Repository Guide

# I. Development environments

- Install Python 3.10.x, PIP and virtual environment.

- Install PostgreSQL or MySQL.

- Create a virtual environment if necessary.

- Install requirements from requirements.txt.

pip install -r requirements.txt

- Install a Python package for PostgreSQL or MySQL.

PostgreSQL case: pip install psycopg

MySQL case: pip install mysqlclient

- Create an environment file from .env.example and configurate it.

- Migrate database with the Django “migrate” command.

python manage.py migrate

- Run the dev server with the Django “runserver” command.

python manage.py runserver

or

python manage.py runserver <port number>

- Execute Celery commands.

beat:

celery -A project beat -l info --scheduler django\_celery\_beat.schedulers:DatabaseScheduler

worker:

celery -A project worker -l info -P gevent

# II. Deployment

- Install Python 3.10.x.

- Install PostgreSQL or MySQL.

- Create a virtual environment if necessary.

- Install requirements from requirements.txt.

pip install -r requirements.txt

- Install a Python package for PostgreSQL or MySQL.

- Create an environment file from .env.example and configurate it.

- Migrate database with the Django “migrate” command.

python manage.py migrate

- Install uwsgi or gunicorn and create a WSGI socket with them. Uwsgi is used

e.g. (on an Ubuntu OS)

1. Install uwsgi with the following command.

pip install uwsgi

1. Create a uwsgi configuration file (\*.ini).

[uwsgi]

uid=<owner of the repository folder>

base=<upper level of repository root folder>

home= <virtual environment path>

chdir=<the path of the folder includes manage.py>

module=project.wsgi:application

env=DJANGO\_SETTINGS\_MODULE=project.settings

master=true

processes=5

socket=%(base)/run/uwsgi.sock

logto=%(base)/logs/uwsgi.log

chown-socket=%(uid):<user-group (e.g. www-data)>

chmod-socket=660

vacuum=true

1. Create a uwsgi service file

e.g. (on an Ubuntu OS)

[Unit]

Description=uWSGI Emperor service

[Service]

ExecStart=<virtual environment path>/bin/uwsgi --emperor <uwsgi conf file path>

User=<owner of the repository folder>

Group=<user-group>

Restart=on-failure

KillSignal=SIGQUIT

Type=notify

NotifyAccess=all

StandardError=syslog

[Install]

WantedBy=multi-user.target

1. Enable the above service

- Enable a uwsgi service

sudo systemctl start uwsgi

sudo systemctl enable uwsgi

- Install nginx

- Configurate nginx for the created WSGI socket.

sudo vim /etc/nginx/sites-available/default

# file beign

upstream django {

server unix:<path for uwsgi.sock>;

}

server {

listen 80 default\_server;

listen [::]:80 default\_server;

charset utf-8;

server\_name \_;

location /static/ {

root <repository root folder>;

}

location /media/ {

root <repository root folder>;

}

location / {

include /etc/nginx/uwsgi\_params;

uwsgi\_pass django;

}

}

# file end

sudo systemctl restart nginx

- Install Supervisor, Redis for Celery tasks .

sudo apt-get install redis-server

sudo systemctl enable redis-server.service

sudo apt-get install supervisor

- Create Supervisor configuration files for Celery beat and worker.

/etc/supervisor/conf.d/celery\_beat.conf

[program:celerybeat]

directory=<repository root folder>

command=<virtual environment path>/bin/celery -A project beat -l info --scheduler django\_celery\_beat.schedulers:DatabaseScheduler

user=<owner of the repository folder>

numprocs=1

stdout\_logfile=<custom celery log path>/beat-access.log

stderr\_logfile=<custom celery log path>/beat-error.log

stdout\_logfile\_maxbytes=50

stderr\_logfile\_maxbytes=50

stdout\_logfile\_backups=10

stderr\_logfile\_backups=10

autostart=true

autorestart=true

startsecs=10

; Causes supervisor to send the termination signal (SIGTERM) to the whole process group.

stopasgroup=true

; if rabbitmq is supervised, set its priority higher

; so it starts first

priority=999

/etc/supervisor/conf.d/celery\_worker.conf

[program:celery]

directory=<repository root folder>

command=<virtual environment path>/bin/celery -A project worker -l info

user=<owner of the repository folder>

numprocs=1

stdout\_logfile=<custom celery log path>/worker-access.log

stderr\_logfile=<custom celery log path>/worker-error.log

stdout\_logfile\_maxbytes=50

stderr\_logfile\_maxbytes=50

stdout\_logfile\_backups=10

stderr\_logfile\_backups=10

autostart=true

autorestart=true

startsecs=10

; Need to wait for currently executing tasks to finish at shutdown.

; Increase this if you have very long running tasks.

stopwaitsecs = 600

; Causes supervisor to send the termination signal (SIGTERM) to the whole process group.

stopasgroup=true

; Set Celery priority higher than default (999)

; so, if rabbitmq is supervised, it will start first.

priority=1000

- Run Supervisor

sudo supervisorctl reread

sudo supervisorctl update

sudo supervisorctl start all

# III. Project Structure

- pyazureboard

It includes DB models, celery tasks for Azure Board.

- pybacklog

It includes DB models, celery tasks for Backlog.

- pychatwork

It includes DB models, celery tasks for Chatwork.

- pyconfluence

It includes DB models, celery tasks for Confluence.

- pygithub

It includes DB models, celery tasks for Github.

- pygitlab

It includes DB models, celery tasks for Gitlab.

- pyjira

It includes DB models, celery tasks for Jira.

- pyredmine

It includes DB models, celery tasks for Redmine.

- pyslack

It includes DB models, celery tasks for Slack.

- pytrello

It includes DB models, celery tasks for Trello.

- dbanalysis

It includes DB models to store necessary data from the dashboard and Rest APIs to send data to the dashboard.

# IV. Admin Panel

1. Create the first super user or super administrator if it doesn’t exist;

- Execute the following the command and input necessary information.

python manage.py createsuperuser

2. Visit the admin panel with a superuser.

- Visit the following link via a web browser.

<website domain>/platform-admin/

- Login with a created superuser

- Click one of db models on the left side bar.

- Do necessary CRUD actions on the main content.

# V. Celery Task Management

1. Visit the admin panel.

2. Click Periodic Tasks on the left side bar.

3. Register Celery tasks.

- Create intervals first.

- Create periodic tasks with created intervals.

**PS:** Necessary tasks are as below.

- Tasks for tools

pyazureboard.tasks.update\_azureboard\_data

pybacklog.tasks.update\_backlog\_data

pychatwork.tasks.update\_trello\_data

pyconfluence.tasks.update\_confluence\_data

pygitlab.tasks.update\_gitlab\_data

pygithub.tasks.update\_github\_data

pyjira.tasks.update\_jira\_data

pyredmine.tasks.update\_redmine\_data

pyslack.tasks.update\_slack\_data

pytrello.tasks.update\_trello\_data

- Tasks to get updates from the dashboard

dbanalysis.tasks.manage\_project\_tools

dbanalysis.tasks.manage\_tools

# VI. Rest APIs

- Tickets

<website domain>/api/v1/tickets/

- Sprints

<website domain>/api/v1/sprints/

- Channels

<website domain>/api/v1/channels/

- Messages

<website domain>/api/v1/messages/

- Mentions

<website domain>/api/v1/mentions/

- Wiki pages

<website domain>/api/v1/documents/

- Pull Requests

<website domain>/api/v1/pull-requests/

- Pull Request Reviewers

<website domain>/api/v1/prequest-reviewers/

- Batch Logs

<website domain>/api/v1/batch-logs/

- Developers

<website domain>/api/v1/developers/