

---

# **Mozilla Developer Services Dashboard Documentation**

***Release 1.0***

**Luke Crouch**

**May 25, 2017**



---

## Contents

---

<b>1</b>	<b>Resources</b>	<b>3</b>
<b>2</b>	<b>Contents</b>	<b>5</b>
2.1	Development . . . . .	5
2.2	Testing . . . . .	8
2.3	Deployment . . . . .	9
2.4	Run-book . . . . .	10



push-dev-dashboard is the code for Mozilla Developer Services dashboard - where web developers can manage how their web apps and sites use services like [Mozilla Push Service](#).



# CHAPTER 1

---

## Resources

---

**Code** <https://github.com/mozilla-services/push-dev-dashboard>

**License** MPL2

**Documentation** <http://push-dev-dashboard.readthedocs.org/>

**Issues** <https://github.com/mozilla-services/push-dev-dashboard/issues>

**CI** <https://travis-ci.org/mozilla-services/push-dev-dashboard> (unit tests)

<https://circleci.com/gh/mozilla-services/push-dev-dashboard> (deployment artifacts)

[https://services-qa-jenkins.stage.mozaws.net:8443/job/push-dashboard\\_e2e-test\\_prod/](https://services-qa-jenkins.stage.mozaws.net:8443/job/push-dashboard_e2e-test_prod/) (selenium/integration tests)

**Servers** <https://pushdevdashboard-default.stage.mozaws.net/> (stage) <https://push-dashboard.services.mozilla.com/> (prod)

**IRC** <irc://irc.mozilla.org/push>





## Development

### Requirements

- `python 2.7`, `virtualenv`, `pip` for app server
- `npm` for front-end testing

### Install Locally

1. Clone and change to the directory:

```
git clone git@github.com:mozilla-services/push-dev-dashboard.git
cd push-dev-dashboard
```

2. Create and activate a `virtual environment` (Can also use `virtualenvwrapper`):

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

3. Install requirements:

```
pip install -r requirements-dev.txt
npm install
npm link stylus yuglify
```

4. Copy the `.env-dist` file to `.env`:

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

5. Source the `.env` file to set environment config vars (Can also use `autoenv`):

```
source .env
```

### 6. **‘Migrate’** DB tables

```
python manage.py migrate
```

### 7. Create a superuser:

```
python manage.py createsuperuser
```

## Run it

#### 1. Source the `.env` file to set environment config vars (Can also use [autoenv](#)):

```
source .env
```

#### 2. Activate the [virtual environment](#) (Can also use [virtualenvwrapper](#)):

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

#### 3. Run it:

```
python manage.py runserver
```

## Enable Firefox Accounts Auth

To enable Firefox Accounts authentication, you can use our local development OAuth client app.

#### 1. Add a [django-allauth social app](#) for Firefox Accounts (Log in as the superuser account you created):

- Provider: Firefox Accounts
- Name: fxa
- Client id: 7a4cd4ca0fb1b5c9
- Secret key: c10059ba24e6715a1b6f2c80f1cc398fb6a39ca18bc7554e894b36ea85b88eeb
- Sites: example.com -> Chosen sites

#### 2. [Log out of the admin account](#)

#### 3. Sign in with a Firefox Account at <http://127.0.0.1:8000>.

## Run in production mode

Follow these steps to emulate production (for example, to test compressed assets). Run all commands from the project root.

1. Stop `runserver` if it's already running
2. In `.env`, set `DJANGO_DEBUG` to `False`
3. Run `python manage.py collectstatic`
4. **Install [stunnel](#)**
  - Mac: `brew install stunnel`

5. Run this command to generate a local cert and key for stunnel (you can use the default values for all prompts): `openssl req -new -x509 -days 9999 -nodes -out stunnel/stunnel.pem -keyout stunnel/stunnel.pem`
6. Run `stunnel stunnel/dev_https`
7. In another terminal, run `HTTPS=1 python manage.py runserver 127.0.0.1:5000`
8. Go to <https://127.0.0.1:8443> to confirm the certificate exception and browse the site

## Working on Docs

Install doc requirements:

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

Building the docs is easy:

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

Read the beautiful docs:

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

## Updating Translations

1. Run `makemessages` to make updated `django.po` files:

```
python manage.py makemessages --keep-pot
```

2. Commit the updates to git:

```
git add locale
git commit -m "Updating translations {YYYY-MM-DD}"
```

## Adding a Translation

1. First, *Update translations*
2. Make the new `{locale}` directory for the new language:

```
mkdir locale/{locale}
```

3. Run `makemessages` to make the `django.po` file for it:

```
python manage.py makemessages -l {locale}
```

4. Add the new directory to git:

```
git add locale/{locale}
git commit -m "Adding {locale} locale"
```

## What to work on

We have [Issues](#).

## Testing

### Back-end python tests

1. Install test requirements:

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

2. Run the test suites:

```
python manage.py test
```

### Back-end style tests

1. Install test requirements:

```
pip install flake8
```

2. Run the test suites:

```
flake8 .
```

### Front-end style tests

1. Install test requirements:

```
npm install
```

2. Run the test suites:

```
npm test
```

### Translation lint tests

1. Install test requirements:

```
pip install -r requirements-l10n.txt
```

2. Run the test suites:

```
cd locale  
dennis-cmd lint .
```

## Selenium/Integration tests

1. Install test requirements:

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

2. Set environment variables in `.env` file:

```
DJANGO_DEBUG_TOOLBAR=False
TESTING_WEBDRIVER_TIMEOUT=10
TESTING_FXA_ACCOUNT_EMAIL=tester@test.com
TESTING_FXA_ACCOUNT_PASSWORD=testpass
```

- **Required** `DJANGO_DEBUG_TOOLBAR` - The django debug toolbar interferes with selenium clicking on the sign-in button; disable it. *NOTE:* Make sure you restart the django process.
- **Required** `TESTING_WEBDRIVER_TIMEOUT` - Number of seconds selenium/Firefox will wait before timing out. Default is 0 which skips selenium test.
- **Required** `TESTING_FXA_ACCOUNT_EMAIL` - Email of Firefox Account to use during tests.
- **Required** `TESTING_FXA_ACCOUNT_PASSWORD` - Password of Firefox Account to use during tests.
- `TESTING_SITE` - The dashboard domain/site that selenium/Firefox will use. Default is `http://127.0.0.1:8000`
- `TEST_PUSH_SERVER_URL` - The `dom.push.serverURL` that selenium/Firefox will use. Default is the dev environment: `wss://benpushstack-1704054003.dev.mozaws.net/` *Note:* Make sure the [Push Messages API](#) server in `PUSH_MESSAGES_API_ENDPOINT` matches this push server.

3. Run the test suites:

```
python manage.py test
```

## Deployment

push-dev-dashboard is designed with [12-factor app philosophy](#), so you can easily deploy your changes to your own app.

### Deploy onto Deis

Note: Mozilla used to run our own Deis cluster, but it has been shut down. The following instructions should work to deploy the code to your own Deis cluster.

1. [Install the deis client](#).
2. [Install deis on AWS](#).
3. Create the application:

```
deis create dev-dashboard-username
```

4. Push code to the deis remote:

```
git push deis master
```

5. [Create an RDS Postgres instance](#) in us-east-1 with default settings except:

- DB Instance Class: db.t2.micro
- Allocated Storage: 5 GB
- VPC: vpc-9c2b0ef8

6. In the RDS Instance configuration details, click the “Security Groups”. (Usually something like “rds-launch-wizard-N (sg-abcdef123)”)
7. In the security group, under the “Inbound” tab, change the source to allow the deis cluster hosts.
8. Set the DATABASE\_URL environment variable to match the RDS DB:

```
deis config:set DATABASE_URL=postgres://username:password@endpoint/dbname
```

9. Migrate DB tables on the new RDS instance:

```
deis run python manage.py migrate
```

10. Dock to app instance to create a superuser:

```
deisctl dock dev-dashboard-username  
/app/.heroku/python/bin/python manage.py createsuperuser
```

11. Open the new deis app:

```
deis open
```

## Enable Firefox Accounts Auth on your Deis app

To enable Firefox Account sign-ins on your Deis app, you will need to create your own Firefox Accounts OAuth Client for your app domain.

1. Go to [register your own Firefox Accounts OAuth Client](#):
  - Client name: dev-dashboard-username
  - Redirect URI: <https://<your-push-dev-dashboard-app-on-deis-domain>/accounts/fxa/login/callback/>
  - Trusted Mozilla Client: **CHECKED**

Be sure to copy the client secret - you can't see it again.

2. Go to <https://<your-push-dev-dashboard-app-on-deis-domain>/admin/socialaccount/socialapp/add/> to *Enable Firefox Accounts Auth* like a local machine; this time using your own new Firefox Accounts OAuth Client ID and Secret
3. Sign in at <https://<your-push-dev-dashboard-app-on-deis-domain>/> with a Firefox Account.

## Run-book

push-dev-dashboard is written as a monolithic django application with 2 processes. It's written with 12 factor methodology, so it is configured almost entirely by environment variables.

[Travis CI](#) is used for unit, docs, l10n, and coding style tests before code lands in master.

[Circle CI](#) is used to build docker containers for deployment.

[Jenkins](#) is used to run the selenium integration tests on deployments to the stage and production servers.

## Processes

### Web

The web process is a django web app run by gunicorn. It is defined as the `CMD` instruction in the `Dockerfile`.

### clock

The clock process is a python script run by the `django_extensions runscript` command defined in `Procfile`. It uses `APScheduler` to call the `start_recording_push_apps` `django` command.

## Environment Variables

- **REQUIRED** `DATABASE_URL` - database connection url. See the `dj-database-url` [URL schema](#) reference
- **REQUIRED** `PUSH_MESSAGES_API_ENDPOINT` - endpoint for [Push Messages API](#).
- **REQUIRED** `FXA_OAUTH_ENDPOINT` - endpoint for FxA oauth provider. See the `django-allauth` [Firefox Accounts](#) reference.
- **REQUIRED** `FXA_PROFILE_ENDPOINT` - endpoint for FxA profile. See the `django-allauth` [Firefox Accounts](#) reference.