Game Server Manager Documentation

Release 0.1.2+4.g24206f4.dirty

Christopher Bailey

Oct 30, 2018

Contents

Contents:

CHAPTER 1

Game Server Manager

Command to manage and control various types of game servers.

- Free software: MIT license
- (Coming soon!) Documentation: https://game-server-manager.readthedocs.io.

1.1 Requirements

- **POSIX Complient System - built and tested on Arch Linux, but should work on any Linux, MAC OSX or Windows Subsy**
  - Uses and requires the following commands:

```
grep
java # optional for Java based servers
ln
nohup
ps
screen # optional for screen based servers
steamcmd # optional for Steam based servers
vim # or whatever your default $EDITOR command is
which
```

- Python - built and tested with 3.6, but for full 1.0 release, unit tests will suppport 2.7 and 3.4+ unless there is a compelling reason not to

1.2 Features

Allows full management of different types of servers with full configuration supported for each. Existing types (so far):

1.2.1 Generic configurable gameserver types

- **Custom Screen (custom_screen)**: Generic gameserver that has an interactive console and can easily be ran via the screen command. Requires additional configuration to work.
- **Custom Steam (custom_steam)**: Generic gameserver that can be installed and updated from Steam. Also, optionally support Steam workshop. Requires additional configuration to work.
- **Custom RCON (custom_rcon)**: Generic Steam gameserver with Source RCON protocol support. Requires additional configuration to work.
- **Java (java)**: Generic Java base gameserver that can be ran with screen. Requires additional configuration to work.

1.2.2 Gameservers for specific games

- **Minecraft (minecraft)**: Java based gameserver ran with screen for Minecraft.
- **ARK (ark)**: Steam based gameserver with RCON support for ARK: Surivial Evolved.

1.3 Quickstart

Install from pip:

```
sudo pip install game_server_manager
gs --help
```

gs will attempt to use *.gs_config.json* as the main configuration file. If this does not exist, you must provide all configuration options via command line. *-t* will speciify type of gameserver and *-s* will save a *.gs_config.json* file based on your commandline parameters.

1.3.1 Generic

1. Generate default config (assuming generic type of *custom_screen*):

   ```
   gs -t custom_screen -s status
   ```

2. Edit *.gs_config.json* with anything that is relevant to your server
3. Start server:

   ```
   gs start
   ```

4. *Optional*: Once you get everything working, make an issue and/or pull request to make a new server type so you do not have to configure in the future!

1.3.2 Minecraft

Existing Install

If you already have an existing install, it is simple to set up *gs* to run with it:

```
gs -t minecraft -s status
```

This will generate a default *.gs_config.json* file. Edit this to match your existing install.

Java

You much have Java installed to run Minecraft. If you need help installing Java, consult the documentation on the Minecraft wiki:

- https://minecraft.gamepedia.com/Tutorials/Setting_up_a_server#Installing_Java_2

Firewall

Open any firewall ports you need as detailed on Minecraft wiki:

- https://minecraft.gamepedia.com/Tutorials/Setting_up_a_server#Firewalling.2C_NATs_and_external_IP_addresses

Install/Start

Assuming you want the latest stable version of Minecraft and the server to run as user *minecraft* with all of the default settings:

```
gs -t minecraft -u minecraft -s install
gs start
gs status
```

See *gs -t minecraft install –help* for more details.

1.3.3 ARK

Existing Install

If you already have an existing install, it is simple to set up *gs* to run with it:

```
gs -t ark -s status
```

This will generate a default *.gs_config.json* file. Edit this to match your existing install.

SteamCMD

Install SteamCMD according to the docs for your OS:

- Valve Docs: https://developer.valvesoftware.com/wiki/SteamCMD
- Arch Linux: https://wiki.archlinux.org/index.php/Steam#SteamCMD

Open File Limit

Increase Open Files Limit as detailed on ARK wiki:

- https://ark.gamepedia.com/Dedicated_Server_Setup#Open_Files_Limit

Firewall

Open any firewall ports you need as detailed on ARK wiki:

- https://ark.gamepedia.com/Dedicated_Server_Setup#Port_Forwarding_and_Firewall

Install/Start

Assuming you want the server to run as user *ark* with all of the default settings and no mods:

```
gs -t ark -u ark -s install
gs start
gs status
```

See *gs -t ark install –help* for more details.

Multiple Instances

It is common to run multiple ARK servers together as a cluster. To do this, you want to use the *instance_overrides* config option. Example .gs_config.json

You can run subcommands against all instances at once with *-ci @all*. You can even run them all in parellel (get for starting and stopping) with *-p*:

```
gs start -ci @all -p
gs status -ci @all
gs stop -ci @all -p
```

1.4 Planned

Stuff planned before the 1.0 release:

- Full Unit Test and code coverage (Python 2.7, 3.4+ support)
- Documentation
- Forge and Curse support for Minecraft servers
- Backup command for all servers
- Staging support to update servers while still running
- Probably more stuff and maybe more server types

1.5 Credits

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

CHAPTER 2

Installation

2.1 Stable release

To install Game Server Manager, run this command in your terminal:

```
$ pip install game_server_manager
```

This is the preferred method to install Game Server Manager, as it will always install the most recent stable release.

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

2.2 From sources

The sources for Game Server Manager can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/AngellusMortis/game_server_manager
```

Or download the tarball:

```
$ curl  -OL https://github.com/AngellusMortis/game_server_manager/tarball/master
```

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

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

CHAPTER 3

Usage

To use Game Server Manager in a project:

```
import game_server_manager
```

CHAPTER 4

Contributing

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

You can contribute in many ways:

4.1 Types of Contributions

4.1.1 Report Bugs

Report bugs at https://github.com/AngellusMortis/game_server_manager/issues.

If you are reporting a bug, please include:

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

4.1.2 Fix Bugs

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

4.1.3 Implement Features

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

4.1.4 Write Documentation

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

4.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/AngellusMortis/game_server_manager/issues.

If you are proposing a feature:

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

4.2 Get Started!

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

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

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

```
$ mkvirtualenv game_server_manager
$ cd game_server_manager/
$ python setup.py develop
```

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, including testing other Python versions with tox:

```
$ flake8 gs_manager tests
$ python setup.py test or py.test
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

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

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

7. Submit a pull request through the GitHub website.

4.3 Pull Request Guidelines

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

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 2.7, 3.3, 3.4, 3.5 and 3.6, including PyPy. Check https://travis-ci.org/AngellusMortis/game_server_manager/pull_requests and make sure that the tests pass for all supported Python versions.

CHAPTER 5

Credits

5.1 Development Lead

- Christopher Bailey <cbailey@mort.is>

5.2 Contributors

None yet. Why not be the first?

CHAPTER 6

History

6.1 0.1.2 (2017-12-16)

- Changes *gs* to re-run main command as user if it is different than existing. Removes all extra logic for indivual commands as another user.

6.2 0.1.0 (2017-12-15)

- First release on PyPI.

CHAPTER 7

Indices and tables

- genindex
- modindex
- search