Taiga

Setup production environment

Table of Contents

1. Introduction

This documentation explains how to deploy a Taiga service (each module is part of the Taiga platform).

The Taiga platform consists of three main components:

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

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

Each component can run on one unique machine or each can be installed in a different machine. In this tutorial we will setup everything in one single machine. This type of setup should suffice for small/medium production environments.

2. Before starting

This tutorial assumes that you are using a clean, recently updated, **ubuntu 16.04** image.

Due the nature of frontend application, you should know that this service will be used through the domain/public-ip. This is because, the frontend application will run on your browser and it should communicate with the backend/api.

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

We assume the following:

- **ip:** `80.88.23.45`
- **hostname:** `example.com` (which points to 80.88.23.45)
- **username:** `taiga`
- **system ram** `>=1GB` (needed for compilation of lxml)

3. Backend installation

This section helps with the installation 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 compiller 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 (>= 9.4) as a database

Install postgresql:

```
sudo apt-get install -y postgresql-9.5 postgresql-contrib-9.5
sudo apt-get install -y postgresql-doc-9.5 postgresql-server-dev-9.5
```

And setup the initial user and database:

```
sudo -u postgres createuser taiga
sudo -u postgres createdb taiga -O taiga
```

3.3. Setup python environment

For run **taiga-back** you should have python (3.4) installed 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 maintain the system clean of third party libraries installed with language package manager, installing them in isolated virtual environment.

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

The next step is to download the code from github and install their dependencies:

Download the code

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

Create new virtualenv named **taiga**

```
mkvirtualenv -p /usr/bin/python3.5 taiga
```

Install dependencies

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

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
```

This creates a new user **admin** with password **123123**.

If you want some example data, you can execute the following command, which populates the database with sample projects and random data; useful for demos:

```
python manage.py sample_data
```

And as final step for setup **taiga-back**, you should create the intial configuration for proper static/media files resolution and optionally, email sending support:

Put this on ~/taiga-back/settings/local.py

```
from .common import *

MEDIA_URL = "http://example.com/media/"
STATIC_URL = "http://example.com/static/"
ADMIN_MEDIA_PREFIX = "http://example.com/static/admin/"
SITES["front"]["scheme"] = "http"
SITES["front"]["domain"] = "example.com"

SECRET_KEY = "theveryultratopsecretkey"

DEBUG = False
TEMPLATE_DEBUG = False
PUBLIC_REGISTER_ENABLED = True

DEFAULT_FROM_EMAIL = "no-reply@example.com"
SERVER_EMAIL = DEFAULT_FROM_EMAIL

# Uncomment and populate with proper connection parameters
# for enable email sending. EMAIL_HOST_USER should end by @domain.tld
#EMAIL_BACKEND = "django.core.mail.backends.smtp.EmailBackend"
#EMAIL_USE_TLS = False
#EMAIL_HOST = "localhost"
#EMAIL_HOST_USER = ""
#EMAIL_HOST_PASSWORD = ""
#EMAIL_PORT = 25

# Uncomment and populate with proper connection parameters
# for enable github login/singin.
#GITHUB_API_CLIENT_ID = "yourgithubclientid"
#GITHUB_API_CLIENT_SECRET = "yourgithubclientsecret"
```

3.4. Verification

To make sure everything is working, you can run the backend in development mode with:

```
workon taiga
python manage.py runserver
```

Then you must be able to see a json representing the list of endpoints in the url http://localhost:8000/api/v1/ .

NOTE

At this stage the backend has been installed successfully. But you're not done yet. Because python in production environments, should run on an application server. The details for this are explained in the final section of this document.

3.5. Async tasks (Optional)

The default behavior in Taiga is to do all tasks in a synchronous way, 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.

There is just an exception related to exported files, if your instance works in asynchronous mode the exported project files will be automatically removed from the storage after 24 hours, if not those media files won't be automatically removed (taiga-back doesn't really know anything about the existance of those files).

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

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

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

```
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 backend. If you need more info about it 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 add celery to circus configuration. See Circus and gunicorn section.

Taiga celery configuration block for circus on /etc/circus/conf.d/taiga-celery.ini

```
[watcher:taiga-celery]
working_dir = /home/taiga/taiga-back
cmd = celery
args = -A taiga worker -c 4
uid = taiga
numprocesses = 1
autostart = true
send_hup = true
stdout_stream.class = FileStream
stdout_stream.filename = /home/taiga/logs/celery.stdout.log
stdout_stream.max_bytes = 10485760
stdout_stream.backup_count = 4
stderr_stream.class = FileStream
stderr_stream.filename = /home/taiga/logs/celery.stderr.log
stderr_stream.max_bytes = 10485760
stderr_stream.backup_count = 4

[env:taiga-celery]
PATH = /home/taiga/.virtualenvs/taiga/bin:$PATH
TERM=rxvt-256color
SHELL=/bin/bash
USER=taiga
LANG=en_US.UTF-8
HOME=/home/taiga
PYTHONPATH=/home/taiga/.virtualenvs/taiga/lib/python3.4/site-packages
```

Then you have to reload your circus configuration, restart taiga and start taiga-celery:

```
circusctl reloadconfig
circusctl restart taiga
circusctl start taiga-celery
```

4. Frontend installation

Download the code from github:

Download the code

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

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

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

```
{
    "api": "http://example.com/api/v1/",
    "eventsUrl": "ws://example.com/events",
    "debug": "true",
    "publicRegisterEnabled": true,
    "feedbackEnabled": true,
    "privacyPolicyUrl": null,
    "termsOfServiceUrl": null,
    "maxUploadFileSize": null,
    "contribPlugins": []
}
```

Now, having **taiga-front-dist** downloaded and configured, the next step is to expose the code (in **dist** directory) under static file web server: we use **nginx**. That process is explained in the final section of this tutorial.

5. Events installation

This step is completelly optional and can be skipped

Taiga-events is the Taiga websocket server, it allows taiga-front showing realtime changes in backlog, taskboard, kanban and issues listing.

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 in your local.py the lines:

```
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 their 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

```
sudo apt-get install -y nodejs nodejs-legacy npm
npm install
sudo npm install -g coffee-script
```

Copy and edit the config.json file you should update your rabbitmq uri and the secret key.

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

Your config.json should be like:

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

Next, your 'secret' in your `config.json` key should be the same as your "SECRET_KEY" in `~/taiga-back/settings/local.py`

Now you have to add taiga-events to circus configuration. See Circus and gunicorn section.

Taiga taiga-events configuration block for circus on /etc/circus/conf.d/taiga-events.ini

```
[watcher:taiga-events]
working_dir = /home/taiga/taiga-events
cmd = /usr/local/bin/coffee
args = index.coffee
uid = taiga
numprocesses = 1
autostart = true
send_hup = true
stdout_stream.class = FileStream
stdout_stream.filename = /home/taiga/logs/taigaevents.stdout.log
stdout_stream.max_bytes = 10485760
stdout_stream.backup_count = 12
stderr_stream.class = FileStream
stderr_stream.filename = /home/taiga/logs/taigaevents.stderr.log
stderr_stream.max_bytes = 10485760
stderr_stream.backup_count = 12
```

Then you have to reload your circus configuration restart the other services and start taiga-events:

```
circusctl reloadconfig
circusctl restart taiga
circusctl restart taiga-celery
circusctl start taiga-events
```

Nginx need extra configuration too for taiga-events

Add specific configuration for **taiga-events** *on /etc/nginx/sites-available/taiga.*

```
server {
    ...
    location /events {
       proxy_pass http://127.0.0.1:8888/events;
       proxy_http_version 1.1;
       proxy_set_header Upgrade $http_upgrade;
       proxy_set_header Connection "upgrade";
       proxy_connect_timeout 7d;
       proxy_send_timeout 7d;
       proxy_read_timeout 7d;
    }
    ...
}
```

And finally, reload nginx with `sudo service nginx reload`

6. Final steps

If you are here, it's probable that you completed the installation of **taiga-back** and **taiga-front-dist**. However, having installed them is insufficient.

taiga-back should run under an application server which in turn should be executed and monitored by a process manager. For this task we will use **gunicorn** and **circus** respectivelly.

taiga-front-dist and **taiga-back** should be exposed to the outside, using good proxy/static-file web server. For this purpose we'll use **nginx**.

6.1. Circus and gunicorn

Circus is a process manager written by **Mozilla** and you will use it to execute **gunicorn**. Circus not only serves to execute processes, it also has utils for monitoring them, collecting logs, restarting processes if something goes wrong, and starting processes on system boot.

Install circus

```
sudo apt-get install circus
```

Initial taiga configuration for circus on /etc/circus/conf.d/taiga.ini

```
[watcher:taiga]
working_dir = /home/taiga/taiga-back
cmd = gunicorn
args = -w 3 -t 60 --pythonpath=. -b 127.0.0.1:8001 taiga.wsgi
uid = taiga
numprocesses = 1
autostart = true
send_hup = true
stdout_stream.class = FileStream
stdout_stream.filename = /home/taiga/logs/gunicorn.stdout.log
stdout_stream.max_bytes = 10485760
stdout_stream.backup_count = 4
stderr_stream.class = FileStream
stderr_stream.filename = /home/taiga/logs/gunicorn.stderr.log
stderr_stream.max_bytes = 10485760
stderr_stream.backup_count = 4

[env:taiga]
PATH = /home/taiga/.virtualenvs/taiga/bin:$PATH
TERM=rxvt-256color
SHELL=/bin/bash
USER=taiga
LANG=en_US.UTF-8
HOME=/home/taiga
PYTHONPATH=/home/taiga/.virtualenvs/taiga/lib/python3.4/site-packages
```

NOTE

Circus stats can generate a high cpu usage without any load you can set statsd in /etc/circus/circusd.conf to false if you don't need them.

Taiga stores logs on the user home, making them available and immediately accessible when you enter a machine. To make everything work, make sure you have the logs directory created.

You can create it with: `mkdir -p ~/logs`

And finally restart circus:

```
sudo service circus restart
```

6.2. Nginx

Nginx is used as a static file web server to serve **taiga-front-dist** and send proxy requests to **taiga-**

back.

First install it:

```
sudo apt-get install -y nginx
```

And now let us start configuring it:

Add specific configuration for **taiga-front-dist** *and* **taiga-back** *on /etc/nginx/sites-available/taiga.*

```
server {
    listen 80 default_server;
    server_name _;

    large_client_header_buffers 4 32k;
    client_max_body_size 50M;
    charset utf-8;

    access_log /home/taiga/logs/nginx.access.log;
    error_log /home/taiga/logs/nginx.error.log;

    # Frontend
    location / {
        root /home/taiga/taiga-front-dist/dist/;
        try_files $uri $uri/ /index.html;
    }

    # Backend
    location /api {
        proxy_set_header Host $http_host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Scheme $scheme;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_pass http://127.0.0.1:8001/api;
        proxy_redirect off;
    }

    # Django admin access (/admin/)
    location /admin {
        proxy_set_header Host $http_host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Scheme $scheme;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_pass http://127.0.0.1:8001$request_uri;
        proxy_redirect off;
```

```
    }

    # Static files
    location /static {
        alias /home/taiga/taiga-back/static;
    }

    # Media files
    location /media {
        alias /home/taiga/taiga-back/media;
    }
}
```

Disable the default nginx site (virtualhost)

```
sudo rm /etc/nginx/sites-enabled/default
```

Enable the recently created Taiga site (virtualhost)

```
sudo ln -s /etc/nginx/sites-available/taiga /etc/nginx/sites-enabled/taiga
```

You can verify the nginx configuration with the following command to track any error preventing the service to start with `sudo nginx -t`.

Finally, restart nginx with `sudo service nginx restart`.

Now you should have the service up and running on `http://example.com/`

6.2.1. Serving HTTPS

Place your SSL certificates in `/etc/nginx/ssl`. It is recommended to replace the original configuration for port 80 so that users are redirected to the HTTPS version automatically.

Second we need to generate a stronger GHE parameter

```
cd /etc/ssl
sudo openssl dhparam -out dhparam.pem 4096
```

New configuration in /etc/nginx/sites-available/taiga

```
server {
    listen 80 default_server;
    server_name _;
    return 301 https://$server_name$request_uri;
}
```

```
server {
    listen 443 ssl default_server;
    server_name _;

    large_client_header_buffers 4 32k;
    client_max_body_size 50M;
    charset utf-8;

    index index.html;

    # Frontend
    location / {
        root /home/taiga/taiga-front-dist/dist/;
        try_files $uri $uri/ /index.html;
    }

    # Backend
    location /api {
        proxy_set_header Host $http_host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Scheme $scheme;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_pass http://127.0.0.1:8001/api;
        proxy_redirect off;
    }

    location /admin {
        proxy_set_header Host $http_host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Scheme $scheme;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_pass http://127.0.0.1:8001$request_uri;
        proxy_redirect off;
    }

    # Static files
    location /static {
        alias /home/taiga/taiga-back/static;
    }

    # Media files
    location /media {
        alias /home/taiga/taiga-back/media;
    }
```

```
    add_header Strict-Transport-Security "max-age=63072000; includeSubdomains; preload";
    add_header Public-Key-Pins 'pin-
sha256="klO23nT2ehFDXCfx3eHTDRESMz3asj1muO+4aIdjiuY="; pin-
sha256="633lt352PKRXbOwf4xSEa1M517scpD3l5f79xMD9r9Q="; max-age=2592000;
includeSubDomains';

    ssl on;
    ssl_certificate /etc/nginx/ssl/example.com/ssl-bundle.crt;
    ssl_certificate_key /etc/nginx/ssl/example.com/example_com.key;
    ssl_session_timeout 5m;
    ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
    ssl_prefer_server_ciphers on;
    ssl_ciphers 'ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-
AES256-GCM-SHA384:ECDHE-ECDSA-AES256-GCM-SHA384:DHE-RSA-AES128-GCM-SHA256:DHE-DSS-AES128-
GCM-SHA256:kEDH+AESGCM:ECDHE-RSA-AES128-SHA256:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-
AES128-SHA:ECDHE-ECDSA-AES128-SHA:ECDHE-RSA-AES256-SHA384:ECDHE-ECDSA-AES256-
SHA384:ECDHE-RSA-AES256-SHA:ECDHE-ECDSA-AES256-SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES128-
SHA:DHE-DSS-AES128-SHA256:DHE-RSA-AES256-SHA256:DHE-DSS-AES256-SHA:DHE-RSA-AES256-
SHA:!aNULL:!eNULL:!EXPORT:!DES:!RC4:!3DES:!MD5:!PSK';
    ssl_session_cache shared:SSL:10m;
    ssl_dhparam /etc/ssl/dhparam.pem;
    ssl_stapling on;
    ssl_stapling_verify on;

}
```

Before activating the HTTPS site, the configuration for the front-end and back-end have to be updated; change the scheme from `http` to `https`.

Update ~/taiga-back/settings/local.py

```
from .common import *

MEDIA_URL = "https://example.com/media/"
STATIC_URL = "https://example.com/static/"
ADMIN_MEDIA_PREFIX = "https://example.com/static/admin/"
SITES["front"]["scheme"] = "https"
SITES["front"]["domain"] = "example.com"

SECRET_KEY = "theveryultratopsecretkey"

DEBUG = False
TEMPLATE_DEBUG = False
PUBLIC_REGISTER_ENABLED = True

DEFAULT_FROM_EMAIL = "no-reply@example.com"
SERVER_EMAIL = DEFAULT_FROM_EMAIL

# Uncomment and populate with proper connection parameters
# for enable email sending.
#EMAIL_BACKEND = "django.core.mail.backends.smtp.EmailBackend"
#EMAIL_USE_TLS = False
#EMAIL_HOST = "localhost"
#EMAIL_HOST_USER = ""
#EMAIL_HOST_PASSWORD = ""
#EMAIL_PORT = 25

# Uncomment and populate with proper connection parameters
# for enable github login/singin.
#GITHUB_API_CLIENT_ID = "yourgithubclientid"
#GITHUB_API_CLIENT_SECRET = "yourgithubclientsecret"
```

Restart circus after updating the configuration

```
sudo service circusd restart
```

Update ~/taiga-front-dist/dist/conf.json

```
{
    "api": "https://example.com/api/v1/",
    "eventsUrl": "wss://example.com/events",
    "debug": "true",
    "publicRegisterEnabled": true,
    "feedbackEnabled": true,
    "privacyPolicyUrl": null,
    "termsOfServiceUrl": null,
    "maxUploadFileSize": null
}
```

And nginx:

Reload the nginx configuration

```
sudo service nginx reload
```