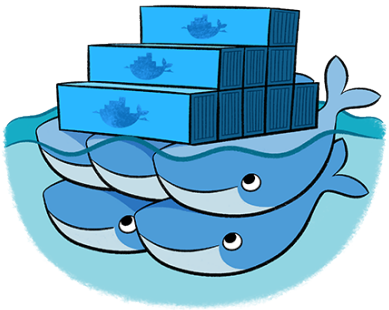
# Setup OIDC Guide for Beginners

# Overview

Please take a look at how OIDC works here if you haven't already: What is OIDC?
This little Wiki Article will guide you through how to get OIDC running. In this example, we will set up OIDC with the service Linkwarden.

---

# Requirements

Make sure you have the following set up:

- **Traefik** as a Reverse Proxy to get certificates and provide access to your Website/Subdomains: Traefik Reverse Proxy.
- **Authelia** as a Middleware/Addon for Traefik, already configured with a "Whoamisecure" test to ensure your Authelia is working before adding OIDC: Getting Started.

---

# Step 1

Open your Authelia configuration file and edit it (in this example, we store it on a GlusterFS mount at `/mnt/glustermount/data/authelia_data/config/configuration.yml`):

```
nano /mnt/glustermount/data/authelia_data/config/configuration.yml
```

---

# Step 2

Scroll to the bottom of your config file and add the following code:

```
identity_providers:
  oidc:
    hmac_secret: 'this_is_a_secret_abc123abc123abc'
    jwks:
     -  key_id: 'F2H5xqbYsa3AssEZTU'
        algorithm: 'RS256'
        use: 'sig'
        key: {{ secret "/secrets/rsa_2048_private.txt" | mindent 10 "|" | msquote }}
    lifespans:
      access_token: '1 hour'
      authorize_code: '1 minute'
      id_token: '1 hour'
      refresh_token: '90 minutes'
    enable_client_debug_messages: false
```

---

# Step 3

By adding the `identity_providers` section, you enable OIDC in Authelia. All settings in that block belong to OIDC. Please note this is only the minimum required configuration for our setup. You can find more details here: Authelia OIDC Introduction.

# HMAC_Secret

The **HMAC Secret** is a random string known only to Authelia. Do not make this public.

**The HMAC secret** is used to sign the JWT. This string is hashed to a SHA256 (RFC6234) byte string.

It's **strongly recommended** you use a random alphanumeric string with 64 or more characters.

Generate a key in Docker CLI:

```
docker run --rm authelia/authelia:latest authelia crypto rand --length 64 --charset alphanumeric
```

For example: `"rzUPr41040tMvw4tg95Ud2HdcvdDMVZPQQPpHAist386QajGftF4lIFSw0yi2gtD"`
Copy it to: `hmac_secret: 'this_is_a_secret_abc123abc123abc'`

# JWKs

The list of issuer JSON Web Keys. At least one of these must be an RSA Private key configured with the RS256 algorithm. You can configure multiple keys or algorithms. The first key for each algorithm is the default if a client isn't configured to use a specific `key_id`.

Below is a contextual example:

```
identity_providers:
  oidc:
    jwks:
      - key_id: 'example'
        algorithm: 'RS256'
        use: 'sig'
        key: |
          -----BEGIN RSA PRIVATE KEY-----
          ...
          -----END RSA PRIVATE KEY-----
        certificate_chain: |
          -----BEGIN CERTIFICATE-----
          ...
          -----END CERTIFICATE-----
          -----BEGIN CERTIFICATE-----
          ...
          -----END CERTIFICATE-----
```

> **RFC Internet Standard:**
> RFC (Request for Comments) documents define internet standards, protocols, and best practices. For example, *RFC7519* outlines how JSON Web Tokens should be structured and validated, ensuring a standard approach to token-based authentication.

## KeyID

This is completely optional unless there's a collision between automatically generated key IDs. If provided, it must be a unique string with fewer than 100 characters, matching the regular expression `^[a-zA-Z0-9](([a-zA-Z0-9._~-]*)([a-zA-Z0-9]))?$`.

The default if this value is omitted is the first 7 characters of the public key SHA256 thumbprint in hex, followed by a hyphen, then the lowercase algorithm value.

```
docker run --rm authelia/authelia:latest authelia crypto rand --length 15 --charset alphanumeric
```

For example: `"F2H5xqbYsa3AssEZTU"`

---

## Use

The key usage. Defaults to `sig`, which is currently the only available option.

---

## Algorithm

The algorithm for this key. Typically optional, as it can be automatically detected based on the type of key. At least one `RS256` key must be provided.

---

## Key

You can generate an RSA keypair using the **Authelia** Docker container:

```
docker run --rm -u "$(id -u):$(id -g)" -v "$(pwd)":/keys authelia/authelia:latest authelia crypto pair rsa generate --directory /keys
```

Assuming your working directory is `/mnt/glustermount/data/authelia_data/`, you'll end up with `private.pem` and `public.pem`.

Then, place `private.pem` somewhere like: `/mnt/glustermount/data/authelia_data/secrets/rsa_2048_private.txt`

Create folders if necessary:

```
mkdir /mnt/glustermount/data/authelia_data/config/secrets
```

Open `private.pem`, copy the content:

```
-----BEGIN RSA PRIVATE KEY-----
...
-----END RSA PRIVATE KEY-----
```

And paste it into a new file:

```
nano /mnt/glustermount/data/authelia_data/config/secrets/rsa_2048_private.txt
```

Save with CTRL+O and exit with CTRL+X.

> If you use different Paths like mentioned in Step 2, please adjust the Configs.

---

# Step 4

Add Clients/Services to OIDC. Here is an example on how to add Linkwarden: Setup OIDC for Linkwarden.

---

# Conclusion

With the above configuration, Authelia can function as an OIDC provider, signing JWT tokens for your clients or services (like Linkwarden). Make sure to correctly configure your OIDC clients with the same settings (client IDs, secrets, scopes) to ensure smooth authentication. Once you've confirmed it's working with a test service, you can reuse these steps for additional applications that support OIDC.

---

Revision #11
Created 1 February 2025 20:08:10 by aeoneros
Updated 4 February 2025 14:23:33 by aeoneros