
MySocket

Andree Toonk

Mar 11, 2021

ABOUT MYSOCKET

1	Introduction	3
2	Quick start	5
3	Features	7
4	Build on a global anycast network	9
5	Example use cases	11
6	Interacting with the Mysocket.io service	13
7	FAQ	15
8	Introduction	17
9	Installing Mysocketctl	19
10	Account management and login	21
11	Quick connect options	23
12	Socket Management	25
13	Tunnel Management	29
14	Index	33
	Index	35

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.

Mysocket provides load balancing and secures application traffic at the edge.

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:

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! Mysocket also provides OpenIDConnect and SAML authentication options, allowing for zero trust deployments.

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](#).

QUICK START

More documentation can be found below; but if you're eager to get started, consider this a quick start. Download the mysocketctl client from our [download page](#)

Accounts can be created online, using the portal here: <https://portal.mysocket.io/register>, or using the cli Create an account:

```
mysocketctl account create \  
  --name "your_name" \  
  --email "your_email_address" \  
  --password "a_secure_password"
```

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

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

or just mysocketctl login:

```
$ mysocketctl login  
Email: atoonk@example.com  
Password:  
Login successful
```

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

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

SOCKET ID	DNS NAME		PORT(S)
↔ TYPE CLOUD AUTH NAME			
6be8287f-cf55-4e29-a7c8-4960166ac609	green-voice-5562.edge.mysocket.io		80, 443
↔ http false my test service			

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

```
Welcome to Mysocket.io!
```

```
my test service - https://green-voice-5562.edge.mysocket.io
```

```
=====
```

(continues on next page)

(continued from previous page)

```
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!

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

Zero trust: Support for OpenIDConnect authentication. protect your sockets with authentication. Login with your favorite identity provider (Google, Facebook, Github)

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

BUILD ON A GLOBAL ANYCAST NETWORK

Mysocket.io is built on a global anycasted network of **94 Points of Presence in 82 cities across 44 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.

EXAMPLE USE CASES

5.1 Zero Trust

With our [Identity Aware sockets](#) you can provide access to your private (on prem) services, without the need for a VPN. Mysocket can act as a VPN alternative. No software is needed on the client, all the while authentication and authorization options are making sure your private resources are only available to those who should have access.

5.2 Kubernetes public load balancer

Provide a load balancer service with a public anycasted IP for your Kubernetes workloads. [As easy as installing the mysocket.io k8 controller.](#)

5.3 Easy Multi-region load balancing

Spin up your origin services over multiple cloud providers and regions and have the mysocket edge network front and secure your traffic. [Load balancing over multiple regions and cloud providers has never been easier.](#)

5.4 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.

5.5 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.

5.6 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.

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](https://api.mysocket.io/). 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.

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?

Mysocket.io, a service that provides you with fast and secure network connectivity whenever you need it, wherever you are. It provides secure and stable public anycasted TCP endpoints for dynamic services or services that are otherwise not publicly reachable!

7.2 What can I do with Mysocket?

There are many examples, but in short you can extend reachability to TCP sockets that run within your network or just your laptop, to a global audience.

7.3 What are identity aware sockets?

These are sockets that are aware of the visitor's identity. If your socket is enabled for "cloud authentication," all visitors will be prompted to authenticate first. Authentication can be completed using the various social Identity providers (Google, Github, Facebook), as well as local accounts. As the owner of the socket, you can then specify authorization rules, allowing only authenticated users with certain email domains or specific email addresses. For more information see this article [identity-aware sockets](#)

7.4 what happened to the Python client?

We changed out the python3 mysocketctl client in favor of the new Golang mysocketctl client. The python code is still available and can be installed using `pip3 install mysocketctl` The python code can be found here: <https://github.com/mysocketio/mysocketctl>

It's recommended, however, to use the Go client, which can be downloaded here: <https://download.edge.mysocket.io/>

7.5 What performance improvement does Mysocket provide?

Because mysocket is an anycast service, both the end-user and the tunnel connection is automatically terminated at the nearest mysocket server. We also use TCP BBR, further improving the performance characteristics of the TCP connection

7.6 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 actual tunnel and api servers are deployed throughout North America, Europe and Asia.

7.7 What kind of support is provided?

Today support is best effort.

7.8 Q: How do I get started with Mysocket?

The best way to get started is to follow the details in this blog: <https://www.mysocket.io/post/introducing-mysocket> or see: <https://mysocket.readthedocs.io/en/latest/about/about.html#quick-start>

7.9 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.

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

Using anycast your users will be routed to our closest proxy service (located in Asia, Europe and North America). From there on we make sure traffic is sent to the tunnel server. So we ingest your users traffic as close to the user as possible. This lower Round Trip time helps improve the user experience.

INTRODUCTION

`mysocketctl` is a cli tool that allows you to easily manage and use the Mysocket services. `mysocketctl` uses the 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
mysocket.io command line interface (CLI)

Usage:
mysocketctl [command]

Available Commands:
account      Create a new account or see account information.
connect      Quickly connect, wrapper around sockets and tunnels
help         Help about any command
login        Login to mysocket and get a token
socket       Manage your global sockets
tunnel       Manage your tunnels
version      check version

Flags:
-h, --help      help for mysocketctl
-v, --version    version for mysocketctl

Use "mysocketctl [command] --help" for more information about a command.
```

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](#).

INSTALLING MYSOCKETCTL

Download the latest mysocketctl from <https://download.edge.mysocket.io/>

The `mysocketctl` client is written in Go. Binaries exist for all major Operating systems. For those interested. The source code for mysocketctl can be found here: <https://github.com/mysocketio/mysocketctl-go>

9.1 Making sure you run the latest version

To check if you're up to date, run:

```
$ mysocketctl version check
You are up to date!
You're running version v1.0-9-g0efd9e3
```

To update to the latest version run:

```
$ mysocketctl version upgrade
```

This will download the latest version, validate the checksum, and replace the current binary with the latest version. A version check is also performed each time the user runs `mysocketctl login`

ACCOUNT MANAGEMENT AND LOGIN

```
$ mysocketctl account --help
Create a new account or see account information.

Usage:
mysocketctl account [command]

Available Commands:
create      Create a new account
show       Show account information

Flags:
-h, --help  help for account

Use "mysocketctl account [command] --help" for more information about a command.
```

10.1 Creating an account

To use mysocket.io users will need to register and create an account. The easiest way to create an account is to use the portal: <https://portal.mysocket.io/register>

Alternatvly, you can also use mysocketctl 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.

```
mysocketctl account create \
  --name "your_name" \
  --email "your_email_address" \
  --password "a_secure_password"
```

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 login
Email: atoonk@example.com
Password:
Login successful
```

or, if you like you can provide the username and/or password directly.

```
mysocketctl 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	addce6c8-cb8d-4ce5-228e-a8fe097434b9	
↪		
SSH Username	addce6c8cb8d4ce5228ea8fe097434b9	
↪		
SSH Key	ssh-rsa <your public key....SNIP TOO LONG>	
↪		

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 connect --help
Quickly connect, wrapper around sockets and tunnels

Usage:
mysocketctl connect [flags]

Flags:
-e, --allowed_email_addresses string    Comma seperated list of allowed Email
    ↪addresses when using cloudatauth
-d, --allowed_email_domains string      comma seperated list of allowed Email domain
    ↪(i.e. 'example.com', when using cloudatauth
-c, --cloudatauth                       Enable oauth/oidc authentication
-h, --help                             help for connect
    --host string                       Target host: Control where inbound traffic
    ↪goes. Default localhost (default "127.0.0.1")
-i, --identity_file string              Identity File
-n, --name string                      Service name
    --password string                   Password, required when protected set to true
-p, --port int                         Port
    --protected                        Protected, default no
-t, --type string                      Socket type: http, https, tcp, tls (default
    ↪"http")
-u, --username string                  Username, required when protected set to true
```

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 connect \
  --port 8000 \
  --name "my test service"
```

(continues on next page)

(continued from previous page)

SOCKET ID	DNS NAME	PORT(S)
↩ TYPE CLOUD AUTH NAME		
db39a501-1010-4b5a-842d-bb6fe1fe4e2d	twilight-sky-7409.edge.mysocket.io	80, 443
↩ http false my test service		

Connecting to Server: ssh.mysocket.io

Welcome to Mysocket.io!

my test service - https://twilight-sky-7409.edge.mysocket.io

=====

Logs

=====

In this case, a socket with the name twilight-sky-7409.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.

```
^Ccleaning up...
```

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
Manage your global sockets

Usage:
mysocketctl socket [command]

Available Commands:
create      Create a new socket
delete      Delete a socket
ls          List your sockets

Flags:
-h, --help  help for socket

Use "mysocketctl socket [command] --help" for more information about a command.
```

12.1 Creating sockets

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

```
mysocketctl socket create \
  --name "my local http service" \
  --type http
```

SOCKET ID	DNS NAME
↪PORT(S) TYPE CLOUD AUTH NAME	
de306718-3315-4445-b9e6-e68fe5cf45d7 ↪	delicate-waterfall-6975.edge.mysocket.io ↪
↪80, 443 http false 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. There are two ways to achieve this: 1) Using the “cloud authentication” feature. This will allow the user to login with OpenIDConnect, which supports Google, Facebook, or Github. As well as local account and even SAML. 2) static username and password using Basic Auth.

Below an example of creating an identity aware socket using the Cloud Authentication feature. With this, we created a socket on the mysocket.io infrastructure, enabled authentication, and provided a list of authorization rules that allow users that have authenticated as [andree@example.io](#), [john@doe.com](#) or anyone with an @mycorp.com email address. for more information about Identity aware socket also see [this article](#).

mysocketctl socket create \
 --name "My Identity aware socket" \
 --cloudauth \
 --allowed_email_domains "mycorp.com" \
 --allowed_email_addresses "andree@example.io,john@doe.com"

SOCKET ID				DNS NAME	
↪PORT(S)	TYPE	CLOUD AUTH	NAME		
↪443	http	true	My Identity aware socket		

fab1357d-acfb-4735-ae4f-0dceb9fcb0ce	wispy-snowflake-5908.edge.mysocket.io	80,
--------------------------------------	---------------------------------------	-----

Cloud Authentication, login details:

ALLOWED EMAIL ADDRESSES	ALLOWED EMAIL DOMAINS	
andree@example.io	mycorp.com	
john@doe.com		

Below an example of creating a password-protected socket, with username john and password secret.

```
mysocketctl socket create \
  --name "my local http service" \
  --type http \
  --protected \
  --username john \
  --password secret
```

SOCKET ID		DNS NAME		PORT(S)
↪TYPE	CLOUD AUTH	NAME		↪
818f3cf8-1ed8-4fbc-af41-3fa6054d5b6b		snowy-sea-4481.edge.mysocket.io		80, 443
↪http	false	my local http service		↪

Protected Socket:

USERNAME	PASSWORD
john	secret

12.2 Listing all sockets

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

```
mysocketctl socket ls
```

SOCKET ID	DNS NAME	
↪PORT(S) TYPE CLOUD AUTH NAME		
cclbfd68-cca7-49ce-b1d8-e4495a43796e	dry-darkness-1814.edge.mysocket.io	↪
↪80, 443 http false Local port 8000		
c28bcd15-7e35-4090-b228-8d154841b699	green-silence-145.edge.mysocket.io	↪
↪80, 443 http false Local port 8888		
d60ca2a1-7215-4a7b-985d-c099ac6d1293	polished-mountain-1373.edge.mysocket.io	↪
↪80, 443 http false Local port 8888		
932b9fab-6d01-4468-84bb-5a1e69170432	restless-voice-3146.edge.mysocket.io	↪
↪80, 443 http false Local port 8888		
69fd1375-313b-4737-bcea-fda60e831f47	rough-bush-1794.edge.mysocket.io	↪
↪80, 443 https false string		
72415de0-65b2-4bb7-b477-96f6ce3603c2	ancient-dust-7286.edge.mysocket.io	↪
↪54858 tls false ssh over tls test		
60d5b3f6-a6fc-4b52-82bf-7538ee18d172	empty-snow-8262.edge.mysocket.io	↪
↪80, 443 http false Local port 80		
de306718-3315-4445-b9e6-e68fe5cf45d7	delicate-waterfall-6975.edge.mysocket.io	↪
↪80, 443 http false my local http service		
818f3cf8-1ed8-4fbc-af41-3fa6054d5b6b	snowy-sea-4481.edge.mysocket.io	↪
↪80, 443 http false my local http service		
fab1357d-acfb-4735-ae4f-0dceb9fcb0ce	wispy-snowflake-5908.edge.mysocket.io	↪
↪80, 443 http true My Identity aware socket		

12.3 Delete sockets

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

```
mysocketctl socket delete \
  --socket_id 5870a362-65d3-474d-bbf6-3341827eaae0

Socket deleted
```


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 tunnel --help
Manage your tunnels

Usage:
mysocketctl tunnel [command]

Available Commands:
connect    Connect a tunnel
create     Create a tunnel
delete     Delete a tunnel
ls         List your tunnels

Flags:
-h, --help  help for tunnel

Use "mysocketctl tunnel [command] --help" for more information about a command.
```

13.1 Creating a tunnel

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

```
mysocketctl tunnel create \
    --socket_id 818f3cf8-1ed8-4fbc-af41-3fa6054d5b6b
```

SOCKET ID	TUNNEL ID
↪ TUNNEL SERVER RELAY PORT	
818f3cf8-1ed8-4fbc-af41-3fa6054d5b6b 3fc93b24-c5b1-4c30-9427-ae6b5738224d	
↪ 7295	

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 `mysocketctl tunnel ls` command like below:

mysocketctl tunnel ls \
--socket_id 818f3cf8-1ed8-4fbc-af41-3fa6054d5b6b

SOCKET ID	TUNNEL ID
↪TUNNEL SERVER RELAY PORT	
818f3cf8-1ed8-4fbc-af41-3fa6054d5b6b 3fc93b24-c5b1-4c30-9427-ae6b5738224d	
↪ 7295	
818f3cf8-1ed8-4fbc-af41-3fa6054d5b6b f0bc2e7f-e22d-4ac0-94ae-ccf5160c0a12	
↪ 7296	

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 tunnel delete \  
--socket_id 818f3cf8-1ed8-4fbc-af41-3fa6054d5b6b \  
--tunnel_id 3fc93b24-c5b1-4c30-9427-ae6b5738224d  
  
Tunnel deleted
```

13.4 Connecting and using a tunnel

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

```
mysocketctl tunnel connect --help  
Connect a tunnel  
  
Usage:  
mysocketctl tunnel connect [flags]  
  
Flags:  
-h, --help                help for connect  
--host string              Target host: Control where inbound traffic goes. Default ↪  
↪localhost (default "127.0.0.1")  
-i, --identity_file string Identity File  
-p, --port int             Port number  
-s, --socket_id string     Socket ID  
-t, --tunnel_id string     Tunnel ID
```

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.

the `--host` parameter defaults to localhost (127.0.0.1). This can be changed, `--host` accepts a DNS name or an IP address. This is useful for when you'd like to forward the traffic to a different host than localhost.

the `--i` parameter allows you to specify a non-standard ssh identity file (private key)

```
mysocketctl tunnel connect \
  --socket_id 818f3cf8-1ed8-4fbc-af41-3fa6054d5b6b \
  --tunnel_id f0bc2e7f-e22d-4ac0-94ae-ccf5160c0a12 \
  --port 8000

Connecting to Server: ssh.mysocket.io

Welcome to Mysocket.io!
my local http service - https://snowy-sea-4481.edge.mysocket.io

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

The tunnel connect command sets up a secure encrypted 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 tunnel sessions, simply create multiple tunnel connections first.

To stop the tunnel session, press `ctrl-c`.

An example of using `mysocketctl` as a side-car / forwarder to, in this case, `google.com:80` can be seen below. In this case, traffic to `https://snowy-sea-4481.edge.mysocket.io` is routed to the tunnel endpoint (the `mysocketctl` client), which will then forward it to `google.com` port 80. This can be useful for cases where the actual target can't run `mysocketctl` like, for example, an appliance or managed database.

```
mysocketctl tunnel connect \
  --socket_id 818f3cf8-1ed8-4fbc-af41-3fa6054d5b6b \
  --tunnel_id f0bc2e7f-e22d-4ac0-94ae-ccf5160c0a12 \
  --port 80 \
  --host google.com

Connecting to Server: ssh.mysocket.io

Welcome to Mysocket.io!
my local http service - https://snowy-sea-4481.edge.mysocket.io

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


INDEX

- `genindex`

INDEX

F

FAQ, [13](#)

Frequently Asked Questions, *see* FAQ

I

Introduction, [1](#), [16](#)