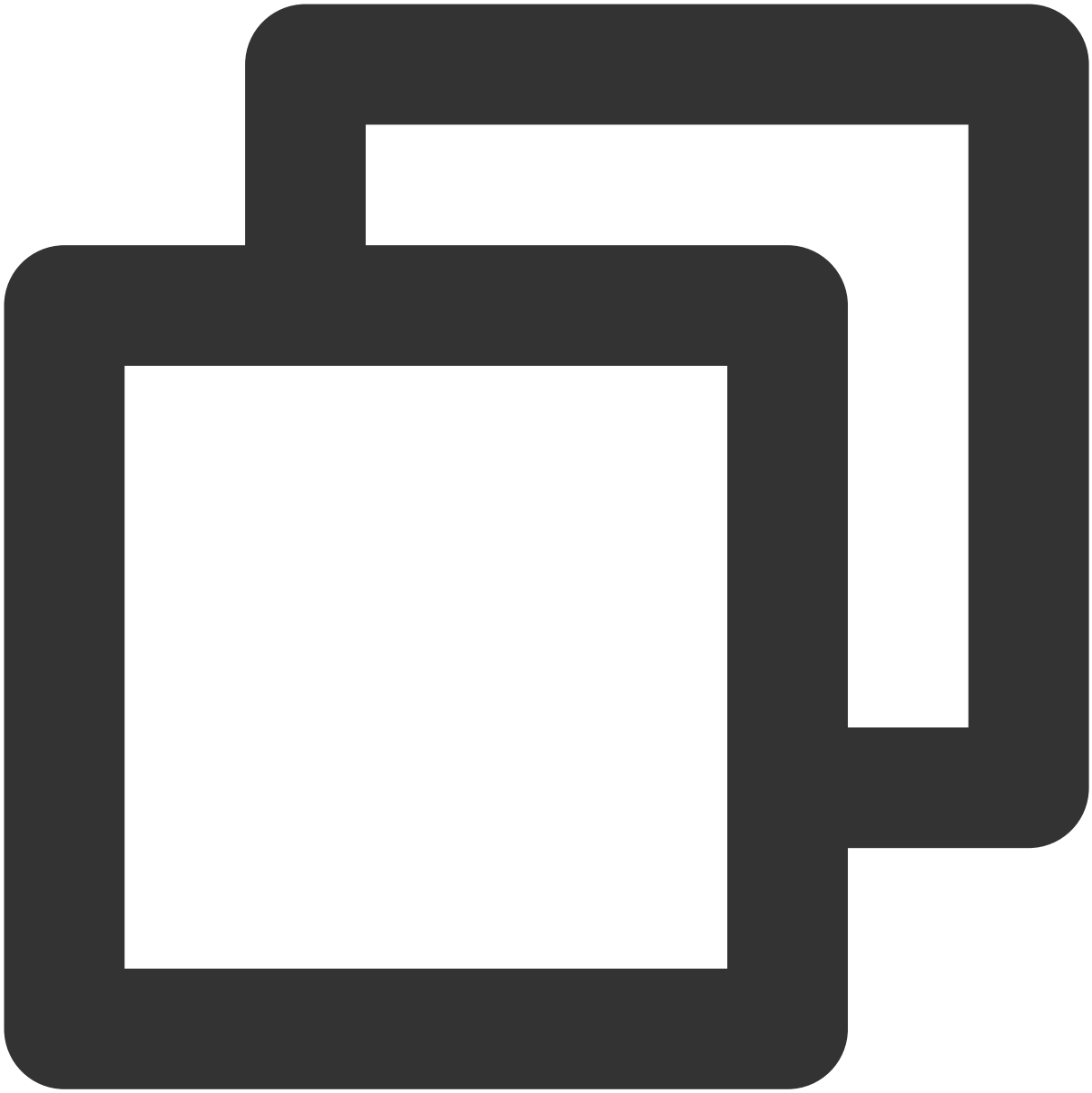
验证安装

执行以下命令，检验命令行工具 TCCLI 是否安装成功，以及是否包含 Batch 相关能力。



```
tccli batch help
```

返回结果如下，则成功安装。

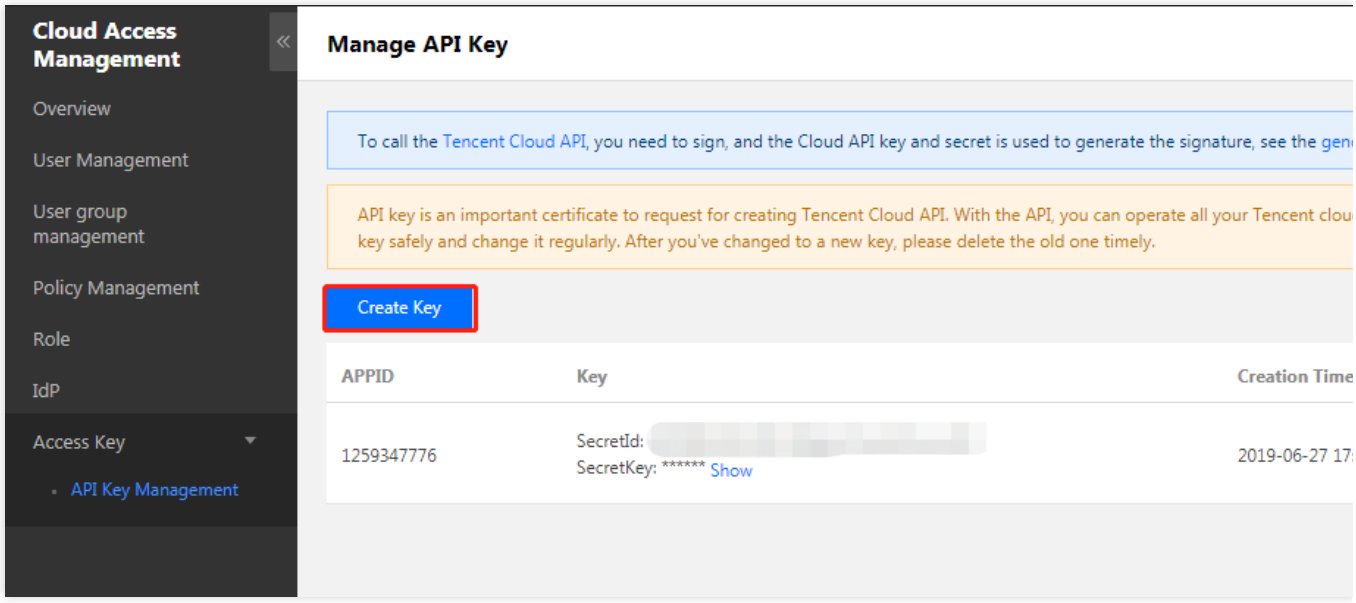


```
NAME
        batch
DESCRIPTION
        batch-2017-03-12
USEAGE
        tccli batch <action> [--param...]
OPTIONS
        help
        show the tccli batch help info
        --version
        specify a batch api version
```

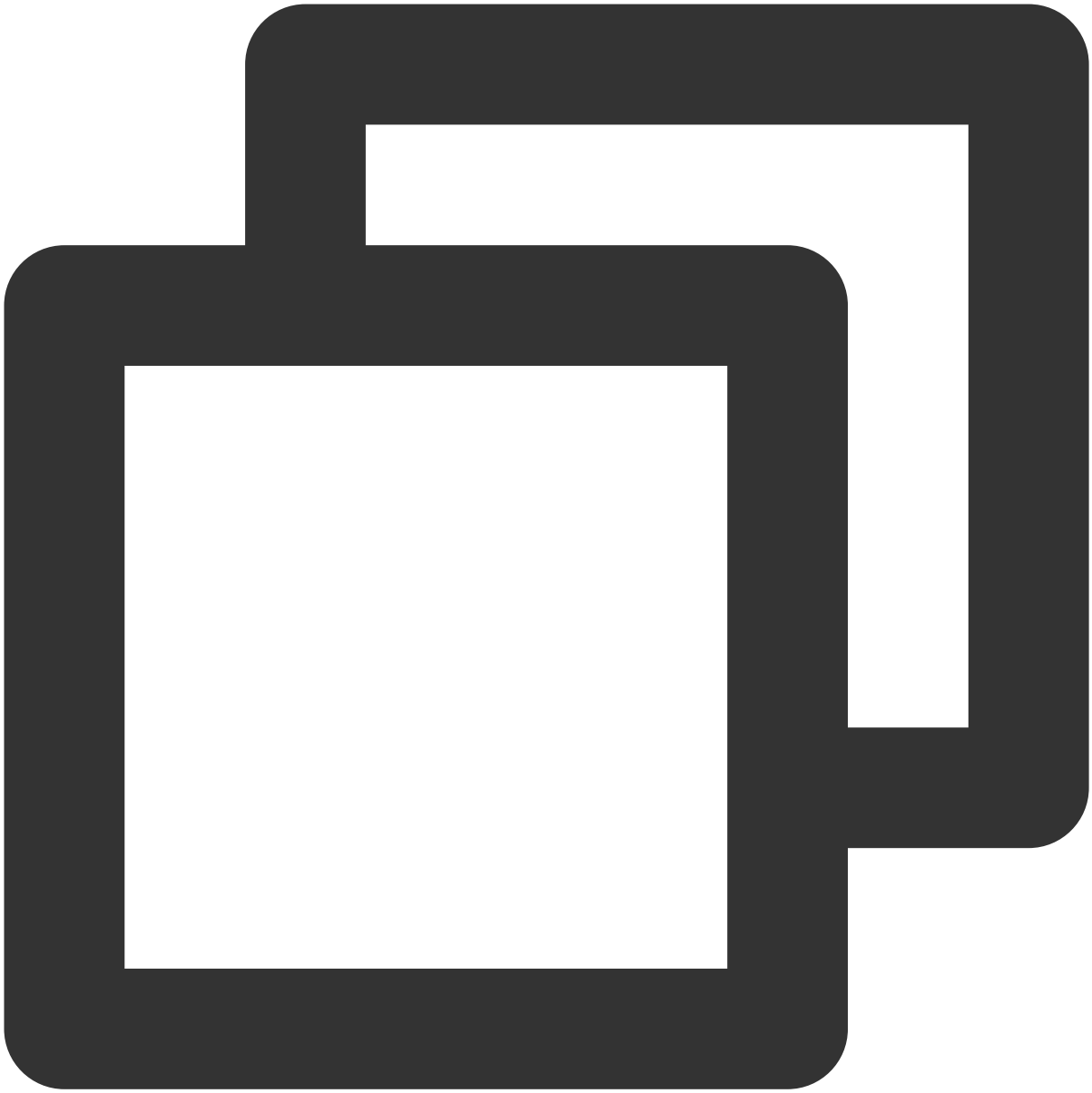
```
AVAILABLE ACTION
        DescribeComputeEnv
        用于查询计算环境的详细信息
        CreateTaskTemplate
        用于创建任务模板
```

步骤2：配置 TCCLI

1. 登录腾讯云 API 密钥控制台。
2. 单击**新建密钥**或使用现有密钥，记录 SecretID 及 SecretKey。如下图所示：



3. 执行 `tccli configure` 命令，并输入 TCCLI 配置信息，详情请参见 配置命令行工具。

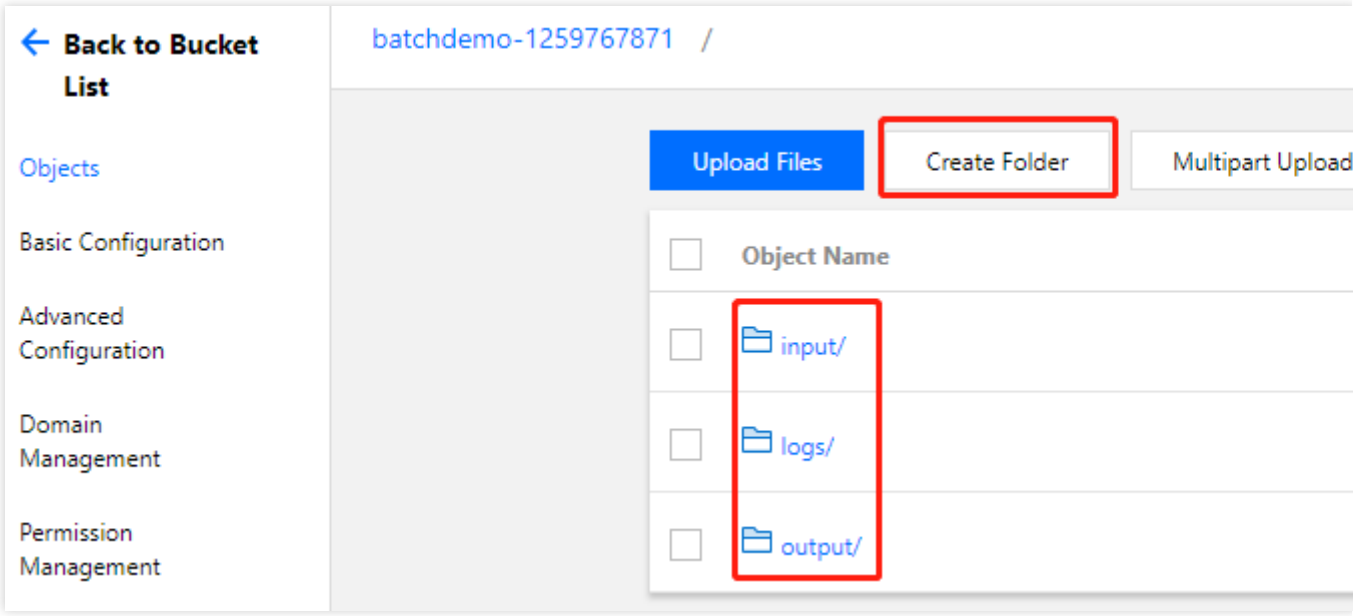


```
$ tccli configure
TencentCloud API secretId[None]:
TencentCloud API secretKey[None]:
region[None]:
output[json]:
```

步骤3：准备 COS 目录

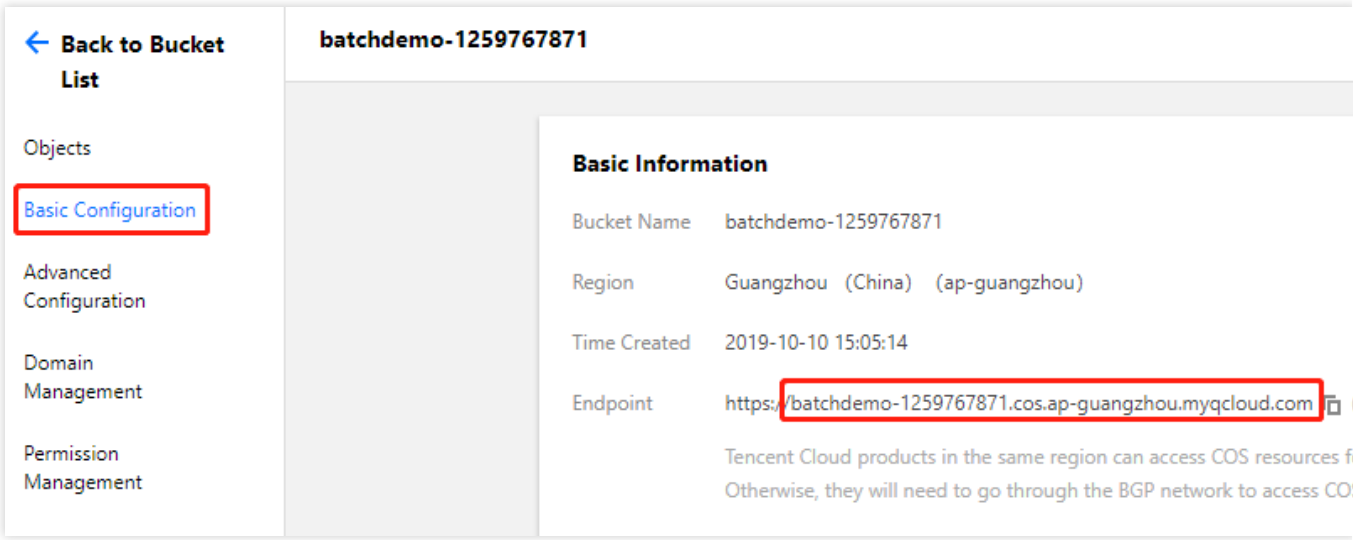
创建 Bucket 及子文件夹

1. 登录对象存储控制台，选择左侧导航栏中的 **存储桶列表**。
2. 创建一个 Bucket，并且在 Bucket 中创建3个文件夹以便后续使用。如下图所示：



获取 COS 相关访问域名

1. 单击 Bucket 左侧的**基础配置**，可在 Bucket 基础信息中查看访问域名。如下图所示：



2. 获取 COS Bucket 子文件夹访问域名。

说明：

请结合您的实际情况，获取 COS 相关域名。

已获得 COS Bucket 的访问域名为：`https://batchdemo-xxxxxxxxx.cos.ap-guangzhou.myqcloud.com`，可通过 `域名+文件夹` 推算出在 创建 Bucket 及子文件夹 中创建的3个文件夹的访

问域名。如下所示：

```
cos://batchdemo-xxxxxxxxx.cos.ap-guangzhou.myqcloud.com/logs/
cos://batchdemo-xxxxxxxxx.cos.ap-guangzhou.myqcloud.com/input/
cos://batchdemo-xxxxxxxxx.cos.ap-guangzhou.myqcloud.com/output/
```

步骤4：下载 Demo 文件

请前往 Batch Demo 下载测试文件并解压。

说明：

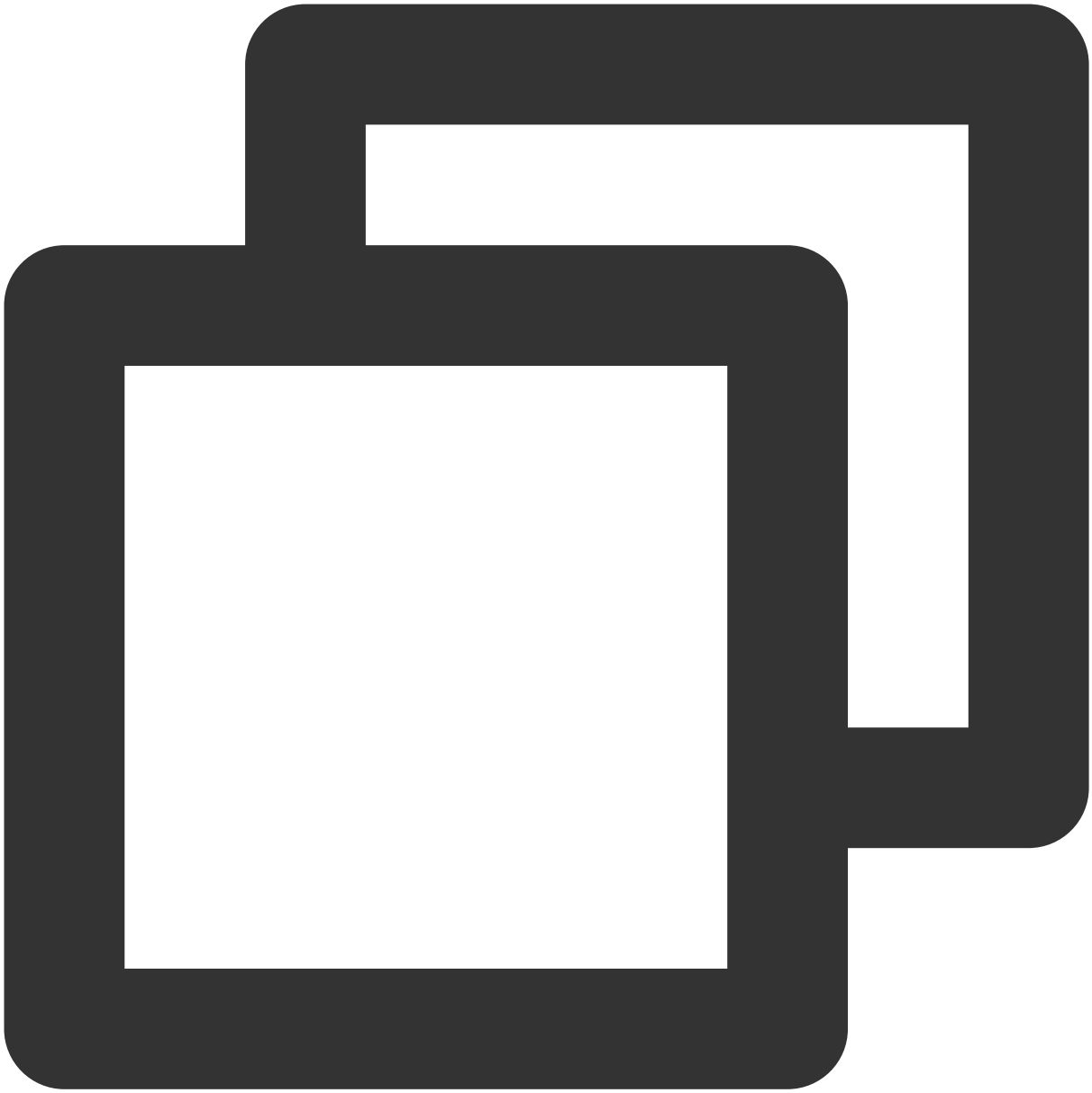
Demo 以 Python + Batch 命令行工具的形式提供，Batch 的能力和可配置项较丰富，通过 Python 脚本可以更便捷的操作。

步骤5：修改 Demo 自定义信息

注意：

Batch Demo 中需替换自定义信息中的通用部分，请参考以下步骤修改 Demo 中的所有文件。

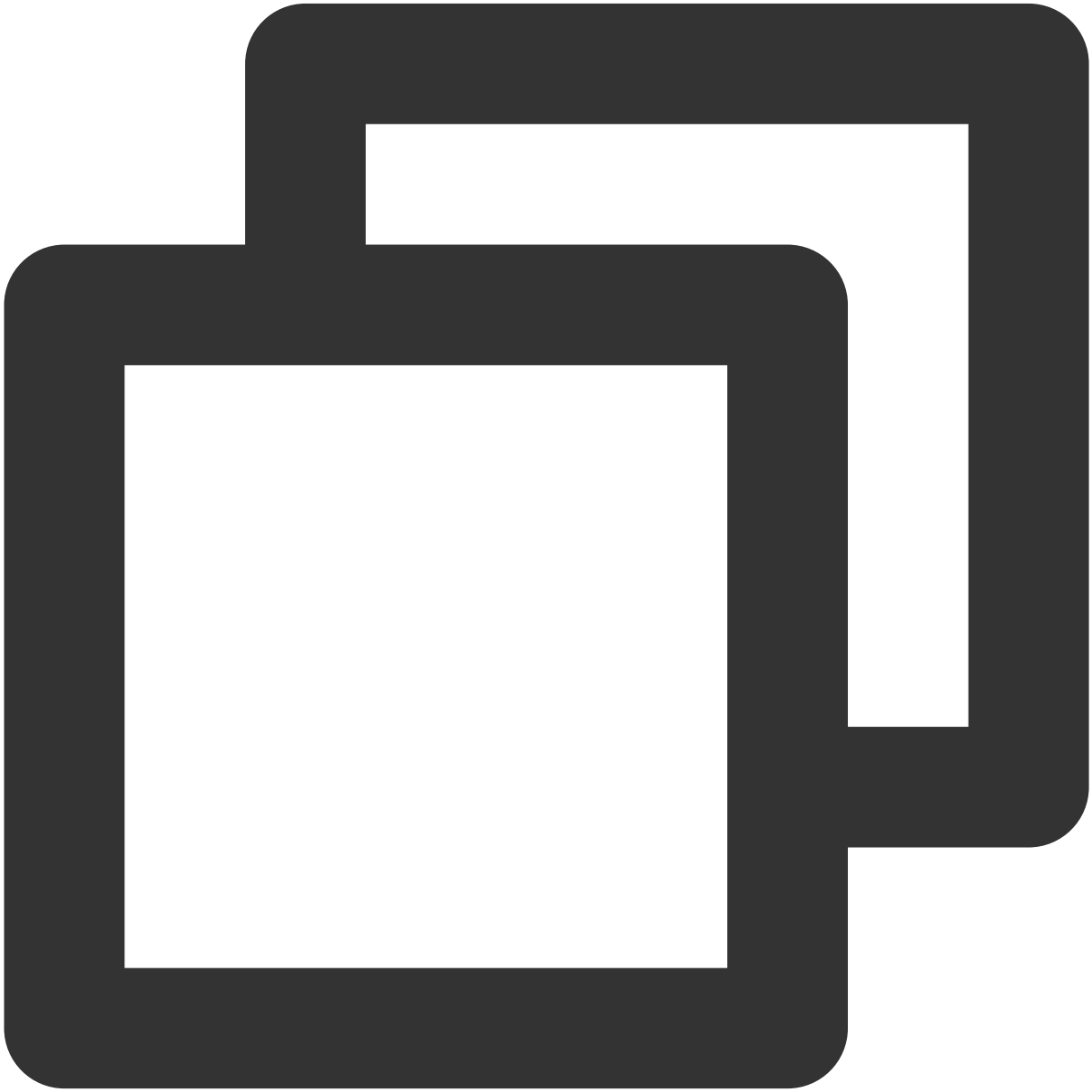
以 `1_SimpleStart.py` 中的自定义信息部分为例：



```
# 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(
}
StdoutRedirectPath = "cos://batchdemo-xxxxxxxxxx.cos.ap-guangzhou.myqcloud.com/logs
StderrRedirectPath = "cos://batchdemo-xxxxxxxxxx.cos.ap-guangzhou.myqcloud.com/logs
```

需要修改的信息如下表所示：

配置项	描述
imageId	自定义镜像需要基于此镜像来制作，可参考 Windows 自定义镜像。
StdoutRedirectPath	请填写 获取 COS 相关访问域名 中获取的 logs 文件夹完整访问域名。
StderrRedirectPath	
Application	启动命令行，保持默认设置。



```
cmd = "tccli batch SubmitJob \\
    --version 2017-03-12 \\
    --Placement '{\\"Zone\\": \\"ap-guangzhou-2\\"}' \\
    --Job ' %s ' "%(json.dumps(testJob))
```

Demo 中指定在广州二区申请资源，您可以根据 TCCLI 中配置的默认地域，选择相应的可用区并申请资源。
地域和可用区的详细信息请查看 地域和可用区。

步骤6：测试

请对应文件参考教程，按照下列顺序体验 Batch 的使用方法及计算能力。

1. 1_SimpleStart.py：简单开始
2. 2_RemoteCodePkg.py：执行远程代码包
3. 3_StoreMapping.py：远程存储映射

简单开始

最近更新时间：2024-01-13 11:19:29

操作场景

您可通过本文快速了解 Batch 的使用方法及计算能力。

前提条件

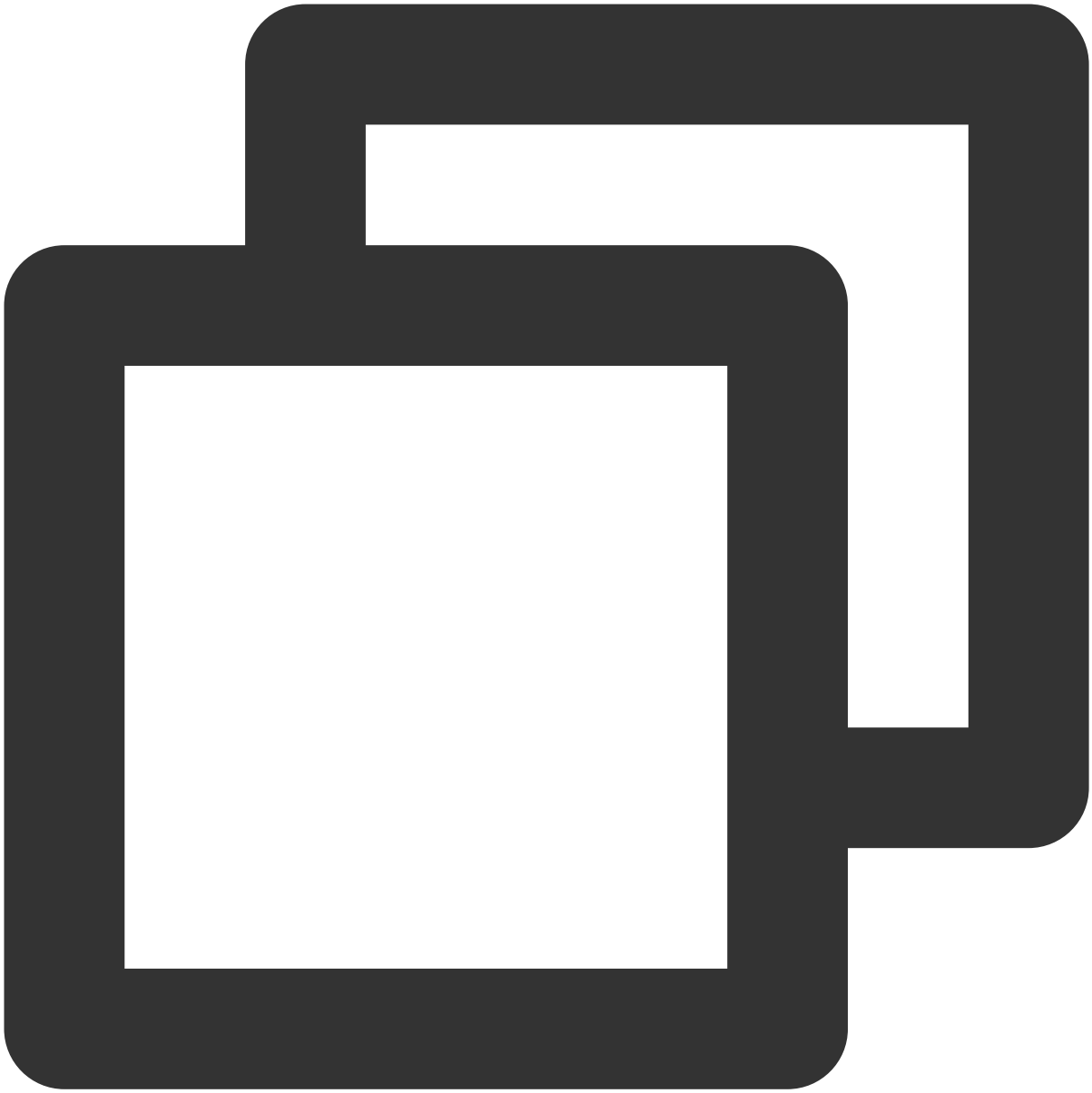
请根据 前置准备 里的说明完成准备，并了解如何配置自定义信息里的通用部分。

查看 Demo

说明：

请在 前置准备 中修改 `1_SimpleStart.py` 文件自定义信息的通用部分。

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(
}
StdoutRedirectPath = "your cos path"
StderrRedirectPath = "your cos path"
```

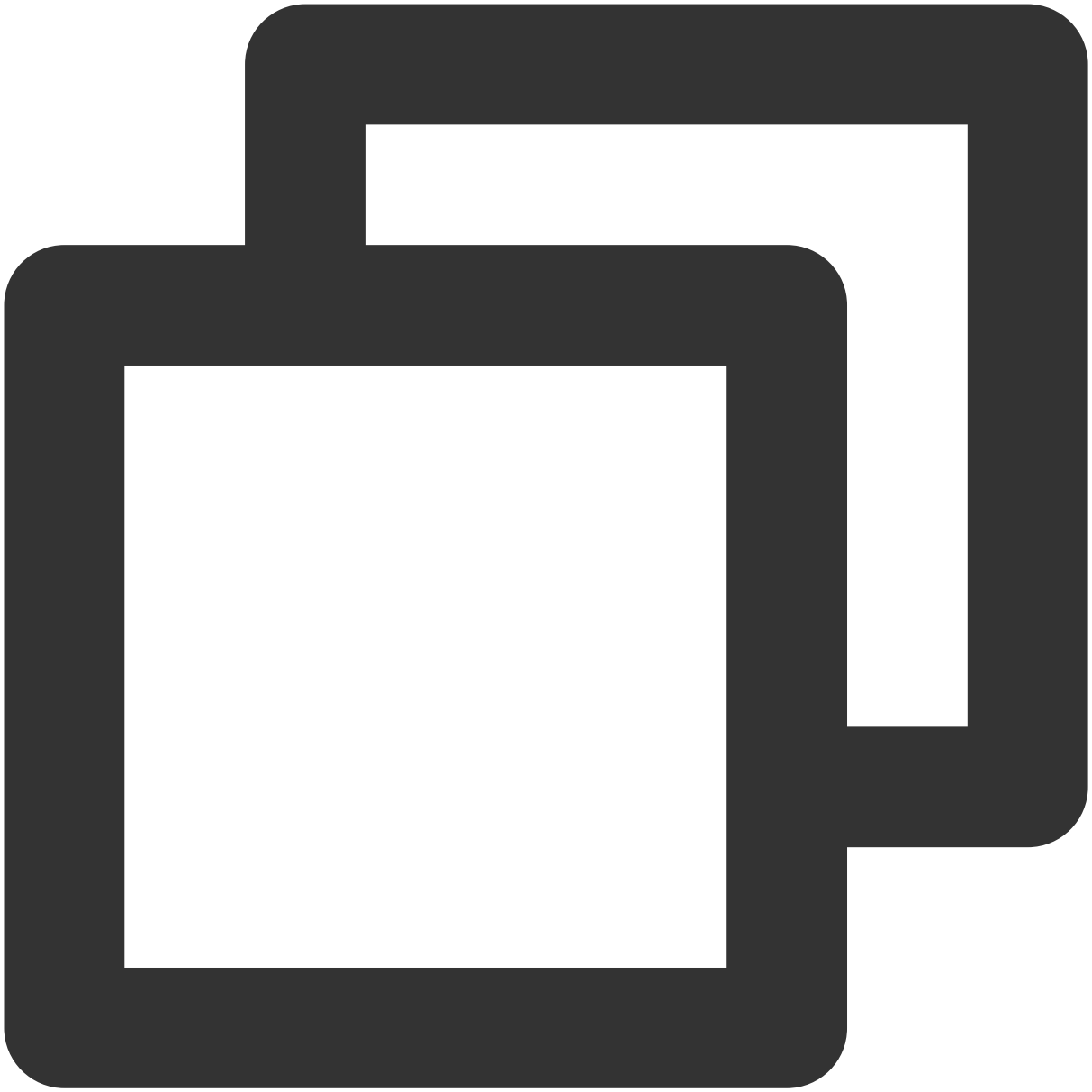
自定义部分除 Application 以外，都已在前置准备中说明， Application 中配置请参考下表：

配置项	描述
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。

提交作业

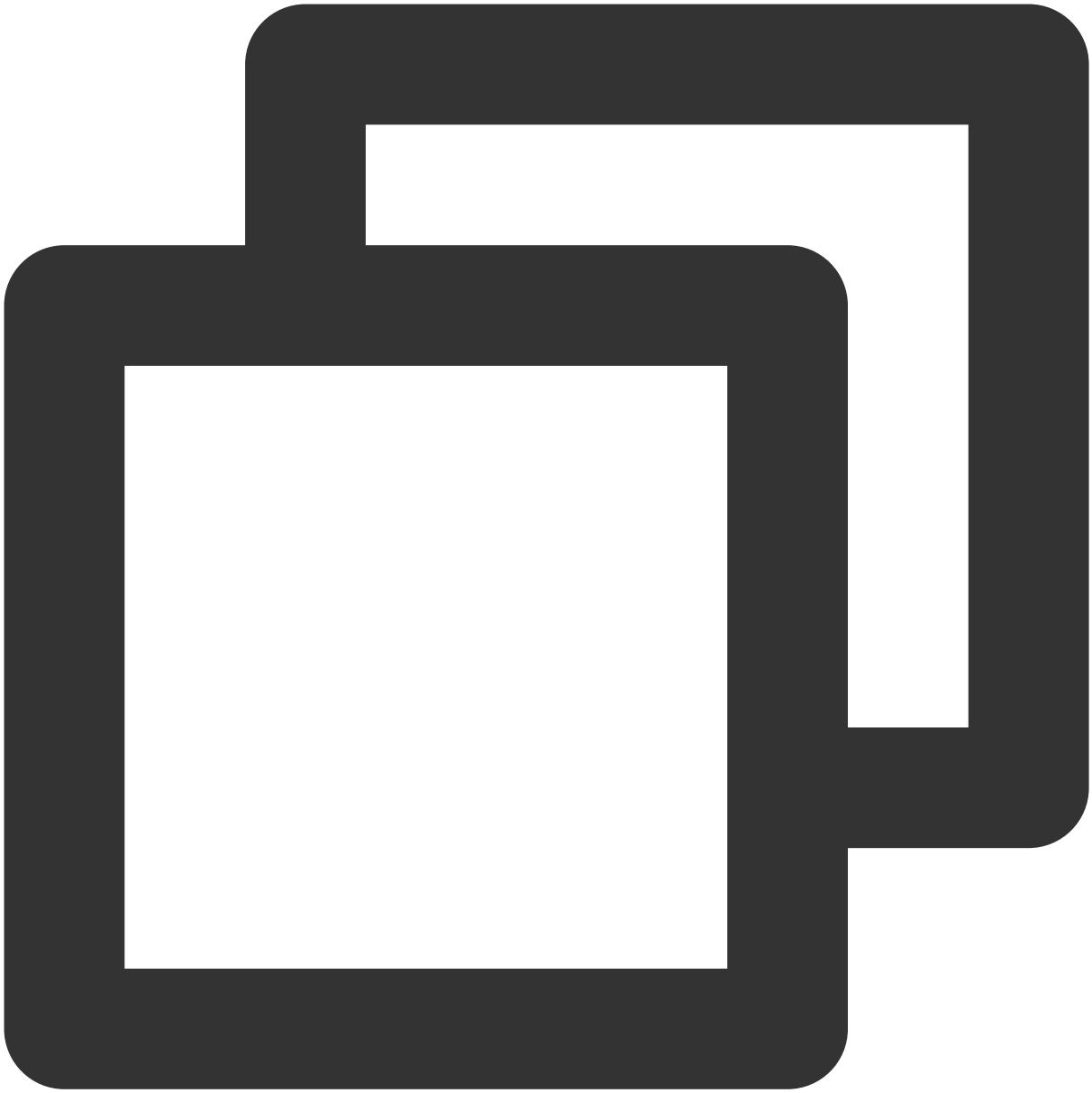
执行以下命令，执行 Python 脚本。

Demo 中已经通过 Python 脚本 + Batch 命令行工具的形式封装了提交作业流程。



```
python 1_SimpleStart.py
```

返回结果如下所示，则表示提交成功。

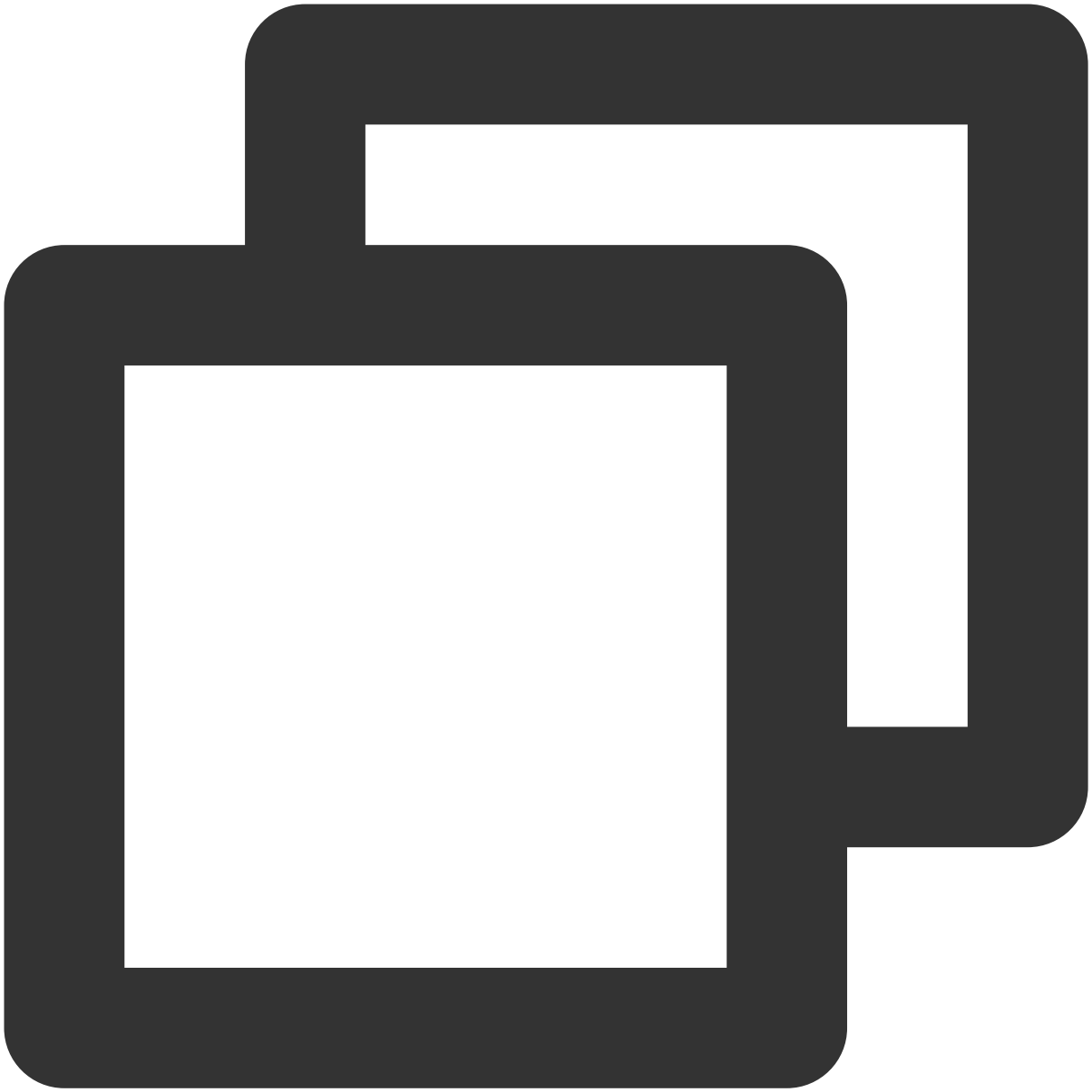


```
{
    "RequestId": "393292f4-5583-48ad-a9f5-f673138ea637",
    "JobId": "job-o0xxxxxq7"
}
```

若未提交成功，请检查返回值排查错误，也可以通过 联系我们 提交工单咨询。

查看状态

执行以下命令，通过 DescribeJob 查看执行状态：

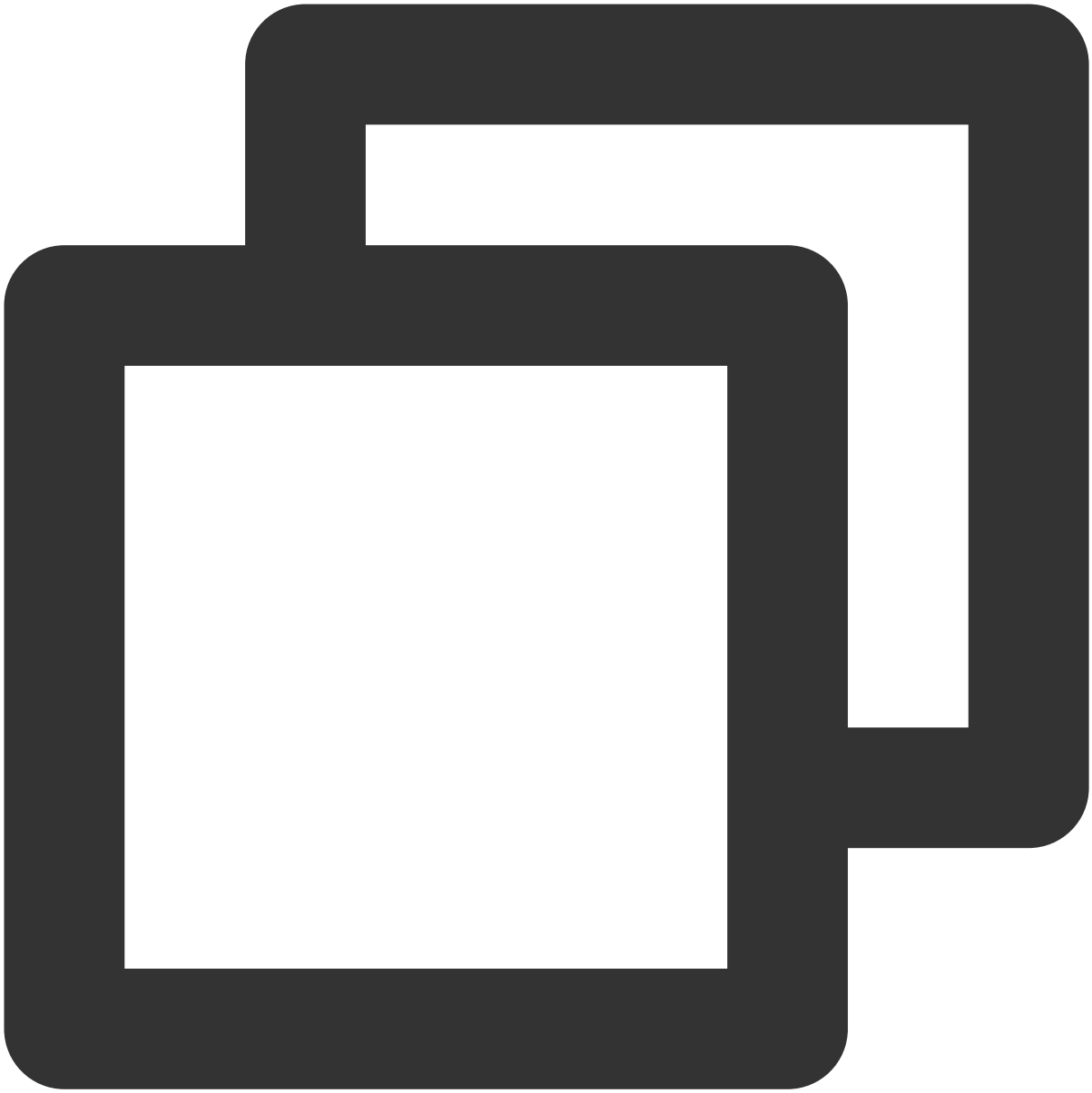


```
$ tccli batch DescribeJob --version 2017-03-12 --JobId job-xxx
```

说明：

--JobId 请替换成 提交作业 中的返回的 JobId。

返回信息如下（部分已省略）：



```
{
    "EndTime": "2019-10-08T04:06:58Z",
    "JobState": "SUCCEED",
    "TaskInstanceMetrics": {
        ...
    },
    "Zone": "ap-guangzhou-2",
    "TaskMetrics": {
        ...
    },
    "JobName": "TestJob",
```

```
    "Priority": 1,
    "RequestId": "7a5f4c94-1357-486c-9c48-8286ba01b5b2",
    "TaskSet": [
        ...
    ],
    "StateReason": null,
    "JobId": "job-o0xxxxxq7",
    "DependenceSet": [],
    "CreateTime": "2019-10-08T04:05:54Z"
}
```

返回信息包含下列常见执行状态：

STARTING：启动中

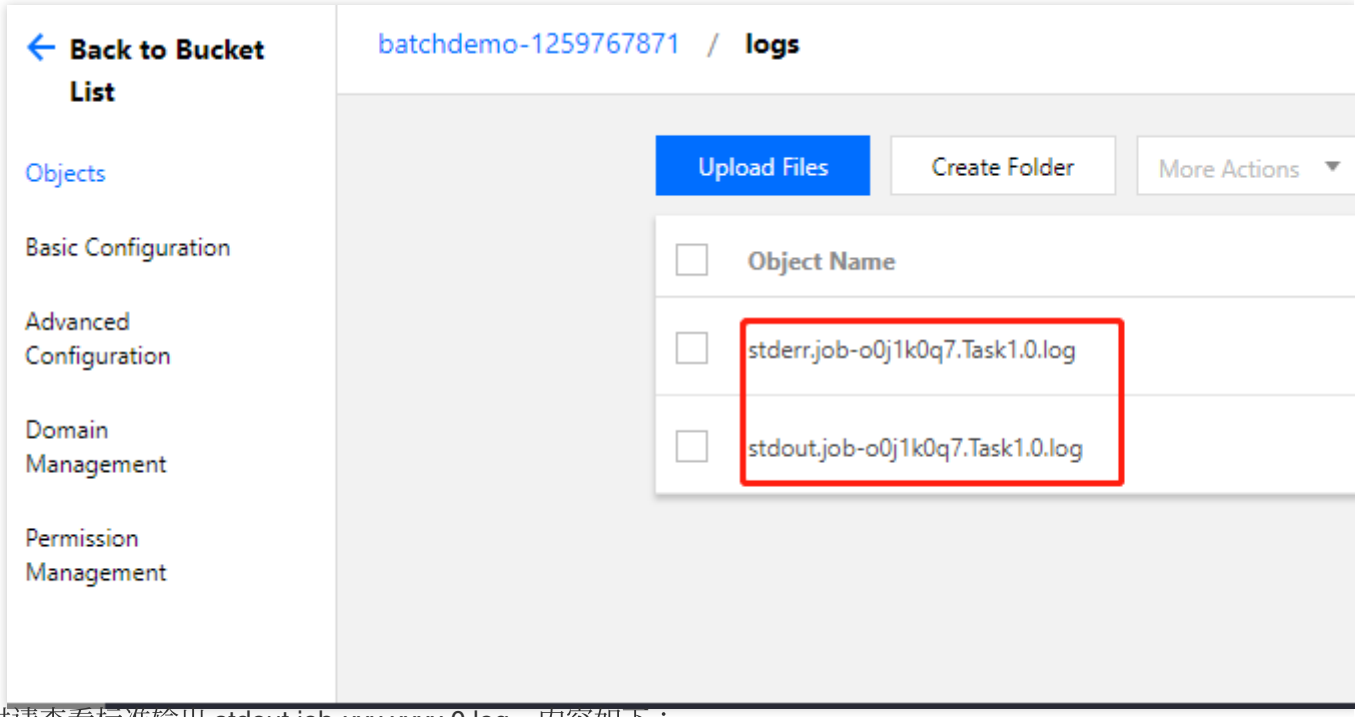
RUNNING：执行中

SUCCEED：执行成功

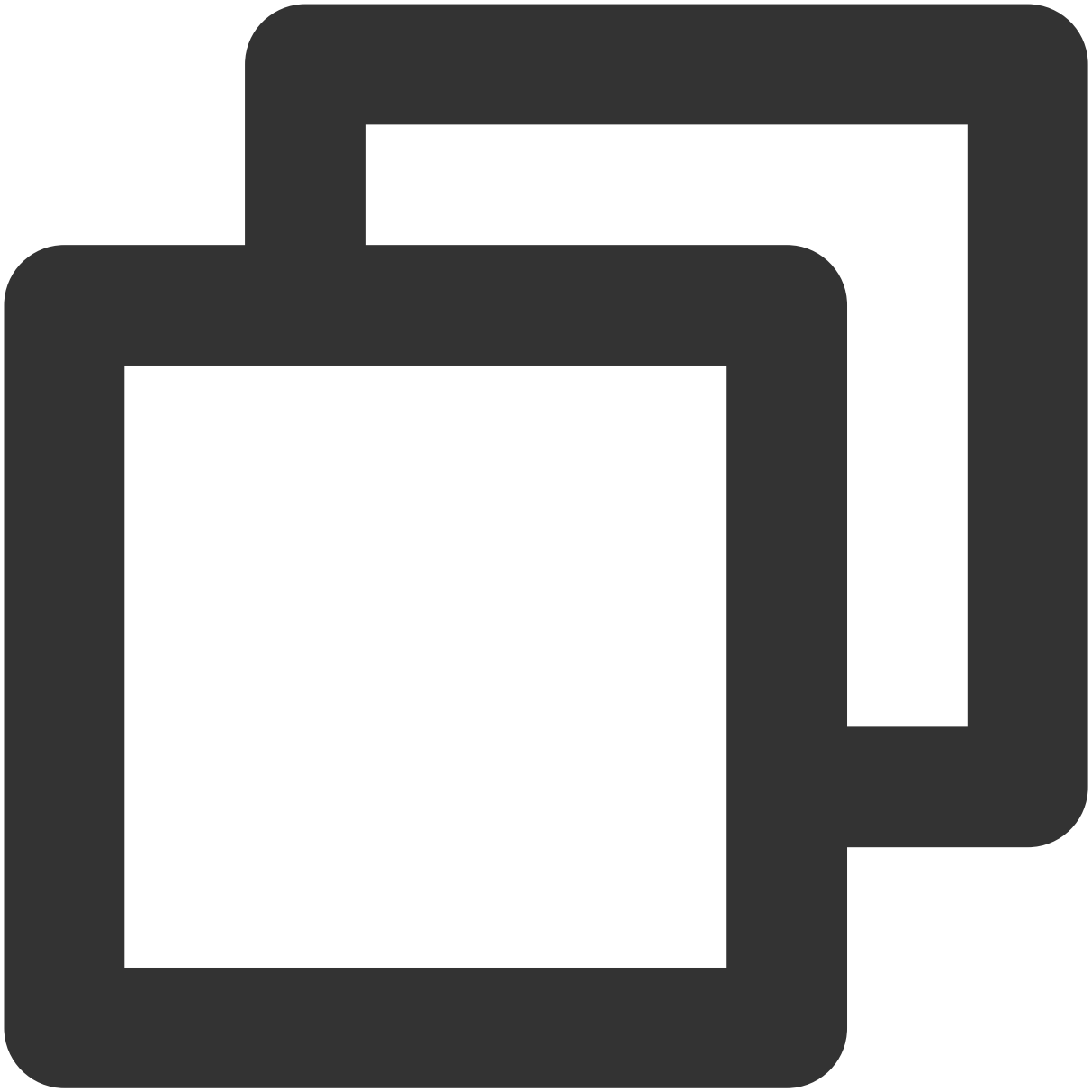
FAILED：执行失败

查看结果

1. 登录对象存储控制台，单击左侧导航栏中的 **存储桶列表**。
2. 选择已创建的 **Bucket ID** >**文件列表** > **logs 文件**，执行结果均保存在 logs 文件中。如下图所示：

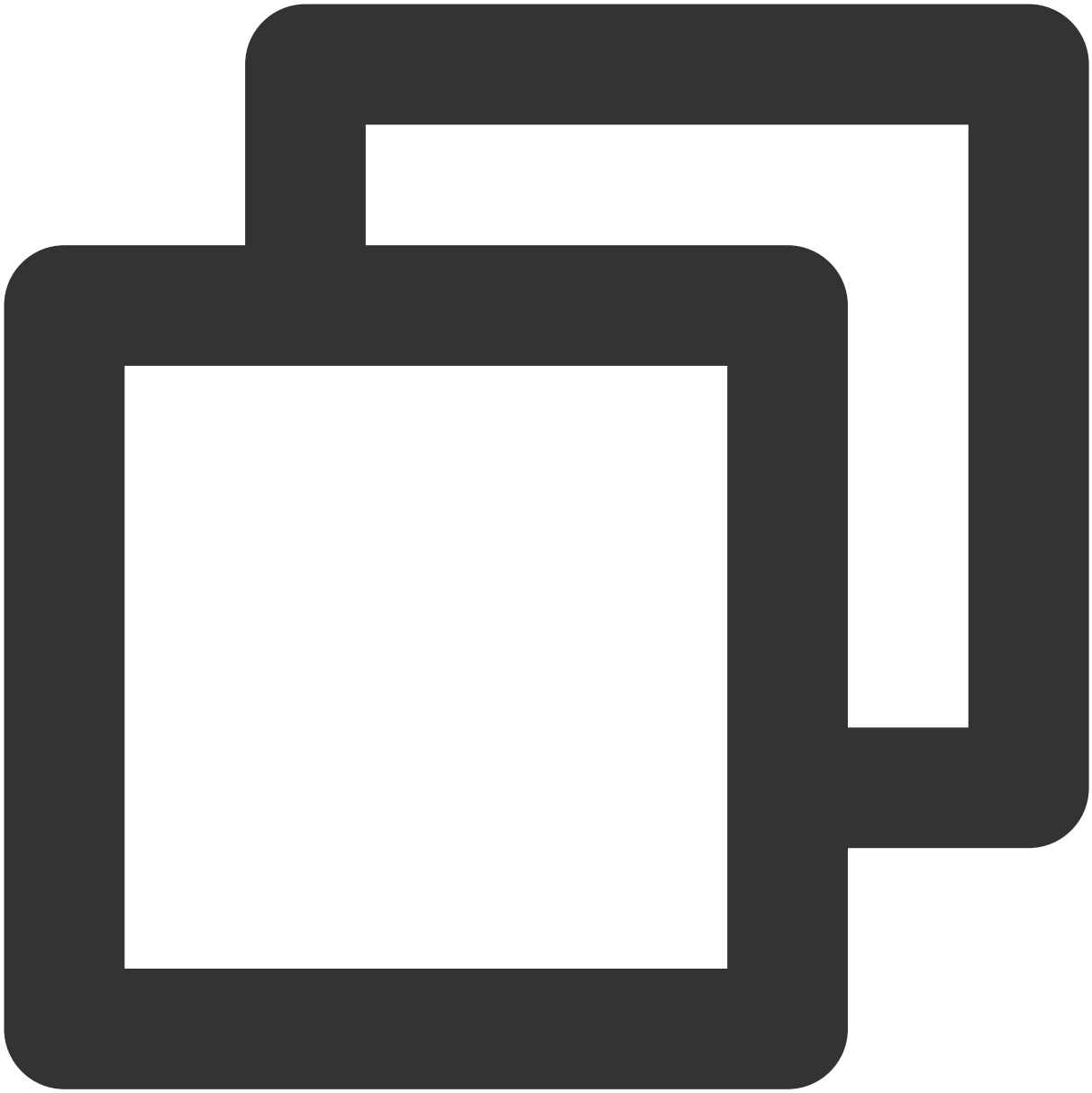


成功时请查看标准输出 stdout.job-xxx.xxxx.0.log，内容如下：



```
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);
```

执行远程代码包

最近更新时间：2024-01-13 11:19:28

操作场景

Batch 支持以 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 模式可以更方便地组织代码。

前提条件

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。

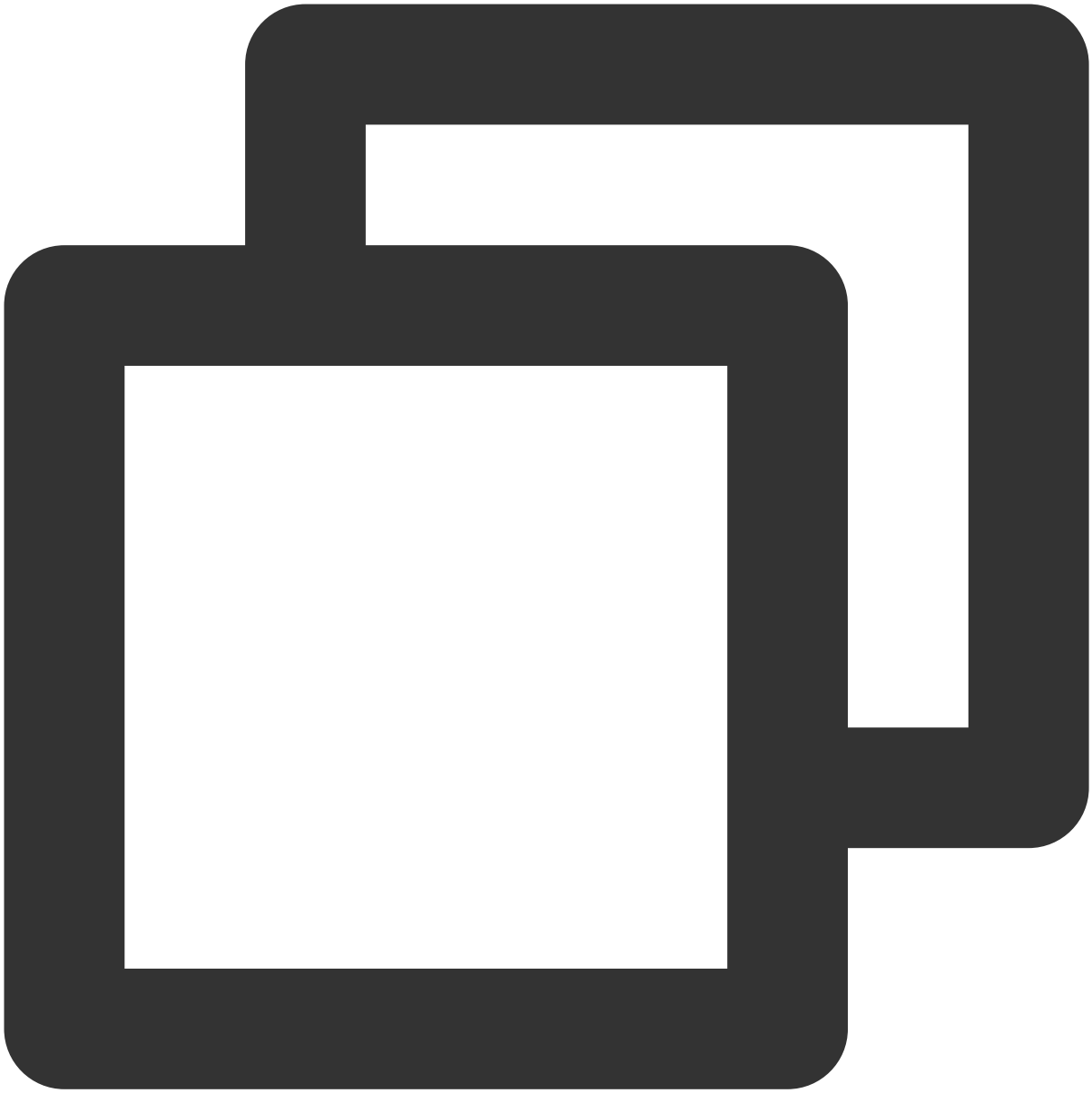
操作步骤

查看 Demo

说明：

请在 前置准备 中修改 `2_RemoteCodePkg.py` 文件自定义信息的通用部分。

使用编辑器打开 `2_RemoteCodePkg.py` 文件。

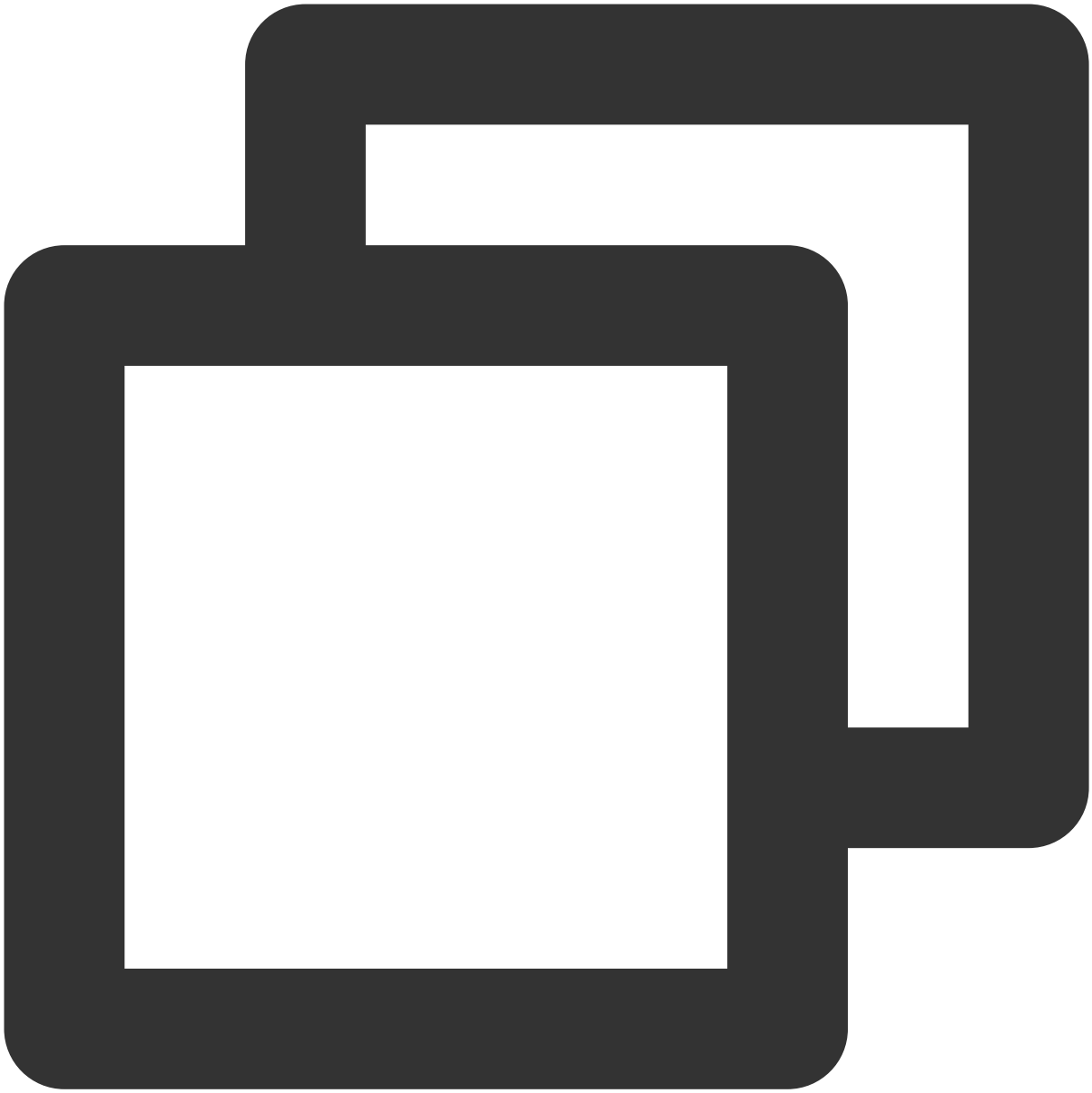


```
# custom (Change to your info)
imageId = "img-m4q71qnf"
Application = {
    "DeliveryForm": "PACKAGE",
    "Command": "python ./codepkg/fib.py",
    "PackagePath": "http://batchdemo-xxxxxxxxx.cos.ap-guangzhou.myqcloud.com/codepk
}
StdoutRedirectPath = "your cos path"
StderrRedirectPath = "your cos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

自定义部分除 Application 以外，都已在前置准备中说明， Application 中配置请参考下表：

配置项	描述
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，您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

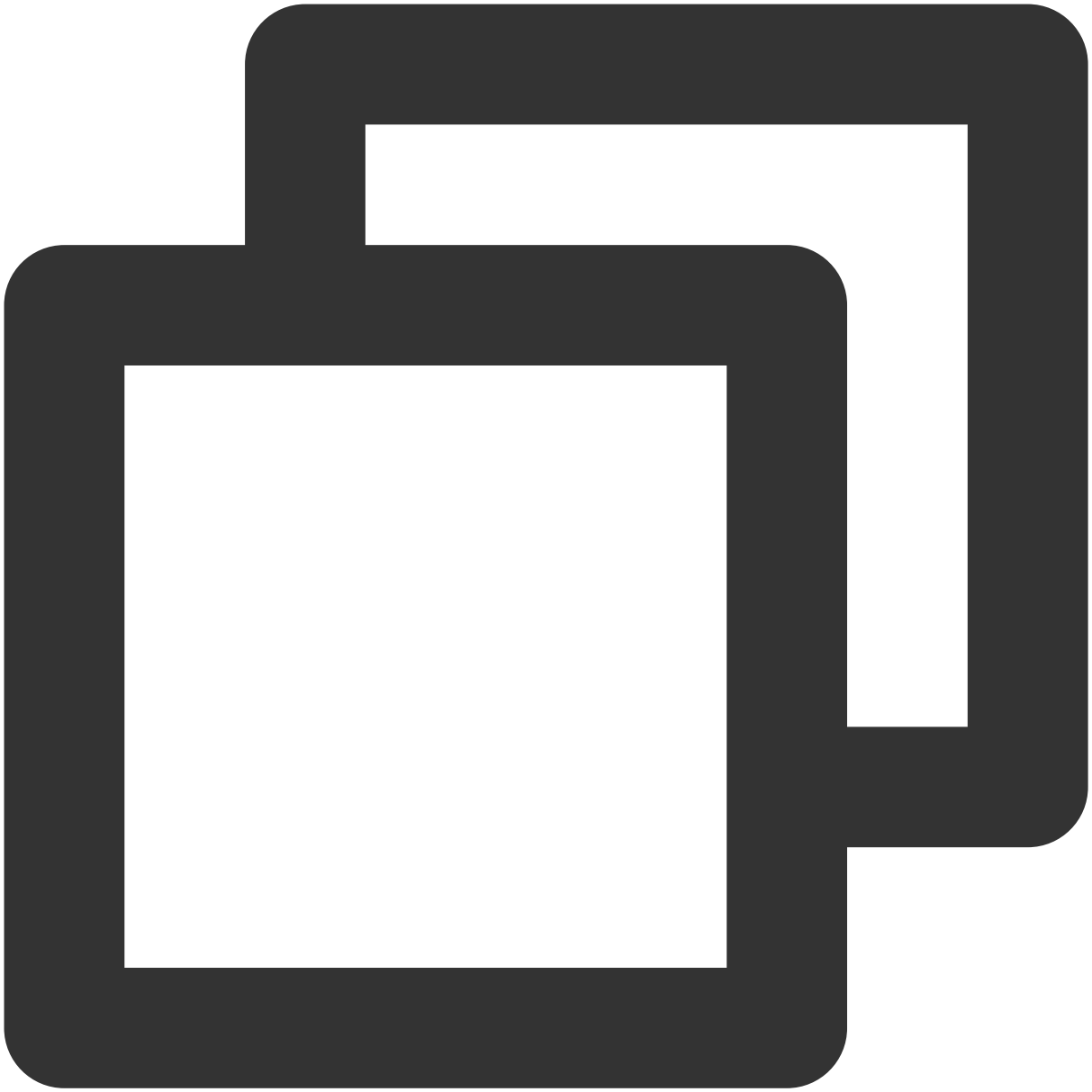


```
fib = lambda n:1 if n<=2 else fib(n-1)+fib(n-2)
print("Remote Code Package : %d"%(fib(20)))
```

提交作业

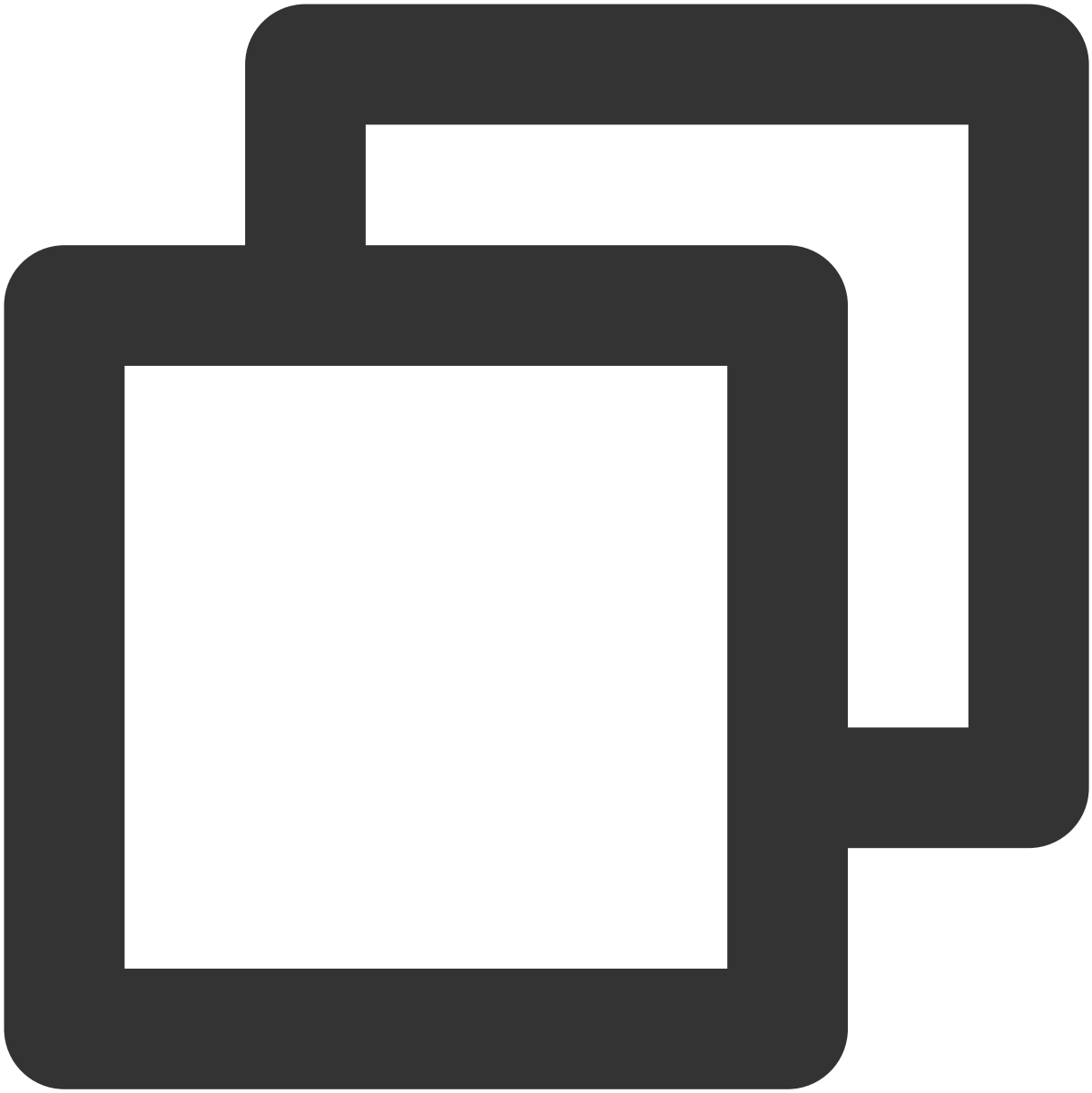
执行以下命令，执行 Python 脚本。

Demo 中已经通过 Python 脚本 + Batch 命令行工具的形式封装了提交作业流程。



```
python 2_RemoteCodePkg.py
```

返回结果如下所示，则表示提交成功。



```
{
    "RequestId": "c09e9291-2661-xxxx-8783-72d36f91ec8a",
    "JobId": "job-7xxxx26l"
}
```

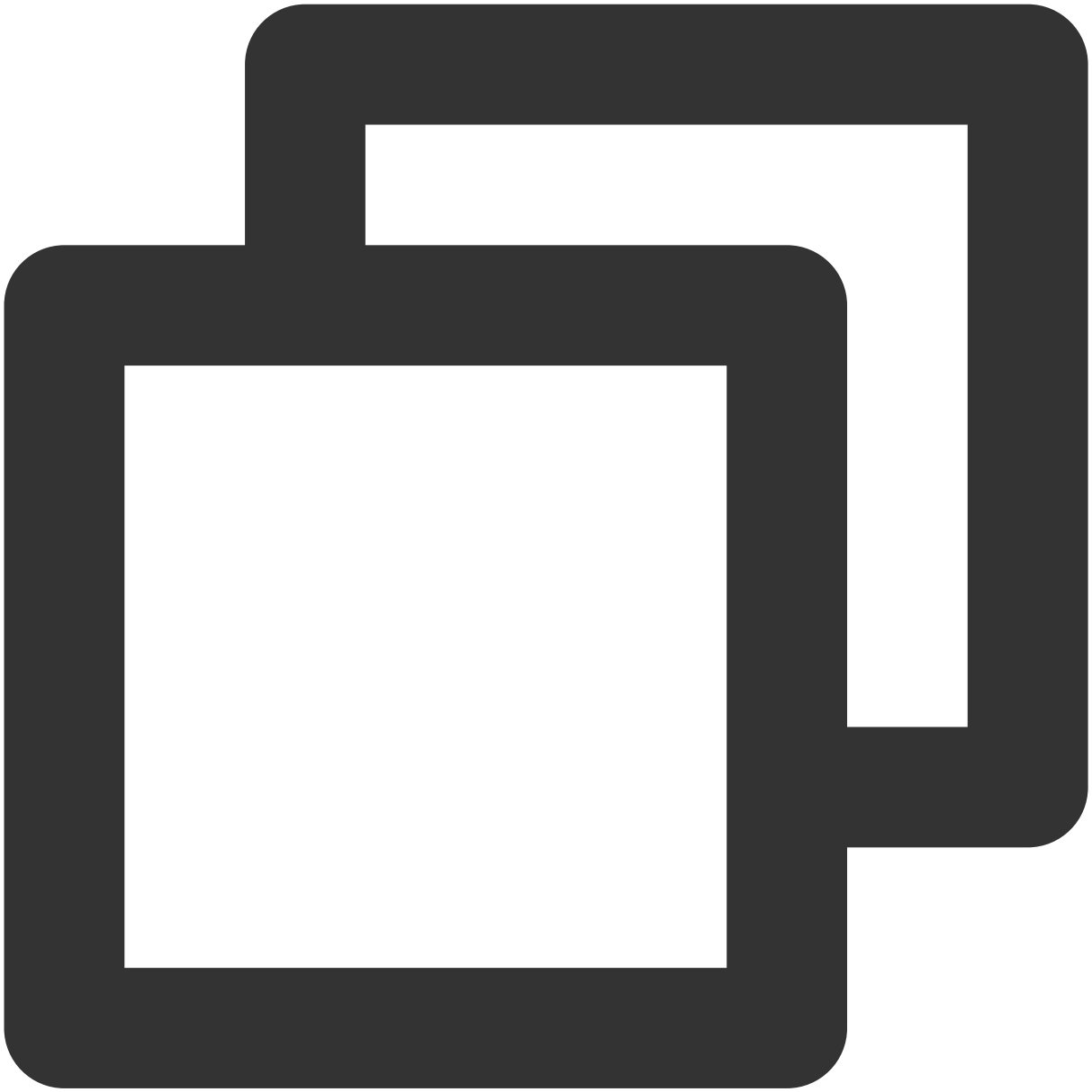
若未提交成功，请检查返回值排查错误，也可以通过 联系我们 。

查看状态

步骤同简单开始中的 查看状态。

查看结果

1. 步骤同简单开始中的 查看结果。
2. `2_RemoteCodePkg.py` 的执行结果如下：



```
Remote Code Package : 6765
```

远程存储映射

最近更新时间：2024-01-13 11:19:28

操作场景

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。

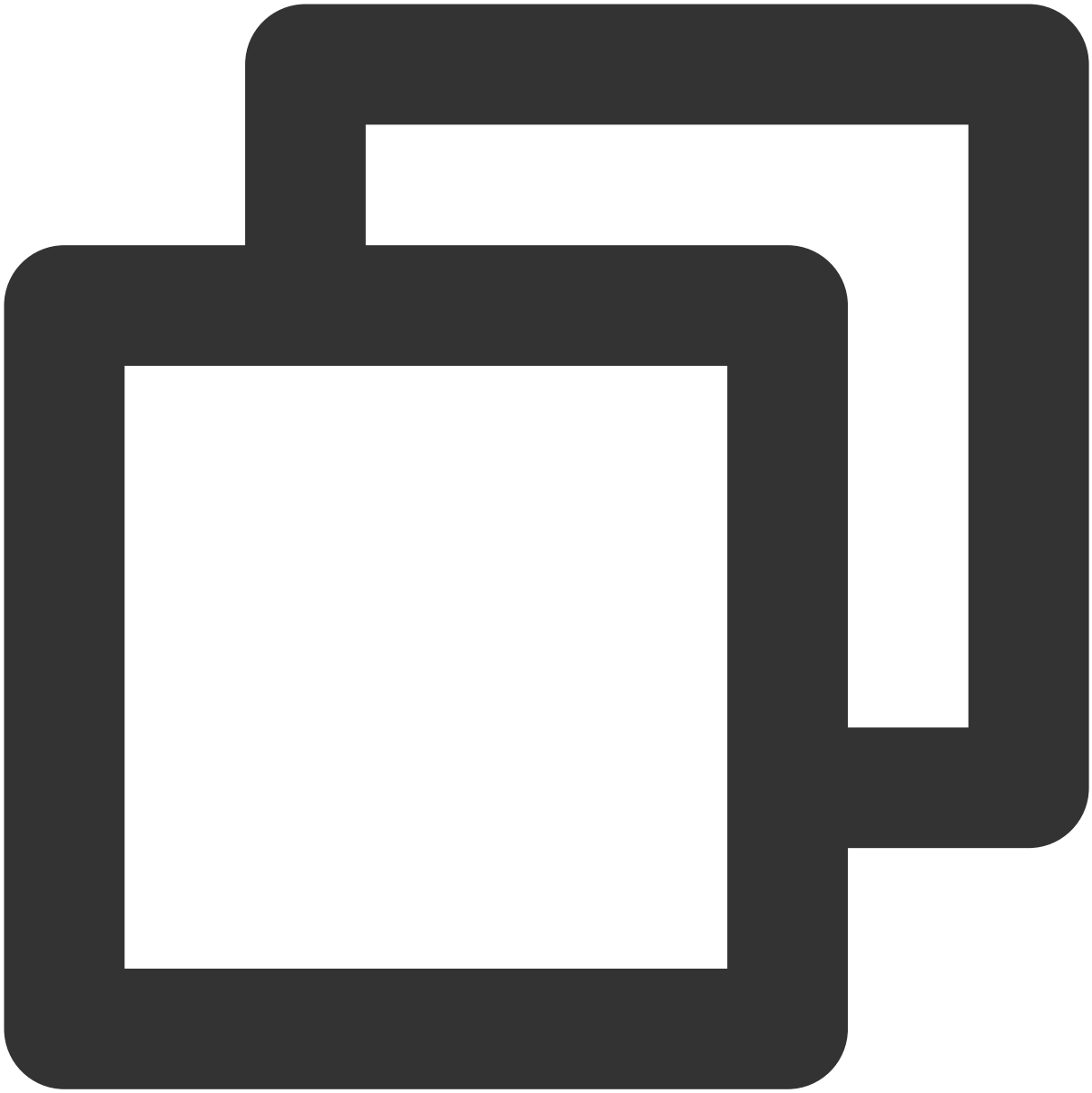
前提条件

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。

操作步骤

上传输入数据文件

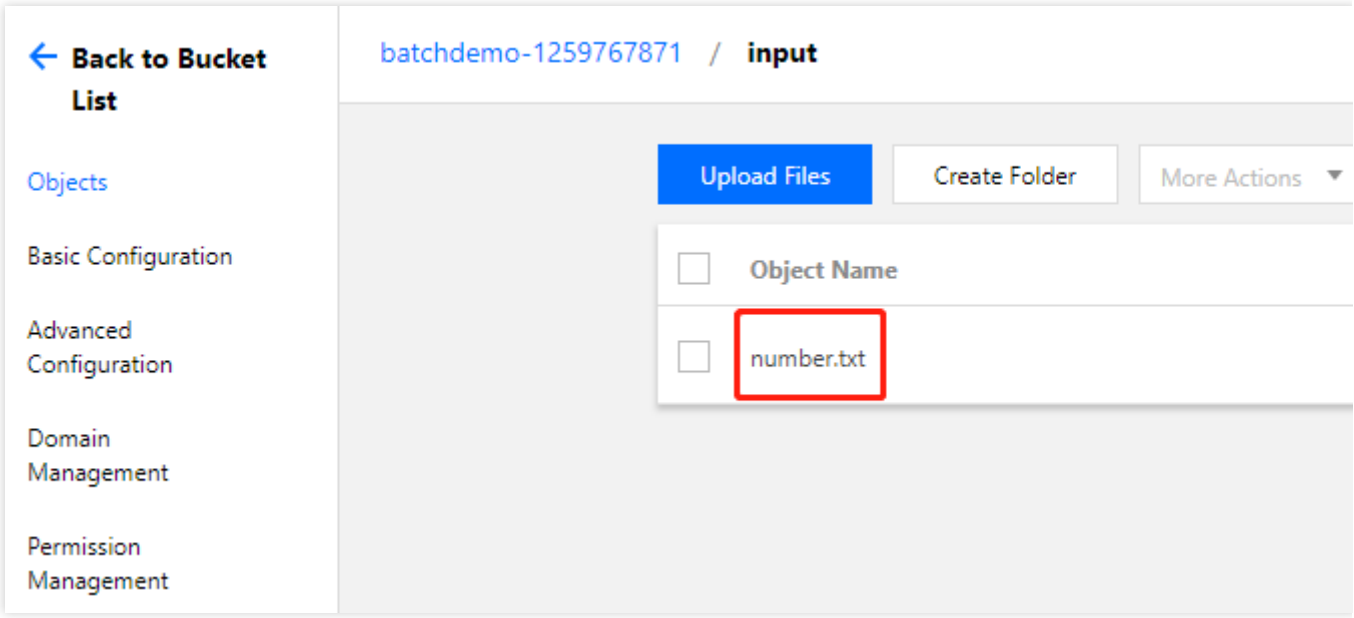
1. 创建 `number.txt` 文件，内容如下：



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

2. 登录对象存储控制台，单击左侧导航栏中的**存储桶列表**。

3. 选择已创建的 Bucket ID>**文件列表**>input 文件，上传 `number.txt` 。如下图所示：

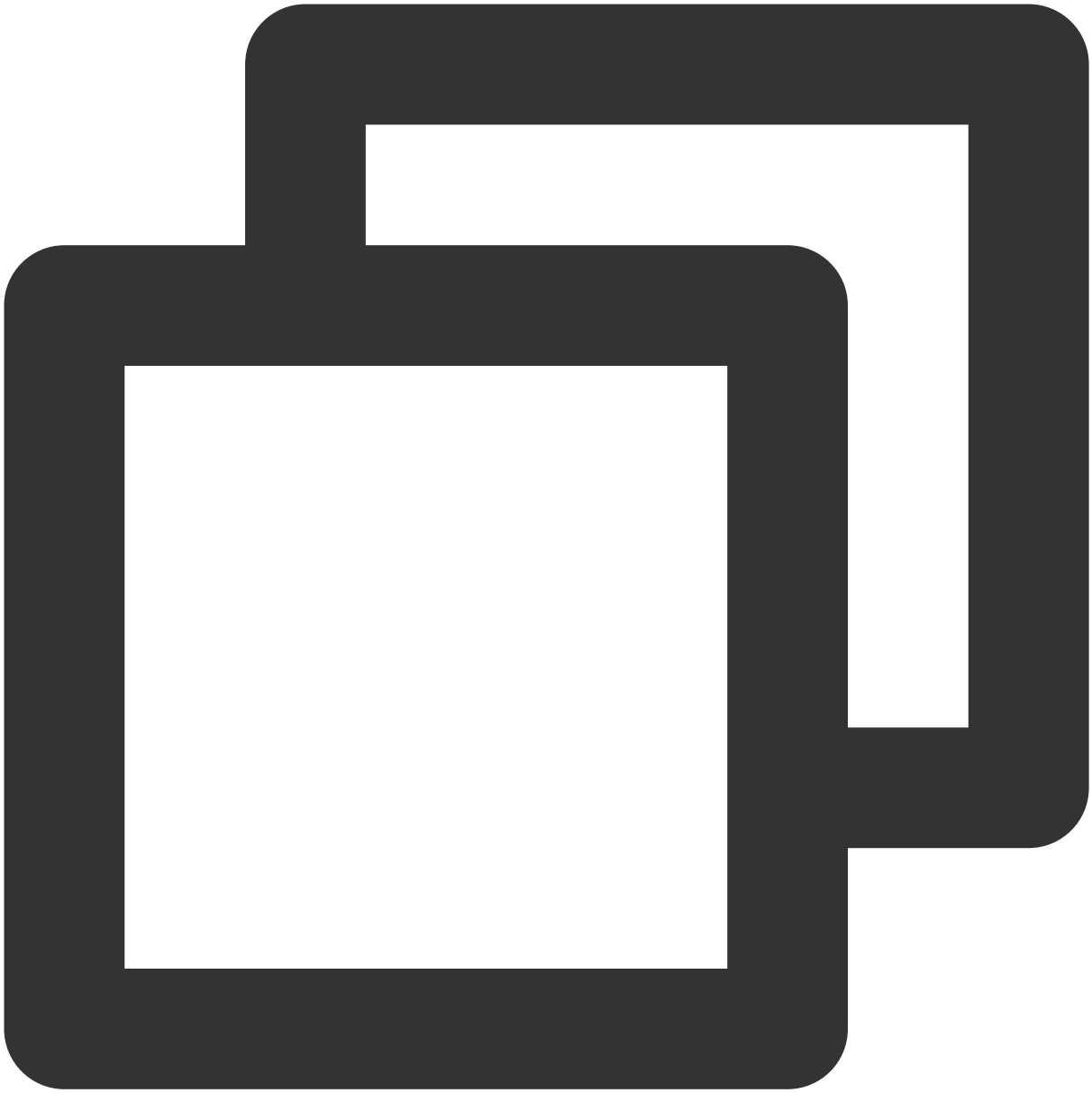


查看和修改 Demo

说明：

请在 前置准备 中修改 `3_StoreMapping.py` 文件自定义信息的通用部分。

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/sumnum.py",
    "PackagePath": "http://batchdemo-xxxxxxxxx.cos.ap-guangzhou.myqcloud.com/codepk
}
StdoutRedirectPath = "your cos path"
StderrRedirectPath = "your cos path"
InputMapping = {
    "SourcePath": "cos://batchdemo-xxxxxxxxx.cos.ap-guangzhou.myqcloud.com/input/",
```

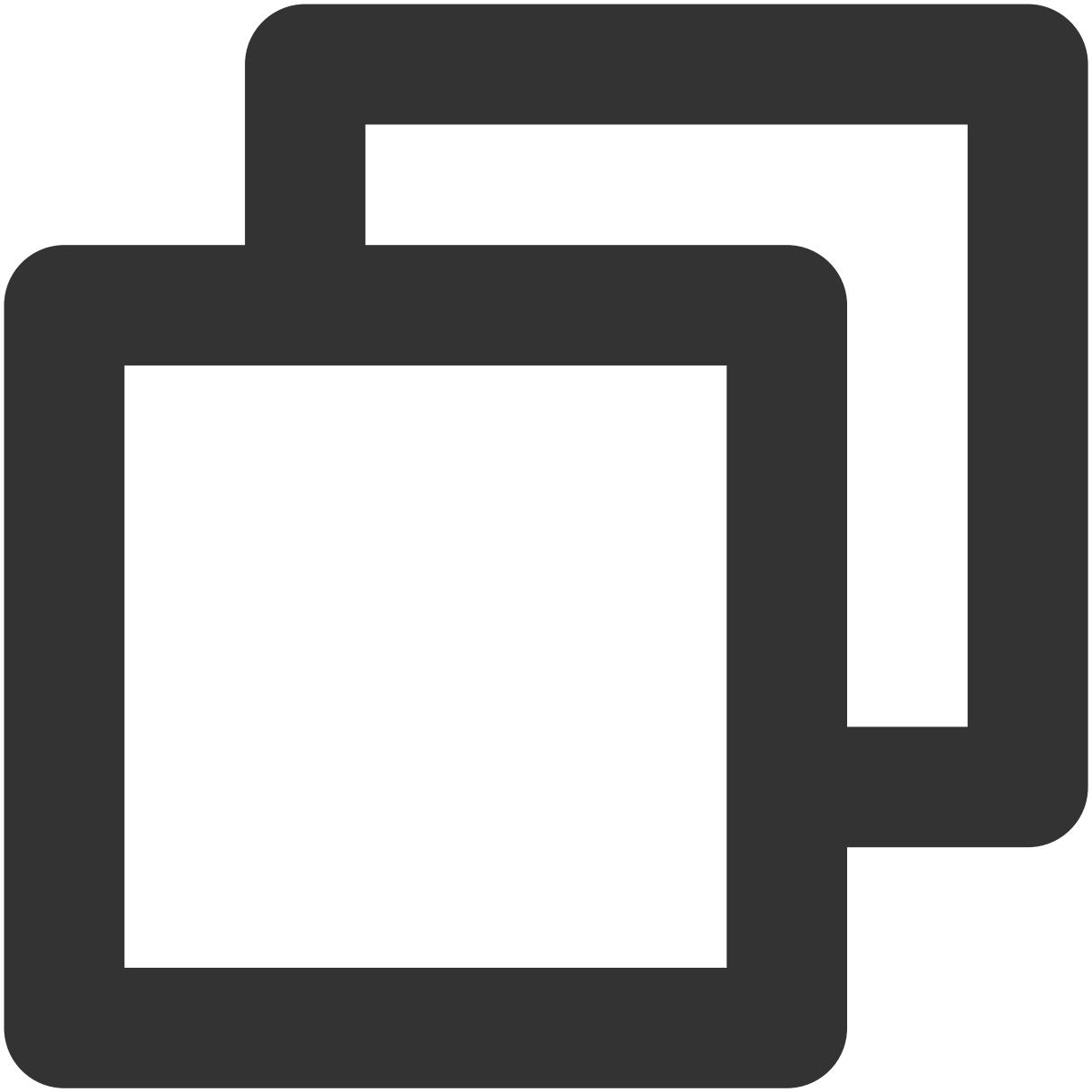
```
    "DestinationPath": "/data/input/"
}
OutputMapping = {
    "SourcePath": "/data/output/",
    "DestinationPath": "your output remote path"
}
```

与 `2_RemoteCodePkg.py` 相比，自定义部分中修改如下表：

配置项	描述
Application	Command 改为执行 sumnum.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，请参见 获取 COS 相关访问域名。 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，请参见 获取 COS 相关访问域名。

sumnum.py 的内容如下：

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。



```
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()
```

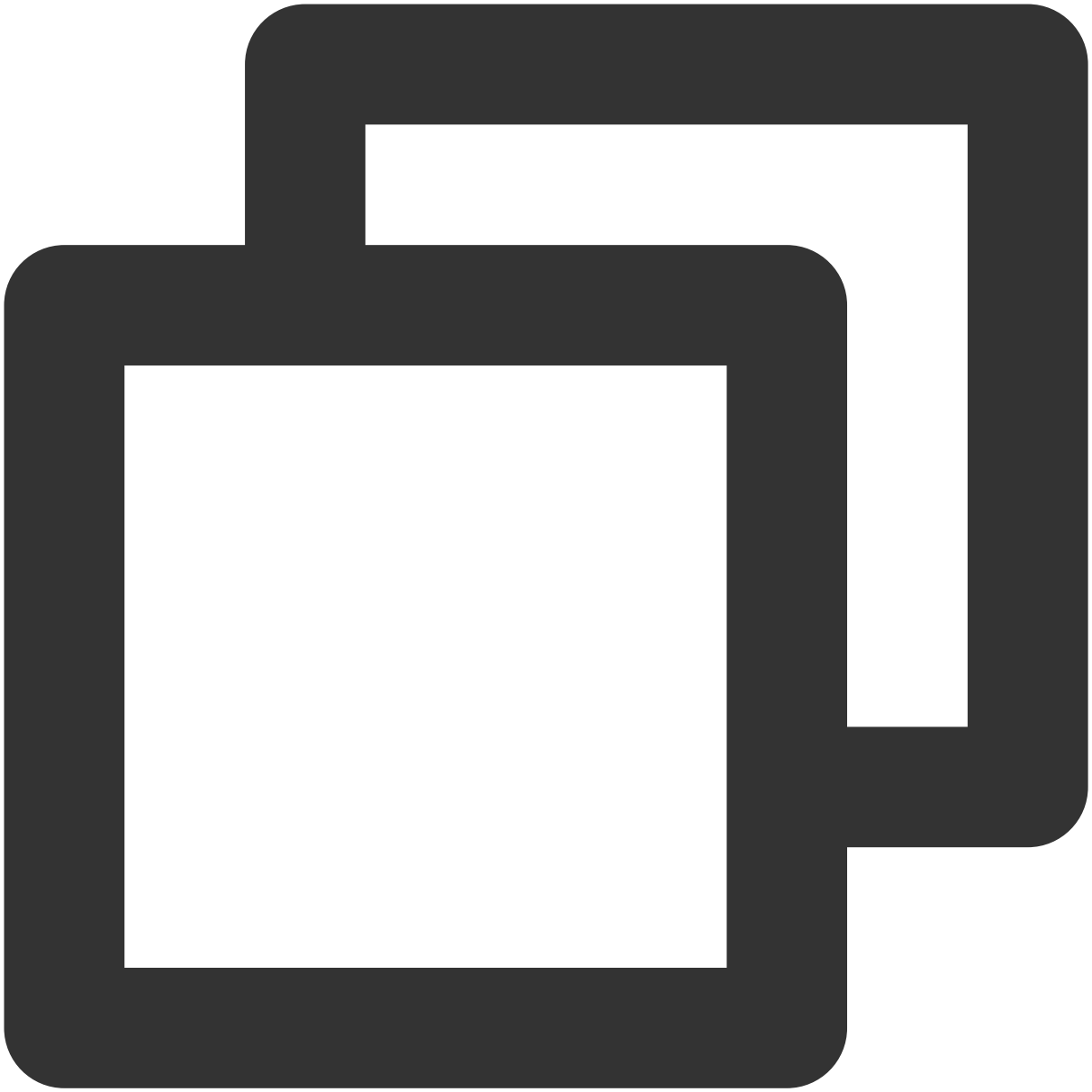
```
    print "total = ",total
    fwrite = open(outputfile, 'w')
    fwrite.write(str(total))
    fwrite.close()

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

提交作业

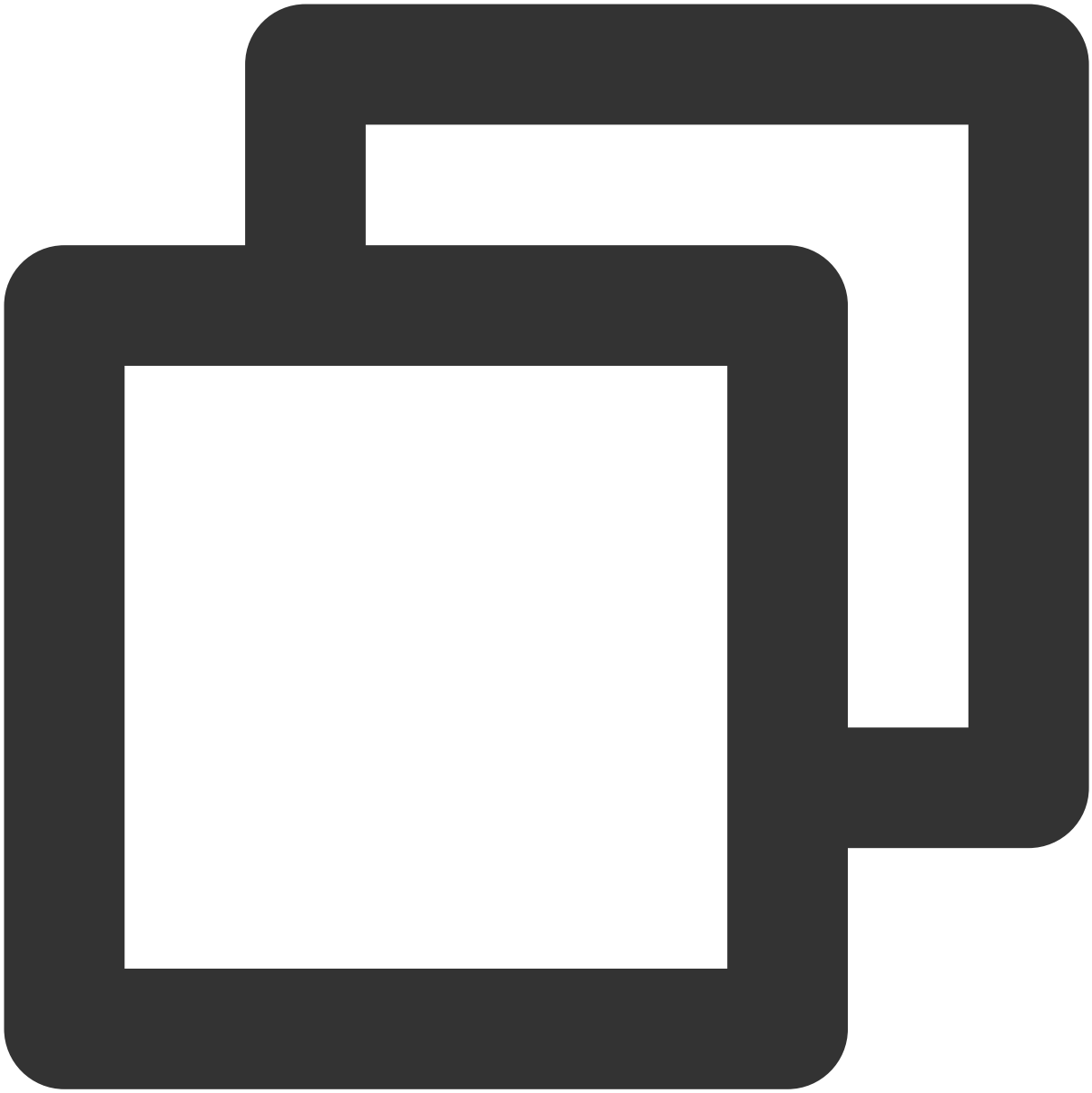
执行以下命令，执行 Python 脚本。

Demo 中已经通过 Python 脚本 + Batch 命令行工具的形式封装了提交作业流程。



```
python 3_StoreMapping.py
```

返回结果如下所示，则表示提交成功。



```
{
    "RequestId": "8eaeb01e-94a6-41a1-b40f-95f15417c0b4",
    "JobId": "job-97smiptb"
}
```

若未提交成功，请检查返回值排查错误，也可以通过 联系我们。

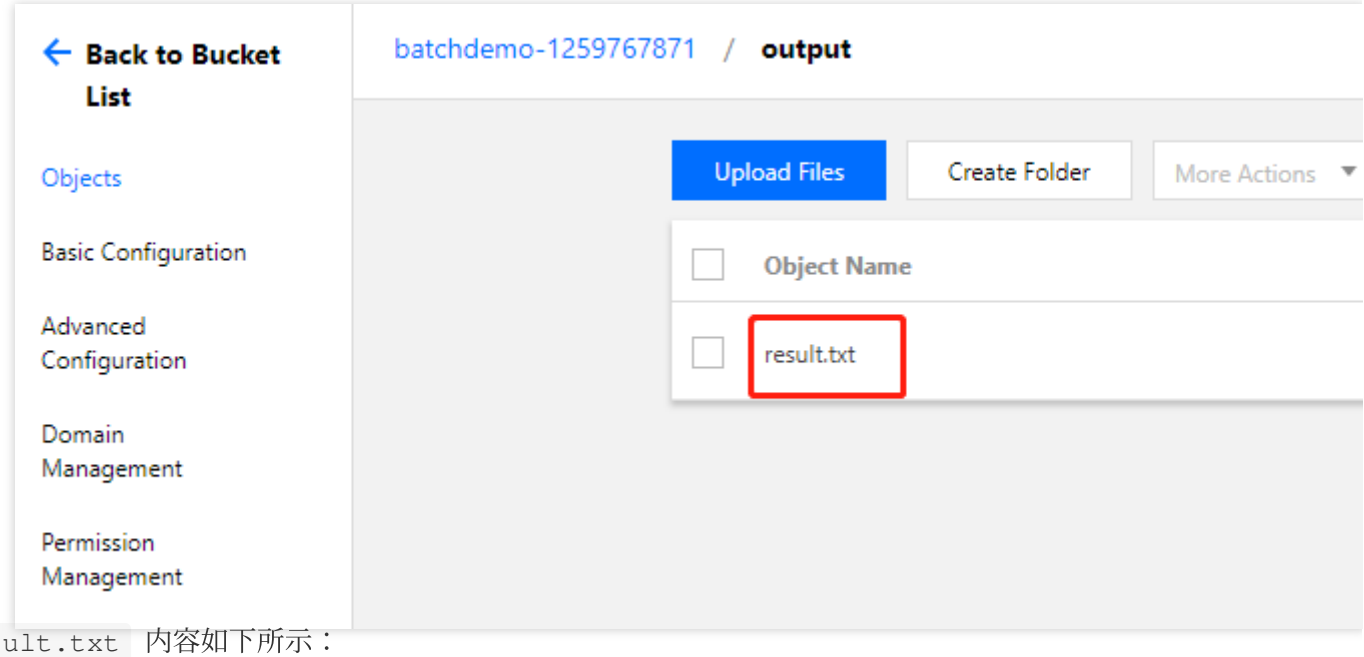
查看状态

步骤同简单开始中的 查看状态。

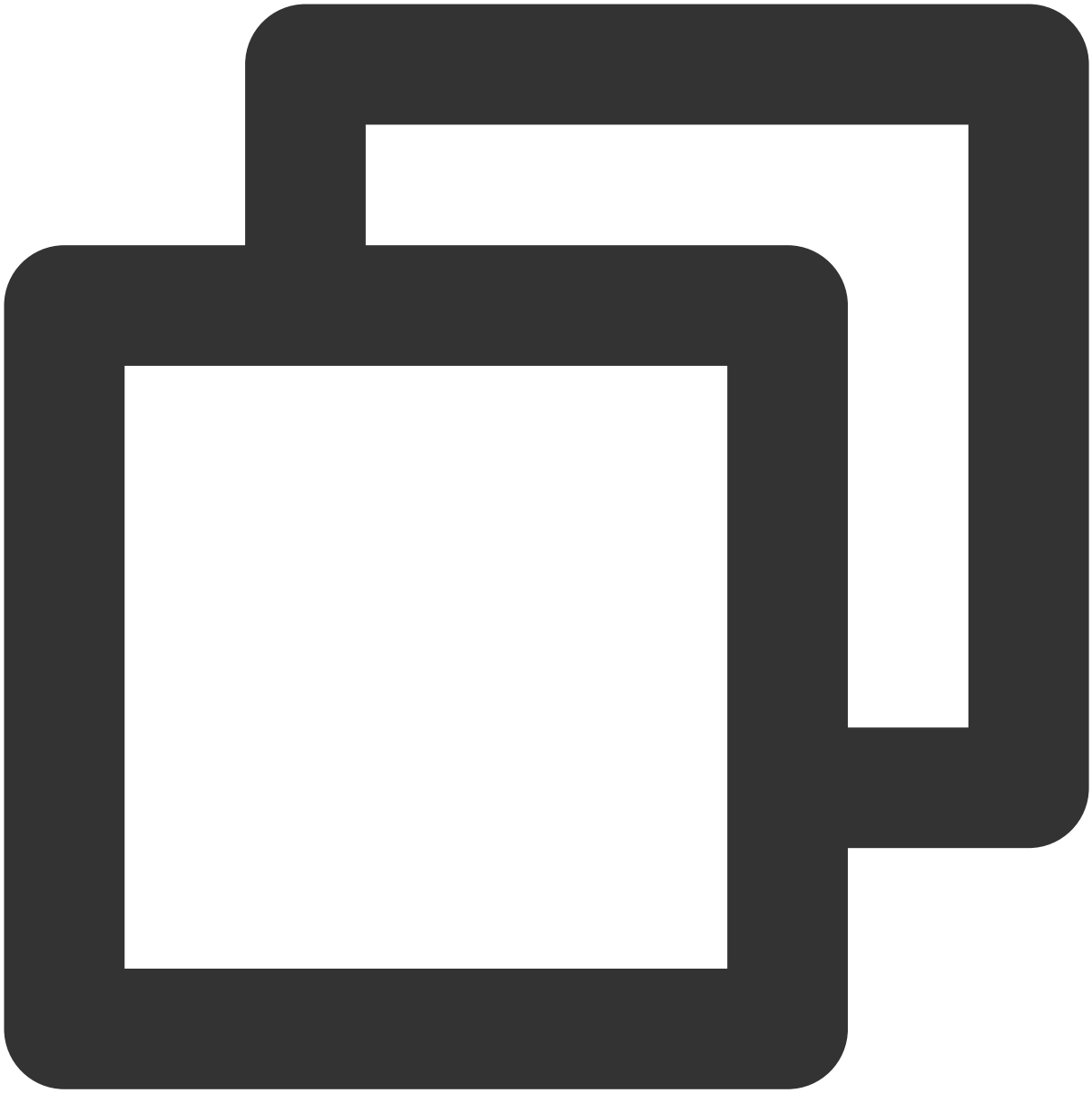
查看结果

1. 登录对象存储控制台，单击左侧导航栏中的**存储桶列表**。
2. 选择已创建的 **Bucket ID>文件列表>output 文件**。如下图所示：

Batch 会将输出数据从本地目录靠白道远程存储目录中，`3_StoreMapping.py` 的执行结果保存在 `result.txt` 中，`result.txt` 将自动同步到 COS 中。



`result.txt` 内容如下所示：



```
45
```