Mirrors Documentation

Release 0.4.0

Ethan House

May 29, 2017

Contents

Contents:

CHAPTER 1

Getting Started

Installation

Pre-Build

```
git clone http://github.com/ehouse/mirrors.git
./setup.py build
```

Install

Running in a virtualenv isn't required but recommended.

```
./setup.py install
```

Uninstall

```
pip uninstall mirrors
```

Configuration

Configuration can be in any file you choose however mirrors.conf is what is included. The file must be included when mirrors is run with the -c flag.

(ex. mirrors -c mirrors.conf)

Global Config Options

```
[GLOBAL]
```

All of the global configuration is stored within the [GLOBAL] block. If it is not present the application will fail to load.

```
async_processes = 4
```

The number of syncs that may be run at any given time. Scheduled syncs will wait until a spot opens up before it will begin running.

```
log_file = ./mirrors.log
```

Application log file. This is were errors, warnings or general information is logged to. Default is mirrors.log

```
check_sleep = 30
```

Time to sleep in between syncs . Default is 30 seconds.

Repo Options

```
[test1]
```

Name of the repo in brackets. This is how it will be accessed within the repl.

```
source = rsync://mirrors.rit.edu/FreeBSD
```

Source of the rsync transfer.

```
rsync_args = -avhz
```

Flags to pass into the rsync process.

```
destination = ./distros/
```

Destination of the rsync file transfer.

```
weight = 0
```

Weight of the sync. Lower numbers will go before higher numbers. Value between -10 and 10. Default is 0

```
pre_command =
```

Shell command to run before the rsync starts.

```
post_command =
```

Shell command to run after the rsync finishes.

```
log_file = ./log/LDP.log
```

Location of the repo log file. Rsync STDOUT and STDERR are piped here.

```
async_sleep = 2h
```

Time to wait after a sync has completed before it is re-queued.

```
hourly_sync = 0,6.5,12,18.5
```

Strict time frame for syncs to run.

CHAPTER 2

Usage

CHAPTER 3

Source Docs

repo.py

class `mirrors.repo.`**`Repo`**`(`*name*, *config*`)`

`__init__``(`*name*, *config*`)`
A repo object which stores info about a single repo.

Parameters

- **name** (*str*) – Name of repo
- **config** (*ConfigParser.ConfigParser*) – running config options

`__weakref__`
list of weak references to the object (if defined)

`is_alive``()`
Bool of syncing status.

`kill``()`
Send SIGKILL To the rsync process.

class **`rsync_thread`**`(`*name*, *config*`)`
Extended threading.Thread class to control rsync via subprocess.

Parameters

- **name** (*str*) – Name of repo
- **config** (*Configparser.Configparser*) – Running config options

`Repo.`**`running_time`**`()`
Total running time of active sync.

Return type int

Returns An int of total syncing time elapsed

Return type None

Returns None if not syncing

`Repo.`**`sleep_time`**`()`
Sleep duration of sleeping sync.

Return type int

Returns A int of time elapsed since sleeping

Return type None

Returns None if not in sleeping state

`Repo.`**`start_sync`**`()`
Run an rsync against the repo source.

`Repo.`**`terminate`**`()`
Send SIGTERM To the rsync process.

`Repo.`**`time_remaining`**`()`
Return time left until sleep is over.

Return type int

Returns A int of time remaining in sleep state

Return type None

Returns None if not in sleeping state

class `mirrors.repo.`**`RepoManager`**`(`*config*`)`

`__init__``(`*config*`)`
Singleton manager of the repositories and threading.

Parameters **config** (*Configparser.Configparser*) – Running config options

`__metaclass__`
alias of `Singleton`

`__weakref__`
list of weak references to the object (if defined)

`activate``(`*name*`)`
Activate repo for syncing.

Parameters **name** (*str*) – Name of Repo

Raises Repo.RepoError if no repo exists by given name

`add_repo``(`*name*`)`
Create a repo for a section in the running config.

Parameters **name** (*str*) – Name of repo

Raises Repo.RepoConfigError if no config exists for given repo name

`deactivate``(`*name*`)`
Deactivate repo from syncing.

Parameters **name** (*str*) – Name of repo

Raises Repo.RepoError if no repo exists by given name

`del_repo``(`*name*`)`
Delete repo object from dict.

Parameters **name** (*str*) – Name of repo

Raises Repo.RepoError if no repo exists by passed in name.

`enqueue``(`*name*`)`
Add repo to the queue.

Parameters **name** (*str*) – Name of repo

Raises Repo.RepoError if repo is already queued or doesn't exist

gen_repo()
Generator for repo_dict.

Return type Repo

Returns Repo Object

get_repo(*name*)
Return repo object if exists.

Parameters **name** (*str*) – name of repo

Return type Repo

Returns Repo Object

Return type None

Returns None if no repo exists by passed in name

status(*name*)
Return status of Repo.

Parameters **name** (*str*) – Name of Repo

Return type str

Returns str status of Repo

libmirrors.py

mirrors.libmirrors.**t2s**(*s*)
Converts human readable time to seconds.

Parameters **s** (*str*) – Human readable time string (ex. 5m or 2h)

Return type int

Returns int time converted to seconds

libcmd.py

class mirrors.cmdline.**Console**(*repo_manager*)

do_activate(**args*)
Activate repo for syncing.

do_add(**args*)
Add repo from running config.

TODO

do_config(**args*)
Edit configuration running settings.

TODO

do_deactivate(**args*)
Deactivate repo from syncing.

do_del(**args*)
Delete repo from running config.

TODO

do_enqueue(**args*)
Add repo onto the end of the async queue.

do_exit(**args*)
Stops all syncs and terminates mirrors.

do_forcekill(**args*)
Send SIGKILL to rsync process.

do_kill(**args*)
Send SIGTERM to rsync process.

do_list(**args*)
List all of the loaded repos.

do_print(**args*)
Print config and status events.

TODO

do_quit(**args*)
Stops all syncs and terminates mirrors.

do_reload(**args*)
Reload either individual or entire config.

TODO

do_start(**args*)
Add repo onto the end of the async queue.

do_status(**args*)
Prints status table or the status of an individual sync.

do_terminate(**args*)
Send SIGTERM to rsync process.

do_write(**args*)
Write configuration settings to file.

TODO

postloop()
postloop cleanup.

CHAPTER 4

Indices and tables

- *genindex*
- *search*

Index