Globus Transfer Documentation

Lawrence Berkeley National Laboratory - HPCS

Summary

This program performs a Globus transfer between two endpoints such that the designated directory at the destination is a superset of the files contained in the designated directory at the source. This can be automated by defining a CRON job that periodically runs the program.

Package Contents

Program files are stored in a directory named `globus_auto`, for the following purposes:

File Name	Description
`config.py`	A list of definitions for the details of transfers, settings and preferences, and constants required for program functionality. This is the only file that the user should have to edit in normal use.
`datastore*`	The mechanism that detects new files or changes to existing files by storing an MD5 hash of the file's path and contents.
`__init__.py`	The means by which Python recognizes files as part of a package.
`main.py`	The main program to be executed by the user, and the only interface the user should have to interact with in normal use.
`refresh_token`	A Globus token that authenticates transfers, generated after the first execution of `main.py`.
`utils.py`	Utility functions.

Requirements

The program uses a package, globus-sdk, that require Python 2.7 or above. It can be installed using `pip install globus-sdk`. The initial working version is globus-sdk 1.5.0.

A client application must be created under the main user's Globus account to authorize transfers. Add the application's ID to `config.py`. Follow Steps 1 and 2 of this tutorial to create one: http://globus-sdk-python.readthedocs.io/en/stable/tutorial/.

How It Works

When the program is initially run, it generates a database file that stores state about the files in the directory being synced and then performs an initial transfer to the destination. If either source or destination endpoint is not ready, an error e-mail is sent to the user.

The Globus transfer syncs the source and destination directories such that the destination contains all of the contents of the source. Deleting objects on either side does not affect the other, nor do transfers affect existing objects in the destination, unless they share the same name with objects in the source. Transfers use a checksum for validation and are encrypted with TLS.

The program only initiates a transfer if a new file is added to or an existing file is changed in the source directory. Consequently, adding an empty directory will not initiate a transfer. In addition, only changing the destination ID will not trigger a transfer.

Configuration

Before running the program, the user should verify the settings in `config.py`.

`SRC_DIR` and `DST_DIR` denote the absolute paths to the directories at the source and destination endpoints. Note that only the contents of the directory at the source endpoint are copied to the destination, so the user should include the name of the source directory in the destination path. In practice, after initial setup, only these variables should need to be edited.

`EMAIL_SENDER` and `EMAIL_RECIPIENTS` refer to the e-mail address from which error e-mails will be sent and the list of recipients of those e-mails.

`SRC_ID` and `DST_ID` denote the Globus IDs of each endpoint. These can be found on the Globus website, labeled UUID.

`CLIENT_ID` is the ID for the Globus client application authorizing transfers. This can be found at https://developers.globus.org, once a client application has been created.

`CODE_PATH` and `PYTHON_PATH` denote the absolute paths to the code and to the installation of Python. `CODE_PATH` must end in a forward slash.

The remaining variables in `config.py` should not be modified.

Running the Program

Ensure that both endpoints are ready for transfer.

To start a transfer, run `python main.py`. If no `datastore` file is found or new files or changes are detected, a transfer is initiated.

To reset the program, delete the `datastore` file. This removes all file state.

Note that the first time the program is run, user authentication will occur. This will generate a `refresh_token`, which will then be used for future authentication without user intervention.

Contact

Please direct any questions and report any bugs to hpcshelp@lbl.gov.