Taiga

Setup development environment

Table of Contents

1. Introduction

This documentation explains how to setup the Taiga development environment.

The Taiga platform consists of three main components:

- **taiga-back** (backend/api)
- **taiga-front** (frontend)
- **taiga-events** (websockets gateway) (optional)

And each one has its own dependencies, at compile time and runtime.

2. Before starting

This tutorial assumes that you are using a clean, recently updated, **ubuntu 14.04** image. Notes for Debian installations are included at the end of the appropriate sections.

Taiga installation must be done with a "normal" user, never with root.

3. Backend environment

This section helps with the download and configuration of the backend (api) Taiga service.

3.1. Install dependencies

The backend is written mainly in python (3.4) but for some third party libraries we need to install a C compiler and development headers.

```
sudo apt-get install -y build-essential binutils-doc autoconf flex bison libjpeg-dev
sudo apt-get install -y libfreetype6-dev zlib1g-dev libzmq3-dev libgdbm-dev libncurses5-
dev
sudo apt-get install -y automake libtool libffi-dev curl git tmux gettext
```

3.2. Setup a database

taiga-back also requires postgresql (>= 9.3) as a database

Install postgresql:

```
sudo apt-get install -y postgresql-9.3 postgresql-contrib-9.3
sudo apt-get install -y postgresql-doc-9.3 postgresql-server-dev-9.3
```

And setup the initial user, database, and permissions:

```
sudo -u postgres psql -c "CREATE ROLE taiga LOGIN PASSWORD 'changeme';"
sudo -u postgres createdb taiga -O taiga
echo 'local all taiga peer' | sudo -u postgres tee -a
/etc/postgresql/9.3/main/pg_hba.conf > /dev/null
sudo service postgresql reload
```

3.3. Setup python environment

To run **taiga-back** you should have python (3.4) installed along with some other third party libraries. As a first step, start installing python and virtualenvwrapper:

```
sudo apt-get install -y python3 python3-pip python-dev python3-dev python-pip
virtualenvwrapper
sudo apt-get install libxml2-dev libxslt-dev
```

NOTE	**virtualenvwrapper** helps keep the system clean from third party libraries installed with the language's package manager by installing them in isolated virtual environments.

Restart the shell or run `bash` again, to reload the bash environment with virtualenvwrapper's variables and functions.

The next step is to download the code from GitHub and install its dependencies:

Download the code

```
cd ~
git clone https://github.com/taigaio/taiga-back.git taiga-back
cd taiga-back
git checkout stable
```

Create a new virtualenv named **taiga**

```
mkvirtualenv -p /usr/bin/python3.4 taiga
```

Install dependencies

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

Adjust Django Configuration

You can tune your own environment configuration by editing a `settings/local.py` configuration file to overwrite any setting in `settings/common.py`.

For a basic configuration that works with these instructions, simply copy settings/local.py.example to settings/local.py.

Otherwise, just put this in your own ~/taiga-back/settings/local.py

```
from .common import *

# YOUR OWN CONFIGURATION HERE
```

Populate the database with initial basic data

```
python manage.py migrate --noinput
python manage.py loaddata initial_user
python manage.py loaddata initial_project_templates
python manage.py loaddata initial_role
python manage.py compilemessages
python manage.py collectstatic --noinput
python manage.py sample_data
```

This creates a new user **admin** with password **123123** and some sample data.

3.4. Run

To run the development environment you can run:

```
workon taiga  # enable the taiga virtualenv
python manage.py runserver
```

Then you should be able to see a json represention of the list of endpoints at the url http://localhost:8000/api/v1/

3.5. Async tasks (Optional)

The default behavior in Taiga is to do all tasks synchronously, but some of them can be completely

asynchronous (for example webhooks or import/export). To do this, you have to configure and install the celery service requirements.

Install `rabbitmq-server` and `redis-server`:

```
sudo apt-get install -y rabbitmq-server redis-server
```

To run celery with Taiga you have to include the following lines in your local.py:

```
from .celery import *

BROKER_URL = 'amqp://guest:guest@localhost:5672//'
CELERY_RESULT_BACKEND = 'redis://localhost:6379/0'
CELERY_ENABLED = True
```

You can configure other broker or results backends as needed. If you need more info about configuration you can check the celery documentation web page: http://docs.celeryproject.org/en/latest/index.html

Once you have configured celery on Taiga, you have to run celery to process the tasks. You can run celery with:

```
workon taiga  # enable the taiga virtualenv
celery -A taiga worker -l info -E
```

3.6. Debian installation notes

Debian stable (Jessie) provides all needed requirements, but old-stable (Wheezy) does not.

The latest Python available from Wheezy's apt repositories is only 3.1 and insufficient for taiga-back. Python 3.4 is available from stable (Jessie) if you are comfortable using mixed versions in your apt sources. Otherwise, you must build Python 3.4 from source (see https://www.python.org/downloads/source/ for links). When building from source, if the bz2 development libraries are not already present on your system, then you must first:

```
sudo apt-get install libbz2-dev
```

Or else Python will build without the bz2 module necessary for some pip installed requirements.

The latest Postgresql available for Wheezy is 9.1, but a fully Wheezy-compatible 9.3 build is available from the official Postgresql apt repositories, however:

```
echo "deb http://apt.postgresql.org/pub/repos/apt/ wheezy-pgdg main" | sudo tee -a
/etc/apt/sources.list
sudo apt-get update
```

4. Frontend installation

This section helps you install the frontend application

4.1. Install dependencies

The frontend application runs entirely in a browser, and thus must be deployed as javascript, css and html. In the case of **taiga-front** we have used other languages. Because of this, you will need to install some additional dependencies that compile **taiga-front** code into something the browser can understand.

4.1.1. Ruby and Gems

Ruby is used mainly for compiling **sass** (css preprocessor). It is also used for sass linting but that is only in development environments.

Install ruby

```
sudo apt-get install -y ruby
```

Install required gems

```
gem install --user-install sass scss-lint
```

*Make gems' scripts available from your path by putting this in your ~/***.bashrc**

```
export PATH=~/.gem/ruby/1.9.1/bin:$PATH
```

Restart the shell, source ~/.bashrc, or run bash again to make the path changes available.

4.1.2. NodeJS and friends

NodeJS is used to execute **gulp** and **bower**:

- **gulp**: task execution tool. Used mainly for executing deployment and compilation tasks.
- **bower**: javascript dependency management tool. Used mainly for downloading third party libraries used by **taiga-front**.

Install nodejs

```
sudo apt-get install -y nodejs npm
```

Make sure your bash responds to the node command to have a smooth installation of gulp and bower

```
node
```

If you get a "Command not found" error, then run

```
sudo ln -s /usr/bin/nodejs /usr/bin/node
```

(If you're on Debian, see the Debian-specific installation notes below.)

Install **gulp** *and* **bower** *using the recently installed npm*

```
sudo npm install -g gulp bower
```

Download the code

```
cd ~
git clone https://github.com/taigaio/taiga-front.git taiga-front
cd taiga-front
git checkout stable
```

Install all dependencies needed to run gulp and compile taiga-front

```
npm install
bower install
```

4.2. Debian installation notes

While Debian stable (Jessie), provides a nodejs package out of the box, old-stable (Wheezy) does not. You can access one via the wheezy-backports apt repository, however, which can be added to your system as follows:

```
echo "deb http://ftp.us.debian.org/debian wheezy-backports main" | sudo tee -a
/etc/apt/sources.list
```

Then, after a:

```
sudo apt-get update
```

You can:

```
sudo apt-get install nodejs
```

Note that Debian installs the executable as nodejs not node, so you will need to provide this alias by issuing the following command:

```
sudo update-alternatives --install /usr/bin/node nodejs /usr/bin/nodejs 100
```

Stable (Jessie) also provides an npm package, but npm is not available for old-stable (Wheezy), not even from wheezy-backports. Thus, you will need to install it manually via:

```
curl https://www.npmjs.com/install.sh | sudo sh
```

4.3. Final steps

Having installed all the dependencies, all you have left to do is to run the code itself.

Run gulp

```
cd ~/taiga-front
gulp
```

Now, you can access http://localhost:9001 for access to taiga-front.

NOTE	If you have npm errors when executing gulp delete the tmp files and install the dependencies again.

```
rm -rf ~/.npm; rm -rf node_modules
npm install
bower install
gulp
```

5. Events installation

This step is completelly optional and can be skipped

Taiga events needs rabbitmq (the message broker) to be installed

Installing rabbitmq

```
sudo  apt-get install rabbitmq-server
```

Creating a taiga user and virtualhost for rabbitmq

```
sudo rabbitmqctl add_user taiga PASSWORD
sudo rabbitmqctl add_vhost taiga
sudo rabbitmqctl set_permissions -p taiga taiga ".*" ".*" ".*"
```

Update your taiga-back settings to include the following lines in your local.py:

```
EVENTS_PUSH_BACKEND = "taiga.events.backends.rabbitmq.EventsPushBackend"
EVENTS_PUSH_BACKEND_OPTIONS = {"url": "amqp://taiga:PASSWORD@localhost:5672/taiga"}
```

The next step is downloading the code from GitHub and installing the dependencies:

Download the code

```
cd ~
git clone https://github.com/taigaio/taiga-events.git taiga-events
cd taiga-events
```

Install all the javascript dependencies needed

```
npm install
sudo npm install -g coffee-script
```

Copy config.example.json to config.json and edit it to update the values for your rabbitmq uri and secret key.

```
cp config.example.json config.json
```

Your config.json should look something like:

```
{
    "url": "amqp://taiga:PASSWORD@localhost:5672/taiga",
    "secret": "mysecret",
    "webSocketServer": {
        "port": 8888
    }
}
```

Now run the taiga events service

```
coffee index.coffee
```