nebulizer Documentation

Release 0.7.0

Peter Briggs

May 17, 2021

Contents

Nebulizer is a Python utility which provides a high-level interactive command-line interface to remotely administer Galaxy servers.

Nebulizer was developed as a tool to help "part-time" Galaxy admins perform day-to-day administrative tasks across multiple Galaxy instances, enabling various admin operations to be executed via the command line as an alternative to using the Galaxy web interface.

Nebulizer is built on top of the Bioblend library and offers an interface and range of functionality that complements the lower-level interfaces offered by Ephemeris and Parsec.

Contents:

nebulizer

Command-line utilities to help with managing users, data libraries and tools in a Galaxy instance, using the Galaxy API via the Bioblend library.

- Free software: Academic Free License version 3.0
- Documentation: https://nebulizer.readthedocs.io
- Code: https://github.com/pjbriggs/nebulizer

Note: Nebulizer is still a work in progress.

Please exercise caution when attempting irreversible operations, especially against production Galaxy instances (for example when creating users or data libraries).

CHAPTER 1

Quick Start

This quick start gives some examples of using `nebulizer` commands to perform remote administration tasks on a Galaxy instance from the command line.

1.1 Getting Nebulizer

It is recommended to install Nebulizer via `pip` in a virtualenv, for example:

```
% virtualenv .venv
% source .venv/bin/activate
% pip install nebulizer
```

This will provide an executable called `nebulizer` with a number of subcommands for performing different tasks remotely on Galaxy instances.

1.2 Nebulizer Basics

Generally Nebulizer commands take the form:

```
nebulizer COMMAND GALAXY [OPTIONS]
```

To interact remotely with a Galaxy instance using Nebulizer requires at minimum the URL of the instance and then either an API key or a user login name.

For example to list the data libraries available on Galaxy Main:

```
nebulizer -k 9b376af2250818d14949b3c list_libraries https://usegalaxy.org
```

or

```
nebulizer -u USER@DOMAIN list_libraries https://usegalaxy.org
```

In this second case Nebulizer will prompt for the Galaxy password to authenticate the user login, unless it's supplied via the `-P` option.

To store the Galaxy URL-API key pair against an alias `main`, to avoid needing full authentication details each time:

```
nebulizer add_key main https://usegalaxy.org 9b376af2250818d14949b3c
```

or alternatively get Nebulizer to fetch the API key itself by supplying the user login:

```
nebulizer -u USER@DOMAIN add_key main https://usegalaxy.org
```

More information on managing API keys in Nebulizer can found here.

The stored alias is then used in subsequent commands, for example to list the data libraries again it is now sufficient to do just:

```
nebulizer list_libraries main
```

The following sections contain examples of how Nebulizer might be used to perform various administrive tasks.

Nebulizer provides subcommands to perform various administrive tasks:

Managing users:

- `list_users`
- `create_user`
- `create_batch_users`
- `create_users_from_file`
- `delete_user`

Managing data libraries:

- `list_libraries`
- `create_library`
- `create_library_folder`
- `add_library_datasets`

Managing tools:

- `list_tools`
- `list_tool_panel`
- `install_tool`
- `update_tool`
- `uninstall_tool`
- `search_toolshed`

Managing quotas:

- `quotas`
- `quota_add`
- `quota_mod`

- `quota_del`

Querying Galaxy instances:

- `ping` (check if a Galaxy instance is alive)
- `config` (fetch configuration for a Galaxy instance)

See the tutorial for a walkthrough some of these commands.

1.3 Contributors

Nebulizer has been developed by Peter Briggs @pjbriggs, with contributions from:

- Hugo van Kemenade (@hugovk)

Thanks to Peter van Heuseden (@pvanheus) for porting Nebulizer into Bioconda.

CHAPTER 2

Installation

2.1 Virtualenv and pip

For a traditional installation of Nebulizer, first set up a Python virtualenv (this example creates a new one in `.venv`) and then install with `pip`:

```
$ virtualenv .venv;
$ source .venv/bin/activate
$ pip install nebulizer
```

When installed this way, Nebulizer can be upgraded as follows:

```
$ . .venv/bin/activate
$ pip install -U nebulizer
```

To install or update to the latest development branch of Nebulizer with `pip`, use the following `pip install` idiom instead:

```
$ pip install -U git+git://github.com/pjbriggs/nebulizer.git@devel
```

Nebulizer should work with recent Python 3 versions.

2.2 Conda

Nebulizer can be installed using Conda (most easily obtained via the Miniconda Python distribution):

```
$ conda config --add channels bioconda
$ conda install nebulizer
```

Note that the version available via `bioconda` may lag the most recent version available via `pip`.

CHAPTER 3

Configuration

Nebulizer doesn't require any additional configuration after installation: it is possible to supply either a Galaxy API key or a Galaxy username/password combination on the command line (alongside the URL of the Galaxy instance) each time a command is executed.

However this is both laborious and potentially insecure. Nebulizer can store Galaxy URL-API key pairs against aliases locally to make this easier, using the `add_key` command.

The simplest way to create a new alias is:

```
nebulizer -u EMAIL add_key ALIAS GALAXY_URL
```

Nebulizer will prompt for the password for the account and will then fetch and store the API key automatically.

Note: For example: to add an alias `main` for Galaxy main at https://usegalaxy:

```
nebulizer -u peter.briggs@manchester.ac.uk add_key main https://usegalaxy.org
```

Alternatively you can supply an API directly on the command line:

```
nebulizer add_key ALIAS GALAXY_URL API_KEY
```

Note: For example:

```
nebulizer add_key main https://usegalaxy.org 9b376af2250818d14949b3c
```

The stored alias can then be used instead of supplying the full Galaxy URL with an email/password combination or an API key; for example to get configuration information from Galaxy main using the `config` command:

```
nebulizer config main
```

See *Storing and managing Galaxy API keys* for more information on managing stored aliases.

CHAPTER 4

Managing Users

4.1 Querying user information

`list_users` displays user emails and names in a Galaxy instance:

```
nebulizer list_users GALAXY
```

- `-l`: returns extended information for each user (status, whether they are an admin user, disk usage and quota size and usage).
- `--status`: filter list on user status, which can be one of `active` (the default), `deleted` (only list deleted users which haven't been purged) or `all` (list all active, deleted, and purged users).
- `--sort`: specify one or more fields to sort the output on; valid fields are `email`, `disk_usage`, `quota`, `quota_usage`. Multiple fields should be separated by commas (e.g. `--sort=quota,disk_usage`).
- `--name`: filter list on user email (can include glob-style wildcards e.g. `--name="*bloggs*"`).

4.2 Creating and deleting individual users

`create_user` makes a new user account:

```
nebulizer create_user GALAXY USER@DOMAIN [ PUBLIC_NAME ]
```

If `PUBLIC_NAME` is not supplied then one will be generated automatically from the email address.

- `-c`: check if the account already exists
- `-p`: supply password for the new account (otherwise Nebulizer will prompt for a password)

`delete_user` removes a user account:

```
nebulizer delete_user GALAXY USER@DOMAIN
```

- `--purge`: purge the account as well as deleting (or purge an unpurged account which has been previously deleted)

Note: Purging a user account overwrites the email and username for that account with random strings; it also marks the datasets and histories associated with that account as deleted. These data can then be removed from disk by running Galaxy's clean-up scripts.

For information on the clean-up scripts see the Galaxy documentation at https://galaxyproject.org/admin/config/performance/purge-histories-and-datasets/

4.3 Creating batches of users from a template name

`create_batch_users` makes a set of user accounts based on a template name:

```
nebulizer create_batch_users GALAXY TEMPLATE [START] END
```

The template email address should include a `#` symbol which acts as as a placeholder for an integer index, e.g.

```
user#@example.org
```

The integer indices are generate from the range `START...END` (if `START` is not supplied then it is set to 1).

For example:

```
nebulizer create_batch_users user#@example.org 1 5
```

creates accounts:

```
user1@galaxy.org
user2@galaxy.org
user3@galaxy.org
user4@galaxy.org
user5@galaxy.org
```

- `-c`: check if the accounts already exist
- `-p`: supply password for the new accounts (otherwise Nebulizer will prompt for a password); the same password will be applied to all the new accounts

4.4 Creating user accounts from a file

`create_users_from_file` makes a set of user accounts using data from a file:

```
nebulizer create_users_from_file GALAXY USERS_FILE
```

`USER_FILE` is a tab-delimited file with the following fields on each line:

```
email|password|public_name
```

defining a new account.

Note: Optionally `public_name` can be left out and will then be generated automatically.

- `-c`: check if the accounts already exist

CHAPTER 5

Managing Data Libraries

5.1 Querying data libraries

`list_libraries` displays information about the data libraries and their contents in a Galaxy instance:

```
nebulizer list_libraries GALAXY [ LIBRARY[/PATH...] ]
```

By default all the libraries in the instance are listed; if the name of a data library (`LIBRARY`) is included then the datasets and folders in that library are listed. If a `PATH` (names of subfolders separated by `/`) is supplied then the contents of the matching folder are displayed.

For example: listing the data libraries in Galaxy main:

```
nebulizer list_libraries https://usegalaxy.org
```

List the contents of the "Tutorials" data library:

```
nebulizer list_libraries https://usegalaxy.org Tutorials
```

List the contents of the "ChIP_seq (Reb1)" folder in the "Tutorials" data library:

```
nebulizer list_libraries https://usegalaxy.org "Tutorial/ChIP_seq (Reb1)"
```

- `-l`: list extended information about libraries, folders and datasets

5.2 Creating and populating data libraries

`create_library` creates a new data library:

```
nebulizer create_library GALAXY LIBRARY_NAME [--description DESCRIPTION] [--synopsis␣
↪SYNOPSIS]
```

`create_library_folder` creates a new folder in an existing data library or folder:

```
nebulizer create_library_folder GALAXY LIBRARY_NAME/[PATH]
```

For example: create a data library called `NGS data` and a folder `Run 21` within it:

```
nebulizer create_library GALAXY "NGS data 2020" \
  --description="Sequencing data analysed in 2020"
nebulizer create_library_folder GALAXY "NGS data 2020/Run 21"
```

`add_library_datasets` uploads datasets to a data library:

```
nebulizer add_library_datasets GALAXY LIBRARY_NAME/PATH FILE [FILE...]
```

- `--file-type`: specify the Galaxy data type to assign to datasets
- `--dbkey`: specify the dbkey for the datasets
- `--server`: upload files from the filesystem of the server that Galaxy is running on (default is to upload from the local filesystem)
- `--link`: create symlinks to the files on the server (if `--server` is also specified)

For example, add Fastq files to a data library folder:

```
nebulizer add_library_datasets galaxy "NGS data/Run 21" ~/Sample1_R*.fq \
  --file-type=fastqsanger --dbkey=hg38
```

CHAPTER 6

Managing Tools

6.1 Querying tool information

By default, `list_tools` displays information on all the tool repositories that are installed in a Galaxy instance:

```
nebulizer list_tools GALAXY
```

For each installed repository the details include: repository name, toolshed, owner, revision id and changeset, and installation status.

Repository details are also preceeded by a single-character 'status' indicator:

- `D` = deprecated;
- `^` = newer revision is also installed;
- `u` = update (newer revision) is available but not installed;
- `U` = upgrade available (new revision with a change of tool version) but not installed;
- `*` = latest revision installed

`list_tools` supports a number of options to modify its behaviour, including:

- `--updateable`: only list tool repositories that have uninstalled available updates or upgrades
- `--built-ins`: include details of the "built-in" tools within the Galaxy instance (i.e. those not installed from a toolshed)

An alternative 'tool-centric' view of the tools in a Galaxy instance can be obtained using the `--mode=tools` option.

Note: This is a new version of the `list_tools` which replaces the old `list_installed_tools` command (which is no longer available). The `--mode=tools` option replicates the output from the old `list_tools` command.

`list_tool_panel` displays information on the tool panel sections in a Galaxy instance:

```
nebulizer list_tool_panel GALAXY
```

New tool panel sections can be created when installing tool repositories.

6.2 Install, update and remove tools

`install_tool` installs tools from a Galaxy toolshed (by default the main Galaxy toolshed at https://toolshed.g2.bx.psu.edu):

```
nebulizer install_tool GALAXY TOOL_REPOSITORY
```

The tool repository can be specified as:

- `[TOOLSHED] OWNER/TOOLNAME [ REVISION ]` e.g. `devteam/fastqc` (defaults to most recent revision from the main toolshed), or `devteam/fastqc e7b2202befea` (specifies revision `e7b2202befea`)
- toolshed URL with or without a revision (e.g. `https://toolshed.g2.bx.psu.edu/view/devteam/fastqc/e7b2202befea`)
- `--tool-panel-section`: specify a tool panel section to install the tools from the repository into (otherwise tools appear at the "top level" of the tool panel)

For example: to install the most recent FastQC tool from the main Galaxy toolshed under the `NGS: QC and manipulation` section of the tool panel:

```
nebulizer install_tool GALAXY devteam/fastqc \
   --tool-panel-section="NGS: QC and manipulation"
```

To install a specific revision of the Trimmer tool from the test toolshed:

```
nebulizer install_tool GALAXY \
   https://testtoolshed.g2.bx.psu.edu/view/devteam/trimmer/dec27ea206c3 \
   --tool-panel-section="Test tools"
```

`update_tool` installs the latest revision of a previously installed tool repository, if a new version is available:

```
nebulizer update_tool GALAXY TOOL_REPOSITORY
```

The tool repository is specified as with `install_tool` except that a revision cannot be included. For example:

```
nebulizer update_tool GALAXY devteam/fastqc
```

It is also possible to include glob-style wildcards in the tool repository name and/or owner e.g. `devteam/*` or `bgruening/deeptools_*`. To request update of all tools:

```
nebulizer update_tool GALAXY '*/*'
```

Note: `update_tool` doesn't uninstall the older versions of the tools that are updated.

Warning: By default checks on the availability of updates for tools performed by the `list_tools` and `update_tool` commands are done using information cached by the Galaxy instance in question. As a result these commands may not always indicate when updates are available.

To force these commands to check the installed revisions against those in the toolshed, add the `--check-toolshed` option. Note however that this can impose a significant overhead which can make the commands much slower.

`uninstall_tool` removes a previously installed tool:

```
nebulizer uninstall_tool GALAXY TOOL_REPOSITORY
```

The tool repository is specified as with `install_tool`, for example to uninstall and deactivate a specific revision of a tool:

```
nebulizer uninstall_tool GALAXY devteam/fastqc/e7b2202befea
```

To uninstall all installed revisions of a tool and remove from disk:

```
nebulizer uninstall_tool localhost devteam/fastqc/* \
   --remove_from_disk
```

6.3 Searching for tool repositories on a Toolshed

`search_toolshed` searches for tools on a toolshed:

```
nebulizer search_toolshed QUERY
```

`QUERY` can include glob-style wildcards. For example, to search the main toolshed for Deeptools related tools:

```
nebulizer search_toolshed "deeptools_*"
```

- `--toolshed`: specify the URL of the toolshed to search.

6.4 Bulk tool repository management

`install_tool --file` installs the tool repositories listed in a tab-delimited file into a Galaxy instance:

```
nebulizer install_tool GALAXY TOOLS_FILE
```

`TOOLS_FILE` must be a tab-delimited list of repositories, one repository per line in the format:

```
TOOLSHED|OWNER|REPOSITORY|REVISON|SECTION
```

For example:

```
toolshed.g2.bx.psu.edu         devteam bowtie_wrappers 9ca609a2a421    NGS: Mapping
```

`list_tools --mode=export` can generate a list of tool repositories already installed in a Galaxy instance in this format, e.g.:

```
nebulizer list_tools GALAXY --mode=export
```

By combining these two commands it is possible to 'clone' the installed tools from one Galaxy instance into another.

For example to replicate the tools installed on the 'Palfinder' instance into a local Galaxy:

```
nebulizer list_tools https://palfinder.ls.manchester.ac.uk --mode=export > palfinder.
↪tsv
nebulizer install_tool http://127.0.0.1 --file palfinder.tsv
```

Warning: Bulk installation of tools in this manner should be used with caution, especially when installing into a Galaxy instance which already has installed tools.

CHAPTER 7

Managing Quotas

7.1 Querying quotas

`quotas` lists the quotas defined in a Galaxy instance:

```
nebulizer quotas GALAXY
```

- `-l`: returns extended information for each quota, including the associated users and groups.
- `--status`: filter list on quota status, which can be one of `active` (the default), `deleted` (only list deleted quotas) or `all` (list all active and deleted quotas).
- `--name`: filter list on quota name (can include glob-style wildcards e.g. `--name="*NGS*"`).

7.2 Creating and deleting quotas

`quota_add` defines a new quota:

```
nebulizer quota_add GALAXY QUOTA_NAME SIZE
```

`SIZE` can either be an amount (e.g. `10GB`, `0.2 T`) or an amount preceeded by an operation (one of +, − or =, e.g. `=300gb`, `+100G`). If an operation isn't specified then = is assumed.

- `-d`/`--description`: set the description for the quota (defaults to the quota name).
- `--default_for`: set the quota as the the default for either 'registered' or 'unregistered' users.

Users and groups can be associated with the new quota using the `-u` and `-g` options:

- `-u`/`--users`: associate one or more users with the quota, as a comma-separated list of email addresses.
- `-g`/`--groups`: associate one or more groups with the quota, as a comma-separated list of group names.

`quota_del` deletes an existing quota:

```
nebulizer quota_del GALAXY QUOTA_NAME
```

Note: A deleted quota can be restored using the `--undelete` option of the `quota_mod` command.

7.3 Modifying quota definitions

`quota_mod` updates a quota definition:

```
nebulizer quota_mod GALAXY QUOTA_NAME ...
```

Options allow various quota properties to be modified:

- `-n`/`--name`: sets a new name for the quota.
- `-d`/`--description`: sets a new description.
- `-q`/`--quota-size`: updates the size of the quota, and how it is applied.
- `--default_for`: set the quota as the the default for either 'registered' or 'unregistered' users.

Users and groups can be associated with the following options:

- `-a`/`--add-users`: associate one or more users with the quota, as a comma-separated list of email addresses.
- `-r`/`--remove-users`: disassociate one or more users from the quota, as a comma-separated list of email addresses.
- `-A`/`--add-groups`: associate one or more groups with the quota, as a comma-separated list of group names.
- `-R`/`--remove-groups`: disassociate one or more groups from the quota, as a comma-separated list of group names.

Previously deleted quotas can be restored using the `-u`/`--undelete` option.

CHAPTER 8

Querying Galaxy instances

8.1 Checking status and configuration of a Galaxy server

'Ping' a Galaxy instance to check it's alive and responding to requests:

```
nebulizer ping GALAXY
```

Get information about an instance's configuration using

:: nebulizer config GALAXY

CHAPTER 9

Storing and managing Galaxy API keys

The majority of Nebulizer's commands require valid credentials in order to interact with a Galaxy instance. These can be supplied explicitly on the command line each time:

```
nebulizer -u USER@DOMAIN [ -P PASSWORD ] ...command...
nebulizer -k API_KEY ...command...
```

Alternatively it is possible to store Galaxy URL-API key pairs in a file called `.nebulizer` located in the user's home directory, with each pair being associated with an alias.

The file is created automatically the first time an alias is created using `add_key` command, and consists of tab-delimited lines with three fields:

```
alias|Galaxy_URL|API_key
```

for example:

```
demo    http://127.0.0.1:8080    4551fbf7cd8b1bc59db....
```

This file can be manually edited using a text editor such as `vi`; however Nebulizer also provides a set of commands for querying and modifying the file contents.

`list_keys` shows the aliases with their associated Galaxy URLs:

```
nebulizer list_keys
```

Note: By default the API keys are not shown by `list_keys`; use the `--show-api-keys` option to include them.

Note: Use the `whomai` command to find out which user is associated with an alias:

```
nebulizer whoami ALIAS
```

`add_key` will store a Galaxy-API key combination under a new alias. If the API key is known then the general form of the command is:

```
nebulizer add_key ALIAS GALAXY_URL API_KEY
```

However it is usually easier to get Nebulizer to fetch the key automatically, by supplying a Galaxy username (email):

```
nebulizer -u USER@DOMAIN add_key ALIAS GALAXY_URL
```

Multiple Galaxy URL-key pairs can be stored; only the associated aliases need to be unique.

`update_key` will update the details stored for an existing alias:

```
nebulizer update_key ALIAS --new-url GALAXY_URL
nebulizer update_key ALIAS --new-api-key API_KEY
nebulizer -u USER@DOMAIN update_key ALIAS --fetch-api-key
```

`remove_key` deletes an existing ALIAS and associated credentials:

```
nebulizer remove_key ALIAS
```

CHAPTER 10

General options

Nebulizer supports a number of general options for managing how connections are made to Galaxy servers, how to authenticate, and what information to output.

10.1 Authenticating without stored credentials

Use `--api_key` (`-k`) to explicitly specify the API key to use for authentication on the command line, e.g.

```
nebulizer -k API_KEY COMMAND...
```

Use the `--username` (`-u`) option to authenticate using the normal Galaxy login credentials (i.e. email and password) instead of the API key, e.g.

```
nebulizer -u USER@DOMAIN COMMAND...
```

You will be prompted to enter the password, but you can also use the `--galaxy_password` (`-P`) option to specify it explicitly on the command line.

10.2 Controlling warnings and debugging output

`--suppress-warnings` (`-q`) suppresses warning messages from Nebulizer; conversely debugging output can be enabled using the `--debug` option.

10.3 Handling SSL certificate verification failures

Nebulizer commands will fail for Galaxy instances which are served over `https` protocol without a valid SSL certificate, reporting an error like:

```
[SSL: CERTIFICATE_VERIFY_FAILED] certificate verify failed (_ssl.c:590), 0 attempts␣
↪left: None
```

In this case adding the `--no-verify` (`-n`) option turns off the certificate verification and should enable a connection to be made.

CHAPTER 11

Nebulizer Tutorial

This tutorial offers a short hands-on introduction to the main functionality offered by Nebulizer.

11.1 0. Preparation: making a temporary local Galaxy instance

Before starting it is recommended to set up a temporary local Galaxy instance to run the tutorial on, rather than trying Nebulizer out on an existing installation.

The following commands should fetch and start up the local instance:

```
# Get the Galaxy source and put into a new directory
# called 'galaxy_for_demo'
git clone -b release_21.01 https://github.com/galaxyproject/galaxy.git galaxy_for_demo

# Move into the source directory
cd galaxy_for_demo

# Make a configuration file
cp config/galaxy.yml.sample config/galaxy.yml
```

You will need to edit the configuration file `config/galaxy.yml` using a text editor (e.g. `vi`, `nano`, `gedit` etc) to specify an admin user, by changing the line:

```
#admin_users: null
```

to e.g.

```
admin_users: admin@localhost.org
```

or whatever email address you want. You will also need to change the line:

```
#allow_user_deletion: false
```

to

```
allow_user_deletion: true
```

Then start Galaxy running:

```
sh run.sh
```

Note: This step can take some time when it's first run; however once Galaxy has installed its dependencies and performed its initial configuration then subsequent startups of the demo Galaxy should be much quicker.

Use `ctrl-C` to stop Galaxy running, and repeat this command to start it up again.

Once Galaxy is running you can connect to it by pointing a web browser to http://127.0.0.1:8080/

Finally you will need to register an account in Galaxy with the same email address as the one used for `admin_users` (`admin@localhost.org` in this example), using the web browser.

Now you're ready to run through the tutorial.

Note: Full instructions for setting up a local Galaxy instance can be found at https://galaxyproject.org/admin/get-galaxy/

11.2 1. Install Nebulizer

Installing Nebulizer is best done in a Python virtual environment - for example:

```
virtualenv -p python3 venv.nebulizer
. venv.nebulizer/bin/activate
pip install nebulizer
```

Note: It is recommended to do this in a different directory to the local Galaxy instance from the previous section.

Once this is done you should have access to the `nebulizer` utility.

To list the available commands:

```
nebulizer --help
```

You use the ping command to check if a Galaxy instance is active. For example to check the main Galaxy server:

```
nebulizer ping https://usegalaxy.org
```

which should produce output like:

```
https://usegalaxy.org: status = ok time = 574.041 (ms)
https://usegalaxy.org: status = ok time = 587.883 (ms)
https://usegalaxy.org: status = ok time = 549.526 (ms)
...
```

Note: Do `control-C` to terminate the "ping".

You can also use the config command to query the details of a Galaxy instance's configuration. For example to query the local Galaxy:

```
nebulizer config http://127.0.0.1:8080
```

11.3 2. Set up aliases for Galaxy API keys

Most Nebulizer commands require you to interact with a Galaxy instance using an account on that instance.

For these commands you can authorise access each time by specifying your registered email address and password or Galaxy API key on the command line. For example:

```
nebulizer -u admin@localhost.org whoami http://127.0.0.1:8080
```

Note: `-u` will prompt you to enter the password for the account before performing the action; `-k` can be used to specify the API key.

Warning: This won't work if you didn't make an account for `admin@localhost.org` when preparing the local Galaxy in the previous step!

This is quite laborious when executing several commands, so Nebulizer allows you to associate Galaxy instances and their API keys with aliases; these are used as shortcuts when running the commands.

To see the aliases and associated Galaxy servers:

```
nebulizer list_keys
```

Note: If you've never used Nebulizer before then nothing will be listed.

To set up a new alias called `local` and associate it with the admin account in our local Galaxy, we can do:

```
nebulizer -u admin@localhost.org add_key local http://127.0.0.1:8080
```

This will prompt you for the password for the account and then create the alias. Once this is done you can repeat the `list_keys` command and see an entry for the local Galaxy:

```
local  http://127.0.0.1:8080
```

In subsequent commands you can use `local` rather than specifying the full Galaxy URL, and won't need to enter your email or password. For example:

```
nebulizer whoami local
```

Now we're ready to do some basic administration of our local Galaxy using Nebulizer.

Note: See *Storing and managing Galaxy API keys* for more details.

11.4 3. Listing, adding and deleting users

We can list the users in our local Galaxy with:

```
nebulizer list_users local
```

There will be just one account (the original admin account we made at the start).

We can add a new user using:

```
nebulizer create_user local ann.onymous@manchester.ac.uk
```

Note: This will prompt you for a password for the new account; use the `-p` option to set the password via the command line.

Do the `list_users` command again to see new user listed. Use the `-l` option to display additional information about each user is displayed, including status and disk usage (and quota usage, if quotas are enabled).

Batches of user accounts can be created from a "template" name using the `create_batch_users` command; this can be useful for example when setting up Galaxy instances for teaching:

```
nebulizer create_batch_users local user#@bcc2020.org 5
```

Note: This will prompt you for a password which will be assigned to all the new accounts.

Use the `list_users` command to see the new accounts:

```
user1@bcc2020.org               user1
user2@bcc2020.org               user2
...
user5@bcc2020.org               user5
```

Accounts can also be deleted:

```
nebulizer delete_user local user5@bcc2020.org
```

The user will no longer be listed by `list_users`.

Warning: If the deletion fails then check that the Galaxy configuration has `allow_user_deletion` set to `true`.

Note: See *Managing Users* for more details.

11.5 4. Creating and populating data libraries

We can list the data libraries in our local Galaxy instance using:

```
nebulizer list_libraries local
```

Initially our local Galaxy doesn't contain any library data; we can create a new data library using:

```
nebulizer create_library local "Example data"
```

Note: Use the `-d` and `-s` options to add description and synopsis information for the new library.

Now this will be listed by the `list_libraries` command. We can list the contents of a library by specifying its name:

```
nebulizer list_libraries local "Example data"
```

Initially the library is empty; we can create a folder within the library:

```
nebulizer create_library_folder local "Example data/Fastqs"
```

To list the contents of a library folder specify the "path" to the folder:

```
nebulizer list_libraries local "Example data/Fastqs"
```

Datasets can be added to libraries and folders from the local workstation:

```
nebulizer add_library_datasets local "Example data/Fastqs" Illumina_SG_R* --dbkey=hg38
```

Note: The example Fastq files can be found here:

- `Illumina_SG_R1.fastq`
- `Illumina_SG_R2.fastq`

When listing the contents of libraries and folders, additional information is reported by specifying the `-l` option:

```
nebulizer list_libraries local "Example data/Fastqs" -l
```

Note: See *Managing Data Libraries* for more details.

11.6 5. Installing and managing tools

We can list the tools installed in our local Galaxy using:

```
nebulizer list_tools local
```

Initially there are no tools installed; we can search the main Galaxy toolshed for the tools we want to install, for example the FastQC tool:

```
nebulizer search_toolshed fastqc
```

Warning: The time taken for searching depends on the speed of the toolshed, so sometimes this can be slow if e.g. the toolshed is experiencing issues.

This will list all the tool repositories and toolshed versions available to install:

```
devteam  fastqc  21:e7b2202befea
devteam  fastqc  19:9da02be9c6cc
devteam  fastqc  16:ff9530579d1f
...
```

We can install the latest version of FastQC with

```
nebulizer install_tool local devteam/fastqc --tool-panel-section="NGS tools"
```

Note: Using `--tool-panel-section` will create a new section in the Galaxy tool panel and put the tools from this repository under it; otherwise tools are not installed under any section. You can use the `list_tool_panel` command to see what tool panel sections are already present.

Running `list_tools` now shows the tool repository is installed:

```
* fastqc  toolshed.g2.bx.psu.edu  devteam  21:e7b2202befea  Installed
```

Note: The `*` next to tool repository indicates that this is most recent version.

Use the `--mode=tools` option to list the associated tools instead.

We can install a specific version of a tool repository, for example the Trimmomatic tool:

```
nebulizer install_tool local pjbriggs/trimmomatic 51b771646466 --tool-panel-section=
↪"NGS tools"
```

Running `list_tools` now shows this tool repository is also installed:

```
* fastqc       toolshed.g2.bx.psu.edu  devteam   21:e7b2202befea  Installed
U trimmomatic  toolshed.g2.bx.psu.edu  pjbriggs  12:51b771646466  Installed
```

Here `U` indicates there is a newer revision available with a new version of the tool (`u` indicates a newer revision without a tool version update).

Rerunning the `list_tools` command with the `--updateable` option filters the list of tool repositories to just those with available updates.

We can update Trimmomatic to the newest version automatically by running the `update_tool` command:

```
nebulizer update_tool local pjbriggs/trimmomatic
```

Note: This installs the most recent version but doesn't remove the older version.

The `uninstall_tool` command removes an installed repository; for example to uninstall the older Trimmomatic tool version:

```
nebulizer uninstall_tool local pjbriggs/trimmomatic 51b771646466
```

Running `list_tools` shows that the older tool repository is no longer present.

Note: See *Managing Tools* for more details.

CHAPTER 12

History

12.1 v0.7.0 (2021-05-17)

Breaking changes:

- Dropped support for Python 2.7: `nebulizer` now needs Python 3.6+ (thanks to Hugo van Kemenade @hugovk) (PR #102)
- Substantial refactoring and simplification of the tool managament commands (PR #113): - Now only `list_tools`, `install_tool`, `update_tool`

  and `delete_tool` commands are supported

  - `list_installed_tools` renamed to `list_tools`
  - old functionality of `list_tools` replaced by `list_tools --mode=tools`
  - `list_repositories` command dropped; functionality replaced by `list_tools --mode=export`
  - `install_repositories` command dropped; functionality replaced by `install_tool --file=...`

New commands:

- New `quota`, `quota_add`, `quota_mod` and `quota_del` commands for managing quotas (PR #66)

Updates to existing commands:

- `-l` option for `search_toolshed` includes the shed URL (PR #91)
- `update_tool` allows use of wildcards (i.e. `*`) when specifying tool repository names and owners, to enable multiple tool repositories to updated at once (PR #92)
- New `--status` option for `list_users` command allows deleted and purged user accounts to also be listed (PR #97)
- New `--sort` option for `list_users` command allows sorting of listed accounts by disk usage, quota and quota usage (PR #104)

Bug fixes:

- Fix to using the `--purge` option of the `delete_user` command (previously it wasn't possible to purge accounts) (PR #98)
- Remove requirement to specify an account or API key on Galaxy server when using the `ping` and `config` commands (PR #100)

Other updates:

- Added support for Python 3.9 (thanks to Hugo van Kemenade @hugovk) (PR #108)

12.2 v0.6.0 (2020-07-14)

New commands:

- New `search_toolshed` command (PR #42)
- New `config` command (PR #57)
- New `delete_user` command (PR #62)
- New `uninstall_tool` command (PR #64)

Updates to existing commands:

- `list_keys` doesn't report API keys unless `--show-api-keys` option is specified (PR #58)
- Additional fields reported by `--long-listing-format` option of `list_users` (disk and quota usage, status); doesn't report ID by default (PR #59)
- Enable flexible tool repository specification syntax for `install_tool` and `update_tool` (PR #60)
- `remove_key` prompts user to confirm API key deletion (PR #72)
- Use spaces rather than tabs to line up fields in output from `list_users`, `list_installed_tools`, `list_tools`, `list_tool_panel`, `list_keys`, `config`, `list_libraries`; use `--show_id` to report Galaxy IDs for users and data libraries (PR #68, PR #69, PR #70)

Documentation:

- Add a tutorial/walkthrough (PR #75)
- Significant overhaul and expansion of documentation (PR #78)

Removed functionality:

- Removed deprecated utilities `manage_users`, `manage_tools` and `manage_libraries` (PR #61)

12.3 v0.5.0 (2020-04-20)

- Add support for Python 3.6, 3.7 and 3.8 (PR #50, PR #51)

12.4 v0.4.3 (2018-10-05)

- Ensure that `click` dependency is version 6.7 or earlier, to avoid subcommand names changing from e.g. `list_users` to `list-users` (PR #49)

12.5 v0.4.2 (2017-08-24)

- Commands now explicitly return appropriate exit code values indicating success (`0`) or failure (non-zero values).
- New option `--check-toolshed` added to `list_installed_tools` and `update_tool` commands, to check installed revisions directly against those available in the toolshed (PR #41)
- Update `install_tool`, `update_tool` and `install_repositories` to install tool dependencies through a resolver (e.g. `conda`) by default (issue #43)
- New options added to `install_tool`, `update_tool` and `install_repositories` commands, to explicit control how tool and repository dependencies should be handled (PR #44):
  - `--install-tool-dependencies [yes|no]`: install tool dependencies via the toolshed, if any are defined (default is `yes`)
  - `--install-tool-dependencies [yes|no]`: install tool dependencies via the toolshed, if any are defined (default is `yes`)
  - `--install-resolver-dependencies [yes|no]`: install dependencies through a resolver that supports installation (e.g. `conda`) (default is `yes`)

12.6 v0.4.1 (2016-12-19)

- Fix broken `update_tool` command (PR #40).

12.7 v0.4.0 (2016-11-18)

- New subcommand `ping`: 'ping' a Galaxy instance to see if it's responsive (PR #33).
- New subcommand `whoami`: reports user associated with the API key (PR #37).
- `add_library_datasets`: refuses to perform upload if using the master API key (essentially API key must have an associated user).
- `install_repositories`: prints a list of all tool repositories that couldn't be installed.
- New `--timeout` and `--nowait` options added for `install_tool`, `update_tool` and `install_repositories` subcommands.
- Fix to treat tool repositories with status `New` as still installing when trying to install tools (PR #31).
- Some improvements to logging (PR #38).

12.8 v0.3.0 (2016-10-26)

- New class `tools.ToolPanel` and updates to existing `tools.ToolPanelSection` class.
- `install_tool`: fix behaviour so that command does nothing if a version is not specified and at least one version of the tool is already installed.
- `list_repositories` and `install_repositories`: new commands to generate a list of installed tool repositories from a Galaxy instance and then reinstall tool repositories from a list with the same format (PR #19).
- `install_tool`: fix incorrect reporting of target tool panel section (PR #20)

- `add_key` and `update_key`: fix automatic retrieval of API key, which only worked previously if connecting user was an admin account (PR #23)
- `list_tool_panel`: shows tools in order they appear in Galaxy when using `--list-tools` option.
- Deprecated utilities (`manage_users`, `manage_tools` and `manage_libraries`) issue warnings when run.
- License updated to Academic Free License (AFL).
- Initial version of documentation also made available via ReadTheDocs (PR #21)

12.9 v0.2.0 (2016-10-17)

- Implemented new `nebulizer` utility which provides all previous functionality via subcommands, plus commands for managing API keys automatically (old `manage_users`, `manage_tools` and `manage_libraries` utilities are still available for backwards-compatibility but are deprecated).
- New general options:
  - `-q`/`--suppress-warnings`: prevent warning messages from `nebulizer` commands.
- Various fixes and improvements to underlying functionality:
  - `install_tools`: now checks if tool is already installed; handles tool revisions that include the revision number; polls Galaxy until tool is installed, or operation times out; exit status reflects the success or failure of the installation.
  - `update_tool`: now works even if original tool isn't in a tool panel section
  - `list_installed_tools`: now groups tools under correct repo revision when using `--list-tools` option.

12.10 v0.1.1 (2016-05-16)

- Add `-u`/`--username` and `-P`/`--galaxy_password` options to all commands to allow interaction with Galaxy instance via API using normal login credentials instead of API key.

12.11 v0.1.0 (2015-11-06)

- Initial release of `nebulizer` utilities for administering Galaxy instances via the command line.