1.5 Authentication

- Introduction
- 1.5.1 Vaultwarden
    - 01. Installation
- 1.5.2 Authenticator Apps
- 1.5.3 Authentik (SSO)
    - 01. Docker Installation
    - 02. Configuration

Introduction

For all the apps that we are using in daily or professional life we need authentication. Hence, before we deploy any apps it is advised to have an authentication system in place that your self hosted apps can use to authenticate. There are basically four ways one can authenticate

1. The classical with username and password
2. With a verified email address and password
3. Multi factor Authentication with an open source Authenticator app
4. Single Sign on by using Authentication providers like Authentik

Now, they all come with their pros and cons as with anything in life ;).

1.5.1 Vaultwarden

Vaultwarden is a password manager with the sole goal of creating random usernames and passwords for all of your applications. It leverages the Bitwarden application for both the web and on the phone. You don't need to remember any of the credentials. That is the job of Vaultwarden. The only thing you need to do is click on sign in with the most complicated usernames and passwords.

01. Installation

1.5.2 Authenticator Apps

Open Source authentication apps to replace Microsoft Authenticator, Google Authenticator, and other proprietary versions.

1.5.3 Authentik (SSO)

Authentik is an open source OIDC that provides single sign on possibilities for all the users of apps in your home lab. It supports a vast amount of authentication tools like Oauth2.0/OpenID, Radius and more. We will setup up Authentik wiht two factor authentication for our users. That means each time a person logs into an application with Authentik they need to use a 2FA system for it.

01. Docker Installation

Official Installation Page: Authentik Docker Installation

Again as usual we will leverage Dockge to install Authentik on our Docker VM that we have created in our Proxmox Server. If you have not done this yet then please go to Docker VM Install.

1. The first thing we do is set the values for our reverse proxy. By default, Authentik listens internally on port 9000 for HTTP and 9443 for HTTPS. However, if you want to change that you can do that in the environment variables file below.

2. Since, Authentik supports HTTPS we will then also use HTTPS for our Caddy reverse proxy entry

> https://authentik.example.org ---> https://IP-Address-Docker-VM:9443

3. According to the documentation we can obtain the docker compose file with the following command

```
wget https://docs.goauthentik.io/compose.yml
```

Docker Compose File

```
services:
  postgresql:
    env_file:
      - .env
    environment:
      POSTGRES_DB: ${PG_DB:-authentik}
      POSTGRES_PASSWORD: ${PG_PASS:?database password required}
      POSTGRES_USER: ${PG_USER:-authentik}
    healthcheck:
      interval: 30s
      retries: 5
      start_period: 20s
      test:
        - CMD-SHELL
```

```
      - pg_isready -d ${POSTGRES_DB} -U ${POSTGRES_USER}
    timeout: 5s
  image: docker.io/library/postgres:16-alpine
  restart: unless-stopped
  volumes:
    - ./database:/var/lib/postgresql/data
server:
  command: server
  depends_on:
    postgresql:
      condition: service_healthy
  env_file:
    - .env
  environment:
    AUTHENTIK_POSTGRESQL__HOST: postgresql
    AUTHENTIK_POSTGRESQL__NAME: ${PG_DB:-authentik}
    AUTHENTIK_POSTGRESQL__PASSWORD: ${PG_PASS}
    AUTHENTIK_POSTGRESQL__USER: ${PG_USER:-authentik}
    AUTHENTIK_SECRET_KEY: ${AUTHENTIK_SECRET_KEY:?secret key required}
  image: ${AUTHENTIK_IMAGE:-ghcr.io/goauthentik/server}:${AUTHENTIK_TAG:-2026.5.2}
  ports:
    - ${COMPOSE_PORT_HTTP:-9000}:9000
    - ${COMPOSE_PORT_HTTPS:-9443}:9443
  restart: unless-stopped
  shm_size: 512mb
  volumes:
    - ./data:/data
    - ./custom-templates:/templates
worker:
  command: worker
  depends_on:
    postgresql:
      condition: service_healthy
  env_file:
    - .env
  environment:
    AUTHENTIK_POSTGRESQL__HOST: postgresql
    AUTHENTIK_POSTGRESQL__NAME: ${PG_DB:-authentik}
    AUTHENTIK_POSTGRESQL__PASSWORD: ${PG_PASS}
```

```
      AUTHENTIK_POSTGRESQL__USER: ${PG_USER:-authentik}
      AUTHENTIK_SECRET_KEY: ${AUTHENTIK_SECRET_KEY:?secret key required}
    image: ${AUTHENTIK_IMAGE:-ghcr.io/goauthentik/server}:${AUTHENTIK_TAG:-2026.5.2}
    restart: unless-stopped
    shm_size: 512mb
    user: root
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
      - ./data:/data
      - ./certs:/certs
      - ./custom-templates:/templates
```

4. We then grab this docker compose file and copy paste it into our Dockge container and change the following. Most of the docker compose file can stay in tact but if you are like me and want to keep all files for one project in one folder then you should remove the docker database volume and replace it with ./database.

5. Next is creating an environment file that serves the values requested in the compose file

```
PG_DB=ksjdflkjslkdfjlkoi3oi
PG_PASS=skdjf23r98ir0u398t34hjf9
PG_USER=soidfj039jf9jf988384j34f
AUTHENTIK_SECRET_KEY=jsjdflkjslfj0i3jf0u90u39u9iu9jjf
AUTHENTIK_IMAGE=ghcr.io/goauthentik/server
AUTHENTIK_TAG=2026.5.2
COMPOSE_PORT_HTTP=9001
COMPOSE_PORT_HTTPS=9444

# SMTP server
AUTHENTIK_EMAIL__HOST=smtp.porkbun.com
AUTHENTIK_EMAIL__PORT=587
# Optionally authenticate (don't add quotation marks to your password)
AUTHENTIK_EMAIL__USERNAME=dp@p2pnode.org
AUTHENTIK_EMAIL__PASSWORD=PuV9#@pUca49C70e2V46&keg
# STARTTLS / explicit TLS, usually on port 587
AUTHENTIK_EMAIL__USE_TLS=true
# Implicit TLS/SSL on the SMTP connection (`USE_SSL` is the variable name), usually on port
465
```

```
AUTHENTIK_EMAIL__USE_SSL=false
AUTHENTIK_EMAIL__TIMEOUT=10
# Sender email address; verify that the domain is valid.
AUTHENTIK_EMAIL__FROM=dp@p2pnode.org
```

6. If that is all set we can deploy our setup and go to the domain name you have set for your Authentik server.

02. Configuration

This document describes briefly how to setup Authentik for the first time and how to implement 2FA for its users.

1. The first thing you need to do is create an admin account by providing an email and a password. It is recommended to use a random generated password by Bitwarden and save it in your vault.

2. We will then go to Admin Interface -> Users -> Change the akadmin username to your preferred username

3. Then finally we can already set up MFA for having access to Authentik by going to Settings -> Credentials -> Enroll. Here you select TOTP.

4. Now, it is time to start using your Authenticator app. This can either be Aegis, Bitwarden Authenticator or Ente Auth. The last one I would recommend since it is the one that keeps all your credentials automatically backed up by your Ente instance. However, if you did not setup Ente yet I would recommend Aegis for now. You can always later transfer the credentials to Ente Auth later.

5. So now you have setup 2FA for your account but how to setup 2FA for all new users that want to create an Authentik account on your instance ?

6. For this go to Admin Interface -> Flows & Stages -> Flows -> default-authentication-flow -> Stage bindings

7.