Ephemeris Documentation

Release 0.7.1.dev0

Galaxy Project and Community

Oct 30, 2017

Contents

Contents:

CHAPTER 1

Introduction

Ephemeris is a small Python library and set of scripts for managing the bootstrapping of Galaxy plugins - tools, index data, and workflows.

- Free software: Academic Free License version 3.0
- Documentation: https://ephemeris.readthedocs.org.
- Code: https://github.com/galaxyproject/ephemeris

CHAPTER 2

Installation

pip

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

```
$ virtualenv .venv; . .venv/bin/activate
$ pip install ephemeris
```

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

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

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

```
$ pip install -U git+git://github.com/galaxyproject/ephemeris.git
```

Conda

Another approach for installing Ephemeris is to use Conda (most easily obtained via the Miniconda Python distribution). Afterwards run the following commands.

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

CHAPTER 3

Commands

Galaxy-wait

Utility to do a blocking sleep until a Galaxy instance is responsive. This is useful in docker images, in RUN steps, where one needs to wait for a currently starting Galaxy to be alive, before API requests can be made successfully.

The script functions by making repeated requests to `http(s)://fqdn/api/version`, an API which requires no authentication to access.

Usage

Script to sleep and wait for Galaxy to be alive.

```
usage: usage: python galaxy-wait <options>
```

Optional Arguments

--timeout=0	Galaxy startup timeout in seconds. The default value of 0 waits forever

Galaxy connection

-v=False, --verbose=False	Increase output verbosity.
-g="http://localhost:8080", --galaxy="http://localhost:8080"	Target Galaxy instance URL/IP address

Galaxy URL

Valid galaxy urls look like:

- https://example.com
- http://example.com/galaxy
- http://localhost:8080/gx

Do not include the trailing slash.

Example Usage

```
$ galaxy-wait -g https://fqdn/galaxy
```

A verbose option is offered which prints out logging statements:

```
$ galaxy-wait -g http://localhost:8080 -v
[00] Galaxy not up yet... HTTPConnectionPool(host='localhost', port=8080): Max␣
↪retries exceeded with url: /api/version (Caused
[01] Galaxy not up yet... HTTPConnectionPool(host='localhost', port=8080): Max␣
↪retries exceeded with url: /api/version (Caused
[02] Galaxy not up yet... HTTPConnectionPool(host='localhost', port=8080): Max␣
↪retries exceeded with url: /api/version (Caused
[03] Galaxy not up yet... HTTPConnectionPool(host='localhost', port=8080): Max␣
↪retries exceeded with url: /api/version (Caused
[04] Galaxy not up yet... HTTPConnectionPool(host='localhost', port=8080): Max␣
↪retries exceeded with url: /api/version (Caused
[05] Galaxy not up yet... HTTPConnectionPool(host='localhost', port=8080): Max␣
↪retries exceeded with url: /api/version (Caused
Galaxy Version: 17.05
```

When the specified Galaxy instance is up, it exits with a code of zero indicating success.

Timeout

By default, the timeout value is `0`, allowing the script to sleep forever for a Galaxy instance to be alive. This may not be desirable behaviour. In that case you can supply the `--timeout` option, and after waiting that number of seconds, the `galaxy-sleep` command will exit `1` if the Galaxy instance could not be contacted.

```
$ galaxy-wait -g https://does-not-exist -v --timeout 3
[00] Galaxy not up yet... HTTPSConnectionPool(host='does-not-exist', port=443): Max␣
↪retries exceeded with url: /api/version (C
[01] Galaxy not up yet... HTTPSConnectionPool(host='does-not-exist', port=443): Max␣
↪retries exceeded with url: /api/version (C
[02] Galaxy not up yet... HTTPSConnectionPool(host='does-not-exist', port=443): Max␣
↪retries exceeded with url: /api/version (C
[03] Galaxy not up yet... HTTPSConnectionPool(host='does-not-exist', port=443): Max␣
↪retries exceeded with url: /api/version (C
Failed to contact Galaxy))))
```

Notes

If the host returns HTML content, or otherwise non-JSON content, the tool will exit with an error.

```
$ galaxy-wait -g https://example.com -v --timeout 3
Traceback (most recent call last):
File "/home/hxr/work-freiburg/ephemeris/.venv/bin/galaxy-wait", line 11, in <module>
    load_entry_point('ephemeris', 'console_scripts', 'galaxy-wait')()
File "/home/hxr/work-freiburg/ephemeris/ephemeris/sleep.py", line 34, in main
    result = requests.get(options.galaxy + '/api/version').json()
File "/home/hxr/work-freiburg/ephemeris/.venv/lib/python3.5/site-packages/requests/
↪models.py", line 886, in json
```

```
    return complexjson.loads(self.text, **kwargs)
File "/usr/lib/python3.5/json/__init__.py", line 319, in loads
    return _default_decoder.decode(s)
File "/usr/lib/python3.5/json/decoder.py", line 339, in decode
    obj, end = self.raw_decode(s, idx=_w(s, 0).end())
File "/usr/lib/python3.5/json/decoder.py", line 357, in raw_decode
    raise JSONDecodeError("Expecting value", s, err.value) from None
json.decoder.JSONDecodeError: Expecting value: line 1 column 1 (char 0)
```

If this behaviour presents an issue for you, please file a bug with ephemeris.

Get-tool-list

Tool to extract a tool list from galaxy.

Usage

```
usage: usage: python get-tool-list <options>
```

Optional Arguments

-o, --output-file tool_list.yml output file

-include_id=False, --include_tool_panel_id=False Include tool_panel_id in tool_list.yml ? Use this only if the tool panel id already exists. See https://github.com/galaxyproject/ansible-galaxy-tools/blob/master/files/tool_list.yaml.sample

-skip_name=False, --skip_tool_panel_name=False Do not include tool_panel_name in tool_list.yml ?

-skip_changeset=False, --skip_changeset_revision=False Do not include the changeset revision when generating the tool list.Use this if you would like to use the list to update all the tools inyour galaxy instance using shed-install.

Galaxy connection

-v=False, --verbose=False Increase output verbosity.

-g=”http://localhost:8080”, --galaxy=”http://localhost:8080” Target Galaxy instance URL/IP address

Example usage: python get_tool_yml_from_gi.py -g https://usegalaxy.org/ -o tool_list.yml

Run-data-managers

Run-data-managers is a tool for provisioning data on a galaxy instance.

Run-data-managers has the ability to reload the datatables after a data manager has finished. It is therefore able to run multiple data managers that are interdependent. When a reference genome is needed for bwa-mem for example, Run-data-managers can first run a data manager to fetch the fasta file, reload the data table and run another data manager that indexes the fasta file for bwa-mem.

Run-data-managers needs a yaml that specifies what data managers are run and with which settings. An example file can be found here.

By default run-data-managers skips entries in the yaml file that have already been run. It checks it in the following way: * If the data manager has input variables "name" or "sequence_name" it will check if the "name" column in the data table already has this entry.

> "name" will take precedence over "sequence_name".

- If the data manager has input variables "value", "sequence_id" or 'dbkey' it will check if the "value" column in the data table already has this entry. Value takes precedence over sequence_id which takes precedence over dbkey.
- If none of the above input variables are specified the data manager will always run.

Usage

Running Galaxy data managers in a defined order with defined parameters.

```
usage: run-data-managers [-h] [-v] [-g GALAXY] [-u USER] [-p PASSWORD]
                         [-a API_KEY] --config CONFIG [--overwrite]
```

Optional Arguments

--config	Path to the YAML config file with the list of data managers and data to install.
--overwrite=False	Disables checking whether the item already exists in the tool data table.

Galaxy connection

-v=False, --verbose=False	Increase output verbosity.
-g="http://localhost:8080", --galaxy="http://localhost:8080"	Target Galaxy instance URL/IP address
-u, --user	Galaxy user name
-p, --password	Password for the Galaxy user
-a, --api_key	Galaxy admin user API key (required if not defined in the tools list file)

Setup-data-libraries

Tool to setup data libraries on a galaxy instance

Usage

Populate the Galaxy data library with test data.

```
usage: setup-data-libraries [-h] [-v] [-g GALAXY] [-u USER] [-p PASSWORD]
                            [-a API_KEY] -i INFILE
```

Optional Arguments

-i, --infile	Undocumented

Galaxy connection

-v=False, --verbose=False	Increase output verbosity.

-g="http://localhost:8080", --galaxy="http://localhost:8080"	Target Galaxy instance URL/IP address
-u, --user	Galaxy user name
-p, --password	Password for the Galaxy user
-a, --api_key	Galaxy admin user API key (required if not defined in the tools list file)

Shed-install

NOTE: *While shed-install can be used to run data managers, it is recommended to use run-data-managers instead.*

A script to automate installation of tool repositories from a Galaxy Tool Shed into an instance of Galaxy. Galaxy instance details and the installed tools can be provided in one of three ways:

1. In the YAML format via dedicated files (a sample can be found here).
2. On the command line as dedicated script options (see the usage help).
3. As a single composite parameter to the script. The parameter must be a single, YAML-formatted string with the keys corresponding to the keys available for use in the YAML formatted file (for example: *–yaml_tool "{'owner': 'kellrott', 'tool_shed_url': 'https://testtoolshed.g2.bx.psu.edu', 'tool_panel_section_id': 'peak_calling', 'name': 'synapse_interface'}"*).

Only one of the methods can be used with each invocation of the script but if more than one are provided are provided, precedence will correspond to order of the items in the list above. When installing tools, Galaxy expects any *tool_panel_section_id* provided when installing a tool to already exist in the configuration. If the section does not exist, the tool will be installed outside any section. See *shed_tool_conf.xml.sample* in this directory for a sample of such file. Before running this script to install the tools, make sure to place such file into Galaxy's configuration directory and set Galaxy configuration option *tool_config_file* to include it.

Usage

```
usage: usage: python shed-install <options>
```

Optional Arguments

-d, --dbkeysfile	Reference genome dbkeys to install (see dbkeys_list.yaml.sample)
-t, --toolsfile	Tools file to use (see tool_list.yaml.sample)
-y, --yaml_tool	Install tool represented by yaml string
--name	The name of the tool to install (only applicable if the tools file is not provided).
--owner	The owner of the tool to install (only applicable if the tools file is not provided).
--section	Galaxy tool panel section ID where the tool will be installed (the section must exist in Galaxy; only applicable if the tools file is not provided).
--section_label	Galaxy tool panel section label where tool will be installed (if the section does not exist, it will be created; only applicable if the tools file is not provided).
--toolshed	The Tool Shed URL where to install the tool from. This is applicable only if the tool info is provided as an option vs. in the tools file.

--skip_install_tool_dependencies=False Skip the installation of tool dependencies using classic toolshed packages. Can be overwritten on a per-tool basis in the tools file.

--install_resolver_dependencies=False Install tool dependencies through resolver (e.g. conda). Will be ignored on galaxy releases older than 16.07. Can be overwritten on a per-tool basis in the tools file

Galaxy connection

-v=False, --verbose=False Increase output verbosity.

-g="http://localhost:8080", --galaxy="http://localhost:8080" Target Galaxy instance URL/IP address

-u, --user Galaxy user name

-p, --password Password for the Galaxy user

-a, --api_key Galaxy admin user API key (required if not defined in the tools list file)

Workflow-install

Tool to install workflows on a Galaxy instance.

Usage

```
usage: workflow-install [-h] [-v] [-g GALAXY] [-u USER] [-p PASSWORD]
                        [-a API_KEY] -w WORKFLOW_PATH
```

Optional Arguments

-w, --workflow_path Path to a workflow file or a directory with multiple workflow files ending with ".ga"

Galaxy connection

-v=False, --verbose=False Increase output verbosity.

-g="http://localhost:8080", --galaxy="http://localhost:8080" Target Galaxy instance URL/IP address

-u, --user Galaxy user name

-p, --password Password for the Galaxy user

-a, --api_key Galaxy admin user API key (required if not defined in the tools list file)

Workflow-to-tools

Tool to generate tools from workflows

Usage

```
usage: python workflow-to-tools <options>
```

Optional Arguments

-w, --workflow	A space separated list of galaxy workflow description files in json format
-o, --output-file	The output file with a yml tool list
-l="Tools from workflows", --panel_label="Tools from workflows"	The name of the panel where the tools will show up in Galaxy.If not specified: "Tools from workflows"

Workflow files must have been exported from Galaxy release 16.04 or newer.

example: python %(prog)s -w workflow1 workflow2 -o mytool_list.yml -l my_panel_label Christophe Antoniewski <drosofff@gmail.com> https://github.com/ARTbio/ansible-artimed/tree/master/extra-files/generate_tool_list_from_ga_workflow_files.py

CHAPTER 4

Code of conduct

Galaxy Project Code of Conduct

This code of conduct outlines our expectations for participants within the Galaxy community, as well as steps to reporting unacceptable behavior. We are committed to providing a welcoming and inspiring community for all and expect our code of conduct to be honored. Anyone who violates this code of conduct may be banned from the community.

Our open source community strives to:

- **Be friendly and patient.**
- **Be welcoming**: We strive to be a community that welcomes and supports people of all backgrounds and identities. This includes, but is not limited to members of any race, ethnicity, culture, national origin, colour, immigration status, social and economic class, educational level, sex, sexual orientation, gender identity and expression, age, size, family status, political belief, religion, and mental and physical ability.
- **Be considerate**: Your work will be used by other people, and you in turn will depend on the work of others. Any decision you take will affect users and colleagues, and you should take those consequences into account when making decisions. Remember that we're a world-wide community, so you might not be communicating in someone else's primary language.
- **Be respectful**: Not all of us will agree all the time, but disagreement is no excuse for poor behavior and poor manners. We might all experience some frustration now and then, but we cannot allow that frustration to turn into a personal attack. It's important to remember that a community where people feel uncomfortable or threatened is not a productive one.
- **Be careful in the words that we choose**: We are a community of professionals, and we conduct ourselves professionally. Be kind to others. Do not insult or put down other participants. Harassment and other exclusionary behavior aren't acceptable. This includes, but is not limited to: Violent threats or language directed against another person, Discriminatory jokes and language, Posting sexually explicit or violent material, Posting (or threatening to post) other people's personally identifying information ("doxing"), Personal insults, especially those using racist or sexist terms, Unwelcome sexual attention, Advocating for, or encouraging, any of the above behavior, Repeated harassment of others. In general, if someone asks you to stop, then stop.

- **Try to understand why we disagree**: Disagreements, both social and technical, happen all the time. It is important that we resolve disagreements and differing views constructively. Remember that we're different. Diversity contributes to the strength of our community, which is composed of people from a wide range of backgrounds. Different people have different perspectives on issues. Being unable to understand why someone holds a viewpoint doesn't mean that they're wrong. Don't forget that it is human to err and blaming each other doesn't get us anywhere. Instead, focus on helping to resolve issues and learning from mistakes.

Diversity Statement

We encourage everyone to participate and are committed to building a community for all. Although we will fail at times, we seek to treat everyone both as fairly and equally as possible. Whenever a participant has made a mistake, we expect them to take responsibility for it. If someone has been harmed or offended, it is our responsibility to listen carefully and respectfully, and do our best to right the wrong.

Although this list cannot be exhaustive, we explicitly honor diversity in age, gender, gender identity or expression, culture, ethnicity, language, national origin, political beliefs, profession, race, religion, sexual orientation, socioeconomic status, and technical ability. We will not tolerate discrimination based on any of the protected characteristics above, including participants with disabilities.

Reporting Issues

If you experience or witness unacceptable behavior, or have any other concerns, please report it by contacting Dave Clements (clementsgalaxy@gmail.com). To report an issue involving Dave Clements please email James Taylor (james@taylorlab.org). All reports will be handled with discretion. In your report please include:

- Your contact information.
- Names (real, nicknames, or pseudonyms) of any individuals involved. If there are additional witnesses, please include them as well. Your account of what occurred, and if you believe the incident is ongoing. If there is a publicly available record (e.g. a mailing list archive or a public IRC logger), please include a link.
- Any additional information that may be helpful.

After filing a report, a representative will contact you personally, review the incident, follow up with any additional questions, and make a decision as to how to respond. If the person who is harassing you is part of the response team, they will recuse themselves from handling your incident. If the complaint originates from a member of the response team, it will be handled by a different member of the response team. We will respect confidentiality requests for the purpose of protecting victims of abuse.

Attribution & Acknowledgements

This code of conduct is based on the Open Code of Conduct from the TODOGroup.

CHAPTER 5

Contributing

Please note that this project is released with a Contributor Code of Conduct. By participating in this project you agree to abide by its terms.

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

You can contribute in many ways:

Types of Contributions

Report Bugs

Report bugs at https://github.com/galaxyproject/ephemeris/issues.

If you are reporting a bug, please include:

- Your operating system name and version, versions of other relevant software such as Galaxy.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

Fix Bugs

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

Implement Features

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

Write Documentation

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

User documentation

User documentation is (partly) automated to contain the first docstring in a module and the usage based on the parser object.

If you want to contribute to the user documentation you can edit the docstring or the parser module or write more information in the commands .rst file. (See galaxy-wait for an example.)

When you add a new command line tool in ephemeris you can add documentation as follows:

1. Go to the source file and:

- Add a docstring that gives general information about the module. (Examples in shed-install and run-data-managers)
- Create a new _parser() method that returns the argument parser.

2. Create a new rst file using shed-install.rst or run-data-managers.rst as a template.
3. Reference the new rst file in commands.rst

To build your documentation to check out how it works before submitting the pull request: 1. Install sphinx in a virtual environment by running *pip install -r docs/requirements.txt* from ephemeris root 2. go to the docs directory and run *make html*

Submit Feedback

The best way to send feedback is to file an issue at https://github.com/galaxyproject/ephemeris/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.
- This will hopefully become a community-driven project and contributions are welcome :)

Get Started!

Ready to contribute? Here's how to set up *ephemeris* for local development.

1. Fork the *ephemeris* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/ephemeris.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ make setup-venv
```

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` and the tests

```
$ make lint
```

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.

Pull Request Guidelines

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

1. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring.
2. The pull request should work for Python 2.7. Check https://travis-ci.org/galaxyproject/ephemeris/pull_requests and make sure that the tests pass for all supported Python versions.

CHAPTER 6

Project Governance

This document informally outlines the organizational structure governing the Ephemeris code base hosted at https://github.com/galaxyproject/ephemeris. This governance extends to code-related activities of this repository such as releases and packaging. This governance does not include any other Galaxy-related projects belonging to the `galaxyproject` organization on GitHub.

Benevolent Dictator for Now (BDFN)

Björn Grüning (@bgruening) is the benevolent dictator for now (BDFN) and is solely responsible for setting project policy. The BDFN is responsible for maintaining the trust of the developer community and so should be consistent and transparent in decision making processes and request comment and build consensus whenever possible.

The BDFN position only exists because the developers of the project believe it is currently too small to support full and open governance at this time. In order to keep things evolving quickly, it is better to keep procedures and process to a minimum and centralize important decisions with a trusted developer. The BDFN is explicitly meant to be replaced with a more formal and democratice process if the project grows to a sufficient size or importance.

The *committers* group is the group of trusted developers and advocates who manage the Ephemeris code base. They assume many roles required to achieve the project's goals, especially those that require a high level of trust.

The BDFN will add committers as he or she see fits, usually after a few successful pull requests. Committers may commit directly or merge pull requests at their discretion, but everyone (including the BDFN) should open pull requests for larger changes.

In order to encourage a shared sense of ownership and openness, any committer may decide at any time to request a open governance model for the project be established and the BDFN must replace this informal policy with a more formal one and work with the project committers to establish a consensus on these procedures.

Committers

- Enis Afgan (@afgane)

- Christophe Antoniewski (@drosofff)
- John Chilton (@jmchilton)
- Björn Grüning (@bgruening)
- Marius van den Beek (@mvdbeek)

CHAPTER 7

Release Checklist

This page describes the process of releasing new versions of Ephemeris.

This release checklist is based on the Pocoo Release Management Workflow.

This assumes `~/.pypirc` file exists with the following fields (variations) are fine.

```
[distutils]
index-servers =
    pypi
    test

[pypi]
username:<username>
password:<password>

[test]
repository:https://testpypi.python.org/pypi
username:<username>
password:<password>
```

- Review `git status` for missing files.
- Verify the latest Travis CI builds pass.
- `make open-docs` and review changelog.
- Ensure the target release is set correctly in `ephemeris/__init__.py` (`version` will be a `devN` variant of target release).
- `make clean && make lint && make test`
- `make release`

  This process will push packages to test PyPI, allow review, publish to production PyPI, tag the git repository, push the tag upstream. If custom changes to this process are needed, the process can be broken down into steps including:

  – `make release-local`

  - `make push-release`

CHAPTER 8

History

0.7.1.dev0

0.7.0 (2017-06-27)

- Many new documentation enhancements (thanks to @rhpvorderman, @erasche, and others) - docs are now published to https://readthedocs.org/projects/ephemeris/.
- Fix problem with empty list options related to running data managers (thanks to @rhpvorderman).
- Enable data managers to run with API keys (thanks to @rhpvorderman).
- Add sleep command to wait for a Galaxy API to become available (thanks to @erasche).
- Preserve readable order of keys while processing tool lists (thanks to @drosofff).

0.6.1 (2017-04-17)

- Add Python 2 and 3 testing for all scripts against galaxy-docker-stable along with various refactoring to reduce code duplication and Python 3 fixes. #36

0.6.0 (2017-04-10)

- Add new connection options for setting up data libraries.

0.5.1 (2017-04-07)

- Fix new `run-data-managers` CLI entrypoint.

0.5.0 (2017-04-06)

- Add `run-data-managers` tool to trigger DM with multiple values and in order. #30
- The workflow install tool now supports a directory of workflows. #27
- enable global options in a tool yaml files, like *install_resolver_dependencies: true* #26
- Mention mimum required galaxy versions. #23 (thanks to @mvdbeek)

0.4.0 (2016-09-07)

- Be more generic in determining a server time-out (thanks to @afgane).
- Get tool list entrypoint and improvements (thanks to @mvdbeek).
- Rename `tool_panel_section_name` to `tool_panel_section_label` like ansible-galaxy-tools (thanks to @nturaga).
- Add missing file `tool_list.yaml.sample` (thanks to @nturaga).

0.3.0 (2016-08-26)

- More robust shed-install script, install dependencies by default, improve logging (thanks to @mvdbeek).

0.2.0 (2016-08-15)

- Add tool generate a tool list from a Galaxy workflow file (thanks to @drosofff).
- Fix various code quality issues including adding beta support for Python 3 (thanks in part to @mvdbeek).

0.1.0 (2016-06-15)

- Setup project, pull in scripts from ansible-galaxy-tools and adapt them for usage as a library.

CHAPTER 9

Indices and tables

- genindex
- modindex
- search

Index