makechat Documentation

Release 0.1.24

Andrew Burdyug

May 28, 2016

Table of Contents

CHAPTER 1

Introduction

There are many chat programs, but almost all of them have the same disadvantages:

- not open source, possible have many hidden bugs/leaks
- you can not run own chat server and use it for own aims, you should use central server which you can not control, it is bad:
  - it may not working properly or restrict connections by country due to sanctions, for example HipChat is blocking Crimean users
  - your private messages/accounts could fall into the wrong hands, because you can't control how the central server is safe

In some cases, if your work is depent on chat systems and you need to have it working without unexpected outage, these desadvantages unacceptable for you. **Makechat** provide you simple chat system, which you might fully control. You might create public/private chat rooms, write your own notifier/bot etc, easy move/run your chat server to anywhere, where Docker works.

1.1 Getting started

1.1.1 System requirements

- Docker
- MongoDb
- Python 3.x

Makechat will store all data(accounts/rooms/messages etc) into *MongoDb*, for easy setup we use docker containers, you do not warry about complecated setup procedures.

1.1.2 Installation

Make these steps:

1. Install docker
2. Run docker containers:

```
$ sudo mkdir -pv /makechat-backups /var/lib/makechat-mongo /var/www/makechat
$ sudo chmod 700 /makechat-backups /var/lib/makechat-mongo
$ echo "172.30.1.1 makechat-mongo" | sudo tee --append /etc/hosts
$ echo "172.30.1.2 makechat" | sudo tee --append /etc/hosts
$ echo "172.30.1.3 makechat-web" | sudo tee --append /etc/hosts
$ docker network create -d bridge --subnet 172.30.0.0/16 makechat_nw
$ docker run --net=makechat_nw --ip=172.30.1.1 -v /var/lib/makechat-mongo:/data/db \
    --name makechat-mongo -d mongo:latest
$ docker run --net=makechat_nw --ip=172.30.1.2 -v /makechat-backups:/backups \
    --name makechat -d buran/makechat:latest
$ docker run --net=makechat_nw --ip=172.30.1.3 --name makechat-web \
    -v /var/www/makechat:/usr/share/nginx/html/makechat/custom \
    -d buran/makechat-web:latest
```

3. Edit `~/makechat.conf`

   Note: Currently `makechat.conf` placed inside home directory of user who installed the **makechat** python package.

4. Restart backend:

```
$ docker restart makechat
```

5. Go to `http://youdomain.com/makechat/admin` and create user accounts/rooms

1.2 Management commands

1.2.1 User management

- Print help about available actions:

```
$ docker exec makechat makechat user -h
usage: makechat user [-h] {create,changepass} ...

positional arguments:
  {create,changepass}
    create              create a new user
    changepass          change user password

optional arguments:
  -h, --help            show this help message and exit
```

- Print help about `user create` action:

```
$ docker exec makechat makechat user create -h
usage: makechat user create [-h] -u USERNAME -p PASSWORD -e EMAIL [-admin]

optional arguments:
  -h, --help   show this help message and exit
  -u USERNAME  specify username
  -p PASSWORD  specify password
  -e EMAIL     specify email address
  -admin       is superuser?
```

- Print help about `user changepass` action:

```
$ docker exec makechat makechat user changepass -h
usage: makechat user changepass [-h] -u USERNAME -p NEW PASSWORD

optional arguments:
  -h, --help       show this help message and exit
  -u USERNAME      specify username
  -p NEW PASSWORD  specify new password
```

- Add a new user account:

```
$ docker exec makechat makechat user create -u test_user -p test_pass -e test@example.com
```

- Add a new superuser(aka admin) account:

```
$ docker exec makechat makechat user create -u admin -p admin_pass -e admin@example.com -admin
```

1.3 Upgrade

Make these steps:

1. Backup **makechat** instance:

```
$ docker exec makechat backup
```

2. Inform users about maintenance:

```
$ docker exec makechat-web maintenance on
```

3. Update docker images:

```
$ docker pull buran/makechat
$ docker pull buran/makechat-web
```

4. Stop **makechat-web** container and remove it:

```
$ docker stop makechat-web && docker rm makechat-web
```

Note: Usually you do not to worry about downtime of frontend, because time of creation new makechat-web instance ~2-5 seconds, so your users may noticed only small lags. But if you want to enable maintenance page for a time of update **makechat-web**, you should use **makechat-web** behind frontend web server(Nginx/Apache etc) and make appropriate changes to its configuration. For example, if you have Nginx as frontend web server for **makechat-web** docker instance, you should make something like this:

```
server {
    listen 80;
    server_name mymakechat.com;
    error_page 503 /maintenance.html;

    location / {
        return 503;
    }

    location = /maintenance.html {
        root /path/to/maintenance.html;
        internal;
```

```
        }
    }
```

5. Create new **makechat-web** container with latest public content and nginx configuration:

```
$ docker run --net=makechat_nw --ip=172.30.1.3 --name makechat-web \
    -v /var/www/makechat:/usr/share/nginx/html/makechat/custom \
    -d buran/makechat-web:latest
```

6. Stop **makechat** container and remove it:

```
$ docker stop makechat && docker rm makechat
```

7. Create new **makechat** container with latest **makechat** package:

```
$ docker run --net=makechat_nw --ip=172.30.1.2 -v /makechat-backups:/backups \
    --name makechat -d buran/makechat:latest
```

8. Stop maintenance:

```
$ docker exec makechat-web maintenance off
```

1.4 API documentation

1.4.1 Read this first

Foremost you need create `token` or use already created when you have completed sign up process. Token is a random phrase with 32 chars(ascii lowercase + digits). For testing of API access you can use curl or httpie command line tools or anything else which can communicate with server over HTTP protocol. You should specify your `token` in HTTP header `X-Auth-Token`.

Playing with API

Example request with `curl`:

```
$ curl -v -H "X-Auth-Token:4388c735cba9455cae341cbf17ed2198" http://makechat-web/api/rooms
* Hostname was NOT found in DNS cache
*   Trying 172.30.1.3...
* Connected to makechat-web (172.30.1.3) port 80 (#0)
> GET /api/rooms HTTP/1.1
> User-Agent: curl/7.35.0
> Host: makechat-web
> Accept: */*
> X-Auth-Token:4388c735cba9455cae341cbf17ed2198
>
< HTTP/1.1 200 OK
* Server nginx/1.9.11 is not blacklisted
< Server: nginx/1.9.11
< Date: Thu, 07 Apr 2016 22:13:14 GMT
< Content-Type: application/json; charset=utf-8
< Content-Length: 304
< Connection: keep-alive
< Cache-Control: max-age=86400, must-revalidate, no-store
< Vary: Accept-Encoding
```

```
<
* Connection #0 to host makechat-web left intact
[{"members": [{"$oid": "5706dac4d55cfe00010944ac"}], "is_open": true, "is_visible": true, "_id": {"$o
```

or less verbose and more readable:

```
$ curl -s -H "X-Auth-Token:4388c735cba9455cae341cbf17ed2198" http://makechat-web/api/rooms| python -m
[
    {
        "_id": {
            "$oid": "5706dac4d55cfe00010944ad"
        },
        "is_open": true,
        "is_visible": true,
        "members": [
            {
                "$oid": "5706dac4d55cfe00010944ac"
            }
        ],
        "name": "room1"
    },
    {
        "_id": {
            "$oid": "5706dac4d55cfe00010944ae"
        },
        "is_open": true,
        "is_visible": true,
        "members": [
            {
                "$oid": "5706dac4d55cfe00010944ac"
            }
        ],
        "name": "room2"
    }
]
```

or golden mean by using `httpie`:

```
http -v GET http://makechat-web/api/rooms X-Auth-Token:4388c735cba9455cae341cbf17ed2198
```

Request:

```
GET /api/rooms HTTP/1.1
Accept: */*
Accept-Encoding: gzip, deflate
Connection: keep-alive
Host: makechat-web
User-Agent: HTTPie/0.9.3
X-Auth-Token: 4388c735cba9455cae341cbf17ed2198
```

Response:

```
HTTP/1.1 200 OK
Cache-Control: max-age=86400, must-revalidate, no-store
Connection: keep-alive
Content-Length: 304
Content-Type: application/json; charset=utf-8
Date: Thu, 07 Apr 2016 22:26:12 GMT
Server: nginx/1.9.11
Vary: Accept-Encoding
```

```
[
    {
        "_id": {
            "$oid": "5706dac4d55cfe00010944ad"
        },
        "is_open": true,
        "is_visible": true,
        "members": [
            {
                "$oid": "5706dac4d55cfe00010944ac"
            }
        ],
        "name": "room1"
    },
    {
        "_id": {
            "$oid": "5706dac4d55cfe00010944ae"
        },
        "is_open": true,
        "is_visible": true,
        "members": [
            {
                "$oid": "5706dac4d55cfe00010944ac"
            }
        ],
        "name": "room2"
    }
]
```


Note: In examples we will use httpie for interaction with API, because it produces beautiful output with indentation aligned of JSON data.

Examples of bad responses

There are many typical mistakes of interaction with API. Do not panic, just inspect your code one more time and look the API responses more closely: almost all (except 405 Method Not Allowed error) they have `title` and `description`, so you can guess what the problem is.

- Request with not valid token or wrong credentials:

  Request:

```
GET /api/rooms HTTP/1.1
Accept: */*
Accept-Encoding: gzip, deflate
Connection: keep-alive
Host: makechat-web
User-Agent: HTTPie/0.9.3
X-Auth-Token: 4388c735cba9455cae341cbf17ed2191
```

  Response:

```
HTTP/1.1 401 Unauthorized
Connection: keep-alive
Content-Length: 99
Content-Type: application/json
```

```
Date: Thu, 07 Apr 2016 22:59:44 GMT
Server: nginx/1.9.11

{
    "description": "Please provide an auth token or login.",
    "title": "Not authentificated"
}
```

- Request with valid token but without admin permissions (for some actions, you need administrator rights):

  Request:

```
GET /api/rooms HTTP/1.1
Accept: */*
Accept-Encoding: gzip, deflate
Connection: keep-alive
Host: makechat-web
User-Agent: HTTPie/0.9.3
X-Auth-Token: 52115268596140298df2a640f37eea57
```

  Response:

```
HTTP/1.1 403 Forbidden
Connection: keep-alive
Content-Length: 74
Content-Type: application/json
Date: Thu, 07 Apr 2016 23:01:49 GMT
Server: nginx/1.9.11

{
    "description": "Admin required.",
    "title": "Permission Denied"
}
```

- Request with not acceptable content type:

  Request:

```
PUT /api/rooms HTTP/1.1
Accept: */*
Accept-Encoding: gzip, deflate
Connection: keep-alive
Content-Length: 0
Host: makechat-web
User-Agent: HTTPie/0.9.3
X-Auth-Token: 52115268596140298df2a640f37eea57
```

  Response:

```
HTTP/1.1 406 Not Acceptable
Connection: keep-alive
Content-Length: 267
Content-Type: application/json
Date: Thu, 07 Apr 2016 23:21:38 GMT
Server: nginx/1.9.11

{
    "description": "This API only supports responses encoded as JSON.",
    "link": {
        "href": "http://docs.examples.com/api/json",
```

```
            "rel": "help",
            "text": "Documentation related to this error"
        },
        "title": "Media type not acceptable"
    }
```

- Request with not allowed method:

  Request:

```
    PUT /api/rooms HTTP/1.1
    Accept: application/json
    Accept-Encoding: gzip, deflate
    Connection: keep-alive
    Content-Length: 18
    Content-Type: application/json
    Host: makechat-web
    User-Agent: HTTPie/0.9.3
    X-Auth-Token: 52115268596140298df2a640f37eea57

    {
        "data": "test"
    }
```

  Response:

```
    HTTP/1.1 405 Method Not Allowed
    Connection: keep-alive
    Content-Length: 0
    Date: Thu, 07 Apr 2016 23:22:09 GMT
    Server: nginx/1.9.11
    allow: GET, POST, OPTIONS
```

- Request without required data:

  Request:

```
    POST /api/rooms HTTP/1.1
    Accept: application/json
    Accept-Encoding: gzip, deflate
    Connection: keep-alive
    Content-Length: 18
    Content-Type: application/json
    Host: makechat-web
    User-Agent: HTTPie/0.9.3
    X-Auth-Token: 4388c735cba9455cae341cbf17ed2198

    {
        "trash": "trash"
    }
```

  Response:

```
    HTTP/1.1 400 Bad Request
    Connection: keep-alive
    Content-Length: 92
    Content-Type: application/json
    Date: Thu, 07 Apr 2016 23:24:52 GMT
    Server: nginx/1.9.11
```

```
        {
            "description": "The 'name' parameter is required.",
            "title": "Missing parameter"
        }
```

1.4.2 API endpoints

/api/rooms

GET /api/rooms
Get all chat rooms.

Request:

```
GET /api/rooms HTTP/1.1
Accept: */*
Accept-Encoding: gzip, deflate
Connection: keep-alive
Host: makechat-web
User-Agent: HTTPie/0.9.3
X-Auth-Token: 4388c735cba9455cae341cbf17ed2198
```

Response:

```
HTTP/1.1 200 OK
Cache-Control: max-age=86400, must-revalidate, no-store
Connection: keep-alive
Content-Length: 304
Content-Type: application/json; charset=utf-8
Date: Thu, 07 Apr 2016 22:26:12 GMT
Server: nginx/1.9.11
Vary: Accept-Encoding

[
    {
        "_id": {
            "$oid": "5706dac4d55cfe00010944ad"
        },
        "is_open": true,
        "is_visible": true,
        "members": [
            {
                "$oid": "5706dac4d55cfe00010944ac"
            }
        ],
        "name": "room1"
    },
    {
        "_id": {
            "$oid": "5706dac4d55cfe00010944ae"
        },
        "is_open": true,
        "is_visible": true,
        "members": [
            {
                "$oid": "5706dac4d55cfe00010944ac"
            }
```

```
            ],
            "name": "room2"
        }
    ]
```

Status Codes

- 200 OK – ok
- 401 Unauthorized – not valid token
- 403 Forbidden – not have admin permissions
- 405 Method Not Allowed – not allowed method
- 406 Not Acceptable – not acceptable content type

POST /api/rooms

Create a new chat room with given name.

Request:

```
POST /api/rooms HTTP/1.1
Accept: application/json
Accept-Encoding: gzip, deflate
Connection: keep-alive
Content-Length: 21
Content-Type: application/json
Host: makechat-web
User-Agent: HTTPie/0.9.3
X-Auth-Token: 4388c735cba9455cae341cbf17ed2198

{
    "name": "test_room"
}
```

Response:

```
HTTP/1.1 201 Created
Cache-Control: max-age=86400, must-revalidate, no-store
Connection: keep-alive
Content-Length: 154
Content-Type: application/json; charset=utf-8
Date: Thu, 07 Apr 2016 23:50:08 GMT
Server: nginx/1.9.11
Vary: Accept-Encoding

{
    "_id": {
        "$oid": "5706f230d55cfe00010944af"
    },
    "is_open": true,
    "is_visible": true,
    "members": [
        {
            "$oid": "5706dac4d55cfe00010944ac"
        }
    ],
    "name": "test_room"
}
```

Request JSON Object

- **name** (*string*) – the name of chat room

Response JSON Object

- **_id** (*object*) – `id` of chat room
- **name** (*string*) – `name` of chat room
- **members** (*array*) – `members` of chat room
- **is_open** (*boolean*) – `true` if chat room is open
- **is_visible** (*boolean*) – `true` if chat room is visible

Status Codes

- 201 Created – ok
- 400 Bad Request – not have required data
- 401 Unauthorized – not valid token
- 403 Forbidden – not have admin permissions
- 405 Method Not Allowed – not allowed method
- 406 Not Acceptable – not acceptable content type

/api/members

/api/messages

1.5 Developers guide

1.5.1 Read this first

makechat.api

makechat.api.utils module

All utils are should be placed here.

This module contain helpers function.

`makechat.api.utils.`**`encrypt_password`**(*password*)
Encrypt plain passowrd.

`makechat.api.utils.`**`session_create`**(*user*)
Create session.

`makechat.api.utils.`**`token_create`**(*user*, *name*)
Cretae a token.

makechat.api.rooms module

All logic of /api/rooms endpoint should be described here.

class makechat.api.rooms.**RoomResource**(*items_per_page*)
Bases: object

Chat room resource.

on_delete(*req*, *resp*, *room_id*)
Process DELETE requests for /api/rooms.

on_get(*req*, *resp*)
Process GET requests for /api/rooms.

on_patch(*req*, *resp*, *room_id*)
Process PUT requests for /api/rooms.

on_post(*req*, *resp*)
Process POST requests for /api/rooms.

on_put(*req*, *resp*, *room_id*)
Process PUT requests for /api/rooms.

m

/api

E

M

O

R

S

T