webscaff Documentation

Release 1.0.0

Igor ‘idle sign‘ Starikov

Dec 09, 2022

Contents

https://github.com/idlesign/webscaff

CHAPTER 1

Description

Remote scaffolding and orchestration for web applications

1.1 Used stack

- **Debian**-based OS (Ubuntu 18.04, 20.04 tested) as a basis.
- **Git** for source code updates.
- **Systemd** to securely run your webservice.
- **PostgreSQL** as a reliable DBMS.
- **uWSGI** as a platform (handling routing, static, background tasks, etc.).
- **Python 3** to cover your needs.
- **Django** as a rich and solid webframework.

And also:

- **Certbot** integration for free HTTPS certificates (webroot plugin).
- SSH Agent forwarding for project code updates on remote using keys from your system.

CHAPTER 2

Requirements

1. Python 3.7+
2. makeapp 1.3.0+ (to streamline project initialization)

CHAPTER 3

Table of Contents

3.1 Quickstart

First steps required for a `webscaff` managed project bootstrap are not really `webscaff` related but rather about setting up thirdparties, so be patient.

3.1.1 1. SSH key

If you don't have one yet, please generate it, because you'll need it in further steps.

Print out a public key part to console using (example for a default key name):

```
$ cat .ssh/id_rsa.pub
```

3.1.2 2. Code hosting

Pick a hosting for your Git repository (e.g. GitHub, GitLab) public or private.

To access your repository you'll use SSH key, so you need to use one from step 1 and register its public part on code hosting service.

Create a project there and **remember SSH address** (*Clone or download* green button on GitHub). We'll need it soon. Example: `git@github.com:idlesign/myproject.git`

3.1.3 3. Project hosting

You'll probably need a VPS hosting for your webservice, so you need to pick one (e.g. Yandex Cloud, DigitalOcean).

There make sure that SSH auth implies using SSH keys, but not password. So you'll need to upload your public key part from step 1, and choose a name. For convenience it's advised to *use the same name you're logged in*.

Remember your IP address. We'll need it soon.

3.1.4 4. Domain name

You can use a free domain name or register a paid one via a provider. Do not forget to target your domain name (DNS settings) to your IP address (see step 3).

Remember your domain name. We'll need it soon.

3.1.5 5. Local bootstrap

`webscaff` is configurable but for convenience it largely relies on various conventions.

You may quickly create a basic project skeleton using `webscaff` template of `makeapp` - https://pypi.org/project/makeapp:

```
$ makeapp new myproject -d "My webscaff project" -t webscaff /home/some/here
```

This will run `myproject` skeleton creation in `/home/some/here`.

> **Warning:** It is advised to *not to use* the same name you're logged in as a project name to avoid confusion.

You'll be asked for information you gathered in previous steps.

`makeapp` will also ask you for a code repository address (step 2).

Note: If you preferred to push your repository manually do not forget to do it before going further.

3.1.6 6. Remote initialization

Now `cd` into your project directory (it'll contain `wscaff.yml`, for example `/home/some/here`).

Let's make sure configuration is success:

```
$ webscaff info
```

If everything is fine and your VPS is up and running this command will print out some basic information about your remote system.

Now we're ready for remote system initialization:

```
$ webscaff run.initialize
```

The initialization process is done in two steps. Please follow the instructions.

If you're lucky after this step you'll be able to access you webservice using your domain name both via HTTP and HTTPS.

3.1.7 7. Basic usage

1. Develop your code locally. Use virtual environment from *venv/* directory.
2. Push a new version to remote repository.
3. Rollout that version on server right from the repository:

```
$ webscaff rollout
```

This will get code from the repository, gather Django static files, apply DB migrations and reload uWSGI.

4. If you want to get a backup (user media, DB, certificates) locally:

```
$ webscaff run.backup
```

Backup archives are stored in *state/dumps* directory.

5. Other basic commads:

```
; Put project into maintenance mode -
; display static page for users.
$ webscaff off
; Quit maintenance mode.
$ webscaff on

; Get project service status.
$ webscaff status
; Restart project service.
$ webscaff restart

; Show current project log.
$ webscaff log
```

5. Run *webscaff* without arguments to get available commands list.

3.2 Q & A

3.2.1 1. Getting *Private key file is encrypted* `paramiko` error

Try the following:

- Upgrade `paramiko` to the latest version.
- https://unix.stackexchange.com/a/12201/32880
- https://askubuntu.com/a/853578/33408

3.2.2 2. How to upgrade OS

This roughly boils to:

1. Restrict service access.

   ```
   $ webscaff off
   ```

2. Backup data.

   ```
   $ webscaff run.backup
   ```

   Note: If you use a cloud service this may also prove disk images/backups.

3. Start OS upgrade procedure.

   ```
   $ webscaff sys.utils.os-upgrade
   ```

4. Upgrade PostgreSQL cluster.

   ```
   $ webscaff sys.pg.cluster-upgrade X
   ```

   Note: Where `X` is a previous PG version to upgrade from, e.g `12`.

5. Cleanup from unused packages.

   ```
   $ webscaff sys.apt.autoremove
   ```

6. Upgrade virtual environment to use a new system Python version.

   ```
   $ webscaff run.venv.populate --reset-py
   ```

7. Restart your service.

   ```
   $ webscaff restart
   ```

   Note: Use `$ webscaff log` in a separate terminal window to see how the restart is happening.