vincenttam.gitlab.io

Staticman API Hosting 2018

Vincent Tam

8-10 minutes

Goal

To host an instance of Staticman v3 server on Heroku.

This post involves *server-side* setup of the commenting system. If you simply want to have a taste of this system on GitLab, you may

- try my demo GitLab Page, and/or
- configure your GitLab repo (for your static web site) with my API instance. See the previous post in this series for a tutorial.

I try to address some concerns about this API service in the introduction of this series to keep this page focused on the *technical* aspects of my customizations against `staticman/dev` branch.

Disclaimer: This post involves creating and transferring confidential info. Store them in a secure place to unavoid unauthorized access to your server, GitHub and/or GitLab accounts. Like *all* community guides, there's *no* guarantee nor liability for any loss of data.

Overview

1. Grab Staticman as a Node.js app from its GitHub repo.
2. Fill in an RSA private key and your personal access token for GitHub/GitLab in the site config JSON file.
3. Deploy the app to your web app host.

Stage 1: prerequisites

1. A clone of Staticman:

```
$ git clone https://github.com/eduardoboucas/staticman
```

2. Access to a web app host. I'm using a free Heroku account.

   If you decide to set up your *own* server, please:

1. understand all the (Linux, networking, database, etc) *basics* (to open up only necessary services/ports to external users).
2. aware of the (fire) safety requirements (as your server is usually online 24 / 7)
3. calculate the annual electricity fee from the workstation's power consumption and compare it with the cost of renting a server.

   If you're using Heroku, download Heroku's CLI. On Debian-based systems, installing it as a snap software should be the *easiest* option.

3. An RSA *private* key (`ssh-keygen` *doesn't* work here.)

   ```
   $ openssl genrsa -out key.pem
   ```

   This is needed for the encryption of secrets so that they can be published (to a remote Git repo).

4. GitLab and/or GitHub personal access token(s) of a *dummy* account (*not* your personal account)

- GitLab: configured with scopes `api` and `read_repositories`.
- GitHub: with write access to user's repo

  You may create a new (personal, work or organization) account for that.

  It's safer to grant access to *some* personal repo's to a bot than to expose them *all* to the API.

  Stage 2: project config

  If you *can* securely copy files (using `scp`, `rsync` with SSH remote, etc), you may directly include your confidential info your JSON config file `config.production.json`.

  ```
  {
    "gitlabToken": "YOUR_GITLAB_TOKEN",
    "githubToken": "YOUR_GITHUB_TOKEN",
    "rsaPrivateKey": "-----BEGIN RSA PRIVATE KEY\n-----
  ```

```
YOUR_KEY-----\nEND RSA PRIVATE KEY-----",
  "port": 8080
}
```

Nicholas Tsim, the developer of Staticman's v3 GitLab support, suggests using a modern text editor to edit `rsaPrivateKey`. (e.g. Sublime Text 3)

Some free web app host (e.g. Heroku) *doesn't* provide such access. To *publish* your JSON config file `config.production.json`, *store your secrets with environment variables*, and access them with `process.env` in your JSON file and Node.js code. (I've learnt this skill thanks to Flying Grizzly's Staticman server guide.)

1. Log in the Heroku CLI.

   You'll be prompted to enter user name and the password.

2. Change to the directory of the cloned repo.

3. Create file `Procfile` *at root level* containing one simple line: `web: npm start`.

4. Create your project on Heroku (for storing environmental variables).

   ```
   $ heroku create <app_name>
   ```

   If `app_name` is omitted, then the system will attribute a random alphanumeric-hyphenated pronounceable name.

5. Set your environment variables for Staticman: `GITHUB_TOKEN`, `GITLAB_TOKEN`, `RSA_PRIVATE_KEY` and `NODE_ENV`.

   ```
   $ heroku config:set key=value
   ```

   For example,

   ```
   $ heroku config:set NODE_ENV="production"
   ```

   During the setup, I find the most difficult point is to *correctly* pass the *multi-lined* RSA *private* key to the shell variable. I tried copying and pasting the string

   ```
   -----BEGIN RSA PRIVATE KEY\n-----YOUR_KEY-----\nEND RSA PRIVATE KEY-----
   ```

   in the shell emulator and issuing the `heroku` command with the pasted string wrapped in double quotes.

```
$ heroku config:set RSA_PRIVATE_KEY="....\n.....\n......."
```

However, `heroku logs --tail` kept complaining that the "RSA key is *not* in the correct format". I almost wanted to give the whole thing up. Last Friday, after a day of fruitless searching, I finally came up with a solution.

```
$ heroku config:set RSA_PRIVATE_KEY="$(cat key.pem)"
Setting RSA_PRIVATE_KEY and restarting ⬢ staticman3... done,
v7
RSA_PRIVATE_KEY: -----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEAvUhs53CsnRfovlgZOg0rcC/lUPXUV2rs9gK3oU+ICzamJX9B
...
EPvidr9qCEQxITeOVj8gZ3MjyyRohxkiqbcSI9ceqh6Ellp82R4NEA==
-----END RSA PRIVATE KEY-----
```

The command in the bracket is executed first. The output (the RSA *private* key) is captured inside the double quotes `"..."`. To make the whole *multi-line* string JSON-friendly, the static function `JSON.stringify` is called.

6. Create your `config.production.json`.

```
{
  "gitlabToken": process.env.GITLAB_TOKEN,
  "githubToken": process.env.GITHUB_TOKEN,
  "rsaPrivateKey":
JSON.stringify(process.env.RSA_PRIVATE_KEY)
}
```

On Heroku, the `port` is *dynamically* attributed to each web app, so it's *useless* to set this parameter. ~~However, I just leave it there and there's no complaint from the machines. You may try omitting that. I would be happy to know if that also works well.~~

For a complete list of API configuration variables, you may consult Staticman's official API documentation. This is useful when you need to incorporate a *third-party* service (e.g. Akismet, Mailgun).

Optional: If you're running on a self-hosted GitHub/GitLab, it's possible to set up `git***BaseUrl`, which defaults to `https://git***.com/`.

7. Create branch `production` from remote branch `dev`. (Staticman v3 is still under

testing and development, but its GitLab support has already been merged against branch `dev`.)

```
$ git checkout -b production origin/dev
```

8. Add `config.production.json` to the exception of `.gitignore`.

```
$ echo "!config.production.json" >> .gitignore
```

9. Commit the changes against the newly created branch.

```
$ git add config.production.json Procfile .gitignore
$ git commit -m "Set up Staticman v3 for deployment to
Heroku"
```

Stage 3: deployment to server

The basic idea is to *securely* transfer the configuration and application files to a web app host, on which `npm start` will be run to start the service. Part of this section is specific to Heroku. You may adapt it according to your own needs.

Push your local `production` branch against the remote `master` branch. (Builds are *limited to `master` branch only*.)

```
$ git push heroku production:master
```

You may refer to the source code of my API instance for details.

If you can transfer your confidential files in a secure way (say, SSH to a self-hosted AWS EC2 Linux instance), you may try

```
$ rsync -auvz ./* <username>@<ssh-remote>:/path/to/your
/directory/
```

That will simplify the file transfer.

- `-a`: preserve the file attributes, access time, etc
- `-u`: only update file if the target file's `-mtime` is less recent than the source file.
- `-v`: verbose, for inspecting our work
- `-z`: compress the data, useful in case of large file transfer

Then on your server, run

Set environment variable `NODE_ENV` to be `production`.

```
$ export NODE_ENV="production"
```

Finally, run the server.

```
$ npm start

> staticman@3.0.0 prestart /app
> if [ ! -d node_modules ]; then npm install; fi



> staticman@3.0.0 start /app
> node index.js

Staticman API running on port 8080
```

Some port forwarding is needed so that the developers using this API *don't* need to key the port number. However, I *don't* know how to do so.

References

1. Staticman PR #219
2. Flying Grizzly's server setup guide
3. Heroku's Node.js deployment guide
