ghau

Release [0.0.1]

Oct 27, 2020

Contents:

CHAPTER 1

Introduction

ghau is a python library aimed to provide easy automatic updates utilizing Github Releases. Its goal is to make adding update capabilities to programs you release much, much easier.

This package is currently in development, so its API may change drastically with each update. If you have any suggestions, questions, or bug reports. Please open an issue at <https://github.com/InValidFire/ghau/issues>

1.1 Tutorial

First create an instance of the Data class:

```
import ghau

VERSION = "v0.0.1"
REPO = "InValidFire/ghau"
update = ghau.Update(version=VERSION, repo=REPO)
```

Then to check for updates and install if necessary, simply do the following:

```
update.update()
```

1.2 Configuration

The *Update* class has a lot of configuration options to tailor the update process for your needs. Examples of how each option is configured can be seen here:

Downloading pre-releases:

```
import ghau

VERSION = "v0.0.1"
```

(continues on next page)

(continued from previous page)

```
REPO = "InValidFire/ghau"
update = ghau.Update(version=VERSION, repo=REPO, pre-releases=True)
update.update()
```

Cleaning and whitelisting files:

```
import ghau

VERSION = "v0.0.1"
REPO = "InValidFire/ghau"

update = ghau.Update(version=VERSION, repo=REPO)

# Whitelisted files are protected from deletion during cleaning.
update.wl_files("README.md","*\*/data/*")

# Cleanlisted files will be deleted before update installation.
# Usually this isn't needed as files will be overwritten if it already exists.
update.cl_files("old_dir/*")
update.update()
```

Downloading Assets:

```
import ghau

VERSION = "v0.0.1"
REPO = "InValidFire/ghau"

#not setting the asset will tell ghau to download the first asset it comes across.
ASSET = "program.exe"

update = ghau.Update(version=VERSION, repo=REPO, download="asset", asset=ASSET)
update.update()
```

Rebooting after updates:

```
import ghau

VERSION = "v0.0.1"
REPO = "InValidFire/ghau"

# rebooting to a python script
REBOOT = ghau.python("main.py")

#rebooting to an executable
REBOOT = ghau.exe("program.exe")

#rebooting using some other command
REBOOT = ghau.cmd("some other command")

update = ghau.Update(version=VERSION, repo=REPO, reboot=REBOOT)
update.update()
```

Authenticating w/ Github API:

```
import os
import ghau
```

(continues on next page)

(continued from previous page)

```
VERSION = "v0.0.1"
REPO = "InValidFire/ghau"

# we store our tokens in an environmental variable in this example.
# this protects it from being compromized, keep your tokens secure.
TOKEN = os.environ['GAPI']

update = ghau.Update(version=VERSION, repo=REPO, auth=TOKEN)
update.update()
```

1.3 Update Loops

Something that can happen if the programmer doesn't keep the version parameter in *ghau.Update* consistent with the release version on Github is an Update Loop.

For example, you release an update "v1.1.0" on Github Releases, but you have "v1.0.0" given to the Update class for that release. This will cause ghau to constantly install updates as it wants the two numbers to match.

These update loops can be prevented in two ways:

1. Ensure before you release an update that the two numbers will match.
2. Using *ghau.python()*, *ghau.exe()*, or *ghau.cmd()* to build the reboot command you use.

These functions will add a parameter that ghau will detect on the next boot, telling it to stop the update process.

1.4 Whitelisting

Whitelisted files will not be overwritten during update installation. Use this to protect data directories or files. Every instance of the *ghau.Update* class has a whitelist and may be added to it using methods *ghau.Update.wl_files()*, and `ghau.Update.wl_exclude()`:

```
import ghau

update = ghau.Update(version="v0.0.1", repo="InValidFire/ghau")
update.wl_files("data.csv")
update.wl_exclude("data/")
update.update()
```

1.5 Licensing

ghau is distributed under the MIT License.

CHAPTER 2

Reference

2.1 Main Class

Most interactions with ghau will be done through the main Update class. Details regarding it can be found below:

class ghau.**Update**(*version: str*, *repo: str*, *pre_releases: bool = False*, *reboot: str = None*, *download: str = 'zip'*, *asset: str = None*, *auth: str = None*, *ratemin: int = 20*, *boot_check: bool = True*, *success_func=None*, *fail_func=None*, *pre_update_func=None*, *fail_args=None*, *success_args=None*, *post_update_func=None*, *pre_update_args=None*, *post_update_args=None*)

Main class used to trigger updates through ghau.

Parameters

- **version** (*str*) – local version to check against online versions.
- **repo** (*str*) – github repository to check for updates in. Must be publicly accessible unless you are using a Github Token.
- **pre-releases** (*bool, optional*) – accept pre-releases as valid updates, defaults to False.
- **reboot** (*str, optional*) – command intended to reboot the program after a successful update installs.
- **download** (*str, optional*) – the type of download you wish to use for updates. Either "zip" (source code) or "asset" (uploaded files), defaults to "zip".
- **asset** (*str, optional.*) – name of asset to download when set to "asset" mode.
- **auth** (*str, optional*) – authentication token used for accessing the Github API, defaults to None.
- **ratemin** (*int, optional.*) – minimum amount of API requests left before updates will stop, defaults to 20. Maximum is 60/hr for unauthorized requests, and 5000 for authorized requests.

restart()
Restarts the program

update()
Check for updates and install if an update is found.

All expected exceptions triggered during the run of this method are automatically handled. They are intentionally raised to stop the update process should it be needed, not the entire program.

An error message will be printed to the console summarizing what occurred when this happens.

Raises ***ghau.errors.InvalidDownloadTypeError*** – an unexpected value was given to the download parameter of *ghau.update.Update*.

update_check()
Returns True if an update is available to download.

wl_files(**args*)
Add files to the whitelist. This protects any listed files from deletion during update installation. Each file should be a string referring to its name.

Parameters **args** (*str*) – list of files to protect.

wl_test()
Test the whitelist and output what's protected.

Useful for testing your whitelist configuration.

2.2 Reboot Functions

There are also a few added functions to make rebooting easier.

These functions build the reboot command for you to place in the reboot parameter of *ghau.Update*. They also provide ghau the ability to stop *Update Loops*.

Because of this, it is highly recommended you utilize these functions if you are rebooting.

ghau.**python**(*file: pathlib.Path*) → str
Builds the command required to run the given python file if it is in the current working directory.

Useful for building the command to place in the reboot parameter of *ghau.update.Update*.

This is the recommended way to reboot into a python file, as it adds an argument ghau detects to stop update loops. This will also ensure the file given is a python script.

See also: *ghau.update.exe()*, *ghau.update.cmd()*

Raises ***ghau.errors.FileNotScriptError*** – raised if the given file is not a python script.

ghau.**exe**(*file: str*) → str
Added for consistency with ghau.update.python.

Useful for building the command to place in the reboot parameter of *ghau.update.Update*.

This is the recommended way to reboot into an executable file, as it adds an argument ghau detects to stop update loops. This will also ensure the file given is an .exe file.

See also: *ghau.update.python()*, *ghau.update.cmd()*

Raises ***ghau.errors.FileNotExeError*** – raised if the given file is not an executable.

ghau.**cmd**(*command: str*) → str

Added for consistency with ghau.update.python.

Useful for building the command to place in the reboot parameter of *ghau.update.Update*.

This is the recommended way to reboot using a command, as it adds an argument ghau detects to stop update loops.

See also: *ghau.update.python()*, *ghau.update.exe()*

CHAPTER 3

Internal Reference

3.1 Update Module

class ghau.update.**Update**(*version: str*, *repo: str*, *pre_releases: bool = False*, *reboot: str = None*, *download: str = 'zip'*, *asset: str = None*, *auth: str = None*, *ratemin: int = 20*, *boot_check: bool = True*, *success_func=None*, *fail_func=None*, *pre_update_func=None*, *fail_args=None*, *success_args=None*, *post_update_func=None*, *pre_update_args=None*, *post_update_args=None*)

Main class used to trigger updates through ghau.

Parameters

- **version** (*str*) – local version to check against online versions.
- **repo** (*str*) – github repository to check for updates in. Must be publicly accessible unless you are using a Github Token.
- **pre-releases** (*bool, optional*) – accept pre-releases as valid updates, defaults to False.
- **reboot** (*str, optional*) – command intended to reboot the program after a successful update installs.
- **download** (*str, optional*) – the type of download you wish to use for updates. Either "zip" (source code) or "asset" (uploaded files), defaults to "zip".
- **asset** (*str, optional.*) – name of asset to download when set to "asset" mode.
- **auth** (*str, optional*) – authentication token used for accessing the Github API, defaults to None.
- **ratemin** (*int, optional.*) – minimum amount of API requests left before updates will stop, defaults to 20. Maximum is 60/hr for unauthorized requests, and 5000 for authorized requests.

restart()
Restarts the program

update()
Check for updates and install if an update is found.

All expected exceptions triggered during the run of this method are automatically handled. They are intentionally raised to stop the update process should it be needed, not the entire program.

An error message will be printed to the console summarizing what occurred when this happens.

Raises ***ghau.errors.InvalidDownloadTypeError*** – an unexpected value was given to the download parameter of *ghau.update.Update*.

update_check()
Returns True if an update is available to download.

wl_files(**args*)
Add files to the whitelist. This protects any listed files from deletion during update installation. Each file should be a string referring to its name.

Parameters **args** (*str*) – list of files to protect.

wl_test()
Test the whitelist and output what's protected.

Useful for testing your whitelist configuration.

ghau.update.**_do_update**(*local*, *online*)
Compares the given versions, returns True if the values are different.

ghau.update.**_find_release_asset**(*release: github.GitRelease.GitRelease*, *asset: str*) → str
Return the requested asset's download url from the given release. If no specific asset is requested, it will return the first one it comes across.

Raises

- ***ghau.errors.ReleaseAssetError*** – No asset by given name was found.
- ***ghau.errors.NoAssetsFoundError*** – No assets found for given release.

ghau.update.**_load_release**(*repo: str*, *pre_releases: bool*, *auth*) → github.GitRelease.GitRelease
Returns the latest release (or pre_release if enabled) for the loaded repository.

Raises

- ***ghau.errors.ReleaseNotFoundError*** – No releases found for given repository.
- ***ghau.errors.GithubRateLimitError*** – Hit the rate limit in the process of loading the release.
- ***ghau.errors.RepositoryNotFoundError*** – Given repository is not found.

ghau.update.**_run_cmd**(*command: str*)
Run the given command and close the python interpreter. If no command is given, it will just close. Does not currently support Windows OS.

ghau.update.**cmd**(*command: str*) → str
Added for consistency with ghau.update.python.

Useful for building the command to place in the reboot parameter of *ghau.update.Update*.

This is the recommended way to reboot using a command, as it adds an argument ghau detects to stop update loops.

See also: *ghau.update.python()*, *ghau.update.exe()*

ghau.update.**exe**(*file: str*) → str
Added for consistency with ghau.update.python.

Useful for building the command to place in the reboot parameter of *ghau.update.Update*.

This is the recommended way to reboot into an executable file, as it adds an argument ghau detects to stop update loops. This will also ensure the file given is an .exe file.

See also: *ghau.update.python()*, *ghau.update.cmd()*

Raises ***ghau.errors.FileNotExeError*** – raised if the given file is not an executable.

ghau.update.**python**(*file: pathlib.Path*) → str
Builds the command required to run the given python file if it is in the current working directory.

Useful for building the command to place in the reboot parameter of *ghau.update.Update*.

This is the recommended way to reboot into a python file, as it adds an argument ghau detects to stop update loops. This will also ensure the file given is a python script.

See also: *ghau.update.exe()*, *ghau.update.cmd()*

Raises ***ghau.errors.FileNotScriptError*** – raised if the given file is not a python script.

3.2 Files Module

ghau.files.**download**(*url: str*, *save_file: str*)
Download a file from the given url and save it to the given save_file.

Parameters

- **url** (*str*) – url of the file to download.
- **save_file** (*str*) – file to save the downloaded to.

ghau.files.**extract_zip**(*extract_path*, *file_path*, *wl: list*)
Extracts files from the given zip file_path into the given extract_path and performs cleanup operations.

Will not overwrite files present in the given whitelist.

Parameters

- **extract_path** (*str*) – path to extract the contents of the given zip to.
- **file_path** (*str*) – path of the zip to extract.
- **wl** (*list*) – whitelist to avoid overwriting files from.

3.3 Errors Module

exception ghau.errors.**FileNotExeError**(*file: str*)
Raised when the file given is not an executable.

exception ghau.errors.**FileNotScriptError**(*file: str*)
Raised when the file given is not a python script.

exception ghau.errors.**GhauError**
Base Exception class

exception ghau.errors.**GitRepositoryFoundError**
Raised when a git repository is detected.

exception ghau.errors.**GithubRateLimitError**(*resettime*)
Raised when exceeding GitHub's API rate.

exception ghau.errors.**InvalidDownloadTypeError**(*download*)
Raised when the download parameter value is not expected.

exception ghau.errors.**LoopPreventionError**
Raised when rebooting after an update, to prevent potential loops if user doesn't bump version number.

exception ghau.errors.**NoAssetsFoundError**(*release*)
Raised when an asset request returns no asset list

exception ghau.errors.**NotAFileOrDirectoryError**(*path*)
Raised when a path leads to neither a file nor a directory

exception ghau.errors.**ReleaseAssetError**(*release*, *asset_name*)
Raised when there is no asset found in the release by the requested name

exception ghau.errors.**ReleaseNotFoundError**(*repo: str*)
Raised when there are no releases found for the requested repository.

exception ghau.errors.**RepositoryNotFoundError**(*repo: str*)
Raised when Github request returns a 404

ghau.errors.**argtest**(*args: list*, *arg: str*)
Raises an error if the specified arg is found in the given args.

Used to determine if booting after an update installation.

Raises ***ghau.errors.LoopPreventionError*** – stops the update process if booting after an update installation.

ghau.errors.**devtest**(*root*)
Tests for an active dev environment.

Raises ***ghau.errors.GitRepositoryFoundError*** – stops the update process if a .git folder is found within the program directory

ghau.errors.**ratetest**(*ratemin: int*, *token=None*)
Tests available Github API rate.

Raises ***ghau.errors.GithubRateLimitError*** – stops the update process if the available rates are below the ratemin.

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

Python Module Index

g

Index

Symbols

A

C

D

E

F

G

I

L

N

P

R

U

W