
codesy Documentation

Release 1.0

Luke Crouch

Aug 12, 2017

Contents

1	Resources	3
2	Contents	5
2.1	Development	5
2.2	Deployment	7

codesy is a pay-what-you-want market for the open source community to encourage coders to fix important bugs.

The code for codesy's API backend combines:

- [GitHub Authentication](#) (via [django-allauth](#))
- [Stripe payments](#) (via [Stripe.js](#))
- codesy variation of [Nate Oostendorp's "Concurrent Sealed Bid Auction"](#) system and [CodePatron](#) concept.

CHAPTER 1

Resources

Code <https://github.com/codesy/codesy>

License AGPLv3; see LICENSE file

Documentation <http://codesy.readthedocs.org/>

Issues <https://github.com/codesy/codesy/issues>

IRC <irc://irc.freenode.net/codesy>

Mailing list <https://groups.google.com/forum/#!forum/codesy-dev>

Servers <https://codesy-stage.herokuapp.com/> (stage)

<https://api.codesy.io/> (prod)

Development

Requirements

- python
- stunnel

Install Locally

codesy's backend tries to be very slim, so starting should be easy. (Especially if you're familiar with Django):

1. Clone and change to the directory:

```
git clone git@github.com:codesy/codesy.git
cd codesy
```

2. Create and activate a virtual environment:

```
virtualenv env
source env/bin/activate
```

3. Install requirements:

```
pip install -r requirements.txt
```

4. Copy decouple config env file:

```
cp .env-dist .env
```

5. Migrate DB tables

```
./manage.py migrate
```

6. Create a superuser:

```
./manage.py createsuperuser
```

Run locally with https

The codesy browser extensions contain content scripts that execute on `https://` domains and request resources from the codesy domain. So, you need to run the backend over `https://`. The easiest way to run `https` connections with Django is to run `stunnel` on `https://127.0.0.1:8443` in front of Django:

1. First, install `stunnel` for your OS (E.g., on Mac OS `brew install stunnel`).
2. Run Django dev server in **HTTPS mode** on port 5000:

```
HTTPS=1 ./manage.py runserver 127.0.0.1:5000
```

3. Generate local cert and key file for `stunnel`:

```
openssl req -new -x509 -days 9999 -nodes -out stunnel/stunnel.pem -keyout stunnel/  
↪stunnel.pem
```

4. Run `stunnel` with the included `dev_https` config:

```
stunnel stunnel/dev_https
```

5. Go to `https://127.0.0.1:8443` and confirm the certificate exception.

Note: You always need to run both `runserver` and `stunnel`.

1. Change the default `django Site` record from `example.com` to `127.0.0.1:8443`

Read the [Chrome Extension docs](#) and the [Firefox Add-on docs](#) too learn how to configure them to use `https://127.0.0.1:8443`.

Finally, you'll need to enable GitHub authentication ...

Enable GitHub Auth

To enable GitHub authentication, you can use our `codesy-local OAuth` app.

Add a `django-allauth social` app for GitHub:

- Provider: `GitHub`
- Name: `codesy-local`
- Client id: `c040becacd90c91a935a`
- Secret key: `08c3da1421bb280e6fa5f61c05afd0c3128a2f9f`
- Sites: `example.com` -> Chosen sites

Now you can sign in with GitHub at `https://127.0.0.1:8443`.

Payments

codesy is pre-configured to use the [Stripe](#) test marketplace. So, you can use the [test credit card numbers](#) from the Stripe docs.

Run the Tests

Install test requirements:

```
pip install -r requirements-test.txt
```

Running the test suite is easy:

```
./manage.py test -s --noinput --logging-clear-handlers
```

Working on Docs

Install dev requirements:

```
pip install -r requirements-dev.txt
```

Building the docs is easy:

```
cd docs
sphinx-build . html
```

Read the beautiful docs:

```
open html/index.html
```

What to work on

We have [Issues](#).

If you are an active codesy user, we love getting pull requests that “scratch your own itch” and help the entire codesy community.

Deployment

codesy is designed with [12-factor app philosophy](#) to run on [heroku](#), so you can easily deploy your changes to your own heroku app with [heroku toolbelt](#).

Deploy your own

1. Create a heroku remote. We suggest naming it `codesy-username`:

```
heroku apps:create codesy-username
```

2. Set a `DJANGO_SECRET_KEY` on heroku that's unique to you.:

```
heroku config:set DJANGO_SECRET_KEY="username-birthdate"
```

3. Set other required environment variables for heroku:

```
heroku config:set DJANGO_DEBUG=True
heroku config:set ACCOUNT_EMAIL_VERIFICATION=none
heroku config:set ACCOUNT_DEFAULT_HTTP_PROTOCOL='https'
```

4. Push code to the heroku remote:

```
git push heroku master
```

5. Migrate DB tables:

```
heroku run python manage.py migrate
```

6. Create a superuser:

```
heroku run python manage.py createsuperuser
```

7. Set the site domain name: Go to `/admin/sites/site/1/change/` and make it match `codesy-username.herokuapp.com`

8. To enable GitHub sign-ins on your heroku domain, use the following settings to register your own GitHub App:

- Application name: codesy-username
- Homepage URL: <https://codesy-username.herokuapp.com/>
- Application description: username's codesy
- Authorization callback URL: <https://codesy-username.herokuapp.com/accounts/github/login/callback/>

Note: You must use *https*

9. Now go to <https://codesy-username.herokuapp.com/admin/socialaccount/socialapp/add/> to **enable GitHub Auth** on your heroku domain, using your new GitHub App Client ID and Secret

Note: Remember to use *https*

10. That's it. <https://codesy-username.herokuapp.com/> should work.

Configure backing services

1. To enable email via SendGrid, set `SENDGRID_USERNAME` and `SENDGRID_PASSWORD` config values:

```
heroku config:set SENDGRID_USERNAME=
heroku config:set SENDGRID_PASSWORD=
```

2. To enable Stripe, you will need to set `STRIPE_SECRET_KEY` and `STRIPE_PUBLIC_KEY` values from Stripe Dashboard API Keys:

```
heroku config:set STRIPE_SECRET_KEY=
heroku config:set STRIPE_PUBLIC_KEY=
```

3. To enable PayPal, you will need to set `PAYPAL_CLIENT_ID` and `PAYPAL_CLIENT_SECRET` value from PayPal Developer Dashboard Apps & Credentials:

```
heroku config:set PAYPAL_CLIENT_ID=  
heroku config:set PAYPAL_CLIENT_SECRET=
```

Configure extensions to use the server

Check the extension docs.

Deploying to production

We use Travis CI for continuous deployment to Heroku. Our `.travis.yml` defines the flow:

1. Commits to `master` are tested on Travis.
2. If/when the build passes, the code is automatically deployed to <https://codesy-stage.herokuapp.com>
3. To deploy changes to production, a repo owner pushes a commit to the `production` branch on GitHub.

This means *anyone* can request a production deployment by submitting a Pull Request from `master` to `production`.