Stream Compute Service

Python Developer Guide

Product Documentation

Python Developer Guide

Last updated：2023-11-08 16:21:54

Overview

With Stream Compute Service, you can use Python to develop Flink jobs and use all Flink features in a Python environment. This makes development easier for you while making the most of Python's advantages in terms of data processing and machine learning.
This document shows you how to develop a Python job with a private cluster. It includes the following two sections:
Environment information
Using Python dependencies

Environment information

Software version

Stream Compute Service supports Python jobs developed using the open-source framework Flink 1.13 and is installed with Python 3.7.

Python software

The Python environment of Stream Compute Service is installed with the following software.

Software	Version
apache-beam	2.27.0
apache-flink	1.13.2
apache-flink-libraries	1.13.2
avro-python3	1.9.1
beautifulsoup4	4,10.0
certifi	2020.12.5
chardet	4.0.0
click	8.0.3
cloudpickle	1.2.2

crcmod	1.7
Cython	0.29.16
dill	0.3.1.1
docopt	0.6.2
fastavro	0.23.6
future	0.18.2
grpcio	1.29.0
hdfs	2.6.0
httplib2	0.17.4
idna	2.10
importlib-metadata	4.10.0
joblib	1.1.0
jsonpickle	1.2
mock	2.0.0
nltk	3.6.7
numpy	1.19.5
oauth2client	3.0.0
pandas	1.0.0
pbr	5.5.1
protobuf	3.15.3
py4j	0.10.8.1
pyarrow	0.17.1
pyasn1	0.4.8
pyasn1-modules	0.2.8
pydot	1.4.2

pymongo	3.11.3
pyparsing	2.4.7
python-dateutil	2.8.0
pytz	2021.1
regex	2021.11.10
requests	2.25.1
rsa	4.7.2
scikit-learn	1.0.2
scipy	1.7.3
six	1.15.0
soupsieve	2.3.1
threadpoolctl	3.0.0
tqdm	4.62.3
typing-extensions	3.7.4.3
urllib3	1.26.3
zipp	3.7.0

Using Python dependencies

You can use dependencies including third-party Python packages, JAR packages, and data files in a Python job.

Third-party Python packages

To use a Python package in your Python job, follow the steps below:

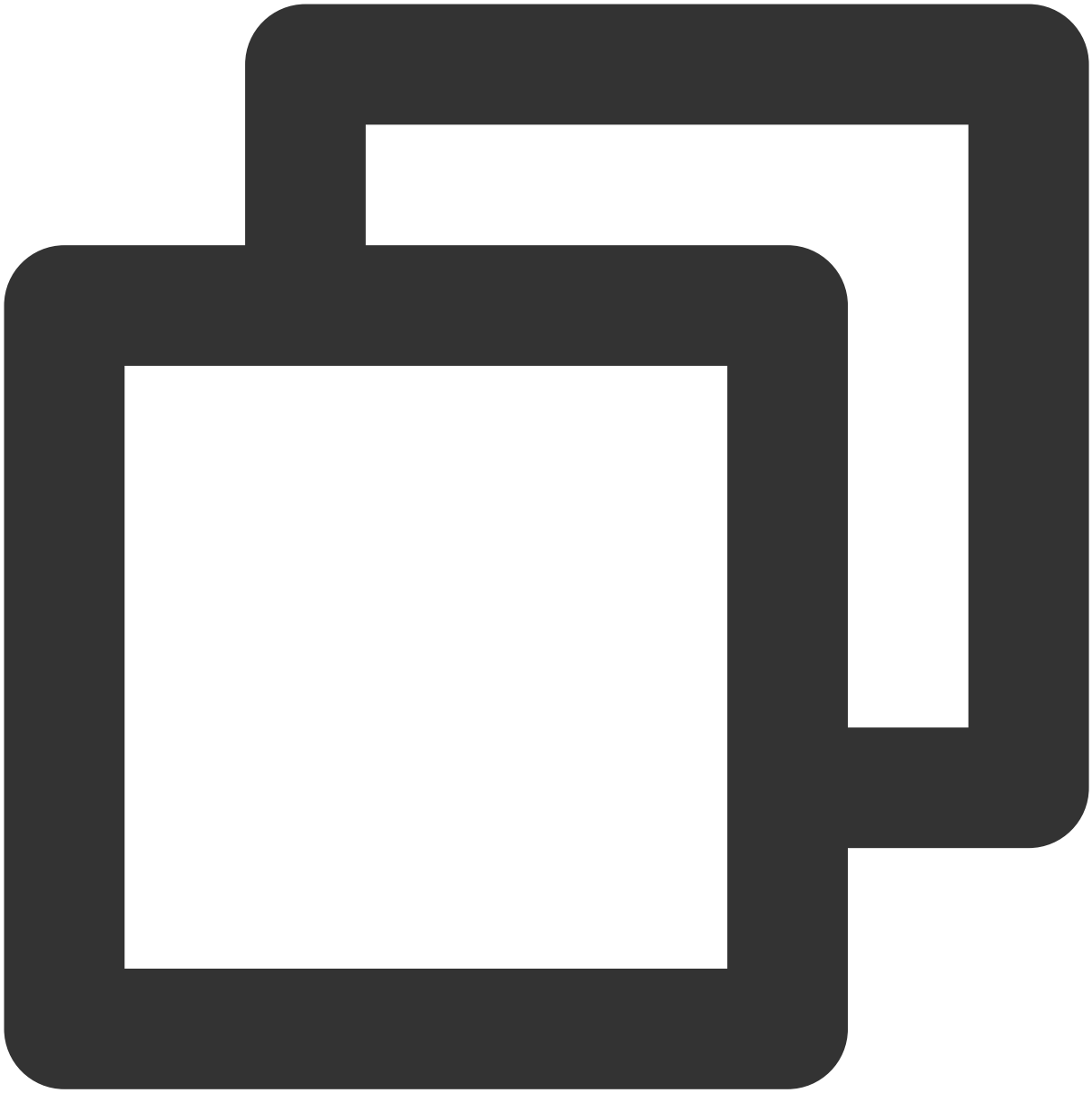
1. Zip the Python package and upload the ZIP file in **Dependencies**.

Use the `pip install xxx -t .` command to install the package to the current directory. Then run `zip -r xxx.zip xxx/*` to generate a ZIP file.

Note

If the package includes SO files, your environment must be Debian 11.1.

The example below shows you how to generate a ZIP file for Requests packages.



```
mkdir /tmp/example
cd /tmp/example
pip install requests -t .
zip -r9 ../pyflink_lib_example.zip ./*
```

2. On the **Development & Testing** page, click **Add Python package** and select the ZIP file you uploaded.

JAR packages

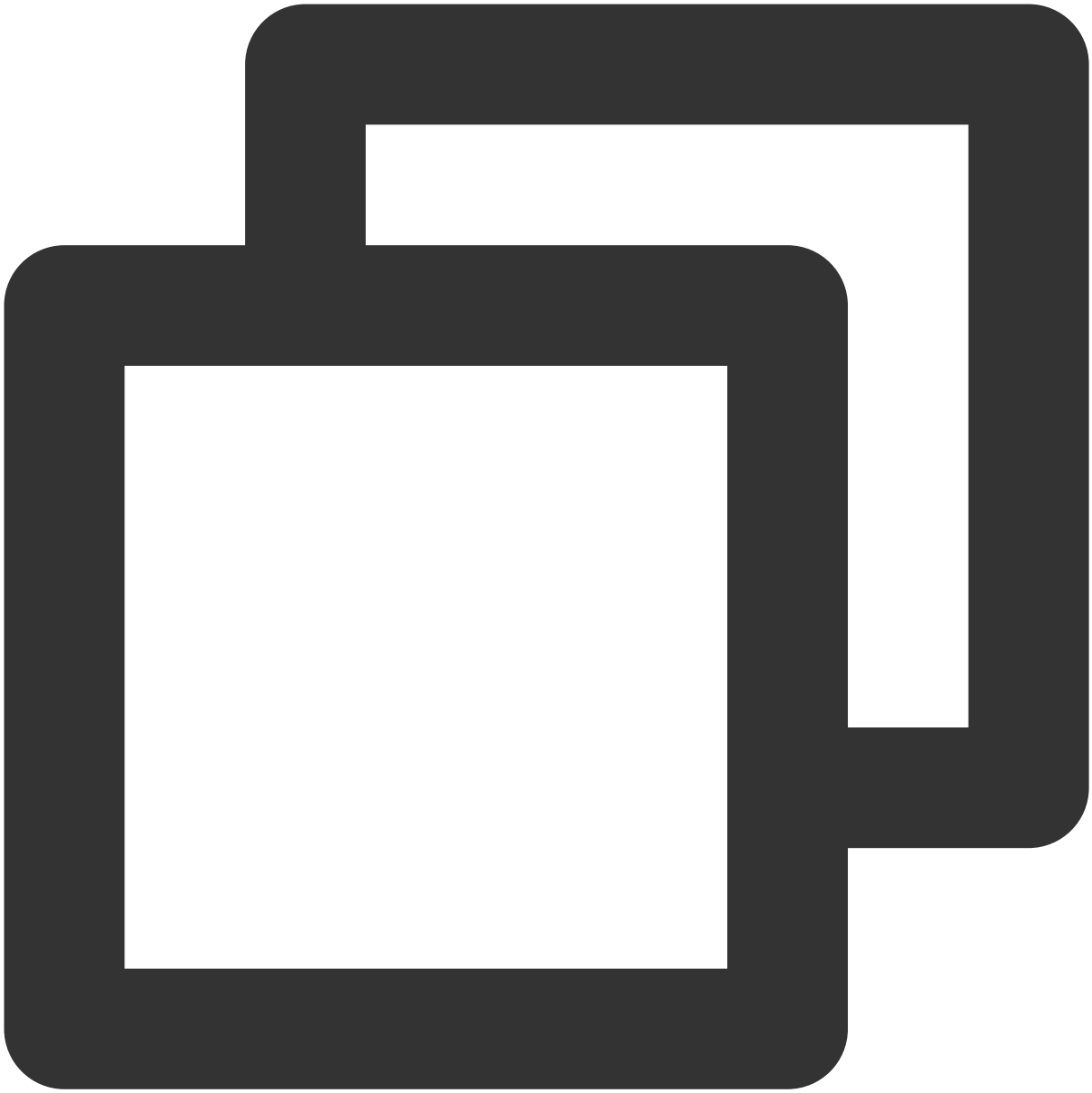
To use Java classes such as connectors or custom Java functions in your Python job, follow the steps below to reference the JAR package.

1. Upload the JAR package in **Dependencies**.
2. On the **Development & Testing** page, click **Referenced JAR package** and select the JAR package you uploaded.

Data files

If you have a lot of data files, you can zip them and use the ZIP file in your Python job.

1. Upload a ZIP package of the data files in **Dependencies**.
2. On the **Development & Testing** page, select the ZIP file you uploaded.
3. Use the data files with a custom Python function. Assume that you zipped your data files and generated the ZIP file `archive.zip`. You can use the following custom Python function to access the data files.



```
def my_udf():
     with open("archive.zip/mydata/data.txt") as f:
     ...
```