silk-deployment Documentation

Release latest

August 12, 2015

Contents

Silk is a Fabric based tool for setting up Python WSGI apps on what I like to call the SNUG stack:

- Supervisord for starting processes and keeping them alive.
- Nginx for proxying between your WSGI app and the big bad web.
- Ubuntu as the OS of choice, enabling resolution of system dependencies with apt. Debian might work as well but hasn't been tested.
- Gunicorn for serving your WSGI app.

(I suppose it could also be the GUNS stack but that sounds far less friendly.)

CHAPTER 1

Key Features

- Deploy your site to one or more servers with a single command ('silk push').
- Automatic configuration of Nginx, Supervisord, and Gunicorn to get your site running.
- Isolation of each site into a separate Virtualenv
- Support for differing app config depending on which role you deploy to (a different DB in staging than production, for example).

CHAPTER 2

Installation

Use pip:

```
pip install silk-deployment
```

You can also install the current development version straight from bitbucket:

```
pip install hg+http://bits.btubbs.com/silk-deployment#egg=silk-deployment
```

CHAPTER 3

Commands

(Almost) all of the commands below require that you specify a role name, like 'silk dosomething -R dev'.

Commands can generally be run from the site root directory or any subdirectory of it.

3.1 push

```
silk push -R rolename
```

This command is the main reason for Silk's existence. It does the work required to get your app running on a host (or set of hosts) given the configuration specified in site.yaml and the selected role .yaml file. 'push' does the following:

1. SSHes to the remote server(s) specified in the role config.
2. Creates a zipped up rollback archive of the old site if there's one there already.
3. Creates a virtualenv for the site.
4. Installs apt and python dependencies.
5. Copies the site from your local machine to a temporary directory on the remote server.
6. Writes config file includes for nginx and supervisord.
7. Moves your code from the temp dir into its production location (/srv/<sitename> by default).
8. Tells nginx and supervisord to reload their configs.

3.2 rollback

```
silk rollback -R rolename
```

This command is for when you have those 'OMG I BROKE THE SITE' moments. It will SSH to the push_hosts specified in your role file and restore the most recent archive of the site. Silk keeps 3 rollback copies of your site, so you could potentially run 'silk rollback' 3 times to go back to the state from 3 deployments ago.

3.3 run

```
silk run -R rolename
```

This command runs the site from the local machine, on port 8000. (Nothing is pushed of copied.) Static directories listed in the *static_dirs* section of site.yaml will also be served. (CherryPy is used for this magic.)

3.4 install_server_deps

```
silk install_server_deps -R rolename
```

When you get a shiny new server with that fresh Ubuntu smell, it needs just a tiny bit of setup before it will know how to serve silk-deployed sites. This command does that. It installs nginx and supervisord, and gives each of them a wildcard include in their configs for loading from /srv/<sitename>/conf.

3.5 pip_deps

```
silk pip_deps
```

This command wraps 'pip install' to install all of the python packages listed in requirements.txt into your local python environment. It's handy for grabbing all the dependencies when you're working with a new virtualenv on an existing project.

3.6 create_virtualenv

```
silk create_virtualenv
```

Creates a virtual environment for the app to be deployed.

3.7 configure_nginx

silk configure_nginx

Configures nginx

3.8 switch_nginx

silk switch_nginx

Reloads nginx config

3.9 configure_supervisor

silk configure_supervisor

Configures supervisor

3.10 start_process

silk start_process

Starts process

3.11 stop_other_versions

silk stop_other_versions

Kills other running proccesses of this app

3.12 skel

```
silk skel sitename
```

Creates a directory with a basic Silk file and directory structure.

CHAPTER 4

Layout

A silk-enabled project should be layed out something like this:

```
mysite.com
-- deps.yaml
-- fabfile.py
-- membrane.py
-- roles
|   -- dev.yaml
|   -- staging.yaml
|   -- production.yaml
-- site.yaml
-- my-django-project
```

Some of those files/folders are required, other are optional:

4.1 Required

1. site.yaml - This is the main config file (comparable to app.yaml in Google App Engine)
2. deps.yaml - Lists Python packages, Ubuntu apt packages, and apt build dependencies that need to be installed on the server running your site.
3. fabfile.py - A Fabric-compatible fabfile that imports Silk's Fabric functions.
4. roles/*.yaml - One or more 'role' files that contain the config to be passed into your app depending on the deployment context.

All of the required files will be created for you with the 'silk skel' command.

4.2 Optional

1. membrane.py - For Django projects it's nice to have a little shim to expose the project as a WSGI app. I like to call mine membrane.py. You can use whatever you like, or nothing at all, depending on your setup.
2. my-django-project - Silk isn't restricted to Django; any valid WSGI app on your Python path should be servable. But for Django projects I think it makes sense to stick them right there.