MySocket

Andree Toonk

Nov 14, 2020

ABOUT MYSOCKET

Welcome to the documentation for Mysocket.io, a service that provides you with fast and secure network connectivity whenever you need it, wherever you are.

Note: This documentation is an open source project. We always appreciate your feedback and improvements.

You can submit an issue or pull request on the GitHub repository,

---

The main documentation is organized into the following sections:

CHAPTER

ONE

INTRODUCTION

Welcome to the documentation for Mysocket.io, a service that provides you with fast and secure network connectivity whenever you need it, wherever you are.

1.1 About Mysocket

Mysocket.io is a service that provides public endpoints for services that are otherwise not publicly reachable. A typical example is a web service running on your laptop, which you'd like to make available to your client. Or ssh access to servers behind NAT or a firewall, like a raspberry pi on your home network. Mysocket.io is a fully managed cloud service, so nothing to run!

1.2 About this Documentation

The goal is for the documentation to be continuously updated and improved.

Note: You can contribute to the documentation by opening an issue or sending patches via pull requests on the GitHub source repository.

CHAPTER

TWO

QUICK START START

More documentation can be found below; but if you're eager to get started, consider this a quick start. Download the mysocketctl (MacOSx and Linux) Create an account:

```
mysocketctl.py account create \
    --name "your_name" \
    --email "your_email_address" \
    --password "a_secure_password" \
    --sshkey "$(cat ~/.ssh/id_rsa.pub)"
```

After confirming your new account (check your email), login and retrieve an access token:

```
mysocketctl.py login --email \
    --email "your_email_address" \
    --password "a_secure_password" \
```

Now you're ready to use the "quick connect" feature to connect your local service listening on port 8000 to the Internet:

```
mysocketctl.py connect \
    --port 8000 \
    --name "my test service"

+--------------------------------------+---------------------------------+------------
↪-----+
|               socket_id              |             dns_name            |        name␣
↪      |
+--------------------------------------+---------------------------------+------------
↪-----+
| 24cd3fa2-b56a-486f-a554-30924bae54d0 | long-hill-8399.edge.mysocket.io | my test␣
↪service |
+--------------------------------------+---------------------------------+------------
↪-----+

Connecting to Server: ssh.mysocket.io

Welcome to Mysocket.io!
my test service - https://long-hill-8399.edge.mysocket.io

=========================================================
Logs
=========================================================
<live stream of your logs here>
```

To quickly test if it's working, you can start a web service on localhost port 8000 like this.

```
python3 -m http.server 8000
```

Now visit the URL (dns_name) using your browser, and you'll see that the localhost service you just started is now globally available!

CHAPTER

THREE

FEATURES

Stable public DNS names and port numbers for your private apps.

Supports various socket types, including:

1. http
2. https
3. TCP
4. TLS

Options for SSL/TLS Encryption for your sockets

All sockets run on a global anycast network, reducing latency and improving the user experience.

Username and Password protected (http/https) Sockets

Live Stream of logs. We show you all requests in real-time, including the latency between our anycasted nodes and your origin server.

Support for multiple origins per socket, ie. Load Balancing

CHAPTER

FOUR

BUILD ON A GLOBAL ANYCAST NETWORK

Mysocket.io is built on a global anycasted network of **91 Points of Presence in 80 cities across 42 countries.** This helps you improve the availability and performance of the applications that you offer to your global users. Mysocket.io application services connect to use anycast network using various servers in North America, Europe, and Asia. All this provides us with the best possible low latency user experience and Instant regional failover, which results in an incredible level of high availability.

CHAPTER

FIVE

EXAMPLE USE CASES

5.1 Make the local web service on your laptop available to your colleagues or client.

You may prefer to do web development on your laptop, and, before publishing it to some public server, would like to share it quickly with your teammate or client. Using Mysocket.io you can make the web app running on localhost, publicly available to anyone on the Internet. Just share the mysocket.io generated URL with those with who you'd like to share it. If you'd like, you can even make it password protected.

5.2 Access your raspberry pi at home from anywhere on the Internet

You have a small lab at home, perhaps with a raspberry pi or Intel nuc. Since these are behind your NAT router you can't normally SSH into them. By using Mysocket.io you can make the SSH services on your home server available by tunneling TCP traffic through the tunnel seamlessly through NAT. Mysocket.io will provide a public DNS name and port number, which can be used to SSH into your server from anywhere.

5.3 A global stable public endpoint for your ephemeral resources.

Your containers come and go, perhaps even distributed over various public clouds as well as your private datacenter. It can be challenging to provide a stable public endpoint for these ephemeral and mobile services. With mysocket.io you can create a public endpoint, either an http/https, or TCP, TLS endpoint. Now each time a new container comes up, it can connect to the mysocket.io service and register as a new origin (backend) server. You can have one, or many of these origin services per public socket.

CHAPTER

SIX

INTERACTING WITH THE MYSOCKET.IO SERVICE

The easiest way to get started with the service is by using the mysocketctl cli tool. More details about that can be found here. All interaction with our services is done using our RESTful API. You can find the API and the API specifications at https://api.mysocket.io/ The mysocketctl tool uses this API to interact with the service.

CHAPTER

SEVEN

FAQ

Note: The contents are available on Github, allowing you to send a pull request with edits or additions, or fork the contents for usage elsewhere.

7.1 What is Mysocket?

xxx

7.2 What can I do with Mysocket?

7.3 What performance improvement does Mysocket provide?

7.4 Where is Mysocket deployed today?

Mysocket.io is built on a global anycast network of 91 Points of Presence in 80 cities across 42 countries. The tunnel and api servers are deployed throughout North America, Europe and Asia.

7.5 What kind of support is provided?

Today support is best effort.

7.6 Q: How do I get started with Mysocket?

7.7 Q: What kind of transport security is used between the mysocket.io and the origin.

We currently support SSHv2 as the transport and tunneling protocol. It encrypts all traffic to eliminate eavesdropping, connection hijacking, and other attacks.

7.8 Q: If I only have one origin server, how do I benefit from the anycast features.

CHAPTER

EIGHT

INTRODUCTION

`mysocketctl` is a cli tool that allows you to easily manage and and use the Mysocket services. `mysocketctl` is a python tool that uses api.mysocket.io REST api to configure the various objects needed to use the services. Using `mysocketctl` users can create and manage their account, as well as manage sockets and tunnels and easily connect to the service.

```
$ mysocketctl.py
Usage: mysocketctl.py [OPTIONS] COMMAND [ARGS]...

Options:
--help  Show this message and exit.

Commands:
account  Create a new account or see account information.
connect  Quckly connect, Wrapper around sockets and tunnels
login    Login to mysocket and get a token
socket   Manage your global sockets
tunnel   Manage your tunnels
```

8.1 About this Documentation

The goal is for the documentation to be continuously updated and improved.

Note: You can contribute to this documentation by opening an issue or sending patches via pull requests on the GitHub source repository.

CHAPTER

NINE

INSTALLING MYSOCKETCTL

The code for mysocketctl can be found here . . .

9.1 The quick way, download the executable

For those who want to get started quickly a binary for Mac OSx and Linux is available.

```
curl ...
chmod +x ./mysocketctl
```

9.2 Using Pip

Or use pip to install mysocketctl

```
pip3 install mysocketctl
```

CHAPTER

TEN

ACCOUNT MANAGEMENT AND LOGIN

```
$ mysocketctl.py account --help
Usage: mysocketctl.py account [OPTIONS] COMMAND [ARGS]...

Create a new account or see account information.

Options:
--help  Show this message and exit.

Commands:
create  Create your new mysocket.io account
show    Show your mysocket.io account information
```

10.1 Creating an account

To use mysocket.io users will need to register and create an account. Use the following command to create an account. Make sure to use a valid email address as we'll use it to send you an email to validate your account.

We need an ssh public key as well, this is what we later use to setup and authenticate tunnels. If you don't know how to create an ssh key pair, please see this link:

https://git-scm.com/book/en/v2/Git-on-the-Server-Generating-Your-SSH-Public-Key

Make sure to upload your public key only.

```
mysocketctl.py account create \
    --name "your_name" \
    --email "your_email_address" \
    --password "a_secure_password" \
    --sshkey "$(cat ~/.ssh/id_rsa.pub)"
```

You should receive an email now with a confirmation link. Please click the link to validate your email account. After that, you can login

10.2 Logging in and get a token

In order to use the service please login like below

```
mysocketctl.py login \
    --email "your_email_address" \
    --password "a_secure_password" \

Logged in! Token stored in /Users/johndoe/.mysocketio_token
```

The login process returns a jwt token that is stored in a `.mysocketio_token` file located in the users home directory. Going forward, `mysocketctl` will use this token to authenticate with the API. Currently, the token is valid for 300 minutes, ie. 5hrs. The user will need to re-issue a login request when the token has expired.

10.3 Account information

To see information about your account, use the following command.

```
mysocketctl account show
+-------------------------------------------------------------------+
| Name          | Andree Toonk                                      |
| Email         | blabla@gmail.com                                  |
| user id       | b2f1b59f-bcba-4286-9818-9f0b6e685e93              |
| ssh username  | b2f1b59fbcba428698189f0b6e685e93                  |
| ssh key       | ssh-rsa <your public key....SNIP TOO lONG>        |
+-------------------------------------------------------------------+
```

CHAPTER

ELEVEN

QUICK CONNECT OPTIONS

The quick-connect function allows users to quickly, ie. in one command:

1. Create a socket
2. Create a tunnel
3. Make a local service available by connecting the tunnel to mysocket.

This quick connect feature is useful for when you want to make a local service available quickly. Later on we'll look at how to configure and manage all the individual components. Every time the connect feature is used, a new socket and, corresponding DNS name is created. If you need more permanent names, please look at creating sockets and tunnels separately.

```
mysocketctl.py connect --help
Usage: mysocketctl.py connect [OPTIONS]

Quckly connect, Wrapper around sockets and tunnels

Options:
--port INTEGER                  Local port to connect  [required]
--name TEXT
--protected TEXT
--protected / --not-protected
--username TEXT
--password TEXT
--type TEXT                     Socket type, http, https, tcp, tls
--help                          Show this message and exit.
```

In the example bellow, we'll connect our local port 8000 to the mysocket service. Mysocket.io will automatically create a socket with a DNS name for you. It will also create a tunnel, which `mysocketctl` will use to connect to automatically.

```
mysocketctl.py connect \
    --port 8000 \
    --name "my test service"
+--------------------------------------+--------------------------------------+-------
↪----------+
|              socket_id               |               dns_name               |      ␣
↪name      |
+--------------------------------------+--------------------------------------+-------
↪----------+
| d84515f7-5c6e-4970-83bb-e25c1ca8cf16 | muddy-darkness-2030.edge.mysocket.io | my␣
↪test service |
+--------------------------------------+--------------------------------------+-------
↪----------+
```

(continues on next page)

(continued from previous page)

```
Connecting to Server: ssh.mysocket.io

Welcome to Mysocket.io!
my test service - https://muddy-darkness-2030.edge.mysocket.io

=========================================================
Logs
=========================================================
....
```

In this case, a socket with the name muddy-darkness-2030.edge.mysocket.io was created. Using your browser, you can now visit this socket which is automatically connected to the http service running on your localhost port 8000. Note, to test this, you can quickly start a localhost http server on port 8000 like this:

```
python3 -m http.server 8000
```

All requests are logged and shown in the `mysocketctl` terminal.

`Ctrl-c` will cause the ssh tunnel to disconnect. Mysocketctl will automatically reconnect the tunnel, this is to recover from possible network issues. To end the quick connect session press `ctrl-c` twice. This will make sure the socket objects are automatically deleted, so you won't hit any of the account limits.

```
^C  (ctr-c)
Connection to ssh.mysocket.io closed.
Disconnected... Automatically reconnecting now..
Press ctrl-c to exit
^C (ctr-c)
Bye
cleaning up...
```

CHAPTER

TWELVE

SOCKET MANAGEMENT

Sockets are the public endpoint that mysocket creates on behalf of users. Each socket will come with a unique DNS name. There are three types of socket supported today:

1. **http/https**. Use this when your local service is a http service.
2. **TCP**. Use this when your local service is a non-http service. In this case mysocket will proxy a raw tcp session. This is used for example for ssh or https services. Note that in this case mysocket will, in addition to a unique DNS name, also create a TCP port number just for your service.
3. **TLS**. This is a TLS encrypted TCP socket. This is great to, for example, make your local mysql service available over TLS.

```
mysocketctl socket --help
Usage: mysocketctl.py socket [OPTIONS] COMMAND [ARGS]...

Manage your global sockets

Options:
--help  Show this message and exit.

Commands:
create
delete
ls
```

12.1 Creating sockets

The command below creates an http socket of type http. It returns the socket_id and dns name.

```
mysocketctl.py socket create \
    --name "my local http service" \
    --type http
+--------------------------------------+-----------------------------------+---------
↪+------+-----------------------+
|              socket_id               |              dns_name             | port(s)␣
↪| type |          name         |
+--------------------------------------+-----------------------------------+---------
↪+------+-----------------------+
| 506182d3-1109-4d94-96f1-3bd7b0de68a9 | frosty-rain-6381.edge.mysocket.io |  80 443␣
↪| http | my local http service |
+--------------------------------------+-----------------------------------+---------
↪+------+-----------------------+
```

For http based services, we can add password protection to the socket. This means that the user will see a username password window before visiting your socket service. Below an example of creating a password-protected socket, with username john and password secret.

```
mysocketctl.py socket create \
    --name "my local http service" \
    --type http \
    --protected \
    --username john \
    --password secret
+--------------------------------------+---------------------------------+---------+--
↪----+-----------------------+
|              socket_id               |             dns_name            | port(s) |␣
↪type |          name         |
+--------------------------------------+---------------------------------+---------+--
↪----+-----------------------+
| 5870a362-65d3-474d-bbf6-3341827eaee0 | dark-darkness-6275.edge.mysocket.io |  80␣
↪443 | http | my local http service |
+--------------------------------------+---------------------------------+---------+--
↪----+-----------------------+

Protected Socket, login details:
+----------+----------+
| username | password |
+----------+----------+
| john     | secret   |
+----------+----------+
```

12.2 Listing all sockets

To see all your socket, issue the socket ls command like below:

```
mysocketctl.py socket ls
+--------------------------------------+-----------------------------------------+----
↪--+---------+-----------------------+
| socket_id                            | dns_name                                |␣
↪type | port(s) | name                  |
+--------------------------------------+-----------------------------------------+----
↪--+---------+-----------------------+
| f441738c-4f77-44d5-bc68-99664f272319 | restless-night-1301.edge.mysocket.io    |␣
↪http | 80 443  | Local port 44         |
| 12967b8a-ccca-4a84-87e6-2443daed5fe5 | frosty-wildflower-4938.edge.mysocket.io |␣
↪http | 80 443  | andree was here       |
| 05da6711-c2c7-4c53-b213-21ea9a3d1db6 | ancient-voice-2982.edge.mysocket.io     |␣
↪http | 80 443  | Local port 8000       |
| 5870a362-65d3-474d-bbf6-3341827eaee0 | wild-pine-1229.edge.mysocket.io         |␣
↪http | 80 443  | my local http service |
+--------------------------------------+-----------------------------------------+----
↪--+---------+-----------------------+
```

12.3 Delete sockets

To delete a socket, issue the socket delete command and provide the socket_id you wish to delete.

```
mysocketctl.py socket delete \
    --socket_id 5870a362-65d3-474d-bbf6-3341827eaee0

Socket 5870a362-65d3-474d-bbf6-3341827eaee0 deleted
```

CHAPTER

THIRTEEN

TUNNEL MANAGEMENT

In the previous section, we looked at managing sockets. Sockets are created on the mysocket servers and serve as the public endpoint for your local services. In order to connect your local service to the mysocket socket we need tunnels. In this section, we'll explain how to manage tunnels and how to connect the tunnels. Tunnels provide the connection between your local service and the globally anycasted public sockets for you. Currently, we support ssh as a transport protocol for secure connectivity between your local services and mysocket. Note that a socket can have multiple tunnels. In that case mysocket will load balance over all available tunnels.

```
mysocketctl.py tunnel --help
Usage: mysocketctl.py tunnel [OPTIONS] COMMAND [ARGS]...

Manage your tunnels

Options:
--help  Show this message and exit.

Commands:
connect
create
delete
ls
```

13.1 Creating a tunnel

The command below creates a new tunnel for a socket we create earlier.

```
mysocketctl.py tunnel create \
    --socket_id 334c2e48-8324-47c0-9b03-c0a69c2c7833
+--------------------------------------+--------------------------------------+-------
↪--------+------------+
| socket_id                            | tunnel_id                            |␣
↪tunnel_server | relay_port |
+--------------------------------------+--------------------------------------+-------
↪--------+------------+
| 334c2e48-8324-47c0-9b03-c0a69c2c7833 | dc620ec5-76d6-455e-865c-eac238472bee |      ␣
↪        | 6054       |
+--------------------------------------+--------------------------------------+-------
↪--------+------------+
```

Note that the mysocket API returned a tunnel_id and a relay port. The relay port is used when connecting the tunnel, it's used as the SSH listener port.

13.2 Listing all tunnels for a socket

To see all tunnels for a socket, issue the `mysocket tunnel ls` command like below:

```
mysocketctl.py tunnel ls \
    --socket_id 334c2e48-8324-47c0-9b03-c0a69c2c7833
+--------------------------------------+--------------------------------------+-------
↪--------+------------+
| socket_id                            | tunnel_id                            |␣
↪tunnel_server | relay_port |
+--------------------------------------+--------------------------------------+-------
↪--------+------------+
| 334c2e48-8324-47c0-9b03-c0a69c2c7833 | 4f1d5c81-1531-4b93-9343-76b5c16194dc | 52.13.
↪204.31  | 6043       |
| 334c2e48-8324-47c0-9b03-c0a69c2c7833 | dc620ec5-76d6-455e-865c-eac238472bee |      ␣
↪        | 6054       |
+--------------------------------------+--------------------------------------+-------
↪--------+------------+
```

The tunnel server field indicates what server the tunnel was last connected to.

13.3 Deleting a tunnel

To delete a tunnel, issue the tunnel delete command and provide the socket_id and tunnel_id you wish to delete.

```
mysocketctl.py tunnel delete \
    --socket_id 334c2e48-8324-47c0-9b03-c0a69c2c7833 \
    --tunnel_id dc620ec5-76d6-455e-865c-eac238472bee

Tunnel dc620ec5-76d6-455e-865c-eac238472bee deleted
```

13.4 Connecting and using a tunnel

In order to spin up your tunnel, the `mysocketctl tunnel connect` feature may be used.

```
mysocketctl.py tunnel connect --help
Usage: mysocketctl.py tunnel connect [OPTIONS]

Options:
--socket_id TEXT  [required]
--tunnel_id TEXT  [required]
--port TEXT       [required]
--help            Show this message and exit.
```

It requires socket_id and tunnel_id as mandatory arguments. It also needs to know what port number the local service listens on. This can be any local TCP port, as long as you have something listening on it. For example, if you have a local webservice, you want to make publicly available using this tunnel in port 8000 then provide 8000 as the `--port` parameter. If you wanted to make ssh available and the socket you created is of type TCP, then provide port 22 as the port parameter.

```
mysocketctl.py tunnel connect \
    --socket_id 334c2e48-8324-47c0-9b03-c0a69c2c7833 \
```

(continues on next page)

(continued from previous page)

```
    --tunnel_id 4f1d5c81-1531-4b93-9343-76b5c16194dc \
    --port 8000

Connecting to Server: ssh.mysocket.io

Welcome to Mysocket.io!
Local port 44 - https://white-dew-2957.edge.mysocket.io

=========================================================
Logs
=========================================================
```

After issuing the tunnel connect command, `mysocketctl` calls ssh and sets up the SSH tunnel to ssh.mysocket.io. This is an anycasted ssh service, so users will always use the closest, lowest latency, mysocket ssh server. Once connected, the mysocket control plane will signal in real-time all other servers where this tunnel is. As a result, you can re-use the tunnel from multiple endpoints, but only the latest login will be used for traffic. If you would like to load balance over multiple ssh sessions, simply create multiple tunnel connections first.

The stop the tunnel session, press `ctr-c`.

CHAPTER

FOURTEEN

INDEX

- genindex

INDEX

F

I