Module 12: Using the Python Connector for Snowflake

This experiment provides instructions for installing and validating the Snowflake Connector for Python. The connector can currently be installed in Linux, macOS, and Windows environments.

The developer notes are hosted on GitHub, along with the source code.

In this Experiment:

Prerequisites

For a list of the operating systems supported by Snowflake clients, see Operating System Support.

The following software packages are required to install the Snowflake Connector for Python.

Python

The Snowflake Connector for Python requires one of the following supported versions of Python:

- 3.6
- 3.7
- 3.8
- 3.9

To verify your version of Python:

```
python --version
```

For more information about installing the required version of Python, see:

- Installing the Required Version of Python

Python Package Installer and Setup Tools

The Snowflake Connector for Python is installed by `pip`, a standard Python package installer and manager.

Use pip version 19.0 or later. Execute the following command to ensure the required version is installed:

```
python -m pip install --upgrade pip
```

Note

- On macOS, use either `virtualenv` or `venv` to install Python and the connector.
- If both Python 2.7.x and Python 3.x are installed, use `pip3` to install the connector with Python 3.x.

Python Packages

The Snowflake Connector for Python uses many Python packages. The connector supports a range of versions for each package. (For details, see Dependency Management Policy for the Python Connector.)

For these packages, Snowflake recommends that you use the same versions that Snowflake used during testing (the minimum supported versions of these

packages). Snowflake also recommends that you avoid overriding pinned dependencies and using applications that might override pinned dependencies.

The package versions that were tested with the connector are documented in these requirements files. Each requirements file applies to a specific version of Python. For example, `requirements_36.reqs` lists the versions that were tested with Python 3.6.

Each requirements file consists of lines like the following:

```
asn1crypto==1.3.0
```

The element to the left of the `==` is the name of the package, and the element to the right of the `==` is the version number of that package.

For instructions on using these requirements files to install the required packages, see Step 1: Install the Connector.

pyOpenSSL (macOS only)

macOS (Yosemite and higher versions) includes Python 2.7.x. If you are using python 2.7.x, and if your version of `pyOpenSSL` is out-of-date, you might encounter the following error if you are not using `virtualenv`:

```
File "/Library/Python/2.7/site-
packages/snowflake/connector/ocsp_pyopenssl.py", line nn, in dump_publickey
  bio = OpenSSL.crypto._new_mem_buf()
AttributeError: 'module' object has no attribute '_new_mem_buf'
```

To fix this issue, use one of the following two options:

- Use either `virtualenv` or `venv` to isolate the Python runtime environments.
- Set the PYTHONPATH environment variable so that the newly-installed `pyOpenSSL` is used instead. For example:
- `export PYTHONPATH=/Library/Python/2.7/site-packages`

Note

Recent and future versions of the Python connector do not support Python 2.x. If you are still using Python 2.7 and an older version of the Python connector, Snowflake encourages you to upgrade both the connector and Python.

OpenSSL and FFI (Linux only)

When the Snowflake Connector for Python is installed, `pip` compiles native codes in the packages on Linux platforms. In order to install it successfully, install the required packages:

- For CentOS, use `yum`:
- `sudo yum install -y libffi-devel openssl-devel`
- For Ubuntu, use `apt-get`:
- `sudo apt-get install -y libssl-dev libffi-dev`

Other platforms do not need the OS packages installed because they are bundled in the Python packages.

Install the Connector

The Snowflake Connector for Python is available in PyPI. A change log is available on the site, so you can determine the changes that have been implemented in each release.

When installing a version of the Snowflake Connector for Python, Snowflake recommends installing the versions of the dependent libraries that have been tested with that version of the connector.

To install the Snowflake Connector for Python and the dependent libraries:

1. Determine the version of the Snowflake Connector for Python that you plan to install.

2. To install the dependent libraries, run the `pip` (or `pip3`) command and point to the requirements file for that version of the connector.

   For example, suppose the latest Snowflake Connector for Python version is 2.7.9 and you are using Python 3.6. To install the dependent libraries for that version of the connector, run the following command:

   ```
   pip install -r
   https://raw.githubusercontent.com/snowflakedb/snowflake-
   connector-python/v2.7.9/tested_requirements/requirements_36.reqs
   ```

   In the example above, the path to the requirements file specifies the version of the connector (“/v2.7.9/”). The requirements filename (“requirements_36.reqs”) specifies the version of Python (Python 3.6).

   Note

   If you need to install a version of the Snowflake Connector for Python that is between 2.2.0 and 2.3.5, replace `.reqs` in the requirements filename with `.txt`. For example, use `requirements_36.txt`, rather than `requirements_36.reqs`.

   If running the `pip` command results in compilation errors, you might need to install the C compiler and Python development package to build some of the required modules, such as PyCryptoDome.

   For more information about installing a C compiler, see http://gcc.gnu.org/(Linux) or https://developer.apple.com/xcode/(macOS).

3. To install the connector, run the following command:

   ```
   pip install snowflake-connector-python==<version>
   ```

   where *`version`* is the version of the connector that you want to install.

For example, to install version 2.7.9 of the Snowflake Connector for Python, run:

```
pip install snowflake-connector-python==2.7.9
```

Note

If you plan to cache connections with browser-based SSO, you must also install the `secure-local-storage` extra. For details, see Using Connection Caching to Minimize the Number of Prompts for Authentication — Optional.

If you plan to use the API support for Pandas DataFrames, you must also install the `pandas` extra. For details, see Using Pandas DataFrames with the Python Connector.

Verify Your Installation

Create a file (e.g. `validate.py`) containing the following Python sample code, which connects to Snowflake and displays the Snowflake version:

```
#!/usr/bin/env python
import snowflake.connector

# Gets the version
ctx = snowflake.connector.connect(
    user='<user_name>',
    password='<password>',
    account='<account_identifier>'
    )
cs = ctx.cursor()
try:
    cs.execute("SELECT current_version()")
    one_row = cs.fetchone()
    print(one_row[0])
finally:
    cs.close()
ctx.close()
```

Before running the example:

- Replace `<account_identifier>` with your account identifier.

  For details and examples, see Usage Notes for the account Parameter (for the connect Method).

- Replace `<user_name>` and `<password>` with the user name and password that you use to connect to Snowflake.

For more information about the Snowflake Python API, see Python Connector API, specifically the `snowflake.connector` methods for details about the supported connector parameters.

Next, execute the sample code. For example, if you created a file named `validate.py`:

```
python validate.py
```

The Snowflake version (e.g. `3.5.0`) should be displayed.

If you see the following error message, your Python installation likely does not have the appropriate security fixes:

```
ERROR: The ssl package installed with your Python package - version n.n.n -
does not have the required security fixes. Upgrade to 3.6.0 or higher.
```

Specify a Temporary Directory

The Snowflake Connector for Python uses a temporary directory to store data for loading and unloading (PUT, GET), as well as other types of temporary data.

The temporary directory can be explicitly specified by setting the `TMPDIR`, `TEMP` or `TMP` environment variables, otherwise the operating system's default temporary directory (i.e. `/tmp`, `C:\temp`) is used.

If the system’s default temporary directory volume is not large enough for the data being processed, you should specify a different directory using any of the supported environment variables.

For example, from a terminal window, execute the following command:

```
export TMPDIR=/large_tmp_volume
```