manifest-checker Documentation

Release 0.0.1

Tony Flury

Mar 12, 2018

Contents

target http://manifest-checker.readthedocs.org/en/latest/?badge=latest

alt Documentation Status

The manifest checker is a suite of easy to use tools which can be used confirm that a directory tree has been correctly copied/deployed.

The checker builds a manifest file, with a checksum against each file. The manifest file is then copied/deployed along with the rest of the directory tree - and the suite can be used to check the destination tree against the manifest.

During the check phase - the suite will report against :

- files which have different checksums in the destination directory tree compared to manifest file - i.e. the files are likely to be different.
- files which are missing from the destination directory tree but which have a checksum within the manifest file - i.e. files which haven't been deployed at all.
- files which exist in the destination tree but which are missing from the manifest file - these record_extra files may impact the behaviour of the deployed code.

These are called exceptions - and using the right command options you can control which exception result in failures

The manifest checker has *default options* which are ideal for checking deployment of a Django deployment, but it can be configured either by command line to look in other places, or other file types.

If you use non default configuration options on the command line to create your manifest, then you will need to ensure that you use the same options when you execute the checker against your destination, to ensure that the right files are checked.

Note: Every care is taken to try to ensure that this code comes to you bug free. If you do find an error - please report the problem on :

- GitHub Issues
- By email to : Tony Flury

CHAPTER 1

Getting Started

1.1 Installation

Manifest checker can be installed using pip :

Manifest checker is compatible with both Python 2 (2.7 and later) and Python 3 (3.5 and later)

1.2 Simple Usage

Using the Manifest checker suite with the default arguments is easy :

To create the manifest

```
$ cd <tree root>
$ manifest create
```

This will create a manifest.txt - detailing all the source code files in the current directory tree, with a sha224 hash created for each file. You can then deploy/copy your source code to the destination and execute

```
$ cd <tree root>
$ manifest check
```

The check subcommand rescursively searches the directory tree under the current directory, and computes the hash for each file - and compares that against the computed has in the manifest file.

Note: Make sure you copy/deploy your manifest.txt file along with your source code - if you don't the `manifest check` will fail immediately.

By default the `manifest check` will detect and report on three different types of issue :

- missing files : files listed in the manifest file, but which don't exist in the local directory tree.

- record_extra files : files which are not listed in the manifest file, but which exist in the local directory tree.
- incorrect hashes : files where there is a discrepancy between the hash created for the local file, and the hash listed in the manifest.

CHAPTER 2

Command Line Options

The full command with options is :

```
manifest [-h] [-v, --verbose {0,1,2,3}]
                [-r,--root ROOT]
                [-o,--out REPORT-OUT]
                [-a,--hash {sha224,md5,sha1,sha256,sha384,sha512}]
                [-o, --out REPORT-OUT]
                [-f,--manifest MANIFEST]
                [-E, --resetExtensions] [-e EXTENSION]
                [-D, --resetDirectories] [-d DIRECTORY]
                [-t/-T]
                [-k/-K]

        create

        check   [-m/-M ]
                [-i/-I ]
                [-e/-E ]
                [-g/-G ]
                [-s, --summary]
```

```
General options for all commands

-h, --help :            Show the full help page

-v, --verbose :         The verbose reporting level from 0 or 1. The default is 1.
                        Level 0 : No output, except execution error messages.
                        Level 1 : Full output.

-a, --hash HASH         Choose an alternative hash algorithm. Can be one of
                        sha224, md5, sha1, sha256, sha384, sha512 - the default is␣
↪sha224.
                        On a standard default installation the above algorithms will␣
↪be
                        available. Some packages - such as OpenSSL will make more
                        algorithms available - check the help page of the manifest␣
↪suite
                        on your installation to check what is available locally.

-f,--manifest MANIFEST  The Manifest file to create or to check against
                        This option will create or use the manifest file with the␣
↪specified
                        name, rather than the default ``manifest.txt``

-r, --root ROOT         The root directory to create the manifest from, or check the␣
↪manifest against.

-E,--clearExtensions    Clears the internal manifest_write of file extensions.
                        Using this option without at least one corresponding
                        ``-e EXTENSION`` option will mean that no files will be
                        included in the manifest.

-e EXTENSION :          Add a file extension to the internal manifest_write.
                        The EXTENSION must start with a dot ``.`` character otherwise
                        an error will be raised.

-D, --clearDirectories  Clears the internal manifest_write of ignored Directories.
                        Using this option without at least on corresponding
                        ``-d DIRECTORY`` option will mean that all top level␣
↪directories
                        will be included in the manifest.

-d DIRECTORY :          Add a directory to the internal manifest_write of ignored top␣
↪level directories.
                        It is not an error if the name given does not exist

Output Flags
    -t/-T               Acts as a flag to enable or disable counting and reporting of␣
↪file extensions.
    -k/-K               Acts as a flag to enable or disable counting and reporting of␣
↪skipped files.

    For these flags the lowercase option enables the report, and an uppercase option␣
↪disables the report.
```

```
General options for the check command - these must occur after the check command.

Exception reporting flags

    -m/-M                Whether or not to report on mismatched files
    -i/-I                Whether or not to report on missing files
    -x/-X                Whether or not to report on record_extra files

    For these flags the lowercase option enables the report, and an uppercase option␣
↪disables the report.
    If a particular exception type is disabled then any occurrence of this type of␣
↪exception will not be reported
    and the command will not exit with a failure status. By default all of these␣
↪reports are enabled.

Other options

    -g/-G               Enables/disables grouping of exception reports files. If␣
↪grouping is
```

```
                        disabled (using -G) then exceptions are reported immediately␣
↪they are found,
```

CHAPTER 3

Configuration file settings

By default the config file is named 'manifest.cfg' and if that file exists the configuration options there are used before those specified on the command line. The values specified in the config file are used for both create and check commands; and thus using a config file can be very useful in ensuring consistency without typing complex command lines.

3.1 Sections

The config file has a number of sections - none are mandatory - although an empty config file is pointless.

Each section has the following format :

```
[title]

<optionline>
```

Sections can contain one or more *optionline*, and the exact form of these option lines depends on the section.

Leading and trailing whitespace within the section is ignored.

Lines within the config file that begin with a # are ignored - and can be used as comments

3.1.1 Manifest Section

Note: This section is equivalent to the *-m/–manifest*, *-h/–hash* and *-r/–root* command line options

The format of this section is :

```
[manifest]

manifest = <file name>
```

```
hash = <hash name>
root = <directory path>
```

where the `<option>=<value>` line can be provided for each option - if an option is repeated then the last value given is used. Available options are :

manifest The name of the manifest file. Equivalent to the `-m/--manifest` command line option. If not provided defaults to `manifest.txt`

hash The name of the hash algorithm to use. The name does not need quotes. Equivalent to the `-h/--hash` command line option. Defaults to using sha224 hash.

root The root directory to use - so that all files under the root will be analysed and catalogued. Equivalent to the `-r/--root` command line option. This can be either a relative or absolute path (although it makes more sense to be relative) Defaults to '.'

Spaces and tabs around the = are optional.

3.1.2 Extensions Section

Note: This section is equivalent to the -E/–clearExtensions and the +e/–add_extension and -e/–rm_extension command line options

The format of this section is :

```
[extensions]

= <new extension list>
+ <additional extension list>
- <remove extension list>
```

The section can have one or more of each possible type of option line - and they can appear in any order. These lines are processed in order to build an list of file extensions which are to be analysed.

= <new extension list> Reset the extension list from it's current value to just those items in the `<new extensions list>`; Equivalent to `-E` followed by one or more `-e` command line options

+ <additional extension list> Add those extensions listed in `<additional extension list>` to the extension list to be used; Equivalent to a one or more `+e` command line options

- <remove extension list> Remove the extensions list in the `<remove extension listt>` from the extension list to be used; Equivalent to a one or more `-e` command line options.

Each of these extension lists are comma separated, with each file extension in the list starting with a **.** (a dot). Spaces and tabs around the individual file extensions are ignored

Spaces after =, +, and − are optional

Lines in this section which start with any character other than =, +, − and # will be reported as an error.

Warning: Within this section ordering matters. The lines are executed strictly in the order they appear; so for instance :

- Using the + and − operators followed by an = operator will result in the changes implemented by the + & − operators being pointless. An error will not be raised.

- Having multiple =' operations within the section will result in only the latest one having an effect. An error will not be raised.

Examples

Using the + operator

Assuming that the default extension list is `.py, .html, .txt, .css, .js, .gif, .png, .jpg, .jpeg` then :

```
[extensions]

+ .htm, .cfg, jsx
```

Would result in the extension list of `.py, .html, .htx, .txt, .css, .cfg, .js, .jsx, .gif, .png, .jpg, .jpeg` being used for both create and check operations.

Note: It is not an error to use the + to add an extension that already exists in the current list.

Using the - operator

Assuming that the default extension list is `.py, .txt, .html, .css, .js, .gif, .png, .jpg, .jpeg` then :

```
[extensions]

- .html, .js, .css
```

Would result in the extension list of `.py, .txt, .gif, .png, .jpg, .jpeg` being used for both create and check operations.

Note: It is not an error to use the − to attempt to remove an extension that doesn't exist in the current list.

Using the = operator

Assuming that the default extension list is `.py, .txt, .html, .css, .js, .gif, .png, .jpg, .jpeg` then :

```
[extensions]

= .html, .js, .css
```

Would result in only files with extensions of `.html, .js` and `.css` only being used for both create and check operations.

3.1.3 Ignore Directories Section

Note: This section is equivalent to the -D/–clearDirectory and -d/–ignoreDirectory command line options

3.1.4 Filter Section

Note: This section is equivalent to the -f/–filter command line option

3.1.5 Reports Section

Note: This section is equivalent to the -v/–verbose, -k/-K, -t/-T, -m/-M, -i/-I, -x/-X and -g/-G command line options

CHAPTER 4

Extra Information

This page contains record_extra information not included in any of the other pages

CHAPTER 5

Defaults

By default the manifest suite with the default options as above will create a manifest for a Django projects (typically these projects exist as a nested set of directories which containt various types of source code, including python css, javascript and html files (and templates), as well as image files (gif, jpeg, png etc). A Django project will also contain directories and files which aren't part of the deployment, which don't need to be included in the integrity check by default.

If you don't use Django, you can still use the manifest suite to check your copies/deployments. Various options command line and Configuration File options exist to make the manifest suite usable including :

- check for different files types
- ignoring different directories

File Extension .py, .html, .txt, .css, .js, .gif, .png, .jpg, .jpeg

Top Level directories which are ignored : static, env, htmlcov, media,build, dist, docs

Default manifest file 'manifest.txt'

Default hash/checksum algorithm. 'sha224' if 'sha224' in hashlib.algorithms else hashlib.algorithms[0]

Reporting options skipped files and file extension totals are not reported on, but they can be included by using -k and -t options on the manifest command.

Checking options mismatched, missing and record_extra files are included in the reports. These checks can be disabled by using the -M, -I and -X options respectively.