Authelia and NPM

This article describes how Authelia and Nginx Proxy Manager (NPM) can be configured to provide both single sign on protection for various internal services via ssl certificate and two factor authentication via passkeys and/or TOTP (time based one time password).

This requires either a domain pointing to a fixed address or a dynamic dns provider that ideally allows wildcard subdomains. This also requires ports 80/443 to be redirected to the internal server running the npm docker instance ports 70 and 743 respectively. Adjust as necessary.

Replace all example.com domains and password references with your domain and random passwords!

The examples use simple local yaml file for authelia users and local file for notifications (used for 2FA setup) and local sqlite database. Any complex SSO, Redis, LDAP, mysql/postgresql database or other integration is out of scope of this article.

Docker setup

docker-npm.yml

```yaml
name: npm
services:
  npm:
    image: 'jc21/nginx-proxy-manager:latest'
    restart: unless-stopped
    container_name: npm

    ports:
      # These ports are in format <host-port>:<container-port>
      - '70:80' # Public HTTP Port
      - '743:443' # Public HTTPS Port
      - '71:81' # Admin Web Port
      # Add any other Stream port you want to expose
      # - '21:21' # FTP

    environment:
      TZ: "Europe/London"

      # Uncomment this if you want to change the location of
      # the SQLite DB file within the container
      # DB_SQLITE_FILE: "/data/database.sqlite"

      # Uncomment this if IPv6 is not enabled on your host
      DISABLE_IPV6: 'true'

      INITIAL_ADMIN_EMAIL: something@example.com
```

```
      INITIAL_ADMIN_PASSWORD: secure_password

    volumes:
      - /opt/npm/data:/data
      - /opt/npm/letsencrypt:/etc/letsencrypt
      - /opt/npm/snippets:/snippets

    healthcheck:
      test: ["CMD", "/usr/bin/check-health"]
      interval: 10s
      timeout: 3s
```

docker-authelia.yml

```
name: authelia
services:
  authelia:
    image: authelia/authelia
    container_name: authelia
    restart: unless-stopped
    volumes:
      - /opt/authelia:/config
    ports:
      - 6091:9091
    environment:
      - TZ=Europe/London
```

Create the data directories:

```
mkdir -p /opt/authelia
mkdir -p /opt/npm/{data,letsencrypt,snippets}
```

Authelia Configuration

This is a stripped down minimal configuration. See the authelia documentation for options and additional functionality.

/opt/authelia/configuration.yml

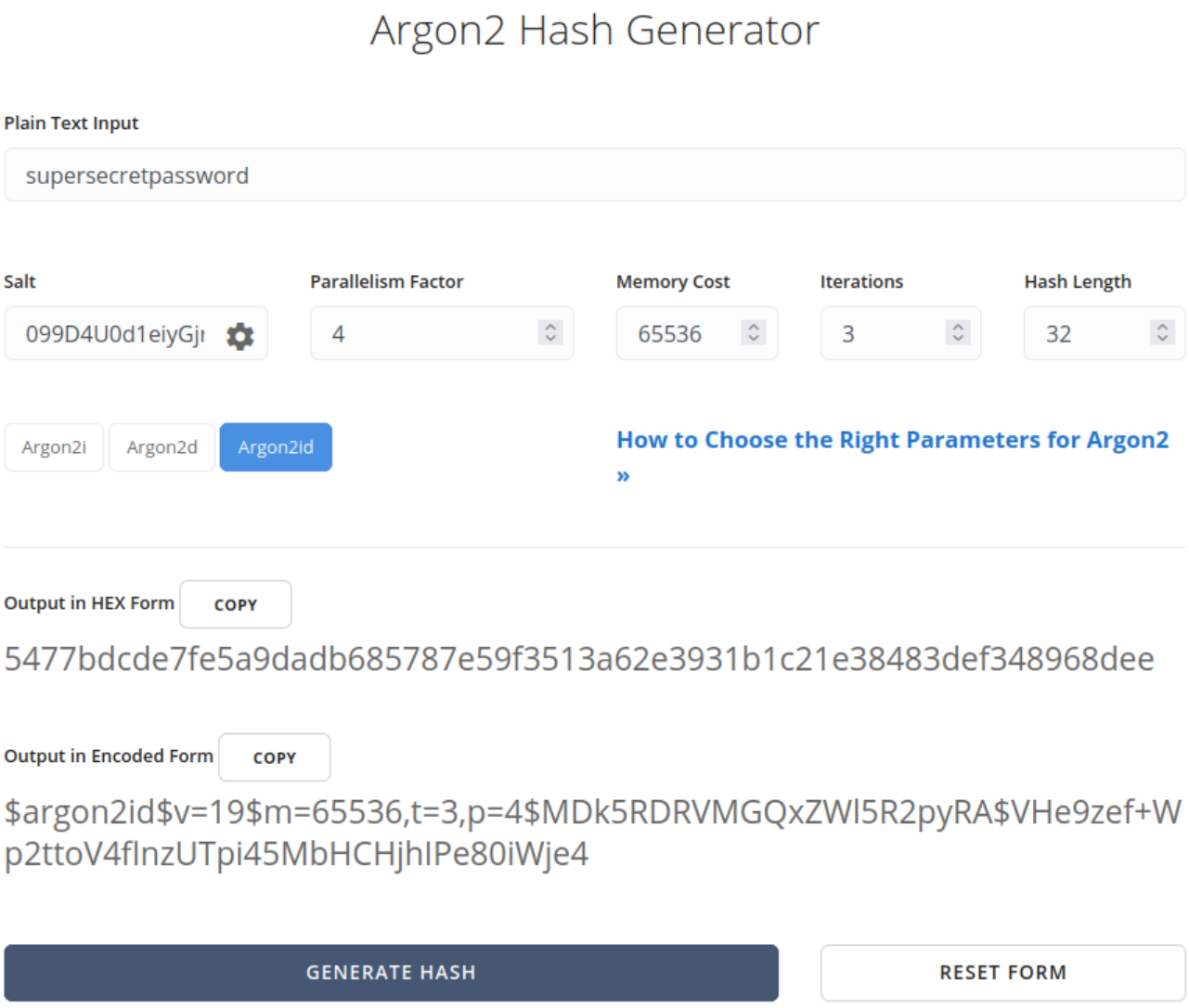
```
# yamllint disable rule:comments-indentation
---
theme: 'auto'
log:
  level: 'info'
  format: 'text'
  file_path: '/config/authelia.log'
  keep_stdout: false
```

```
totp:
  disable: false
  issuer: 'domain.example.com'
  period: 30
  skew: 1
webauthn:
  disable: false
  enable_passkey_login: true
  display_name: 'Authelia'
  attestation_conveyance_preference: 'indirect'
  timeout: '60 seconds'
  filtering:
    prohibit_backup_eligibility: false
    permitted_aaguids: []
    prohibited_aaguids: []
  selection_criteria:
    attachment: 'platform'
    discoverability: 'preferred'
    user_verification: 'preferred'
identity_validation:
  reset_password:
    jwt_secret: 'set_to_secure_password'
authentication_backend:
  file:
    path: '/config/users_database.yml'
access_control:
  rules:
    - domain:
        - "auth.example.com"
      policy: bypass
    - domain: "example.com"
      policy: bypass
    - domain:
       - 'sub1.example.com'
       - 'sub2.example.com'
       - 'sub3.example.com'
      policy: 'two_factor'
    - domain: "*.example.com"
      policy: one_factor
session:
  secret: 'insecure_session_secret'
  cookies:
     -
      name: 'authelia_session'
      domain: 'example.com'
      authelia_url: 'https://auth.example.com'
      default_redirection_url: 'https://example.com'
regulation:
  modes:
   - 'user'
  max_retries: 3
```

```
  find_time: '2 minutes'
  ban_time: '10 minutes'
storage:
  encryption_key: 'very_secure_encryption_key'
  local:
    path: '/config/db.sqlite3'
notifier:
  disable_startup_check: true
  filesystem:
    filename: '/config/notification.txt'
```

Authelia password file

For the user passwords, go to https://argon2.online/ to generate your passwords, use the settings you see below:



Enter your password into the “Plain Text Input”

Click the gear in “Salt” to generate a random string of characters.

Be sure to have “Argon2id” activated.

Other settings:

```
Parallelism: 4
Memory Cost: 65536
Iterations: 3
Hash Length: 32
```

Click “Generate Hash”

Copy the string that starts with $argon2id into the associated user password in the users_database.yml

/opt/authelia/users_database.yml

```
users:
  user1: #username for user 1. change to whatever you'd like
    displayname: "User Name 1" #whatever you want the display name to
be
    password:
"$argon2i$v=19$m=1024,t=1,p=8$eTQ3MXdqOGFiaDZoMUtMVw$OeHWQSg9zGKslOepe5
t4D1T9BZJjHA1Z+doxZrZYDgI" #generated at https://argon2.online/
    email: youremail1@example.com #whatever your email address is
    groups: #enter the groups you want the user to be part of below
      - admins
      - dev
    disabled: false
  user2: #username for user 2. change to whatever you'd like. Or delete
this section if you only have 1 user
    displayname: "User Name 2" #whatever you want the display name to
be
    password:
"$argon2i$v=19$m=1024,t=1,p=8$eTQ3MXdqOGFiaDZoMUtMVw$OeHWQSg9zGKslOepe5
t4D1T9BZJjHA1Z+doxZrZYDgI" #generated at https://argon2.online/
    email: youremail2@example.com #whatever your email address is
    groups: #enter the groups you want the user to be part of below
      - dev
    disabled: true
```

Config Snippets

authelia-authrequest.conf

```
## Send a subrequest to Authelia to verify if the user is authenticated
and has permission to access the resource.
```

```
auth_request /internal/authelia/authz;
## Save the upstream metadata response headers from Authelia to
variables.
auth_request_set $user $upstream_http_remote_user;
auth_request_set $groups $upstream_http_remote_groups;
auth_request_set $name $upstream_http_remote_name;
auth_request_set $email $upstream_http_remote_email;
## Inject the metadata response headers from the variables into the
request made to the backend.
proxy_set_header Remote-User $user;
proxy_set_header Remote-Groups $groups;
proxy_set_header Remote-Email $email;
proxy_set_header Remote-Name $name;
## Configure the redirection when the authz failure occurs. Lines
starting with 'Modern Method' and 'Legacy Method'
## should be commented / uncommented as pairs. The modern method uses
the session cookies configuration's authelia_url
## value to determine the redirection URL here. It's much simpler and
compatible with the mutli-cookie domain easily.
## Modern Method: Set the $redirection_url to the Location header of
the response to the Authz endpoint.
auth_request_set $redirection_url $upstream_http_location;
## Modern Method: When there is a 401 response code from the authz
endpoint redirect to the $redirection_url.
error_page 401 =302 $redirection_url;
## Legacy Method: Set $target_url to the original requested URL.
## This requires http_set_misc module, replace 'set_escape_uri' with
'set' if you don't have this module.
# set_escape_uri $target_url $scheme://$http_host$request_uri;
## Legacy Method: When there is a 401 response code from the authz
endpoint redirect to the portal with the 'rd'
## URL parameter set to $target_url. This requires users update
'auth.yourdomain.com/' with their external authelia URL.
# error_page 401 =302 https://auth.yourdomain.com/?rd=$target_url;
```

Adjust internal IP and port to your system:

authelia-location.conf

```
set $upstream_authelia http://192.168.1.2:6091/api/authz/auth-request;
location /internal/authelia/authz {
    internal;
    proxy_pass $upstream_authelia;
    proxy_set_header X-Original-Method $request_method;
    proxy_set_header X-Original-URL $scheme://$http_host$request_uri;
    proxy_set_header X-Forwarded-For $remote_addr;
    proxy_set_header Content-Length "";
    proxy_set_header Connection "";
    proxy_pass_request_body off;
    proxy_next_upstream error timeout invalid_header http_500 http_502
```

```
http_503;
    proxy_redirect http:// $scheme://;
    proxy_http_version 1.1;
    proxy_cache_bypass $cookie_session;
    proxy_no_cache $cookie_session;
    proxy_buffers 4 32k;
    client_body_buffer_size 128k;
    send_timeout 5m;
    proxy_read_timeout 240;
    proxy_send_timeout 240;
    proxy_connect_timeout 240;
```

proxy.conf

```
proxy_set_header Host $host;
proxy_set_header X-Original-URL $scheme://$http_host$request_uri;
proxy_set_header X-Forwarded-Proto $scheme;
proxy_set_header X-Forwarded-Host $http_host;
proxy_set_header X-Forwarded-URI $request_uri;
proxy_set_header X-Forwarded-Ssl on;
proxy_set_header X-Forwarded-For $remote_addr;
proxy_set_header X-Real-IP $remote_addr;
client_body_buffer_size 128k;
proxy_next_upstream error timeout invalid_header http_500 http_502
http_503;
proxy_redirect http:// $scheme://;
proxy_http_version 1.1;
proxy_cache_bypass $cookie_session;
proxy_no_cache $cookie_session;
proxy_buffers 64 256k;
real_ip_header X-Forwarded-For;
real_ip_recursive on;
send_timeout 5m;
proxy_read_timeout 360;
proxy_send_timeout 360;
proxy_connect_timeout 360;
```

websocket.conf

```
proxy_set_header Upgrade $http_upgrade;
proxy_set_header Connection "upgrade";
```

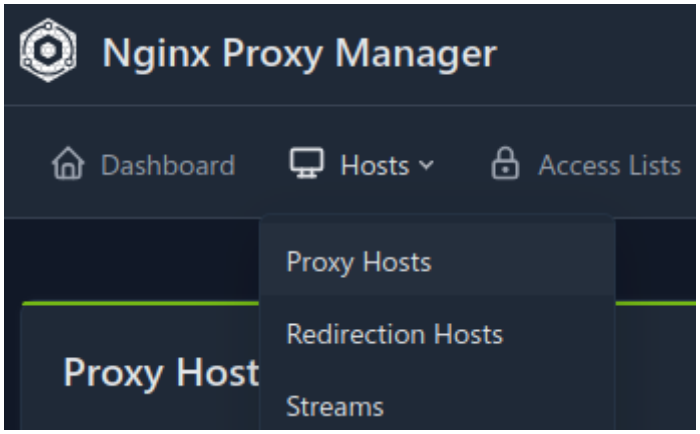
Docker startup

Start the docker containers:

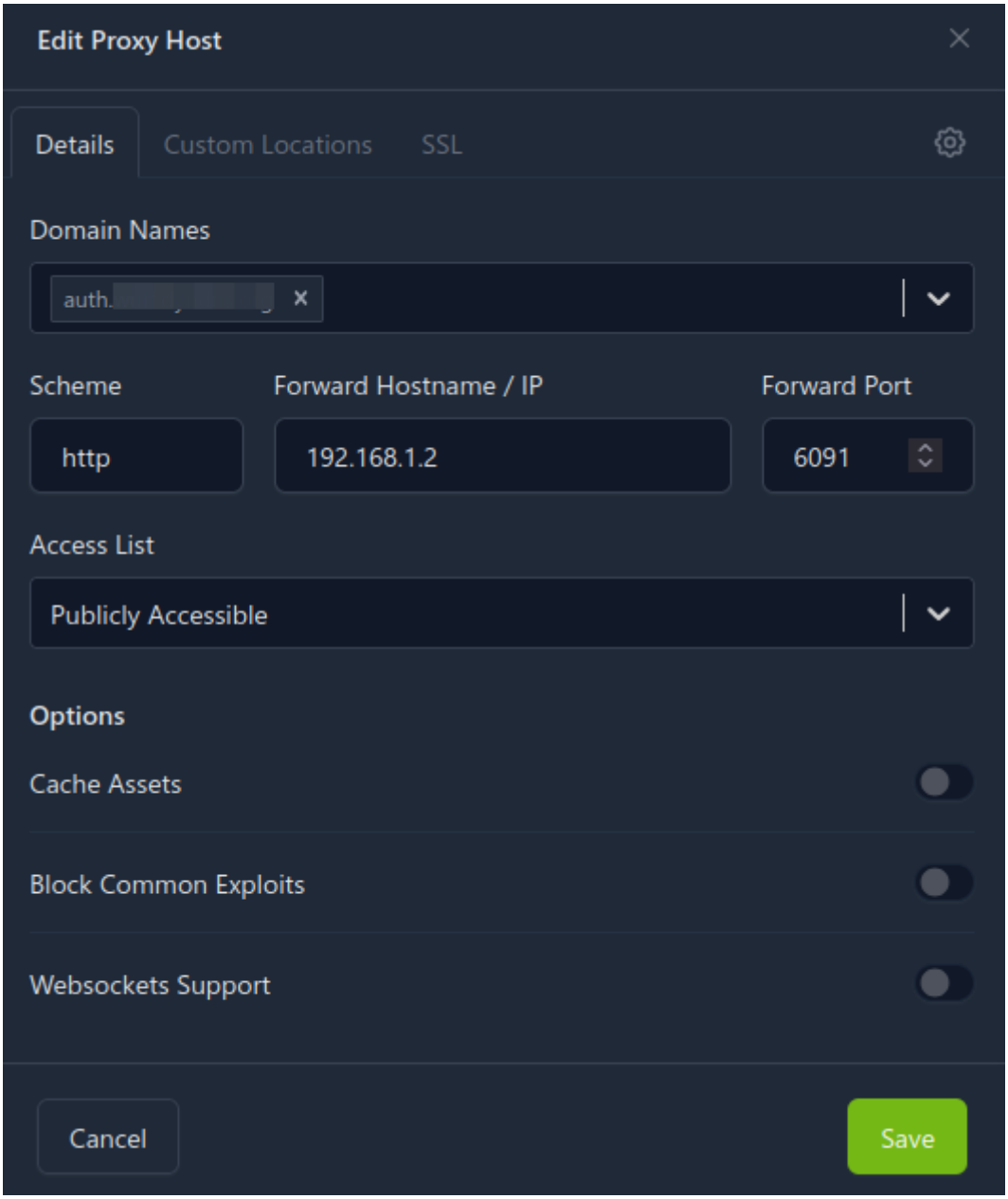
```
docker compose -f docker-npm.yaml up -d
docker compose -f docker-authelia.yaml up -d
```

NPM GUI Configuration

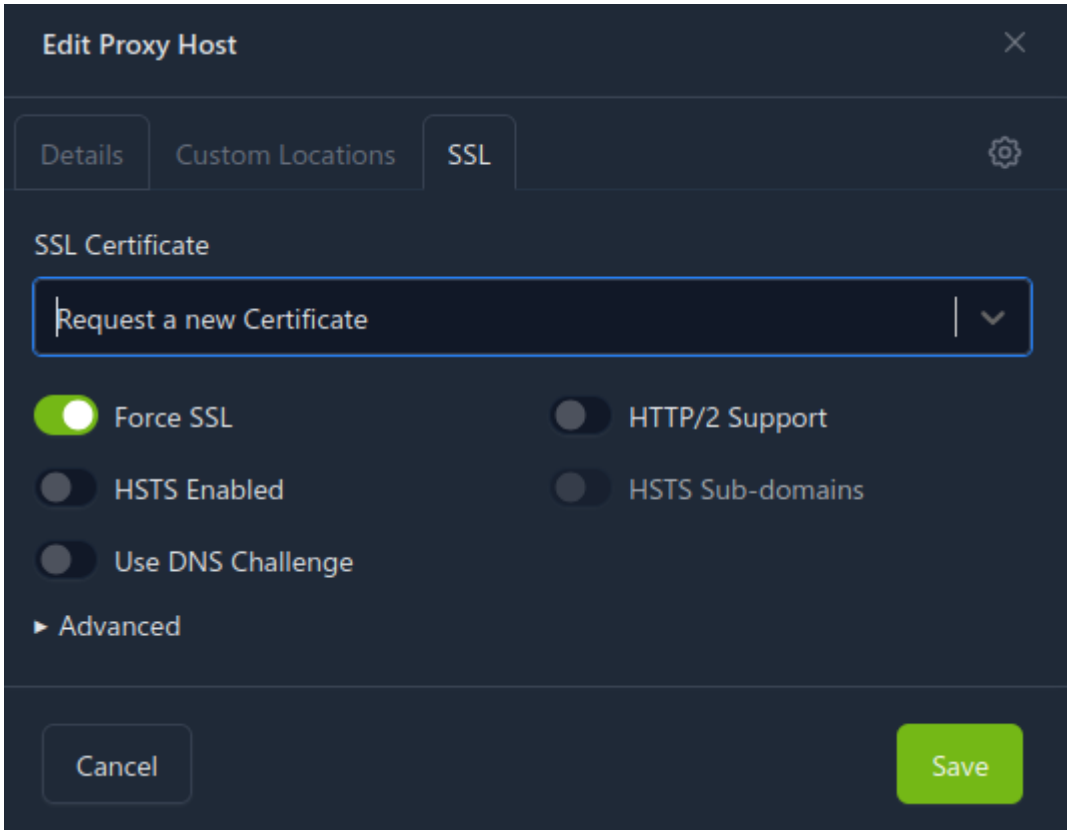
The default NPM GUI is internally accessible on http://192.168.1.2:71 (replace with other IP/port if different). Log in using the initial admin email and password as configured in the npm docker yaml file.

Go to Hosts → Proxy Hosts 

Set up the first subdomain 'auth' for Authelia, enter auth.example.com with your domain and click create in the domain names, set it to publicly accessible and point it to your internal IP and Authelia



port: On the SSL Certificate tab, set it to 'Request a new Certificate' and Force SSL

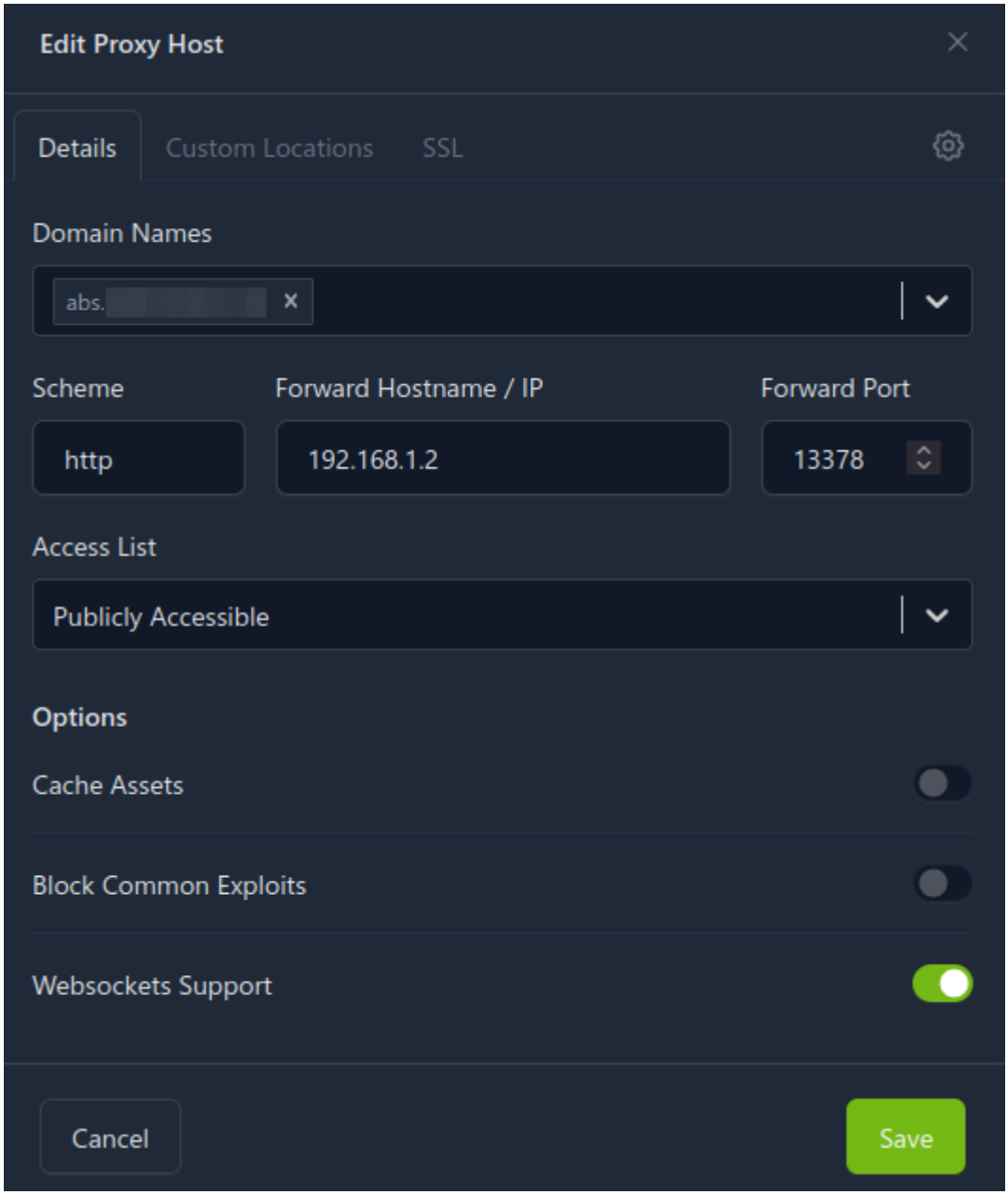


On the cog icon for advanced configuration paste this:

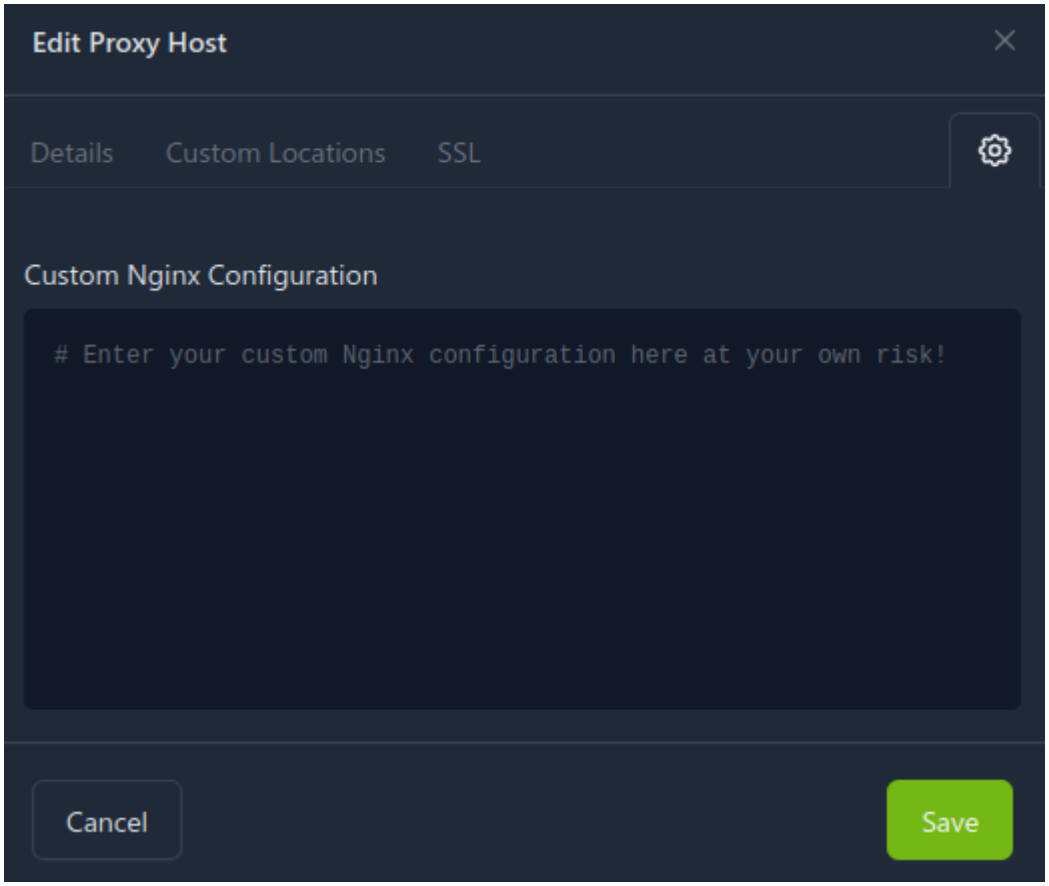
```
location / {
    include /snippets/proxy.conf;
    proxy_pass $forward_scheme://$server:$port;
}
```

Click save.

Then proceed with the first real subdomain.



Request a new ssl certificate as in the auth subdomain. To use Authelia for the main authentication, click the cog icon and paste the below configuration and adjust it to your setup:



Advanced nginx config for Authelia authentication.

```
include /snippets/authelia-location.conf;
location / {
    include /snippets/proxy.conf;
    include /snippets/websocket.conf;
    include /snippets/authelia-authrequest.conf;
    proxy_pass $forward_scheme://$server:$port;
}
```

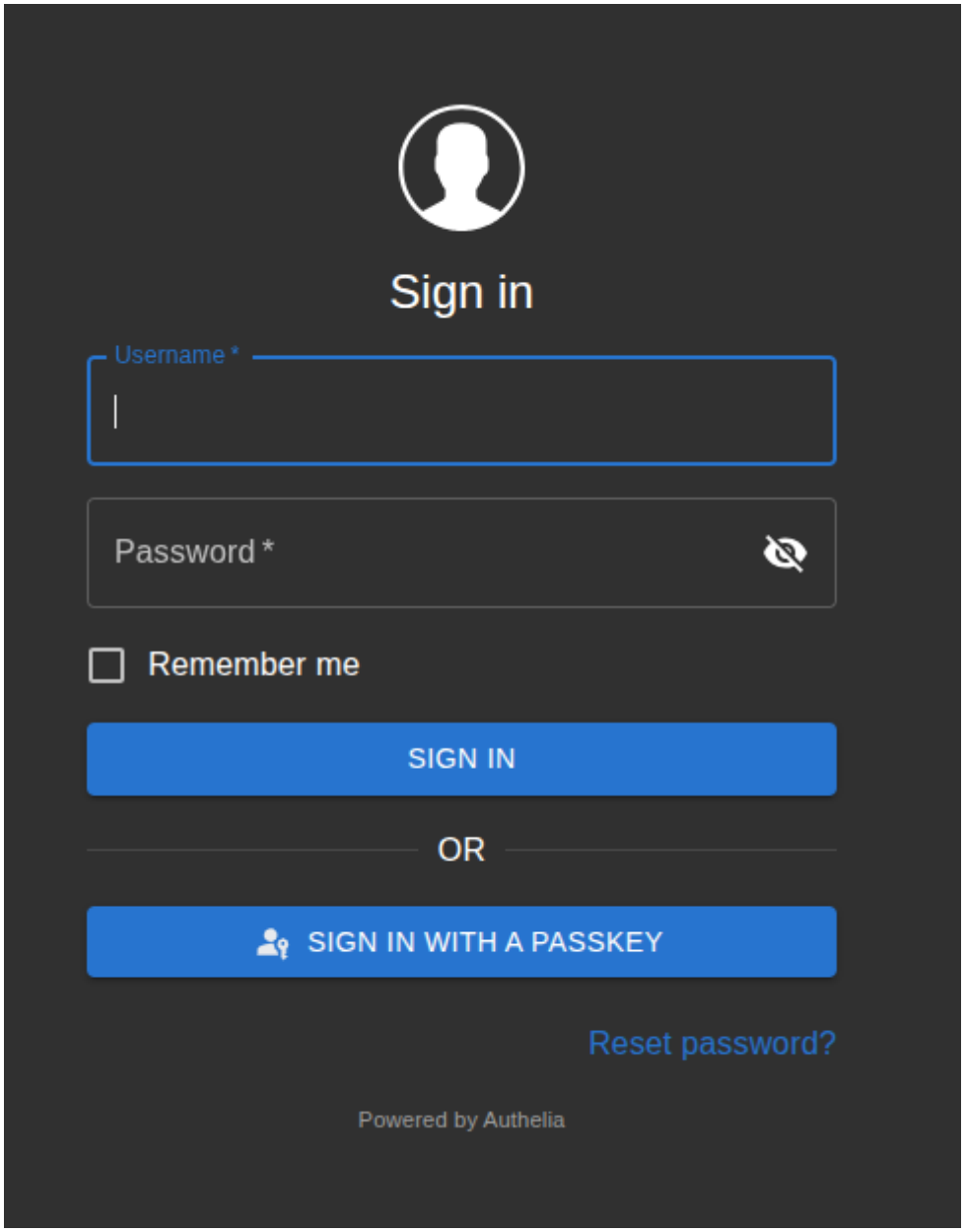
Click save.

Repeat for all additional subdomains as required.

Authelia login

Going to https://auth.example.com (replace with your domain) will show the authelia login prompt now. You can log in and set up 2FA, see your authentication status and change your password and registered devices.

This configuration enables 2FA by default, so logging in to any of the services will prompt 2FA setup. The auth code will be stored in /opt/authelia/notification.txt and not emailed to the user!



By default, authentication is valid for 1 hour or 30 days when the remember me is checked on login. This can be changed in the authelia configuration.yml

Authelia email backend

gmail example for email backend. Requires google mail app password.

```
notifier:
  ## You can disable the notifier startup check by setting this to true.
  disable_startup_check: true

  ##
  ## File System (Notification Provider)
  ##
  #filesystem:
  #  filename: '/config/notification.txt'
```

```
  smtp:
    address: 'smtp://smtp.gmail.com:587'
    username: 'email address'
    password: 'app password'
    sender: 'Authelia <noreply@domain>'
    subject: '[Authelia] {title}'
    startup_check_address: 'email address'
```

Authelia SSO backend

Single sign on through authelia for booklore:

```
server:
  endpoints:
    authz:
      forward-auth:
        implementation: 'ForwardAuth'
        authn_strategies:
          - name: 'HeaderAuthorization'
            schemes:
              - 'Basic'
              - 'Bearer'
          - name: 'CookieSession'
      ext-authz:
        implementation: 'ExtAuthz'
        authn_strategies:
          - name: 'HeaderAuthorization'
            schemes:
              - 'Basic'
              - 'Bearer'
          - name: 'CookieSession'
      auth-request:
        implementation: 'AuthRequest'
        authn_strategies:
          - name: 'HeaderAuthRequestProxyAuthorization'
            schemes:
              - 'Basic'
              - 'Bearer'
          - name: 'CookieSession'
      legacy:
        implementation: 'Legacy'
        authn_strategies:
          - name: 'HeaderLegacy'
          - name: 'CookieSession'



##
## Identity Providers
##
```

```
#Valid keys can be generated using the following commands:
#openssl genrsa -out private.pem 2048
#openssl rsa -in private.pem -outform PEM -pubout -out public.pem

identity_providers:
  oidc:
    jwks:
      - algorithm: 'RS256'
        use: 'sig'
        key: |
          -----BEGIN PRIVATE KEY-----
          MIIEvQI....
          iulbfHg....
          -----END PRIVATE KEY-----

    clients:
      -
        client_id: 'booklore'
        client_name: 'BookLore'
        public: true
        authorization_policy: 'two_factor'
        require_pkce: true
        pkce_challenge_method: 'S256'
        redirect_uris:
          - 'https://booklore.domain/oauth2-callback'
        scopes:
          - 'openid'
          - 'offline_access'
          - 'profile'
          - 'email'
          - 'groups'
        response_types:
          - 'code'
        grant_types:
          - 'authorization_code'
          - 'refresh_token'
        access_token_signed_response_alg: 'none'
        userinfo_signed_response_alg: 'none'
        token_endpoint_auth_method: 'none'
```

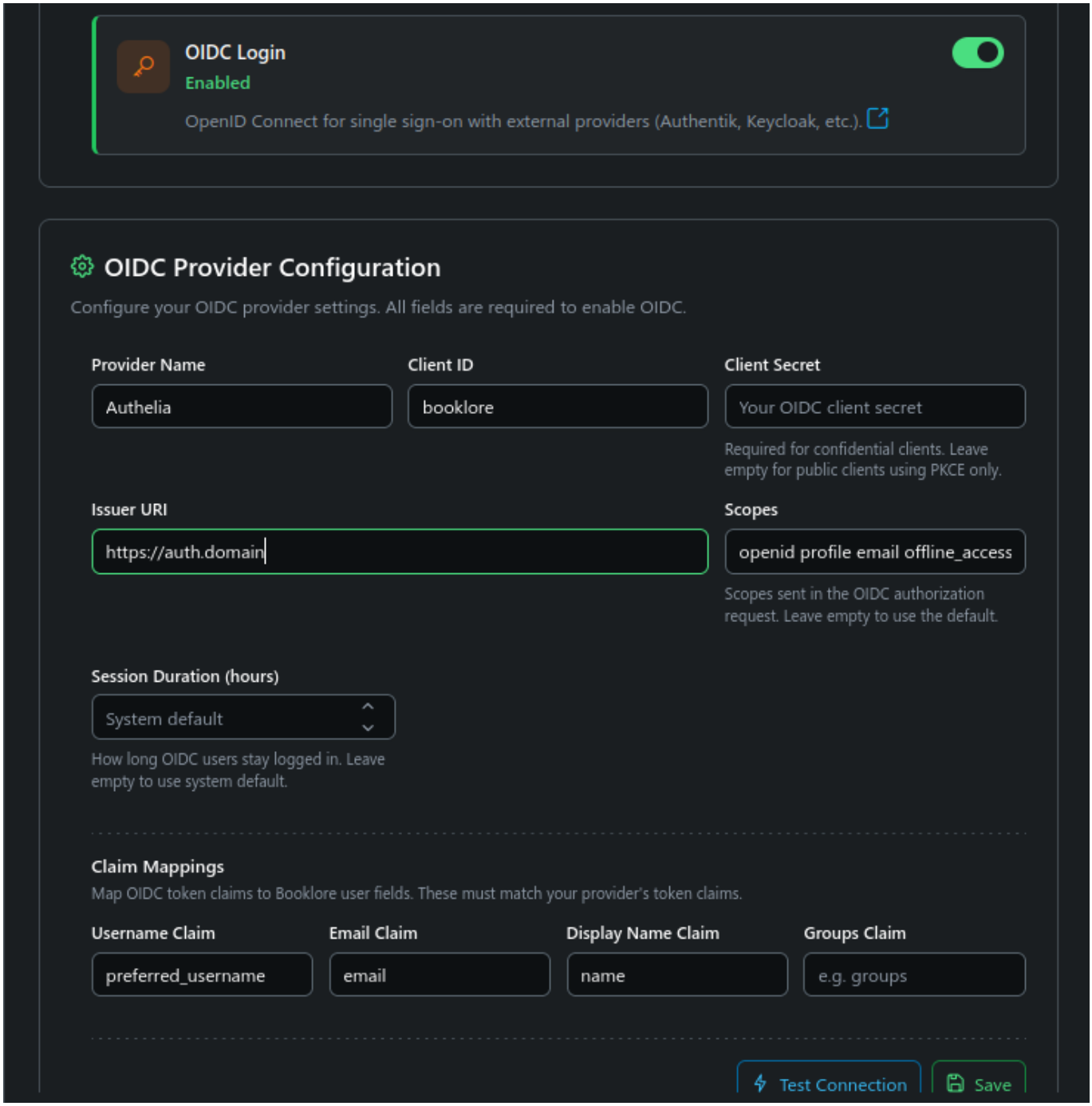
Restart Authelia for the changes to take effect!

In booklore gui: Settings → OIDC Settings:

- In the top right click the settings icon (looks like a cog)
- Provider Name: Authelia
- Client ID: booklore
- Issuer URI: https://auth.example.com
- Scope: openid profile email offline_access
- Username Claim: preferred_username
- Email Claim: email

- Display Name Claim: name
- Test, then click Save Settings.
- OIDC Enabled: Toggle to the on Position above the settings.

Further down on the page, “OIDC-Only Mode” can be enabled to enforce authelia and prevent local user login. Admins can still log in with password only using /login?local=true



Make sure to remove the advanced nginx config in NPM and enable websocket support in NPM. The OIDC integration replaces the proxy authelia protection.