DLHub_CLI Documentation

Release 0.6.0

Ryan Chard

Mar 07, 2019

Contents:

1 Quickstart **3**
- 1.1 Installation . . . 3
- 1.2 For Developers . . . 4
- 1.3 Requirements . . . 4

2 Tutorial **5**
- 2.1 Authentication . . . 5
- 2.2 Finding Servables . . . 6
- 2.3 Describing Servables . . . 7
- 2.4 Running Servables . . . 7
- 2.5 Publishing Servables . . . 7

3 FAQ **9**

4 Reference **11**
- 4.1 dlhub_cli package . . . 11

5 Indices and tables **15**

Python Module Index **17**

A Command Line Interface for DLHub

Source Code: https://github.com/DLHub-Argonne/dlhub_cli

CHAPTER 1

Quickstart

Welcome to the DLHub Command Line Interface (CLI). This CLI simplifies interacting with the DLHub service and the deployed servables.

1.1 Installation

The CLI is available on PyPI, but first make sure you have Python3.5+

```
>>> python3 --version
```

The CLI has been tested on Linux.

1.1.1 Installation using Pip

While `pip` and `pip3` can be used to install the CLI we suggest the following approach for reliable installation when many Python environments are avaialble.:

```
$ python3 -m pip install dlhub_cli

(to update a previously installed cli to a newer version, use: python3 -m pip install␣
↪-U dlhub_cli)
```

1.1.2 Installation using Conda

1. Install Conda and setup python3.6 following the instructions here:

   ```
   $ conda create --name dlhub_py36 python=3.6
   $ source activate dlhub_py36
   ```

2. Install the CLI:

```
$ python3 -m pip install dlhub_cli

(to update a previously installed the cli to a newer version, use: python3 -m pip␣
↪install -U dlhub_cli)
```

1.2 For Developers

1. Download the CLI:

```
$ git clone https://github.com/DLHub-Argonne/dlhub_cli
```

2. Install:

```
$ cd dlhub_cli
$ python3 setup.py install
```

3. Use the CLI:

```
$ dlhub
```

1.3 Requirements

DLHub_CLI requires the following:

- Python 3.5+

For building documentation:

- nbsphinx
- sphinx
- sphinx_rtd_theme

CHAPTER 2

Tutorial

The DLHub CLI provides an easy-to-use interface for interacting with the DLHub serving to publish, find, and run servables. This tutorial showcases many of the functions available through the CLI. The CLI thinly wraps the DLHub SDK. Further documentation on the SDK can be found here. Once installed, the CLI can be accessed via the command 'dlhub':

```
$ dlhub

Usage: dlhub [OPTIONS] COMMAND [ARGS]...

  CLI Client to the DLHub API

Options:
  -h, --help     Show this message and exit.
  -v, --version  Show the version and exit.

Commands:
  describe   Get the description of a servable
  init       Initialize a DLHub servable.
  login      Log into Globus to get credentials for the DLHub CLI
  logout     Logout of the DLHub CLI
  methods    Print method information
  publish    Publish a servable to DLHub.
  run        Invoke a servable
  search     Search the servable index
  servables  List the available servables.
  status     Check the status of a DLHub task.
  whoami     Get the username of logged in user
```

2.1 Authentication

Before using the DLHub CLI to publish, find, or use models you must login. DLHub is underpinned by Globus Auth and uses a Globus Native App login flow to authenticate users. Using various CLI commands (e.g., run and publish)

will initiate a DLHubClient and automatically start a login flow if you are not already logged in. Once logged in the CLI will create a ~/.globus.cfg file to store your tokens. The login flow will first check whether valid credentials exist in this file and will start a flow if they cannot be used.

2.1.1 Logging In

The login flow will present a URL for you to visit. This URL will ask you to login with Globus or a compatible identitiy provider. Once logged in the authentication flow will present the requested scopes for the DLHub service. Agreeing to these scopes will then generate a temporary token for you to copy and paste into the terminal.

Logging in can also be explicitly invoked with the 'login' command:

```
$ dlhub login --force
```

The `--force` flag will initiate a login flow even when the user is already logged in.

2.1.2 Logging Out

The logout process is initiated using the logout command. This will invalidate and remove the credentials stored in the `~/.dlhub/credentials/` directory and will force subsequent commands to initiate a new login flow.:

```
$ dlhub logout
```

2.2 Finding Servables

The CLI can be used to list the available servables accessible to the user. This command will provide a list of servables that are currently available in DLHub:

```
$ dlhub servables
```

You can also search the index of DLHub servables with the 'search' command. The 'search' command supports tags for common options, such as the owner:

```
$ dlhub search --owner blaiszik_globusid

Model Name       Owner              Publication Date    Type
---------------  -----------------  ------------------  -----------
cherukara_phase  blaiszik_globusid  2019-02-19 15:21    Keras Model
```

Call `dlhub search --help` for a full listing of search tags.

You can also provide a full query string using the DLHub query syntax:

```
$ dlhub search dlhub.owner:blaiszik_globusid AND servable.type:"Keras Model"

Model Name       Owner              Publication Date    Type
---------------  -----------------  ------------------  -----------
cherukara_phase  blaiszik_globusid  2019-02-19 15:21    Keras Model
```

2.3 Describing Servables

The 'describe' command queries the service for additional information about a servable. This information includes details on how to invoke the servable, required inputs, and expected outputs. The describe command requires the owner name and user name of the servable:

```
$ dlhub describe dlhub.test_gmail 1d_norm
```

Use the 'methods' to return only information about the methods implemented by a servable:

```
$ dlhub methods dlhub.test_gmail 1d_norm

run:
  input:
    description: Array to be normed
    shape:
    - None
    type: ndarray
  output:
    description: Norm of the array
    type: number
```

2.4 Running Servables

Servables can be invoked through the CLI using the run command. The run command accepts flags to specify the servable and input parameters. The servable flag requires the identifier of the servable. Input parameters should be correctly-formatted JSON strings. The run command first attempts to json.loads() the input before using the DLHub SDK to invoke the servable. Output will be returned as well formatted JSON documents.:

```
$ dlhub run --servable 50358d8c-be7a-41bf-af76-a460223907fe --input '[{"composition":
↪"Al"}]'

Outputs:
   [
     {
       "composition": "Al",
       "composition_object":
↪"gANjcHltYXRnZW4uY29yZS5jb21wb3NpdGlvbgpDb21wb3NpdGlvbgpxACmBcQF9cQIoWA4AAABh\nbGxvd19uZWdhdGl2ZXEI
↪wAAAAAAAAWAUAAABfZGF0YXEFfXEGY3B5\nbWF0Z2VuLmNvcmUucGVyaW9kaWNfdGFibGUKRWxlbWVudApxB1gCAAAAQWxxCIVx
↪8AAA\nAAAAAHN1Yi4=\n"
     }
   ]
```

2.5 Publishing Servables

Publishing a servable can be achieved by issuing a publish command using either a GitHub repository or a local servable definition file.

2.5.1 Description Files

Publishing a servable relies on a compatible metadata document. The publication process uses the metadata document to determine which shim to use when loading and interacting the servable.

A guide for describing servables can be found in the DLHub SDK documentation.

2.5.2 Publishing a Repository

Publishing a model can also be achieved by specifying a compliant GitHub repository. The repository will need to include the `dlhub.json` file already. The publication flow relies on repo2docker to construct a container with all of the required dependencies.

An example repository can be found here: https://github.com/ryanchard/dlhub_publish_example

The publication command will return a task identifier that can subsequently be used to query the status of publication tasks.::

```
$ dlhub publish --repository https://github.com/ryanchard/dlhub_publish_example

 Task_id: ff56599e-3377-4475-9684-0afd7f563aeb
```

2.5.3 Publishing a Local Servable

Publishing a local servable requires first generating the `dlhub.json` file and storing it on your system. Once that file has been generated you can use the `--local` flag to initiate a publication for the local model. Files mentioned within the dlhub.json document will be packaged into a temporary zip file then transmitted to the DLHub service using HTTP:

```
$ dlhub publish --local
```

2.5.4 Checking Publication Status

The status of a publication task can be queried using the status command. The status command requires the task id and will return a JSON status document.:

```
$ dlhub status --task ff56599e-3377-4475-9684-0afd7f563aeb

 ff56599e-3377-4475-9684-0afd7f563aeb: {'status': 'COMPLETE'}
```

CHAPTER 3

FAQ

CHAPTER 4

Reference

4.1 dlhub_cli package

4.1.1 Subpackages

dlhub_cli.commands package

Module contents

dlhub_cli.parsing package

Submodules

dlhub_cli.parsing.click_wrappers module

Mostly copied from globus-cli and globus-search-cli We want the niceties of parsing improvements worked out in that project.

class dlhub_cli.parsing.click_wrappers.**CommandState**
Bases: object

class dlhub_cli.parsing.click_wrappers.**DLHubCommandGroup**(*name=None*, *commands=None*, ***attrs*)
Bases: click.core.Group

This is a click.Group with any customizations which we deem necessary *everywhere*. In particular, at present it provides a better form of handling for no_args_is_help. If that flag is set, helptext will be triggered not only off of cases where there are no arguments at all, but also cases where there are options, but no subcommand (positional arg) is given.

invoke(*ctx*)
Given a context, this invokes the attached callback (if it exists) in the right way.

class dlhub_cli.parsing.click_wrappers.**HiddenOption**(*param_decls=None*, *show_default=False*, *prompt=False*, *confirmation_prompt=False*, *hide_input=False*, *is_flag=None*, *flag_value=None*, *multiple=False*, *count=False*, *allow_from_autoenv=True*, *type=None*, *help=None*, ***attrs*)

Bases: click.core.Option

HiddenOption – absent from Help text. Supported in latest and greatest version of Click, but not old versions, so use generic 'cls=HiddenOption' to get the desired behavior.

get_help_record(*ctx*)

Has "None" as its help record. All that's needed.

Parameters **ctx** –

Returns

dlhub_cli.parsing.click_wrappers.**common_options**(*f*)

Global/shared options decorator.

Parameters **f** –

Returns

dlhub_cli.parsing.click_wrappers.**debug_option**(*f*)

Enable debugging for commands.

Parameters **f** –

Returns

dlhub_cli.parsing.click_wrappers.**dlhub_cmd**(**args*, ***kwargs*)

Wrapper over click.command which sets common opts

Parameters

- **args** –
- **kwargs** –

Returns

dlhub_cli.parsing.click_wrappers.**dlhub_group**(**args*, ***kwargs*)

Wrapper over click.group which sets GlobusCommandGroup as the Class

Parameters

- **args** –
- **kwargs** –

Returns

dlhub_cli.parsing.click_wrappers.**index_argument**(*f*)

Click indexing for arguments.

Parameters **f** –

Returns

dlhub_cli.parsing.click_wrappers.**setup_logging**(*level='DEBUG'*)
Configure the logger.

Parameters **level** –

Returns

dlhub_cli.parsing.main module

dlhub_cli.parsing.main.**main_func**(*f*)
Wrap root command func in common opts and make it a command group

Parameters **f** –

Returns

Module contents

dlhub_cli.parsing.**index_argument**(*f*)
Click indexing for arguments.

Parameters **f** –

Returns

dlhub_cli.parsing.**dlhub_group**(**args*, ***kwargs*)
Wrapper over click.group which sets GlobusCommandGroup as the Class

Parameters

- **args** –
- **kwargs** –

Returns

dlhub_cli.parsing.**dlhub_cmd**(**args*, ***kwargs*)
Wrapper over click.command which sets common opts

Parameters

- **args** –
- **kwargs** –

Returns

dlhub_cli.parsing.**main_func**(*f*)
Wrap root command func in common opts and make it a command group

Parameters **f** –

Returns

4.1.2 Submodules

4.1.3 dlhub_cli.config module

dlhub_cli.config.**get_dlhub_client**(*force=False*)
Get the DLHub client

Returns DLHubClient with the proper credentials

Return type DLHubClient

4.1.4 dlhub_cli.main module

4.1.5 dlhub_cli.printing module

Functions related to cleaner printing of the outputs

dlhub_cli.printing.**format_output**(*dataobject*)
Use safe print to make sure jobs are correctly printed.

Parameters **dataobject** (*string*) – String to print.

dlhub_cli.printing.**safeprint**(*s*)
Catch print errors.

Parameters **s** (*string*) – String to print.

4.1.6 dlhub_cli.version module

4.1.7 Module contents

CHAPTER 5

Indices and tables

- genindex
- modindex
- search

Python Module Index

d

dlhub_cli, 14
dlhub_cli.commands, 11
dlhub_cli.config, 13
dlhub_cli.main, 14
dlhub_cli.parsing, 13
dlhub_cli.parsing.click_wrappers, 11
dlhub_cli.parsing.main, 13
dlhub_cli.printing, 14
dlhub_cli.version, 14

Index

C

CommandState (class in dlhub_cli.parsing.click_wrappers), 11
common_options() (in module dlhub_cli.parsing.click_wrappers), 12

D

debug_option() (in module dlhub_cli.parsing.click_wrappers), 12
dlhub_cli (module), 14
dlhub_cli.commands (module), 11
dlhub_cli.config (module), 13
dlhub_cli.main (module), 14
dlhub_cli.parsing (module), 13
dlhub_cli.parsing.click_wrappers (module), 11
dlhub_cli.parsing.main (module), 13
dlhub_cli.printing (module), 14
dlhub_cli.version (module), 14
dlhub_cmd() (in module dlhub_cli.parsing), 13
dlhub_cmd() (in module dlhub_cli.parsing.click_wrappers), 12
dlhub_group() (in module dlhub_cli.parsing), 13
dlhub_group() (in module dlhub_cli.parsing.click_wrappers), 12
DLHubCommandGroup (class in dlhub_cli.parsing.click_wrappers), 11

F

format_output() (in module dlhub_cli.printing), 14

G

get_dlhub_client() (in module dlhub_cli.config), 13
get_help_record() (dlhub_cli.parsing.click_wrappers.HiddenOption method), 12

H

HiddenOption (class in dlhub_cli.parsing.click_wrappers), 11

I

index_argument() (in module dlhub_cli.parsing), 13
index_argument() (in module dlhub_cli.parsing.click_wrappers), 12
invoke() (dlhub_cli.parsing.click_wrappers.DLHubCommandGroup method), 11

M

main_func() (in module dlhub_cli.parsing), 13
main_func() (in module dlhub_cli.parsing.main), 13

S

safeprint() (in module dlhub_cli.printing), 14
setup_logging() (in module dlhub_cli.parsing.click_wrappers), 12