Inboxen Documentation

Release deploy-2020-08-14T22-38-11-3-gf407dfe

Jessica Tallon & Matt Molyneaux

Aug 18, 2020

CONTENTS:

CHAPTER

ONE

GETTING STARTED

There are too many options and too much difference between various distributions (e.g. Fedora, Debian, any one of the BSDs) for us to docment from start to finish how to deploy Inboxen for a majority for people wishing to deploy this software for themselves. We have often been frustrated by projects for listing Debian package names when I'm using Fedora, or listing outdated package names entirely. As such, you're expected to be familiar with the following:

- A webserver and deploying WSGI applications to that server.
- A mail-server and forwarding mail to another server.
- Setting up a message queue with either RabbitMQ or Redis.
- Managing a cache like Memcache.

We understand that this will make deploying Inboxen a far more daunting task, but we find that preferable to masses of documentation of unknown quality.

1.1 Requirements

To use Inboxen, you'll need the following:

- Python 3, including the following:
  - Development headers (usually in a package called `python3-devel` or `python-dev`. If you installed via `brew` or from source, these headers will already be there.
  - `pip`
  - `virtualenv`
- Git
- NodeJS and `npm`
- GCC
- GNU Make
- PostgreSQL
  - `pg_config` needs to be in your `$PATH`

1.2 Setup

Let's get started!

```
$ git clone https://github.com/Inboxen/Inboxen.git
$ cd Inboxen
$ virtualenv-3 env
$ . env/bin/activate
(env) $ pip install -r requirements.txt
(env) $ npm install
(env) $ touch inboxen.config
```

At this point we should add some basic configuration. Open `inboxen.config` with your favourite text editor and add the following:

```
[general]
# some_random_string should be replaced by an actual random string, it is
# used for various cryptographic functions and should be kept secret
secret_key: some_random_string
```

Now we've got some configuration, let's finish the setup:

```
(env) $ ./manage.py migrate
(env) $ make static
(env) $ mkdir -p run logs
(env) $ make salmon-start celery-start
```

Finally, there are some external services that you will need to configure:

- Your WSGI daemon needs to be configured to use your virtualenv (found in `env/`) and use the script `inboxen/wsgi.py`. You should also set your current working directory to same folder the contains `setup.py` Refer to your WSGI daemon's documentation for details on how to do that.
- Your webserver or your WSGI daemon (depending on your configuration) should serve `/static/` from `static_content`.
- Your mailserver should forward mail to `localhost:8823` via SMTP

1.3 Additional configuration

There are a number of other configuration options that you can use. See *The Setting File* for all available settings.

1.3.1 Additional Python packages

You can also install additional Python packages to enable certain features. For example, let's say that we want to use Memcache as our cahce backend. Create a file called `local-reqs.in` and add the following:

```
-r requirements.txt
pylibmc
```

Note: As well as the Memcache backend, if you're not using RabbitMQ for your task queue you will need to install extra package for Celery. Those packages should be added to `local-reqs.in` as well. Refer to the Celery documentation for details.

Note: You'll have to enable Memcache in your `inboxen.config` file before using it. The same applies to using a different Celery broker.

Always pin your dependencies!

```
(env) $ pip-compile -U --output-file local-reqs.txt local-reqs.in
(env) $ pip-sync local-reqs.txt
```

1.3.2 make rules

As you've seen already, we provide a number of make rules for common tasks. You can add your own in `local.mk`. For example, you might want to have a rule to install dependencies:

```
.PHONY: install-local-deps
install-local: install-js-deps
    pip-sync local-reqs.txt
```

This would allow you to run the following:

```
(env) $ make install-local-deps
```

1.4 Upgrading

```
(env) $ make salmon-stop celery-setop
(env) $ git pull
```

If you specified additional Python packages, then update your pinned dependencies:

```
(env) $ pip-compile -U --output-file local-reqs.txt local-reqs.in
```

Otherwise, skip this step.

Install updated packages and compile various assets:

```
(env) $ pip-sync local-reqs.txt || pip-sync requirements.txt
(env) $ npm install
(env) $ ./manage.py migrate
(env) $ make static
```

Finally, restart services:

```
(env) $ make salmon-start celery-start
(env) $ touch inboxen/wsgi.py
```

CHAPTER

TWO

THE SETTING FILE

The settings file is required to set basic option for your Inboxen instance. It contains some secret information such as the `secret_key`, you should ensure the permissions are set correctly and also that this file is kept safe as without it sessions, cookie storage and along with other things rely on this will not work[0].

The Inboxen settings file can be located in several places on a system, it will use the first one it finds. Inboxen looks for the files in this order:

1. The path specified in the environment variable: `INBOXEN_CONFIG`
2. ~/.config/inboxen/inboxen.config
3. inboxen.config in the current working directory
4. inboxen.config inside the base directory of the Inboxen project

If you're familiar with Django and would like to use your own settings module, you can set `DJANGO_SETTINGS_MODULE` in the usual way[1].

2.1 Example configuration

This also contains the default values for all configuration values.

```
admins: []
allowed_hosts: []
cache:
  backend: "django.core.cache.backends.filebased.FileBasedCache"
  location: "inboxen_cache"
  timeout: 300
database:
    host: ""
    name: "inboxen"
    password: ""
    port: ""
    user: ""
debug: false
enable_registration: false
inbox_length: 6
language_code: "en-gb"
media_root: "media_content"
per_user_email_quota: 0
ratelimits:
```

(continues on next page)

[0] https://docs.djangoproject.com/en/2.2/ref/settings/#secret-key
[1] https://docs.djangoproject.com/en/2.2/topics/settings/#envvar-DJANGO_SETTINGS_MODULE

(continued from previous page)

```
    inbox:
        count: 100
        window: 1440
    login:
        count: 5
        window: 60
    register:
        count: 5
        window: 30
    single_email:
        count: 100
        window: 60
server_email: "django@localhost"
site_name: "LazyAdmin.com's Inboxen"
source_link: "https://github.com/Inboxen/Inboxen"
static_root: "static_content"
tasks:
    always_eager: false
    broker_url: "amqp://guest:guest@localhost:5672//"
    concurrency: 3
    liberation:
        path: "liberation_store"
        sendfile_backend: "django_sendfile.backends.simple"
time_zone: "UTC"
```

2.2 Minimum Development Configuration

To hit the ground running, the minimum you need to setup a development instance is:

```
secret_key: something-secret
debug: true
```

2.3 Options

2.3.1 secret_key

This is used as the global salt for cryptographic signing throughout Inboxen. This is security sensitive and should be generated using a random number generator. It's strongly suggested you use at least 50 characters of numbers, both case characters and symbols from a high entropy source.

2.3.2 admins

This should be pairs of values denoting the name and email address of your admins, like so:

```
admins:
  - - Me
    - me@example.com
  - - You
    - you@example.com
```

2.3.3 allowed_hosts

This is a list of domains and/or IPs that Django will serve Inboxen on. There is support for wildcards, the syntax of which can be found in the Django documentation.

2.3.4 debug

Enabling this puts Inboxen into debug mode, this should never be used in a production environment as it exposes the state of some calls in Inboxen including the settings file. This should be used when developing on Inboxen as it allows for tracebacks to be displayed instead of emailed and disables `allowed_hosts` checking.

2.3.5 enable_registration

A boolean flag which controls if the Inboxen instance permits registration, if disabled the site will not allow new users to be created through the public facing site and disables the links to the registration page.

2.3.6 language_code

This specifies the language code that is used as a fallback when one can't be detected by Django's locale middleware (or if the middleware is disabled). This should be set to a standard language ID format[2].

2.3.7 static_root

This specifies where the directory is for serving static files. Django will use this directory to place static files when using:

```
python manage.py collectstatic
```

2.3.8 meida_root

This specifies where the directory is for uploading media via the CMS. It should be writable by the Django app.

[2] https://docs.djangoproject.com/en/2.2/topics/i18n/#term-language-code

2.3.9 server_email

The email the server uses when sending emails.

2.3.10 site_name

The name of the site as displayed in page titles.

2.3.11 source_link

The link to the source code for the current instance. If you change any code in Inboxen this must be shared back under the terms of the AGPL v3, you should populate this with the link to the source code.

2.3.12 time_zone

The timezone used for the site, this is used for example when storing dates in the database.

2.3.13 per_user_email_quota

If not `0`, this is the maximum number of emails a user can have before they need to delete some. This deletion can be done automatically if the user prefers.

2.3.14 ratelimits

Rate limits control various parts of Inboxen. Each rate limit section has a window (the timeframe a rate limit should be considering) and a count (the maximum number of times whatever that rate limit is protecting can happen with a window).

The following rate limits are available:

inbox

Controls how often a single user can create an inbox. Useful to prevent someone from exhausting all available inboxes.

login

Controls how often a user can try to login. This slows down password guessing attempts, but can block users who genuinely can't remember their passwords.

register

Controls how often a given IP can register a new account. Prevents circumventing of the inbox ratelimit.

single_email

Controls how often a user can download a single email. This is quite an intense workload for the server, so it is ratelimited to prevent the instance becoming overloaded.

2.3.15 tasks

broker_url

The URL that celery will look at to find tasks and to store results.

concurrency

The number of celery processes to start

liberation

path

Specifies the path where to store the liberation data. This needs to be kept secure as it will contain user data.

sendfile_backend

Which method should be used to accelerate liberation data downloads. This should be a dotted path to the django_sendfile2 backend you wish to use.

2.3.16 database

name

The name of the database.

user

User used when connecting to PostgreSQL.

password

The password used when connecting to PostgreSQL.

host

The host name or IP address to connect to for PostgreSQL.

port

The port to connect to for PostgreSQL.

2.3.17 Cache

backend

The dotted path of the cache module you'd like to use.

timeout

The number of seconds before a cache entry is considered stale.

location

This is either the host and port for the `memcached` backend or the path of the cache directory.