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 20.04** 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 (>= 3.8, < 3.12) 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 libssl-dev curl git tmux gettext
```

3.2. Setup a database

taiga-back also requires postgresql (>= 9.4, < 14) as a database

Install postgresql:

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

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 --encoding='utf-8' --locale=en_US.utf8
--template=template0
echo 'local all taiga peer' | sudo -u postgres tee -a $(sudo -u postgres psql -t -P
format=unaligned -c 'show hba_file') > /dev/null
sudo service postgresql reload
```

3.3. Setup python environment

To run **taiga-back** you should have python (>=3.8, < 3.12) installed along with some other third party libraries. As a first step, start installing python:

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

NOTE	Between November 2018 and May 2020, taiga-back relied on pipenv to manage dependencies and its virtualenv. Currently, taiga-back uses `pip` and pip-tools to

> manage dependecies and `venv` to create its virtualenv.

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

Download the code

```
cd ~
git clone https://github.com/kaleidos-ventures/taiga-back.git taiga-back
cd taiga-back
git checkout main
```

Create a new virtualenv, activate it and install taiga-back base requirements

```
python -m venv .venv
source .venv/bin/activate
python -m pip install --upgrade pip setuptools wheel
python -m pip install --upgrade pip-tools
```

NOTE We are using the `venv` module, included in the standard library, to manage the taiga-back virtual environmnent but you can choose other ways to manage the virtual environment like virtualenvwrapper or pew.

Install dependencies

```
python -m pip install -r requirements.txt -r requirements-devel.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` (but remember to edit your postgresql password, to match what you picked for the `CREATE ROLE taiga LOGIN PASSWORD` command above).

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 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:

```
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 line in your local.py:

```
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:

```
python -m 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. You must build Python (>= 3.8, < 3.12) 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 9.4 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 environment

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. NodeJS and friends

NodeJS is used to execute **gulp**, a task execution tool used mainly for executing deployment and compilation tasks.

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

```
node
```

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

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

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

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

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

Download the code

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

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

```
npm 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

```
cd ~/taiga-front
npm start
```

And now, you can configure it copying the `dist/conf.example.json` to `dist/conf.json` and editing it.

Copy and edit initial configuration on ~/taiga-front/dist/conf.json

```
{
    "api": "http://localhost:8000/api/v1/",
    "eventsUrl": null,
    "eventsMaxMissedHeartbeats": 5,
    "eventsHeartbeatIntervalTime": 60000,
    "debug": true,
    "debugInfo": false,
    "defaultLanguage": "en",
    "themes": ["taiga"],
    "defaultTheme": "taiga",
    "publicRegisterEnabled": true,
    "feedbackEnabled": true,
    "privacyPolicyUrl": null,
    "termsOfServiceUrl": null,
    "maxUploadFileSize": null,
    "contribPlugins": []

}
```

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
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 rabbitmquser named taiga and virtualhost for rabbitmq

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

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://rabbitmquser:rabbitmqpassword@rabbitmqhost:5672/taiga"}
```

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

Download the code

```
cd ~
git clone https://github.com/kaleidos-ventures/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://rabbitmquser:rabbitmqpassword@rabbitmqhost:5672/taiga",
    "secret": "taiga-back-secret-key",
    "webSocketServer": {
        "port": 8888
    }
}
```

Now run the taiga events service

```
coffee index.coffee
```