thunderpush Documentation

Release 0.9.7

Krzysztof Jagiello

Aug 03, 2017

Contents

This documentation contains everything you need to know about Thunderpush.

CHAPTER 1

What is Thunderpush?

Have you ever wanted to push out data to web browsers in real-time? Thunderpush is what you are looking for! Fire up Thunderpush, include a simple JavaScript file on your site and you are ready to go. Sending out messages to clients is a piece of cake. Thunderpush provides you a HTTP API you can use to send out messages in JSON format. Thunderpush is powered by the awesome SockJS library that makes sure that your users get messages in real-time regardless of the browser being used.

CHAPTER 2

First steps

Quick start

Installing Thunderpush

To install Thunderpush server using pip:

```
pip install thunderpush
```

Starting the server

Note: Read *Deploying to production* for a recommended way to run Thunderpush in production environment.

Help message for Thunderpush:

```
usage: thunderpush [-h] [-p PORT] [-H HOST] [-v] [-d] [-V] clientkey apikey

positional arguments:
  clientkey             client key
  apikey                server API key

optional arguments:
  -h, --help            show this help message and exit
  -p PORT, --port PORT  binds server to custom port
  -H HOST, --host HOST  binds server to custom address
  -v, --verbose         verbose mode
  -d, --debug           debug mode (useful for development)
  -V, --version         show program's version number and exit
```

To start Thunderpush on *localhost:8000* with *publickey* as apikey and *secret* as apisecret you would do following:

```
thunderpush -H localhost -p 8000 publickey secret
```

Deploying to production

When running an application in production, you want to make sure that it stays alive, even if a crash occurs. A way to do it is using supervisord which is a great Python process management tool.

Assuming that you have already installed supervisord, add following snippet of code to the configuration file:

```
[program:thunderpush]
command=/usr/local/bin/thunderpush -p 8000 apikey apisecret
user=thunderpush
```

Note: We recommend creating a separate user for running Thunderpush, although it's entirely possible to run it as *root*, but simply removing *user=thunderpush*.

Now we're ready to start the server:

```
supervisorctl start thunderpush
```

The server is now running at port 8000, but we want ideally to run it on port 80 alongside your web server.

To Be Continued...

Quick start A quick guide to running Thunderpush.

CHAPTER 3

Reference

Server HTTP API

Example of interacting with Thunderpush API using cURL:

```
curl \
    -X POST \
    -H "Content-Type: application/json" \
    -H "X-Thunder-Secret-Key: secretkey" \
    --data-ascii "\"Hello World!\"" \
    http://thunder.example.com/api/1.0.0/[API key]/channels/[channel]/
```

All requests to the HTTP API must include *X-Thunder-Secret-Key* header that should contain the private API key.

Available endpoints

Sending a message to a channel

```
POST /api/1.0.0/[API key]/channels/[channel]/
```

Message should be sent as the body of the request. Only valid JSON body will be accepted.

Getting number of users online

```
GET /api/1.0.0/[API key]/users/
```

Checking presence of a user

```
GET /api/1.0.0/[API key]/users/[user id]/
```

Sending a message to a user

```
POST /api/1.0.0/[API key]/users/[user id]/
```

Message should be sent as the body of the request. Only valid JSON body will be accepted.

Forcing logout of a user

```
DELETE /api/1.0.0/[API key]/users/[user id]/
```

Always returns 204 http code.

Retrieving list of users in a channel

```
GET /api/1.0.0/[API key]/channels/[channel]/
```

Server HTTP API Learn how to use the HTTP API and see the list of all the available endpoints.