

Authenticating with a User and Password



- [Connecting with a User/Password](#)
- [Connecting with a User/Password in the URL](#)

For this example, start the server using:

```
nats-server --user myname --pass password
```

You can encrypt passwords to pass to `nats-server` using the simple [NATS CLI tool](#):

```
nats server passwd
```

```
? Enter password [? for help] *****
? Reenter password [? for help] *****

$2a$11$qbtrnb0mSG2eV55xoyPqH0Zx/LLB1ryHRhU3LK2o0PFRwGF/5rtGK
```

and use the hashed password in the server config. The client still uses the plain text version.

The code uses localhost:4222 so that you can start the server on your machine to try them out.

Connecting with a User/Password

When logging in with a password `nats-server` will take either a plain text password or an encrypted password.

Go

```
// Set a user and plain text password
nc, err := nats.Connect("127.0.0.1", nats.UserInfo("myname", "password"))
```

```
if err != nil {
    log.Fatal(err)
}
defer nc.Close()

// Do something with the connection
```

Java

```
Options options = new Options.Builder()
    .server("nats://localhost:4222")
    .userInfo("myname","password") // Set a user and plain text password
    .build();
Connection nc = Nats.connect(options);

// Do something with the connection

nc.close();
```

JavaScript

```
const nc = await connect({
  port: ns.port,
  user: "byname",
  pass: "password",
});
```

Python

```
nc = NATS()

await nc.connect(servers=["nats://myname:password@demo.nats.io:4222"])

# Do something with the connection.
```

C#

```
// dotnet add package NATS.Net
using NATS.Net;
using NATS.Client.Core;
```

```

await using var client = new NatsClient(new NatsOpts
{
    Url = "nats://localhost:4222",
    AuthOpts = new NatsAuthOpts
    {
        Username = "myname",
        Password = "password",
    }
});

```

Ruby

```

require 'nats/client'

NATS.start(servers:["nats://myname:password@127.0.0.1:4222"], name: "my-connection") do |nc|
  nc.on_error do |e|
    puts "Error: #{e}"
  end

  nc.on_reconnect do
    puts "Got reconnected to #{nc.connected_server}"
  end

  nc.on_disconnect do |reason|
    puts "Got disconnected! #{reason}"
  end

  nc.close
end

```

C

```

natsConnection      *conn      = NULL;
natsOptions          *opts      = NULL;
natsStatus           s          = NATS_OK;

s = natsOptions_Create(&opts);
if (s == NATS_OK)
    s = natsOptions_SetUserInfo(opts, "myname", "password");
if (s == NATS_OK)

```

```
s = natsConnection_Connect(&conn, opts);

(...)

// Destroy objects that were created
natsConnection_Destroy(conn);
natsOptions_Destroy(opts);
```

Connecting with a User/Password in the URL

Most clients make it easy to pass the user name and password by accepting them in the URL for the server. This standard format is:

```
“ nats://user:password@server:port
```

Using this format, you can connect to a server using authentication as easily as you connected with a URL:

Go

```
// Set a user and plain text password
nc, err := nats.Connect("myname:password@127.0.0.1")
if err != nil {
    log.Fatal(err)
}
defer nc.Close()

// Do something with the connection
```

Java

```
Connection nc = Nats.connect("nats://myname:password@localhost:4222");

// Do something with the connection
```

```
nc.close();
```

JavaScript

```
// JavaScript clients don't support username/password in urls use `user` and `pass` options.
```

Python

```
nc = NATS()

await nc.connect(servers=["nats://myname:password@demo.nats.io:4222"])

# Do something with the connection.
```

C#

```
// dotnet add package NATS.Net
using NATS.Net;
using NATS.Client.Core;

await using var nc = new NatsClient(new NatsOpts
{
    // .NET client doesn't support username/password in URLs
    // use `Username` and `Password` options.
    Url = "nats://demo.nats.io:4222",
    AuthOpts = new NatsAuthOpts
    {
        Username = "myname",
        Password = "password",
    }
});
```

Ruby

```
require 'nats/client'

NATS.start(servers: ["nats://myname:password@127.0.0.1:4222"], name: "my-connection") do |nc|
  nc.on_error do |e|
    puts "Error: #{e}"
  end
end
```

```

nc.on_reconnect do
  puts "Got reconnected to #{nc.connected_server}"
end

nc.on_disconnect do |reason|
  puts "Got disconnected! #{reason}"
end

nc.close
end

```

C

```

natsConnection      *conn      = NULL;
natsOptions          *opts      = NULL;
natsStatus           s          = NATS_OK;

s = natsOptions_Create(&opts);
if (s == NATS_OK)
  s = natsOptions_SetURL(opts, "nats://myname:password@127.0.0.1:4222");
if (s == NATS_OK)
  s = natsConnection_Connect(&conn, opts);

(...)

// Destroy objects that were created
natsConnection_Destroy(conn);
natsOptions_Destroy(opts);

```

```

[] #3
[] cocobook-sync [] 2026-10-28 11:11:08 UTC
[] cocobook-sync [] 2026-10-29 01:42:35 UTC

```