Dask Azure Blob FileSystem Documentation

Release 0.1.1.dev3+ge72bd3b

Manish Sinha

Dec 03, 2018

Contents:

1 Dask Azure Blob FileSystem **1**
- 1.1 Features . . . 1
- 1.2 Usage . . . 1
- 1.3 Installation . . . 2
- 1.4 Credits . . . 2

2 Installation **3**
- 2.1 Stable release . . . 3
- 2.2 From sources . . . 3

3 Usage **5**

4 Contributing **7**
- 4.1 Types of Contributions . . . 7
- 4.2 Get Started! . . . 8
- 4.3 Pull Request Guidelines . . . 9
- 4.4 Tips . . . 9
- 4.5 Deploying . . . 9

5 Credits **11**
- 5.1 Development Lead . . . 11
- 5.2 Contributors . . . 11

6 History **13**
- 6.1 0.1.0 (2018-11-18) . . . 13

7 Indices and tables **15**

CHAPTER 1

Dask Azure Blob FileSystem

Azure Blob Storage Backend for Dask

1.1 Features

- Supports dask when your data files are stored in the cloud.
  - Import `DaskAzureBlobFileSystem`
  - Use `abfs://` as protocol prefix and you are good to do.
- For authentication, please read more on Usage.
- Support for key-value storage which is backed by azure storage. Create an instance of `AzureBlobMap`

1.2 Usage

Make the right imports:

```
from azureblobfs.dask import DaskAzureBlobFileSystem
import dask.dataframe as dd
```

then put all data files in an azure storage container say `clippy`, then you can read it:

```
data = dd.read_csv("abfs://noaa/clippy/weather*.csv")
max_by_state = data.groupby("states").max().compute()
```

you would need to set your azure account name in environment variable `AZURE_BLOB_ACCOUNT_NAME` (which in our above example is `noaa`) and the account key in `AZURE_BLOB_ACCOUNT_KEY`.

If you don't want to use account key and instead want to use SAS, set it in the environment variable `AZURE_BLOB_SAS_TOKEN` along with the connection string in the environment variable `AZURE_BLOB_CONNECTION_STRING`.

1.3 Installation

Just:

```
pip install dask-azureblobfs
```

or get the development version if you love to live dangerously:

```
pip install git+https://github.com/manish/dask-azureblobfs@master#egg=dask-azureblobfs
```

1.4 Credits

This package was created with Cookiecutter and the audreyr/cookiecutter-pypackage project template.

CHAPTER 2

Installation

2.1 Stable release

To install Dask Azure Blob FileSystem, run this command in your terminal:

```
$ pip install dask-azureblobfs
```

This is the preferred method to install Dask Azure Blob FileSystem, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

If you want to install a specific development version of *dask-azureblobfs* using *pipenv*, run this command in your terminal

```
$ pipenv install -e git+https://github.com/manish/dask-azureblobfs#egg=dask-
↪azureblobfs
```

This will install the package from github and update your *Pipfile* and *Pipfile.lock*

2.2 From sources

The sources for Dask Azure Blob FileSystem can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/manish/dask-azureblobfs
```

Or download the tarball:

```
$ curl  -OL https://github.com/manish/dask-azureblobfs/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

CHAPTER 3

Usage

To use Dask Azure Blob FileSystem in a project:

```
from azureblobfs.dask import DaskAzureBlobFileSystem
```

This import makes sure that dask is aware of azure blob filesystem. Next we import dask to read our data:

```
import dask.dataframe as dd
```

Then you load your data as usual:

```
data = dd.read_csv("abfs://account_name/mycontainer/weather*.csv",
    storage_options={"account_name": account_name,
        "account_key": account_key})
```

If you don't provide *account_name* or *account_key*, you would need to set them via environment variables *AZURE_BLOB_ACCOUNT_NAME* and *AZURE_BLOB_ACCOUNT_KEY* respectively. In which case your code would be much simpler:

```
data = dd.read_csv("abfs://account_name/mycontainer/weather*.csv")
```

The *account_name* in the URL is the same as *AZURE_BLOB_ACCOUNT_NAME*, so you can remove a lot more of the hardcoding:

```
data = dd.read_csv("abfs://{account_name}/mycontainer/weather*.csv"
    .format(account_name=os.environ.get("AZURE_BLOB_ACCOUNT_NAME")
```

You won't even have to hardcode *abfs://* if you want to use it from *DaskAzureBlobFileSystem.protocol*. Now our code becomes more verbose, but has even fewer hardcoding:

```
data = dd.read_csv("{protocol}://{account_name}/mycontainer/weather*.csv"
    .format(protocol=DaskAzureBlobFileSystem.protocol,
        account_name=os.environ.get("AZURE_BLOB_ACCOUNT_NAME")
```

CHAPTER 4

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

4.1 Types of Contributions

4.1.1 Report Bugs

Report bugs at https://github.com/manish/dask-azureblobfs/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

4.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

4.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

4.1.4 Write Documentation

Dask Azure Blob FileSystem could always use more documentation, whether as part of the official Dask Azure Blob FileSystem docs, in docstrings, or even on the web in blog posts, articles, and such.

4.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/manish/dask-azureblobfs/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

4.2 Get Started!

Ready to contribute? Here's how to set up *dask-azureblobfs* for local development.

1. Fork the *dask-azureblobfs* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/dask-azureblobfs.git
```

3. Install docker as we use *docker* and *pipenv* for creating a repeatable environment, this is how you set up your fork for local development:

```
$ cd dask-azureblobfs/
$ docker-compose build
```

If you get an error which looks like:

```
WARNING: The AZURE_BLOB_ACCOUNT_NAME variable is not set. Defaulting to a blank␣
↪string.
WARNING: The AZURE_BLOB_ACCOUNT_KEY variable is not set. Defaulting to a blank string.
```

then you will need to create an Azure account, create an instance of *Azure Blob Storage* and set the environment variables with the respective azure blob storage account name and the key associated with that account.

If you are not comfortable using the key, you can generate a shared access signature and set *AZURE_BLOB_SAS_TOKEN*.

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8, the tests pass and the docs can be generated successfully.:

```
$ ./drun_app make lint
$ ./drun_app make test
$ ./drun_app make docs
```

You don't need to install any package as *docker-compose build* command takes care of installing all the required packages and creating the container.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

4.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. Check https://travis-ci.org/manish/dask-azureblobfs/pull_requests and make sure that the tests pass for all supported Python versions.

4.4 Tips

To run a subset of tests:

```
$ ./drun_app pytest tests/test_AzureBlobMap.py
```

which will run all the test fixtures in that file:

```
$ ./drun_app pytest tests/test_AzureBlobMap.py::AzureBlobMapTest
```

will run only the specific fixture:

```
$ ./drun_app pytest tests/test_AzureBlobMap.py::AzureBlobMapTest::test_keys
```

will run only the specific test

4.5 Deploying

A reminder for the maintainers on how to deploy. Make sure all your changes are committed (including an entry in HISTORY.rst). Then run:

```
$ git push
$ git push --tags
```

Travis will then deploy to PyPI if tests pass.

CHAPTER 5

Credits

5.1 Development Lead

- Manish Sinha <masinha@microsoft.com>

5.2 Contributors

None yet. Why not be the first?

CHAPTER 6

History

6.1 0.1.0 (2018-11-18)

- First release on PyPI.

CHAPTER 7

Indices and tables

- genindex
- modindex
- search