naive puppets marching npm private modules npm Enterprise documentation blog

find packages

n up or log in

grunt-ftp-deploy public

Deployment over FTP

This is a **grunt** task for code deployment over the *ftp* protocol.

These days *git* is not only our goto code management tool but in many cases our deployment tool as well. But there are many cases where *git* is not really fit for deployment:

- we deploy to servers with only *ftp* access
- the production code is a result of a build process producing files that we do not necessarily track with *git*

This is why a *grunt* task like this would be very useful.

For simplicity purposes this task avoids deleting any files and it is not trying to do any size or time stamp comparison. It simply transfers all the files (and folder structure) from your dev / build location to a location on your server.

# Getting Started

This plugin requires Grunt `~0.4.0`

If you haven't used **Grunt** before, be sure to check out the **Getting Started** guide, as it explains how to create a **Gruntfile** as well as install and use Grunt plugins. Once you're familiar with that process, you may install this plugin with this command:

```
npm install grunt-ftp-deploy --save-dev
```

and load the task:

`npm i grunt-ftp-depl`

zonak published 7 mo…

0.1.10 is the latest of 21 rel…

github.com/zonak/grunt-f…

MIT license

Collaborators

Stats

199 downloads in the last day

1,088 downloads in the las…

3,857 downloads in the las…

One open issue on GitHub

One open pull request on …

Keywords

gruntplugin

Dependencies (5)

async, lodash, grunt, jsftp,

```
grunt.loadNpmTasks('grunt-ftp-deploy');
```

# Usage

To use this task you will need to include the following configuration in your *grunt* file:

```
'ftp-deploy': {
  build: {
    auth: {
      host: 'server.com',
      port: 21,
      authKey: 'key1'
    },
    src: 'path/to/source/folder',
    dest: '/path/to/destination/folder',
    exclusions: ['path/to/source/folder/**/.DS_St
  }
}
```

Please note that when defining paths for sources, destinations, exclusions e.t.c they need to be defined having the root of the project as a reference point.

The parameters in our configuration are:

- **host** - the name or the IP address of the server we are deploying to
- **port** - the port that the *ftp* service is running on
- **authPath** - an optional path to a file with credentials that defaults to `.ftppass` in the project folder if not provided
- **authKey** - a key for looking up credentials saved in a file (see next section). If no value is defined, the `host` parameter will be used
- **src** - the source location, the local folder that we are transferring to the server

prompt

Dependents

spm-demo02, spm-ftp, frank-styleguide, grunt-workflow

NewsCred is hiring. View more…

- **dest** - the destination location, the folder on the server we are deploying to
- **exclusions** - an optional parameter allowing us to exclude files and folders by utilizing grunt's support for minimatch. The `matchBase` minimatch option is enabled, so `.git*` would match the path `/foo/bar/.gitignore`.
- **forceVerbose** - if set to `true` forces the output verbosity.

# Authentication parameters

Usernames and passwords can be stored in an optional JSON file (`.ftppass` in the project folder or optionaly defined in`authPath`). The credentials file should have the following format:

```
{
  "key1": {
    "username": "username1",
    "password": "password1"
  },
  "key2": {
    "username": "username2",
    "password": "password2"
  }
}
```

This way we can save as many username / password combinations as we want and look them up by the `authKey` value defined in the *grunt* config file where the rest of the target parameters are defined.

The task prompts for credentials that are not found in the credentials file and it prompts for all credentials if a credentials file does not exist.

IMPORTANT: make sure that the credentials file uses double quotes (which is the proper *JSON* syntax) instead of single quotes for the names of the keys and the string values.

# Dependencies

This task is built by taking advantage of the great work of Sergi Mansilla and his **jsftp** *node.js* module and suited for the **0.4.x** branch of *grunt*.

# Release History

- 2015-02-04 v0.1.10 An option to force output verbosity.
- 2014-10-22 v0.1.9 Log successful uploads only in verbose mode.
- 2014-10-13 v0.1.8 Allow empty strings to be used as login details.
- 2014-09-03 v0.1.7 Restructured the code deailing with the authentication values to address some issues.
- 2014-08-20 v0.1.6 Bug fix with the modules updates.
- 2014-08-20 v0.1.5 Refresh of versions of used modules.
- 2014-07-28 v0.1.4 Added a `authPath` configuration option.
- 2014-05-05 v0.1.3 Added warning if an `authKey` is provided and no `.ftppass` is found.
- 2013-11-22 v0.1.1 Added compatibility with `grunt` *0.4.2* and switched to `jsftp` *1.2.x*.
- 2013-08-26 v0.1.0 Switched to `jsftp` *1.1.x*.

You Need Help

Documentation

Support / Contact Us

Registry Status

Website Issues

CLI Issues

Security

About npm

About npm, Inc

Jobs

npm private modules

Blog

Twitter

GitHub

Legal Stuff

Code of Conduct

Package Name Disputes

npm License

Privacy Policy

Reporting Abuse

Other policies

npm loves you