Taskstack Documentation

(work in progress)

[Tech Stack 4](#_Toc48563988)

[Python, JavaScript (vanilla), HTML, CSS 4](#_Toc48563989)

[Flask as a web application framework 4](#_Toc48563990)

[Flask-SocketIO on the server + SOCKET.IO on client-side for bi-directional communication between the clients and the server(s) 4](#_Toc48563991)

[uWSGI as the application server 4](#_Toc48563992)

[NGINX web server as a reverse proxy and load balancer (with sticky sessions) for the application server(s) 4](#_Toc48563993)

[MySQL database + SQLAlchemy 4](#_Toc48563994)

[Redis as a message queue (for Flask-SocketIO to support multiple servers) 4](#_Toc48563995)

[Docker (+ Docker-Compose) 4](#_Toc48563996)

[AWS EC2 to host the server(s) 4](#_Toc48563997)

[AWS S3 to host static files + client uploaded files 4](#_Toc48563998)

[AWS RDS to host the database 4](#_Toc48563999)

[AWS SES for email sending 4](#_Toc48564000)

[Namecheap 4](#_Toc48564001)

[Notes for development 5](#_Toc48564002)

[Running for development 6](#_Toc48564003)

[Requirements 6](#_Toc48564004)

[Start 6](#_Toc48564005)

[Set environment variables -> /taskstack/.env 6](#_Toc48564006)

[Run the development server 6](#_Toc48564007)

[Deployment 7](#_Toc48564008)

[Server requirements 7](#_Toc48564009)

[Initial deployment 7](#_Toc48564010)

[Get the code 7](#_Toc48564011)

[Set environment variables -> /taskstack/.env 7](#_Toc48564012)

[Disable THP 7](#_Toc48564013)

[Build and start 7](#_Toc48564014)

[Enable https 7](#_Toc48564015)

[Updating 7](#_Toc48564016)

[Stop affected service (taskstack |nginx |redis) 7](#_Toc48564017)

[Get the code 7](#_Toc48564018)

[Re-build and start affected service (taskstack |nginx |redis) 7](#_Toc48564019)

[Code style: naming conventions 8](#_Toc48564020)

[Python 8](#_Toc48564021)

[Javascript 8](#_Toc48564022)

[Db: table and column names 8](#_Toc48564023)

[SocketIO/ Fetching 8](#_Toc48564024)

[HTML/ CSS 8](#_Toc48564025)

[UX Design 9](#_Toc48564026)

[Button placement and order 9](#_Toc48564027)

[Dialog 9](#_Toc48564028)

[Page 9](#_Toc48564029)

# Tech Stack

## Python, JavaScript (vanilla), HTML, CSS

## Flask as a web application framework

## Flask-SocketIO on the server + SOCKET.IO on client-side for bi-directional communication between the clients and the server(s)

## uWSGI as the application server

## NGINX web server as a reverse proxy and load balancer (with sticky sessions) for the application server(s)

## MySQL database + SQLAlchemy

## Redis as a message queue (for Flask-SocketIO to support multiple servers)

## Docker (+ Docker-Compose)

## AWS EC2 to host the server(s)

## AWS S3 to host static files + client uploaded files

## AWS RDS to host the database

## AWS SES for email sending

## Namecheap

# Notes for development

* always import the render\_template func from taskstack/app
* gevent monkey-patching is done in production environment (by uWSGI)

# Running for development

## Requirements

* Python 3.\*
* Python modules: /taskstack/requirements.txt (except uwsgi, gevent, redis)
* Local MySQL database
* AWS CLI
* Port 5000 open

## Start

### Set environment variables -> /taskstack/.env

FLASK\_ENV=development  
TASKSTACK\_DB\_URI=mysql+pymysql://[username]:[password]@127.0.0.1/[db name]   
TASKSTACK\_SECRET\_KEY=?  
AWS\_ACCESS\_KEY\_ID=?  
AWS\_SECRET\_ACCESS\_KEY=?  
AWS\_DEFAULT\_REGION=eu-central-1

### Run the development server

* Execute /taskstack/wsgi.py  
  Access the server -> <http://127.0.0.1:5000>

# Deployment

## Server requirements

* OS: Ubuntu
* Docker (+ Docker-Compose)
* Git
* Port 80 and 443 open for all incoming and outgoing connections

## Initial deployment

### Get the code

* git clone https://github.com/juliuskrahn/taskstack

### Set environment variables -> /taskstack/.env

FLASK\_ENV=production  
TASKSTACK\_DB\_URI=?  
TASKSTACK\_SECRET\_KEY=?  
AWS\_ACCESS\_KEY\_ID=?  
AWS\_SECRET\_ACCESS\_KEY=?  
AWS\_DEFAULT\_REGION=eu-central-1

### Disable THP

* echo never > /sys/kernel/mm/transparent\_hugepage/enabled

(needs to be done again if the server is shut down)

### Build and start

* docker-compose build
* docker-compose up -d

### Enable https

* docker exec -it nginx bash
* sudo certbot –nginx

## Updating

Stop affected service (taskstack |nginx |redis)

* docker-compose stop [service]

Get the code

* git pull

Re-build and start affected service (taskstack |nginx |redis)

* docker-compose build [service]
* docker-compose up -d [service]

(If you need to rebuild the nginx service, make sure to enable https again!)

# Code style: naming conventions

## Python

* Globals (hard coded string/ number): uppercase; words separated by underscores (A\_GLOBAL\_VARIABLE)
* Variable/ function name: lowercase; words separated by underscores (my\_variable)
* Class: upper camel case (MyClass)
* Data dict for client side rendering - keys: camel case (myKey)

## Javascript

* Globals (hard coded string/ number): uppercase; words separated by underscores (A\_GLOBAL\_VARIABLE)
* Local variable:
  + If refers to something: style of the name referred to (suffix/ prefix: lowercase; words separated by underscores)
  + Else: lowercase; words separated by underscores
* Variable/ function name: camel case (myVariable) (if refers to something -> suffix/ prefix: lowercase; words separated by underscores)
* Func collection object: upper camel case (MyFuncCollection)
* Class: upper camel case (MyClass)

## Db: table and column names

* lowercase; words separated by underscores (my\_table)

## SocketIO/ Fetching

* SocketIO event name: lowercase; words separated by underscores (my\_event)
* Json/ dict data key naming: camel case (myKey)

## HTML/ CSS

* Id/ class name: camel case (myTextBox) (prefixes: lowercase; separated by a dash)

# UX Design

## Button placement and order

### Dialog

Main

Additional

Cancel

### Page

Back

Main

Additional