Gravity

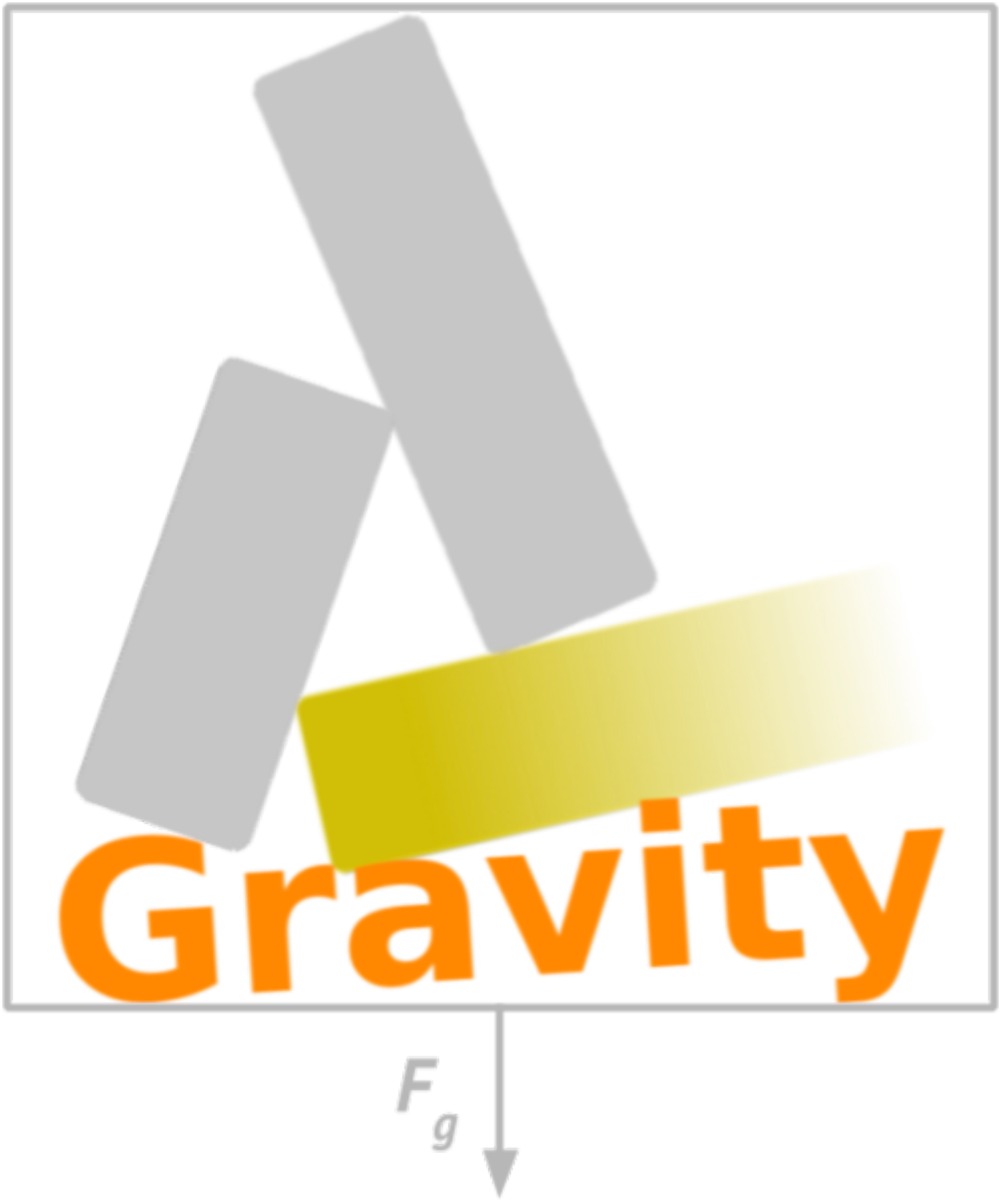
Release 1.0.3

Sep 20, 2023

Contents:

Process management for Galaxy servers.

- License: MIT
- Documentation: https://gravity.readthedocs.io
- Code: https://github.com/galaxyproject/gravity


Gravity
F_g

CHAPTER 1

Overview

Modern Galaxy servers run multiple disparate processes: gunicorn for serving the web application, celery for asynchronous tasks, tusd for fault-tolerant uploads, standalone Galaxy processes for job handling, and more. Gravity is Galaxy's process manager, to make configuring and running these services simple.

Installing Gravity will give you two executables, `galaxyctl` which is used to manage the starting, stopping, and logging of Galaxy's various processes, and `galaxy`, which can be used to run a Galaxy server in the foreground.

CHAPTER 2

Quick Start

2.1 Installation

Python 3.7 or later is required. Gravity can be installed independently of Galaxy, but it is also a dependency of Galaxy since Galaxy 22.01. If you've installed Galaxy, then Gravity is already installed in Galaxy's virtualenv.

To install independently:

```
$ pip install gravity
```

2.2 Usage

From the root directory of a source checkout of Galaxy, after first run (or running Galaxy's `./scripts/common_startup.sh`), activate Galaxy's virtualenv, which will put Gravity's `galaxyctl` and `galaxy` commands on your `$PATH`:

```
$ . ./.venv/bin/activate
$ galaxyctl --help
Usage: galaxyctl [OPTIONS] COMMAND [ARGS]...

  Manage Galaxy server configurations and processes.

... additional help output
```

You can start and run Galaxy in the foreground using the `galaxy` command:

```
$ galaxy
Registered galaxy config: /srv/galaxy/config/galaxy.yml
Creating or updating service gunicorn
Creating or updating service celery
Creating or updating service celery-beat
celery: added process group
```

(continues on next page)

(continued from previous page)

```
2022-01-20 14:44:24,619 INFO spawned: 'celery' with pid 291651
celery-beat: added process group
2022-01-20 14:44:24,620 INFO spawned: 'celery-beat' with pid 291652
gunicorn: added process group
2022-01-20 14:44:24,622 INFO spawned: 'gunicorn' with pid 291653
celery                           STARTING
celery-beat                      STARTING
gunicorn                         STARTING
==> /srv/galaxy/var/gravity/log/gunicorn.log <==
...log output follows...
```

Galaxy will continue to run and output logs to stdout until terminated with `CTRL+C`.

More detailed configuration and usage examples, especially those concerning production Galaxy servers, can be found in the full documentation.

CHAPTER 3

Documentation Conventions

Examples in this documentation assume a Galaxy layout like the one used in the Galaxy Installation with Ansible tutorial:

```
/srv/galaxy/server   # Galaxy code
/srv/galaxy/config   # config files
/srv/galaxy/venv     # virtualenv
```

CHAPTER 4

Installation and Configuration

4.1 Installation

Python 3.7 or later is required. Gravity can be installed independently of Galaxy, but it is also a dependency of Galaxy since Galaxy 22.01. If you've installed Galaxy, then Gravity is already installed in Galaxy's virtualenv.

To install independently:

```
$ pip install gravity
```

To make your life easier, you are encourged to install into a virtualenv. The easiest way to do this is with Python's built-in venv module:

```
$ python3 -m venv ~/gravity
$ . ~/gravity/bin/activate
```

4.2 Configuration

Gravity needs to know where your Galaxy configuration file is, and depending on your Galaxy layout, some additional details like the paths to its virtualenv and root directory. By default, Gravity's configuration is defined in Galaxy's configuration file (`galaxy.yml`) to be easy and familiar for Galaxy administrators. Gravity's configuration is defined underneath the `gravity` key, and Galaxy's configuration is defined underneath the `galaxy` key. For example:

```
---
gravity:
  gunicorn:
    bind: localhost:8192
galaxy:
  database_connection: postgresql:///galaxy
```

4.2.1 Configuration Search Paths

If you run `galaxy` or `galaxyctl` from the root of a Galaxy source checkout and do not specify the config file option, `config/galaxy.yml` or `config/galaxy.yml.sample` will be automatically used. To avoid having to run from the Galaxy root directory or to work with a config file in a different location, you can explicitly point Gravity at your Galaxy configuration file with the `--config-file` (`-c`) option or the `$GRAVITY_CONFIG_FILE` (or `$GALAXY_CONFIG_FILE`, as set by Galaxy's `run.sh` script) environment variable. Then it's possible to run the `galaxyctl` command from anywhere.

Often times it's convenient to put the environment variable in the Galaxy user's shell environment file, e.g.:

```
$ echo "export GRAVITY_CONFIG_FILE='/srv/galaxy/config/galaxy.yml'" >> ~/.bash_profile
```

When running Gravity as root, the following configuration files will automatically be searched for and read, unless `--config-file` is specified or `$GRAVITY_CONFIG_FILE` is set:

- `/etc/galaxy/gravity.yml`
- `/etc/galaxy/galaxy.yml`
- `/etc/galaxy/gravity.d/*.y(a?)ml`

4.2.2 Splitting Gravity and Galaxy Configurations

For more advanced deployments, it is *not* necessary to write your entire Galaxy configuration to the Gravity config file. You can write only the Gravity configuration, and then point to your Galaxy config file with the `galaxy_config_file` option in the Gravity config. This can be useful for cases such as your Galaxy server being split across multiple hosts.

For example, on a deployment where the web (gunicorn) and job handler processes run on different hosts, one might have:

In `gravity.yml` on the web host:

```
---
gravity:
  galaxy_config_file: galaxy.yml
  log_dir: /var/log/galaxy
  gunicorn:
    bind: localhost:8888
  celery:
    enable: false
    enable_beat: false
```

In `gravity.yml` on the job handler host:

```
---
gravity:
  galaxy_config_file: galaxy.yml
  log_dir: /var/log/galaxy
  gunicorn:
    enable: false
  celery:
    enable: true
    enable_beat: true
  handlers:
    handler:
      processes: 2
```

See the *Managing Multiple Galaxies* section for additional examples.

4.3 Configuration Options

The following options in the `gravity` section of `galaxy.yml` can be used to configure Gravity:

```
# Configuration for Gravity process manager.
# ``uwsgi:`` section will be ignored if Galaxy is started via Gravity commands (e.g. ``./run.sh``, ``galaxy`` or ``galaxyctl``).
gravity:

  # Process manager to use.
  # ``supervisor`` is the default process manager when Gravity is invoked as a non-root user.
  # ``systemd`` is the default when Gravity is invoked as root.
  # Valid options are: supervisor, systemd
  # process_manager:

  # What command to write to the process manager configs
  # `gravity` (`galaxyctl exec <service-name>`) is the default
  # `direct` (each service's actual command) is also supported.
  # Valid options are: gravity, direct
  # service_command_style: gravity

  # Use the process manager's *service instance* functionality for services that can run multiple instances.
  # Presently this includes services like gunicorn and Galaxy dynamic job handlers. Service instances are only supported if
  # ``service_command_style`` is ``gravity``, and so this option is automatically set to ``false`` if
  # ``service_command_style`` is set to ``direct``.
  # use_service_instances: true

  # umask under which services should be executed. Setting ``umask`` on an individual service overrides this value.
  # umask: '022'

  # Memory limit (in GB), processes exceeding the limit will be killed. Default is no limit. If set, this is default value
  # for all services. Setting ``memory_limit`` on an individual service overrides this value. Ignored if ``process_manager``
  # is ``supervisor``.
  # memory_limit:

  # Specify Galaxy config file (galaxy.yml), if the Gravity config is separate from the Galaxy config. Assumed to be the
  # same file as the Gravity config if a ``galaxy`` key exists at the root level, otherwise, this option is required.
  # galaxy_config_file:

  # Specify Galaxy's root directory.
  # Gravity will attempt to find the root directory, but you can set the directory explicitly with this option.
  # galaxy_root:

  # User to run Galaxy as, required when using the systemd process manager as root.
```

(continues on next page)

(continued from previous page)

```
  # Ignored if ``process_manager`` is ``supervisor`` or user-mode (non-root)␣
↪``systemd``.
  # galaxy_user:

  # Group to run Galaxy as, optional when using the systemd process manager as root.
  # Ignored if ``process_manager`` is ``supervisor`` or user-mode (non-root)␣
↪``systemd``.
  # galaxy_group:

  # Set to a directory that should contain log files for the processes controlled by␣
↪Gravity.
  # If not specified defaults to ``<galaxy_data_dir>/gravity/log``.
  # log_dir:

  # Set to Galaxy's virtualenv directory.
  # If not specified, Gravity assumes all processes are on PATH. This option is␣
↪required in most circumstances when using
  # the ``systemd`` process manager.
  # virtualenv:

  # Select the application server.
  # ``gunicorn`` is the default application server.
  # ``unicornherder`` is a production-oriented manager for (G)unicorn servers that␣
↪automates zero-downtime Galaxy server restarts,
  # similar to uWSGI Zerg Mode used in the past.
  # Valid options are: gunicorn, unicornherder
  # app_server: gunicorn

  # Override the default instance name.
  # this is hidden from you when running a single instance.
  # instance_name: _default_

  # Configuration for Gunicorn. Can be a list to run multiple gunicorns for rolling␣
↪restarts.
  gunicorn:

    # Enable Galaxy gunicorn server.
    # enable: true

    # The socket to bind. A string of the form: ``HOST``, ``HOST:PORT``,␣
↪``unix:PATH``, ``fd://FD``. An IP is a valid HOST.
    # bind: localhost:8080

    # Controls the number of Galaxy application processes Gunicorn will spawn.
    # Increased web performance can be attained by increasing this value.
    # If Gunicorn is the only application on the server, a good starting value is the␣
↪number of CPUs * 2 + 1.
    # 4-12 workers should be able to handle hundreds if not thousands of requests per␣
↪second.
    # workers: 1

    # Gunicorn workers silent for more than this many seconds are killed and␣
↪restarted.
    # Value is a positive number or 0. Setting it to 0 has the effect of infinite␣
↪timeouts by disabling timeouts for all workers entirely.
    # If you disable the ``preload`` option workers need to have finished booting␣
↪within the timeout.
```

(continues on next page)

(continued from previous page)

```
    # timeout: 300

    # Extra arguments to pass to Gunicorn command line.
    # extra_args:

    # Use Gunicorn's --preload option to fork workers after loading the Galaxy␣
↪Application.
    # Consumes less memory when multiple processes are configured. Default is␣
↪``false`` if using unicornherder, else ``true``.
    # preload:

    # umask under which service should be executed
    # umask:

    # Value of supervisor startsecs, systemd TimeoutStartSec
    # start_timeout: 15

    # Value of supervisor stopwaitsecs, systemd TimeoutStopSec
    # stop_timeout: 65

    # Amount of time to wait for a server to become alive when performing rolling␣
↪restarts.
    # restart_timeout: 300

    # Memory limit (in GB). If the service exceeds the limit, it will be killed.␣
↪Default is no limit or the value of the
    # ``memory_limit`` setting at the top level of the Gravity configuration, if set.␣
↪Ignored if ``process_manager`` is
    # ``supervisor``.
    # memory_limit:

    # Extra environment variables and their values to set when running the service. A␣
↪dictionary where keys are the variable
    # names.
    # environment: {}

  # Configuration for Celery Processes.
  celery:

    # Enable Celery distributed task queue.
    # enable: true

    # Enable Celery Beat periodic task runner.
    # enable_beat: true

    # Number of Celery Workers to start.
    # concurrency: 2

    # Log Level to use for Celery Worker.
    # Valid options are: DEBUG, INFO, WARNING, ERROR
    # loglevel: DEBUG

    # Queues to join
    # queues: celery,galaxy.internal,galaxy.external

    # Pool implementation
    # Valid options are: prefork, eventlet, gevent, solo, processes, threads
```

(continues on next page)

(continued from previous page)

```
    # pool: threads

    # Extra arguments to pass to Celery command line.
    # extra_args:

    # umask under which service should be executed
    # umask:

    # Value of supervisor startsecs, systemd TimeoutStartSec
    # start_timeout: 10

    # Value of supervisor stopwaitsecs, systemd TimeoutStopSec
    # stop_timeout: 10

    # Memory limit (in GB). If the service exceeds the limit, it will be killed.␣
↪Default is no limit or the value of the
    # ``memory_limit`` setting at the top level of the Gravity configuration, if set.␣
↪Ignored if ``process_manager`` is
    # ``supervisor``.
    # memory_limit:

    # Extra environment variables and their values to set when running the service. A␣
↪dictionary where keys are the variable
    # names.
    # environment: {}

  # Configuration for gx-it-proxy.
  gx_it_proxy:

    # Set to true to start gx-it-proxy
    # enable: false

    # gx-it-proxy version
    # version: '>=0.0.5'

    # Public-facing IP of the proxy
    # ip: localhost

    # Public-facing port of the proxy
    # port: 4002

    # Routes file to monitor.
    # Should be set to the same path as ``interactivetools_map`` in the ``galaxy:``␣
↪section. This is ignored if
    # ``interactivetools_map is set``.
    # sessions: database/interactivetools_map.sqlite

    # Include verbose messages in gx-it-proxy
    # verbose: true

    # Forward all requests to IP.
    # This is an advanced option that is only needed when proxying to remote␣
↪interactive tool container that cannot be reached through the local network.
    # forward_ip:

    # Forward all requests to port.
    # This is an advanced option that is only needed when proxying to remote␣
↪interactive tool container that cannot be reached through the local network.
```

(continues on next page)

(continued from previous page)

```
    # forward_port:

    # Rewrite location blocks with proxy port.
    # This is an advanced option that is only needed when proxying to remote␣
↪interactive tool container that cannot be reached through the local network.
    # reverse_proxy: false

    # umask under which service should be executed
    # umask:

    # Value of supervisor startsecs, systemd TimeoutStartSec
    # start_timeout: 10

    # Value of supervisor stopwaitsecs, systemd TimeoutStopSec
    # stop_timeout: 10

    # Memory limit (in GB). If the service exceeds the limit, it will be killed.␣
↪Default is no limit or the value of the
    # ``memory_limit`` setting at the top level of the Gravity configuration, if set.␣
↪Ignored if ``process_manager`` is
    # ``supervisor``.
    # memory_limit:

    # Extra environment variables and their values to set when running the service. A␣
↪dictionary where keys are the variable
    # names.
    # environment: {}

  # Configuration for tusd server (https://github.com/tus/tusd).
  # The ``tusd`` binary must be installed manually and made available on PATH (e.g in␣
↪galaxy's .venv/bin directory).
  tusd:

    # Enable tusd server.
    # If enabled, you also need to set up your proxy as outlined in https://docs.
↪galaxyproject.org/en/latest/admin/nginx.html#receiving-files-via-the-tus-protocol.
    # enable: false

    # Path to tusd binary
    # tusd_path: tusd

    # Host to bind the tusd server to
    # host: localhost

    # Port to bind the tusd server to
    # port: 1080

    # Directory to store uploads in.
    # Must match ``tus_upload_store`` setting in ``galaxy:`` section.
    # upload_dir:

    # Comma-separated string of enabled tusd hooks.
    #
    # Leave at the default value to require authorization at upload creation time.
    # This means Galaxy's web process does not need to be running after creating the␣
↪initial
    # upload request.
```

(continues on next page)

(continued from previous page)

```
    #
    # Set to empty string to disable all authorization. This means data can be␣
↪uploaded (but not processed)
    # without the Galaxy web process being available.
    #
    # You can find a list of available hooks at https://github.com/tus/tusd/blob/
↪master/docs/hooks.md#list-of-available-hooks.
    # hooks_enabled_events: pre-create

    # Extra arguments to pass to tusd command line.
    # extra_args:

    # umask under which service should be executed
    # umask:

    # Value of supervisor startsecs, systemd TimeoutStartSec
    # start_timeout: 10

    # Value of supervisor stopwaitsecs, systemd TimeoutStopSec
    # stop_timeout: 10

    # Memory limit (in GB). If the service exceeds the limit, it will be killed.␣
↪Default is no limit or the value of the
    # ``memory_limit`` setting at the top level of the Gravity configuration, if set.␣
↪Ignored if ``process_manager`` is
    # ``supervisor``.
    # memory_limit:

    # Extra environment variables and their values to set when running the service. A␣
↪dictionary where keys are the variable
    # names.
    # environment: {}

  # Configuration for Galaxy Reports.
  reports:

    # Enable Galaxy Reports server.
    # enable: false

    # Path to reports.yml, relative to galaxy.yml if not absolute
    # config_file: reports.yml

    # The socket to bind. A string of the form: ``HOST``, ``HOST:PORT``,␣
↪``unix:PATH``, ``fd://FD``. An IP is a valid HOST.
    # bind: localhost:9001

    # Controls the number of Galaxy Reports application processes Gunicorn will spawn.
    # It is not generally necessary to increase this for the low-traffic Reports␣
↪server.
    # workers: 1

    # Gunicorn workers silent for more than this many seconds are killed and␣
↪restarted.
    # Value is a positive number or 0. Setting it to 0 has the effect of infinite␣
↪timeouts by disabling timeouts for all workers entirely.
    # timeout: 300
```

(continues on next page)

(continued from previous page)

```
    # URL prefix to serve from.
    # The corresponding nginx configuration is (replace <url_prefix> and <bind> with␣
↪the values from these options):
    #
    # location /<url_prefix>/ {
    #     proxy_pass http://<bind>/;
    # }
    #
    # If <bind> is a unix socket, you will need a ``:`` after the socket path but␣
↪before the trailing slash like so:
    #     proxy_pass http://unix:/run/reports.sock:/;
    # url_prefix:

    # Extra arguments to pass to Gunicorn command line.
    # extra_args:

    # umask under which service should be executed
    # umask:

    # Value of supervisor startsecs, systemd TimeoutStartSec
    # start_timeout: 10

    # Value of supervisor stopwaitsecs, systemd TimeoutStopSec
    # stop_timeout: 10

    # Memory limit (in GB). If the service exceeds the limit, it will be killed.␣
↪Default is no limit or the value of the
    # ``memory_limit`` setting at the top level of the Gravity configuration, if set.␣
↪Ignored if ``process_manager`` is
    # ``supervisor``.
    # memory_limit:

    # Extra environment variables and their values to set when running the service. A␣
↪dictionary where keys are the variable
    # names.
    # environment: {}

  # Configure dynamic handlers in this section.
  # See https://docs.galaxyproject.org/en/latest/admin/scaling.html#dynamically-
↪defined-handlers for details.
  # handlers: {}
```

4.4 Galaxy Job Handlers

Gravity has support for reading Galaxy's job configuration: it can read statically configured job handlers in the `job_conf.yml` or `job_conf.xml` files, or the job configuration inline from the `job_config` option in `galaxy.yml`. However, unless you need to statically define handlers, it is simpler to configure Gravity to run dynamically defined handlers as detailed in the Galaxy scaling documentation.

When using dynamically defined handlers, be sure to explicitly set the job handler assignment method to `db-skip-locked` or `db-transaction-isolation` to prevent the web process from also handling jobs.

4.5 Gravity State

Older versions of Gravity stored a considerable amount of *config state* in `$GRAVITY_STATE_DIR/configstate.yaml`. As of version 1.0.0, Gravity does not store state information, and this file can be removed if left over from an older installation.

Although Gravity no longer uses the config state file, it does still use a state directory for storing supervisor configs, the default log directory (if `log_dir` is unchanged), and the celery-beat database. This directory defaults to `<galaxy_root>/database/gravity/` by way of the `data_dir` option in the `galaxy` section of `galaxy.yml` (which defaults to `<galaxy_root>/database/`).

If running multiple Galaxy servers with the same Gravity configuration as described in *Managing Multiple Galaxies* and if doing so using supervisor rather than systemd, the supervisor configurations will be stored in `$XDG_CONFIG_HOME/galaxy-gravity` (`$XDG_CONFIG_HOME` defaults to `~/.config/galaxy-gravity`)

In any case, you can override the path to the state directory using the `--state-dir` option, or the `$GRAVITY_STATE_DIR` environment variable.

Note: Galaxy 22.01 and 22.05 automatically set `$GRAVITY_STATE_DIR` to `<galaxy_root>/database/gravity` in the virtualenv's activation script, `activate`. This can be removed from the activate script when using Gravity 1.0.0 or later.

CHAPTER 5

Basic Usage

A basic example of starting and running a simple Galaxy server from a source clone in the foreground is provided in the ref:*Quick Start* guide. This section covers more typical usage for production Galaxy servers.

5.1 Managing a Single Galaxy

If you have not installed Gravity separate from the Galaxy virtualenv, simply activate Galaxy's virtualenv, which will put Gravity's `galaxyctl` and `galaxy` commands on your `$PATH`:

```
$ . /srv/galaxy/venv/bin/activate
$ galaxyctl --help
Usage: galaxyctl [OPTIONS] COMMAND [ARGS]...

  Manage Galaxy server configurations and processes.

Options:
  -d, --debug             Enables debug mode.
  -c, --config-file FILE  Gravity (or Galaxy) config file to operate on. Can
                          also be set with $GRAVITY_CONFIG_FILE or
                          $GALAXY_CONFIG_FILE
  --state-dir DIRECTORY   Where process management configs and state will be
                          stored.
  -h, --help              Show this message and exit.

Commands:
  configs     List registered config files.
  deregister  Deregister config file(s).
  exec        Run a single Galaxy service in the foreground, with logging...
  follow      Follow log files of configured instances.
  graceful    Gracefully reload configured services.
  instances   List all known instances.
  pm          Invoke process manager (supervisorctl, systemctl) directly.
  register    Register config file(s).
```

(continues on next page)

(continued from previous page)

```
  rename      Update path of registered config file.
  restart     Restart configured services.
  show        Show details of registered config.
  shutdown    Shut down process manager.
  start       Start configured services.
  status      Display server status.
  stop        Stop configured services.
  update      Update process manager from config changes.
```

If you run `galaxy` or `galaxyctl` from the root of a Galaxy source checkout and do not specify the config file option, `config/galaxy.yml` or `config/galaxy.yml.sample` will be automatically used. This is handy for working with local clones of Galaxy for testing or development. You can skip Galaxy's lengthy and repetitive `run.sh` configuration steps when starting and stopping Galaxy in between code updates (you should still run `run.sh` after performing a `git pull` to make sure your dependencies are up to date).

Gravity can either run Galaxy via the supervisor process manager (the default) or systemd. For production servers, **it is recommended that you run Gravity as root in systemd mode**. See the *Managing a Production Galaxy* section for details.

As shown in the Quick Start, the `galaxy` command will run a Galaxy server in the foreground. The `galaxy` command is actually a shortcut for two separate steps: 1. read the provided `galaxy.yml` and write out the corresponding process manager configurations, and 2. start and run Galaxy in the foreground using the process manager (supervisor). You can perform these steps separately (and in this example, start Galaxy as a backgrounded daemon instead of in the foreground):

```
$ galaxyctl update
Adding service gunicorn
Adding service celery
Adding service celery-beat
$ galaxyctl start
celery                           STARTING
celery-beat                      STARTING
gunicorn                         STARTING
Log files are in /srv/galaxy/var/gravity/log
```

When running as a daemon, the `stop` subcommand stops your Galaxy server:

```
$ galaxyctl stop
celery-beat: stopped
gunicorn: stopped
celery: stopped
All processes stopped, supervisord will exit
Shut down
```

Most Gravity subcommands (such as `stop`, `start`, `restart`, ...) are straightforward, but a few subcommands are worth pointing out: *update*, *graceful*, and *exec*. All subcommands are documented in the *Subcommands* section and their respective `--help` output.

5.2 Managing a Production Galaxy

By default, Gravity runs Galaxy processes under supervisor, but setting the `process_manager` option to `systemd` in Gravity's configuration will cause it to run under systemd instead. systemd is the default init system under most modern Linux distributions, and using systemd is strongly encouraged for production Galaxy deployments.

Gravity manages systemd service unit files corresponding to all of the Galaxy services that it is aware of, much like how it manages supervisor program config files in supervisor mode. If you run `galaxyctl update` as a non-root user, the unit files will be installed in `~/.config/systemd/user` and run via systemd user mode. This can be useful for testing and development, but in production it is recommended to run Gravity as root, so that it installs the service units in `/etc/systemd/system` and are managed by the privileged systemd instance. Even when Gravity is run as root, Galaxy itself still runs as a non-root user, specified by the `galaxy_user` option in the Gravity configuration.

It is also recommended, when running as root, that you install Gravity independent of Galaxy, rather than use the copy installed in Galaxy's virtualenv:

```
# python3 -m venv /opt/gravity
# /opt/gravity/bin/pip install gravity
```

Caution: Because systemd unit file names have semantic meaning (the filename is the service's name) and systemd does not have a facility for isolating unit files controlled by an application, Gravity considers all unit files in the unit dir (`/etc/systemd/system`) that are named like `galaxy-*` to be controlled by Gravity. **If you have existing unit files that are named as such, Gravity will overwrite or remove them.**

In systemd mode, and especially when run as root, some Gravity options are required:

```
gravity:
  process_manager: systemd

  # required if running as root
  galaxy_user: GALAXY-USERNAME
  # optional, defaults to primary group of the user set above
  galaxy_group: GALAXY-GROUPNAME

  # required
  virtualenv: /srv/galaxy/venv
  # probably necessary if your galaxy.yml is not in galaxy_root/config
  galaxy_root: /srv/galaxy/server
```

See the *Configuration* section for more details on these options and others.

The `log_dir` option is ignored when using systemd. Logs are instead captured by systemd's logging facility, `journald`.

You can use `galaxyctl` to manage Galaxy process starts/stops/restarts/etc. and follow the logs, just as you do under supervisor, but you can also use `systemctl` and `journalctl` directly to manage process states and inspect logs (respectively). Only `galaxyctl update` is necessary, in order to write and/or remove the appropriate systemd service units based on your configuration. For example:

```
# export GRAVITY_CONFIG_FILE=/srv/galaxy/config/galaxy.yml
# . /srv/galaxy/venv/bin/activate
(venv) # galaxyctl update
Adding service galaxy-gunicorn.service
Adding service galaxy-celery.service
Adding service galaxy-celery-beat.service
```

After this point, operations can be performed with either `galaxyctl` or `systemctl`. Some examples of equivalent commands:

Gravity	systemd
`galaxy`	`systemctl start galaxy.target && journalctl -f -u 'galaxy-*'`
`galaxyctl start`	`systemctl start galaxy.target`
`galaxyctl start SERVICE ...`	`systemctl start galaxy-SERVICE.service galaxy-...`
`galaxyctl restart`	`systemctl restart galaxy.target`
`galaxyctl restart SERVICE ...`	`systemctl restart galaxy-SERVICE.service galaxy-...`
`galaxyctl graceful`	`systemctl reload-or-restart galaxy.target`
`galaxyctl graceful SERVICE ...`	`systemctl reload-or-restart galaxy-SERVICE.service galaxy-...`
`galaxyctl stop`	`systemctl start galaxy.target`
`galayxctl follow`	`journalctl -f -u 'galaxy-*'`

CHAPTER 6

Advanced Usage

6.1 Zero-Downtime Restarts

Prior to Gravity 1.0, the preferred solution for performing zero-downtime restarts was unicornherder. However, due to limitations in the unicornherder software, it does not always successfully perform zero-downtime restarts. Because of this, Gravity is now able to perform rolling restarts of gunicorn services if more than one gunicorn is configured.

To run multiple gunicorn processes, configure the `gunicorn` section of the Gravity configuration as a *list*. Each item in the list is a gunicorn configuration, and can have all of the same parameters as a single gunicorn configuration:

```
gravity:
  gunicorn:
    - bind: unix:/srv/galaxy/var/gunicorn0.sock
      workers: 4
    - bind: unix:/srv/galaxy/var/gunicorn1.sock
      workers: 4
```

Caution: This will start multiple Galaxy servers with the same `server_name`. If you have not configured separate Galaxy processes to act as job handlers, your gunicorn processes will handle them, resulting in job errors due to handling the same job multiple times. See the Gravity and Galaxy documentation on configuring handlers.

Your proxy server can balance load between the two gunicorns. For example, with nginx:

```
upstream galaxy {
    server unix:/srv/galaxy/var/gunicorn0.sock;
    server unix:/srv/galaxy/var/gunicorn1.sock;
}

http {
    location / {
        proxy_pass http://galaxy;
```

(continues on next page)

(continued from previous page)

```
    }
}
```

By default, Gravity will wait 300 seconds for the gunicorn server to respond to web requests after initiating the restart. To change this timeout this, set the `restart_timeout` option on each configured `gunicorn` instance.

6.2 Service Instances

In the case of multiple gunicorn instances as described in *Zero-Downtime Restarts* and multiple dynamic handlers as described in *Galaxy Job Handlers*, Gravity will create multiple *service instances* of each service. This allows multiple processes to be run from a single service definition.

In supervisor, this means that the service names as presented by supervisor are appended with `:INSTANCE_NUMBER`, e.g.:

```
$ galaxyctl status
celery                           RUNNING   pid 121363, uptime 0:02:33
celery-beat                      RUNNING   pid 121364, uptime 0:02:33
gunicorn:gunicorn_0              RUNNING   pid 121365, uptime 0:02:33
gunicorn:gunicorn_1              RUNNING   pid 121366, uptime 0:02:33
```

However, `galaxyctl` commands that take a service name still use the base service name, e.g.:

```
$ galaxyctl stop gunicorn
gunicorn:gunicorn_0: stopped
gunicorn:gunicorn_1: stopped
Not all processes stopped, supervisord not shut down (hint: see `galaxyctl status`)
```

In systemd, the service names as presented by systemd are appended with `@INSTANCE_NUMBER`, e.g.:

```
$ galaxyctl status
  UNIT                        LOAD   ACTIVE SUB     DESCRIPTION
  galaxy-celery-beat.service loaded active running Galaxy celery-beat
  galaxy-celery.service      loaded active running Galaxy celery
  galaxy-gunicorn@0.service  loaded active running Galaxy gunicorn (process 0)
  galaxy-gunicorn@1.service  loaded active running Galaxy gunicorn (process 1)
  galaxy.target              loaded active active  Galaxy
```

As with supervisor, `galaxyctl` commands that take a service name still use the base service name.

If you prefer not to work with service instances and want Galaxy to write a service configuration file for each instance of each service, you can do so by setting `use_service_instances` in the Gravity configuration to `false`.

6.3 Managing Multiple Galaxies

Gravity can manage multiple instances of Galaxy simultaneously. This is useful especially in the case where you have multiple production Galaxy instances on a single server and are managing them with Gravity installed outside of a Galaxy virtualenv, as root. There are multiple ways to achieve this:

1. Pass multiple `--config-file` options to `galaxyctl`, or set a list of colon-separated config paths in `$GRAVITY_CONFIG_FILE`:

```
$ galaxyctl --config-file /srv/galaxy/test/config/galaxy.yml \
            --config-file /srv/galaxy/main/config/galaxy.yml list --
↪version
TYPE      INSTANCE NAME       VERSION       CONFIG PATH
galaxy    test                22.05         /srv/galaxy/test/config/
↪galaxy.yml
galaxy    main                22.09.dev0    /srv/galaxy/main/config/
↪galaxy.yml
$ export GRAVITY_CONFIG_FILE='/srv/galaxy/test/config/galaxy.yml:/srv/
↪galaxy/main/config/galaxy.yml'
$ galaxyctl list --version
TYPE      INSTANCE NAME       VERSION       CONFIG PATH
galaxy    test                22.05         /srv/galaxy/test/config/
↪galaxy.yml
galaxy    main                22.09.dev0    /srv/galaxy/main/config/
↪galaxy.yml
```

2. If running as root, any config files located in `/etc/galaxy/gravity.d` will automatically be loaded.
3. Specify multiple Gravity configurations in a single config file, as a list. In this case, the Galaxy and Gravity configurations must be in separate files as described in *Splitting Gravity and Galaxy Configurations*:

```
gravity:
  - instance_name: test
    process_manager: systemd
    galaxy_config_file: /srv/galaxy/test/config/galaxy.yml
    galaxy_root: /srv/galaxy/test/server
    virtualenv: /srv/galaxy/test/venv
    galaxy_user: gxtest
    gunicorn:
      bind: unix:/srv/galaxy/test/var/gunicorn.sock
    handlers:
      handler:
        pools:
          - job-handlers
          - workflow-schedulers

  - instance_name: main
    process_manager: systemd
    galaxy_config_file: /srv/galaxy/main/config/galaxy.yml
    galaxy_root: /srv/galaxy/main/server
    virtualenv: /srv/galaxy/main/venv
    galaxy_user: gxmain
    gunicorn:
      bind: unix:/srv/galaxy/main/var/gunicorn.sock
      workers: 8
    handlers:
      handler:
        processes: 4
        pools:
          - job-handlers
          - workflow-schedulers
```

In all cases, when using multiple Gravity instances, each Galaxy instance managed by Gravity must have a unique **instance name**. When working with a single instance, the default name `_default_` is used automatically and mostly hidden from you. When working with multiple instances, set the `instance_name` option in each instance's Gravity config to a unique name.

Although it is strongly encouraged to use systemd for running multiple instances, it is possible to use supervisor. Please see the *Gravity State* section for important details on how and where Gravity stores the supervisor configuration and log files.

CHAPTER 7

Subcommands

Use `galaxyctl --help` for help. Subcommands also support `--help`, e.g. `galaxy register --help`

7.1 start

Start and run Galaxy and associated processes in daemonized (background) mode, or `-f` to run in the foreground and follow log files. The `galaxy` command is a shortcut for `galaxyctl start -f`.

7.2 stop

Stop daemonized Galaxy server processes. If using supervisor mode and no processes remain running after this step (which should be the case when working with a single Galaxy instance), `supervisord` will terminate.

7.3 restart

Restart Galaxy server processes. This is done in a relatively "brutal" fashion: processes are signaled (by the process manager) to exit, and then are restarted. See the `graceful` subcommand to restart gracefully.

7.4 graceful

Restart Galaxy with minimal interruption.

If running with a single gunicorn without `preload`, this means holding the web socket open while restarting (connections to Galaxy will block). With `preload`, gunicorn is restarted and some clients may experience connection failures.

If running with multiple gunicorns, a rolling restart is performed, where Gravity restarts each gunicorn, waits for it to respond to requests after restarting, and then moves to the next one. This process should be transparent to clients. See *Zero-Downtime Restarts* for configuration details.

If running with unicornherder, a new Galaxy application will be started and the old one shut down only once the new one is accepting connections. This should also be transparent to clients, but limitations in the unicornherder software may allow interruptions to occur.

7.5 update

Figure out what has changed in the Galaxy/Gravity config(s), which could be:

- changes to the Gravity configuration options in `galaxy.yml`
- adding or removing handlers in `job_conf.yml` or `job_conf.xml`

This may cause service restarts if there are any changes.

Any needed changes to supervisor or systemd configs will be performed and then `supervisorctl update` or `systemctl daemon-reload` will be called.

If you wish to *remove* any existing process manager configurations for Galaxy servers managed by Gravity, the `--clean` flag to `update` can be used for this purpose.

7.6 shutdown

Stop all processes and cause `supervisord` to terminate. Similar to `stop` but there is no ambiguity as to whether `supervisord` remains running. The equivalent of `stop` when using systemd.

7.7 follow

Follow (e.g. using `tail -f` (supervisor) or `journalctl -f` (systemd)) log files of all Galaxy services, or a subset (if named as arguments).

7.8 list

List config files known to Gravity.

7.9 show

Show Gravity configuration details for a Galaxy instance.

7.10 pm

Pass through directly to the process manager (e.g. supervisor). Run `galaxyctl pm` to invoke the supervisorctl shell, or `galaxyctl pm [command]` to call a supervisorctl or systemctl command directly. See the supervisor documentation or `galaxyctl pm help` for help.

7.11 exec

Directly execute a single Galaxy service in the foreground, e.g. `galaxyctl exec gunicorn`, `galaxyctl exec tusd`, etc.

When Gravity writes out configs for the underlying process manager, it must provide a *command* (program and arguments) to execute and some number of *environment variables* that must be set for each individual Galaxy service (gunicorn, celery, etc.) to execute. By default, rather than write this information directly to the process manager configuration, Gravity sets the command to `galaxyctl exec --config-file=<gravity-config-path> <service-name>`. The `exec` subcommand instructs Gravity to use the exec(3) system call to execute the actual service command with its proper arguments and environment.

This is done so that it is is not necesary to rewrite the process manager configs and update the process manager every time a parameter is changed, only when services are added or removed entirely. Gravity can instead be instructed to write the actual service command and environment variables directly to the process manager configurations by setting `service_command_style` to `direct`.

Thus, although `exec` is mostly an internal subcommand, developers and admins may find it useful when debugging in order to quickly and easily start just a single service and view only that service's logs in the foreground.

CHAPTER 8

History

8.1 1.0.3

- Don't create supervisor conf dir unless necessary, create the gravity data dir as the correct user by @natefoo in https://github.com/galaxyproject/gravity/pull/105

8.2 1.0.2

- Pin a minimum package version of gx-it-proxy by @natefoo in https://github.com/galaxyproject/gravity/pull/102

8.3 1.0.1

- Added configuration of gx-it-proxy to support path-based proxying by @sveinugu in https://github.com/galaxyproject/gravity/pull/100

8.4 1.0.0

Version 1.0.0 represents a significant update to Gravity, its features and functionality. Although Gravity 1.x is intended to be backwards compatible with 0.x, you are strongly encouraged to [read the documentation](https://gravity.readthedocs.io/en/latest/) if upgrading to Gravity 1.x or to Galaxy 23.0 (which depends on Gravity 1.x) in order to get the most out of the new features.

- Support systemd as a process manager by @natefoo in https://github.com/galaxyproject/gravity/pull/77
- Full stateless mode when working with single instances and other improvements for 1.0 by @natefoo in https://github.com/galaxyproject/gravity/pull/80

- Multi-unicorn rolling restart and general multi-instance service support by @natefoo in https://github.com/galaxyproject/gravity/pull/81
- Don't clobber other Galaxies' systemd units when managed by different Gravity config files by @natefoo in https://github.com/galaxyproject/gravity/pull/83
- Don't restart tusd on graceful by @natefoo in https://github.com/galaxyproject/gravity/pull/85
- Read job_conf.yml by default if job_config_file is unset by @natefoo in https://github.com/galaxyproject/gravity/pull/86
- Fixes for spaces in the galaxy root path, fix the *galaxy* entrypoint by @natefoo in https://github.com/galaxyproject/gravity/pull/87
- Update existing env with program env when running exec, rather than the other way around by @natefoo in https://github.com/galaxyproject/gravity/pull/93
- Hide the "exec" ServiceCommandStyle from documentation since it is only used internally by @natefoo in https://github.com/galaxyproject/gravity/pull/94
- Updates for settings documentation generation by @natefoo in https://github.com/galaxyproject/gravity/pull/95
- Set *$VIRTUAL_ENV* if *virtualenv* is set in config by @natefoo in https://github.com/galaxyproject/gravity/pull/97
- Always add venv bin dir to *$PATH* if *virtualenv* is set by @natefoo in https://github.com/galaxyproject/gravity/pull/98

8.5 0.13.6

- Fix graceful method for gunicorn `--preload` by @Slugger70 in https://github.com/galaxyproject/gravity/pull/76
- Add `--version` option to get Gravity version by @natefoo in https://github.com/galaxyproject/gravity/pull/79
- Fix stopping of gx-it-proxy by @abretaud in https://github.com/galaxyproject/gravity/pull/91

8.6 0.13.5

- If virtualenv is set in the Gravity config, automatically add its bin dir to $PATH if the gx-it-proxy is enabled by @natefoo in https://github.com/galaxyproject/gravity/pull/71
- Support converting settings to command line arguments in a generalized way by @natefoo in https://github.com/galaxyproject/gravity/pull/73

8.7 0.13.4

- Fixes for startup test by @natefoo in https://github.com/galaxyproject/gravity/pull/68
- Fix setting environment vars on handlers by @natefoo in https://github.com/galaxyproject/gravity/pull/70

8.8 0.13.3

- Don't use gunicorn logging options with unicornherder by @natefoo in https://github.com/galaxyproject/gravity/pull/65

8.9 0.13.2

- Don't override PATH in subprocess call by @jdavcs in https://github.com/galaxyproject/gravity/pull/62
- Only send pre create hook by @mvdbeek in https://github.com/galaxyproject/gravity/pull/64

8.10 0.13.1

- Set correct default for environment settings by @natefoo in https://github.com/galaxyproject/gravity/pull/58
- Don't catch exceptions in the deregister, register, and rename subcommands by @natefoo in https://github.com/galaxyproject/gravity/pull/59
- `processes` in the `handling` dict in the job conf dict is a dict, not a list by @natefoo in https://github.com/galaxyproject/gravity/pull/60

8.11 0.13.0

- Add options to enable/disable gunicorn, celery, and celery-beat services by @natefoo in https://github.com/galaxyproject/gravity/pull/47
- Add ability to include gravity config from a separate file and document by @natefoo in https://github.com/galaxyproject/gravity/pull/48
- Only default to preload = true for gunicorn if not using unicornherder by @natefoo in https://github.com/galaxyproject/gravity/pull/49
- Add option to specify tusd path by @natefoo in https://github.com/galaxyproject/gravity/pull/50
- Support setting per-service environment variables by @natefoo in https://github.com/galaxyproject/gravity/pull/56

8.12 0.12.0

- Fix typo in `log_dir` description by @nsoranzo in https://github.com/galaxyproject/gravity/pull/44
- Shortcut individual services fix by @natefoo in https://github.com/galaxyproject/gravity/pull/45
- Add additional options to celery beat / celery workers by @mvdbeek in https://github.com/galaxyproject/gravity/pull/46

8.13 0.11.0

- Allow setting supervisor socket path via environment variable by @mvdbeek in https://github.com/galaxyproject/gravity/pull/36
- Automatically switch to non-sample galaxy.yml if it exists by @mvdbeek in https://github.com/galaxyproject/gravity/pull/39
- Add pydantic config schema by @mvdbeek in https://github.com/galaxyproject/gravity/pull/42
- Add –quiet option to galaxy and galaxyctl start by @mvdbeek in https://github.com/galaxyproject/gravity/pull/40
- Add support for yaml job config by @mvdbeek in https://github.com/galaxyproject/gravity/pull/37
- Add –preload support for gunicorn by @mvdbeek in https://github.com/galaxyproject/gravity/pull/41
- Support running tusd by @natefoo in https://github.com/galaxyproject/gravity/pull/23

8.14 0.10.0

- Fix for the case where a job_conf.xml exists but no handlers are defined by @natefoo in https://github.com/galaxyproject/gravity/pull/24
- Do not raise error if config file section is empty by @nsoranzo in https://github.com/galaxyproject/gravity/pull/25
- Add tests for static handlers and a defined job_conf.xml with no handlers by @natefoo in https://github.com/galaxyproject/gravity/pull/26
- Fix minor typos in readme by @ic4f in https://github.com/galaxyproject/gravity/pull/27
- Move configuration to gravity key of galaxy.yml file by @mvdbeek in https://github.com/galaxyproject/gravity/pull/28
- Fix for resolved galaxy.yml.sample symlink by @mvdbeek in https://github.com/galaxyproject/gravity/pull/31
- Support managing gx-it-proxy via gravity by @mvdbeek in https://github.com/galaxyproject/gravity/pull/32

8.15 0.9

- Gunicorn/fastAPI support, click support, tests by @mvdbeek in https://github.com/galaxyproject/gravity/pull/14
- Don't test on Python 3.6, which is unsupported by @natefoo in https://github.com/galaxyproject/gravity/pull/17
- Update README. Also some various small bugfixes and fixes for other stuff mentioned in the README by @natefoo in https://github.com/galaxyproject/gravity/pull/18
- Add unicornherder support by @natefoo in https://github.com/galaxyproject/gravity/pull/15
- Expose the log following used by *start -f* as its own subcommand. by @natefoo in https://github.com/galaxyproject/gravity/pull/16
- Better integration with Galaxy's run.sh by @natefoo in https://github.com/galaxyproject/gravity/pull/19
- Use relative paths in supervisord by @natefoo in https://github.com/galaxyproject/gravity/pull/21
- Converted CLI from argparse to click.
- Stole ideas and code from planemo in general.

- Improve the AttributeDict so that it can have "hidden" items (anything that starts with a _) that won't be serialized. Also, it serializes itself and can be created via deserialization from a classmethod. This simplifies using it to persist state data in the new GravityState subclass.

8.16 0.8.3

- Merge `galaxycfg` and `galaxyadm` commands to `galaxy`.

8.17 0.8.2

- Allow for passing names of individual services directly to `supervisorctl` via the `start`, `stop`, and `restart` methods.
- Fix a bug where uWSGI would not start when using the automatic virtualenv install method.

8.18 0.8.1

- Version bump because I deleted the 0.8 files from PyPI, and despite the fact that it lets you delete them, it doesn't let you upload once they have been uploaded once...

8.19 0.8

- Add auto-register to `galaxy start` if it's called from the root (or subdirectory) of a Galaxy root directory.
- Make `galaxycfg remove` accept instance names as params in addition to config file paths.
- Use the same hash generated for an instance name as the hash for a generated virtualenv name, so virtualenvs are more easily identified as belonging to a config.
- Renamed from `galaxyadmin` to `gravity` (thanks John Chilton).

8.20 0.7

- Added the `galaxyadm` subcommand `graceful` on a suggestion from Nicola Soranzo.
- Install uWSGI into the config's virtualenv if requested.
- Removed any dependence on Galaxy and eggs.
- Moved project to its own repository from the Galaxy clone I'd been working from.

8.21 Older

- Works in progress as part of the Galaxy code.

CHAPTER 9

Indices and tables

- genindex
- search