Pask Documentation

Release 0.1.0

Daniel Jay Haskin

Oct 31, 2017

Contents

CHAPTER 1

Why Pask?

It's PAckages and tASKs.

Pask is given a list of Pask packages and is told either to install the packages or run a specific task which is associated with all the installed packages. Packages are installed relative to a project (build or deploy) root.

Here's the real power of pask: not only will the packages be installed in the order that they appear in the list, but will also run the tasks associated with the packages in the order that they appear in the given list.

Imagine running `pask run deploy`, `pask run blue-green`, or `pask run compile`, and the task associated with each package runs in order.

Well, at least *I* think it's pretty cool. :)

CHAPTER 2

Get Pask

- Using Golang: `go get github.com/djhaskin987/pask`
- Binaries are downloadable from GitHub.

CHAPTER 3

Quickstart

1. Create a pask package called `a` with files in it:

```
mkdir pkga
cd pkga
mkdir -p pask/tasks
cat > pask/tasks/mytask << MYTASK
#!/bin/sh

echo compile task A
MYTASK
chmod a+x pask/tasks/mytask
mkdir files
touch files/a
tar cJf ../a-1.0.tar.xz $(ls -1A)
cd ..
```

2. Do the same thing to create a package called `b`:

```
mkdir pkgb
cd pkgb
mkdir -p pask/tasks
cat > pask/tasks/mytask << MYTASK
#!/bin/sh

echo compile task B
MYTASK
chmod a+x pask/tasks/mytask
mkdir files
touch files/b
tar cJf ../b-1.3.tar.xz $(ls -1A)
cd ..
```

3. Write out a `pask/spec.yml` file relative to the root of your project:

```
packages:
  - name: a
    version: 1.0
    location: file://<path-to-where-a-was-made>/a-1.0.tar.xz
  - name: b
    version: 1.3
    location: file://<path-to-where-b-was-made>/b-1.3.tar.xz
```

Pask also accepts HTTP and HTTPS URLs.

4. Have pask install the contents of the packages:

```
pask install
find .
```

5. Run `pask run mytask`:

```
pask run mytask
```

CHAPTER 4

Command Reference for Pask

Root CLI Reference

The basic idea is that a pask package is simply an archive that contains files and tasks. These archives can be installed and their tasks can be run in order of a list given to pask when it starts its run. All packages will be installed and run relative to the project root directory, presumably of a build or deployment.

If you were to run `pask help` today you might see this output:

```
Install files and run tasks. Make your build
less painful and more fun!

Usage:
  pask [command]

Available Commands:
  help        Help about any command
  install     Install or update a package or packages
  run         Run a packaged task

Flags:
  -p, --base string     Base project path (default "./")
  -c, --config string   config file (default is $HOME/.pask)
  -h, --help            help for pask
  -s, --spec string     Pask spec file

Use "pask [command] --help" for more information about a command.
```

Explanation of root command options:

- `-p BASE`, `--base BASE`: Specify base project path. This is the path relative to which pask will try to install files and/or run tasks. By default this is the present working directory (`./`).
- `-c FILE`, `--config FILE`: Specify config file for pask. This file is in TOML format. By default it can be found at `~/.pask`. This file currently only allows the operator to specify values for the base project path

and spec file location relative to that path, using the `base` and `spec` file keys. Arguably configuring these is useless, but the option to use a configuration file is there to facilitate growth of pask and its configuration values for future versions of pask.

- `-s SPEC, --spec SPEC`: Specify pask spec file location. By default this location is `<base>/pask/spec.yml`. This file is intended to be committed to project source code and specifies, under the `package` key, the list of packages that pask will use for its operations. Each entry in the list under the `package` key itself has at least three keys: `name`, `version`, and `location`. Example:

```
packages:
  - name: a
    version: 1.0
    location: file://<path-to-where-a-was-made>/a-1.0.tar.xz
  - name: b
    version: 1.3
    location: file://<path-to-where-b-was-made>/b-1.3.tar.xz
```

CLI Reference for `pask install`

The command `pask install` has no command-specific options in the current version.

When *pask install* is run, this list of packages is installed in order of the list. Each package is simply a XZ-compressed tarball is downloaded and unpacked relative to the build project root. If any files are found in the archive under the folder *./pask*, those files are installed under *./pask/packages/<package-name>/<package-version>/* relative to the build project root.

CLI Reference for `pask run`

The command `pask run` has no command-specific options in the current version.

When *pask run <task>* is run, the file *./pask/packages/<package-name>/<package-version>/tasks/<task>* is assumed to be executable. It is run with no arguments for each package in the list found in the file *./pask/spec.yml* relative to the build project root. Any environment variables set when pask is run are passed through. If, for any of the packages in the list of the spec file, the named task does not exist, is not executable, or exits abnormally, an error is printed and pask stops. The task for each package in the list are run in the order that the packages found in the list.

Compatibility with Degasolv

Pask was built to work natively with Degasolv, version 1.10.0 or greater. By using Degasolv to resolve dependencies between pask packages, you get a complete package management system for your builds.

When you run `degasolv resolve-locations`, Degasolv will print out a list of packages in order of "dependance". Packages with many dependants appear first in the list, while packages which have many dependencies appear last in the list.

The operator can create a valid spec for Pask by using Degasolv's `--output-format` CLI option, like so:

```
cd <build-project-dir>
mkdir -p ./pask
degasolv resolve-locations --output-format json > pask/spec.yml
```

The Pask spec can then serve as Degasolv's lock file for the build project. Then, Pask would run tasks associated with these tasks in order of dependance (more dependers = installed first, more dependencies = installed last), and will also run the tasks associated with those packages in order of dependance.

CHAPTER 5

Changelog

All notable changes to this project will be documented here.

The format is based on Keep a Changelog and this project adheres to Semantic Versioning.

Unreleased

Added

Changed

Fixed

0.1.0

Added

- Added *pask install*
- Added *pask run*
- Added root CLI

CHAPTER 6

Indices and tables

- genindex
- modindex
- search