Authentik

- Setup Authentik
  - Install and Setup Authentik

Setup Authentik

Install and Setup Authentik

https://www.youtube.com/embed/KlDJ4K45M_o

Introduction

As the internet and services have become more and more the norm in our society protecting our services with authentication has become ever more important as well. This is where amazing projects like Authentik, and Authelia come in. These solutions give you the opportunity to setup a Single Sign On system for all of your services. That means you just need to have one very secure login, and you can access any of the services you use everyday.

What You'll Need

- Docker and Docker Compose Installed on a Host that you can access from the Internet
- A domain or sub-domain that you own and can set an A or CNAME Record for.
- NGinX Proxy Manager (or other reverse proxy of your choice)
- an SMTP Email Server
- About 30 minutes of your time

Installation

Install Docker, Docker Compose, and NGinX Proxy Manager via a Simple Script

You can easily install Docker-CE, Docker-Compose, Portainer-CE, and NGinX Proxy manager by using this quick install script I created and maintain on Github. Just use the command:

```
wget https://gitlab.com/bmcgonag/docker_installs/-/raw/main/install_docker_nproxyman.sh
```

To download the script to your desired host.

Change the permissions to make the script executable:

```
chmod +x ./install_docker_nproxyman.sh
```

and then run the script with the command:

`./install_docker_nproxyman.sh`

When run, the script will prompt you to select your host operating system, then will ask you which bits of software you want to install.

Simply enter 'y' for each thing you want to install.

At some point, you may be asked for your super user (sudo) password as well.

Allow the script to complete installation.

At this point, you might want to log out and back in, as this will allow you to use the `docker` and `docker-compose` commands without the need of `sudo` in front of them.

Configure and Install Authentik

Let's create our folder structure. First, we want to create a parent 'docker' folder, and inside that place any / all of our application folders we may want to run on this host. In this case our application folder will be 'authentik'.

`mkdir -p docker/authentik`

Next, we'll move into our new folder and create our docker-compose.yml file. This file defines our application containers, and helps put them all into a private network on our host so the various parts of the overall application can communicate securely.

`nano docker-compose.yml`

Copy the block of yaml code from below, and paste it into the document we just opened. You can paste in the linux terminal by right clicking, and selecting paste, or by using CTRL + Shift + V.

```
version: "3.4"

services:
  postgresql:
    image: docker.io/library/postgres:16-alpine
    restart: unless-stopped
    healthcheck:
      test: ["CMD-SHELL", "pg_isready -d ${POSTGRES_DB} -U ${POSTGRES_USER}"]
      start_period: 20s
      interval: 30s
      retries: 5
      timeout: 5s
```

```
    volumes:
      - ./database:/var/lib/postgresql/data
    environment:
      POSTGRES_PASSWORD: ${PG_PASS:?database password required}
      POSTGRES_USER: ${PG_USER:-authentik}
      POSTGRES_DB: ${PG_DB:-authentik}
    env_file:
      - .env
  redis:
    image: docker.io/library/redis:alpine
    command: --save 60 1 --loglevel warning
    restart: unless-stopped
    healthcheck:
      test: ["CMD-SHELL", "redis-cli ping | grep PONG"]
      start_period: 20s
      interval: 30s
      retries: 5
      timeout: 3s
    volumes:
      - ./redis:/data
  server:
    image: ${AUTHENTIK_IMAGE:-ghcr.io/goauthentik/server}:${AUTHENTIK_TAG:-2025.6.3}
    restart: unless-stopped
    command: server
    environment:
      AUTHENTIK_REDIS__HOST: redis
      AUTHENTIK_POSTGRESQL__HOST: postgresql
      AUTHENTIK_POSTGRESQL__USER: ${PG_USER:-authentik}
      AUTHENTIK_POSTGRESQL__NAME: ${PG_DB:-authentik}
      AUTHENTIK_POSTGRESQL__PASSWORD: ${PG_PASS}
    volumes:
      - ./media:/media
      - ./custom-templates:/templates
    env_file:
      - .env
    ports:
      - "${COMPOSE_PORT_HTTP:-9000}:9000"
      - "${COMPOSE_PORT_HTTPS:-9443}:9443"
    depends_on:
      - postgresql
```

```
      - redis
  worker:
    image: ${AUTHENTIK_IMAGE:-ghcr.io/goauthentik/server}:${AUTHENTIK_TAG:-2025.6.3}
    restart: unless-stopped
    command: worker
    environment:
      AUTHENTIK_REDIS__HOST: redis
      AUTHENTIK_POSTGRESQL__HOST: postgresql
      AUTHENTIK_POSTGRESQL__USER: ${PG_USER:-authentik}
      AUTHENTIK_POSTGRESQL__NAME: ${PG_DB:-authentik}
      AUTHENTIK_POSTGRESQL__PASSWORD: ${PG_PASS}
    # `user: root` and the docker socket volume are optional.
    # See more for the docker socket integration here:
    # https://goauthentik.io/docs/outposts/integrations/docker
    # Removing `user: root` also prevents the worker from fixing the permissions
    # on the mounted folders, so when removing this make sure the folders have the correct
UID/GID
    # (1000:1000 by default)
    # user: root   # <--- changed as of 9/15/2025 due to errors in console. Suggested by
Authentik community support.
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
      - ./media:/media
      - ./certs:/certs
      - ./custom-templates:/templates
    env_file:
      - .env
    depends_on:
      - postgresql
      - redis
volumes:
  database:
    driver: local
  redis:
    driver: local
```

> NOTE: In the above, if you used it in the past, I have adjusted it to remove (comment out) one line which said `user: root` based on errors after an upgrade to 2025.8.2. Many people had the same error, and this was determined to be the cause.

Once you've pasted in the code, save the file with CTRL + O, then press Enter to confirm, and exit the nano editor with CTRL + X.

Next we need to create an environment variable file that the dockeer-compose.yml file will read. This file allows us to customize our variables for the container setup without having to repeatedly enter like values throughout the docker-compose file.

`nano .env`

> Note: The period in front of the filename. In unix based systems a filename starting with a period is a hidden file, and will not be shown with commands like `ls` alone. Instead you must use the flag `-a` (all) to show everything in a folder including hidden files, like this `ls -a`.

Once open, copy the file contents below, and paste them into the .env file.

```
PG_USER=authentik
PG_PASS=aReallyLongStrongPasswordShouldBePutHere
AUTHENTIK_SECRET_KEY=someincrediblylongcomplexkeygoeshere
AUTHENTIK_ERROR_REPORTING__ENABLED=true
# SMTP Host Emails are sent to
AUTHENTIK_EMAIL__HOST=smtp.example.com
AUTHENTIK_EMAIL__PORT=587
# Optionally authenticate (don't add quotation marks to your password)
AUTHENTIK_EMAIL__USERNAME=auth@example.com
AUTHENTIK_EMAIL__PASSWORD=a-L0n6-Strong_password_should_go_here
# Use StartTLS
AUTHENTIK_EMAIL__USE_TLS=true
# Use SSL
AUTHENTIK_EMAIL__USE_SSL=false
AUTHENTIK_EMAIL__TIMEOUT=10
# Email address authentik will send from, should have a correct @domain
AUTHENTIK_EMAIL__FROM=auth@example.com
COMPOSE_PORT_HTTP=80
COMPOSE_PORT_HTTPS=443
# Authentik Version to Pull
ATHENTIK_TAG=2025.6.3
```

I have intentionally added placeholders to most of the values. You'll want to go through and update these placeholders to be actual values for your installation. The values you must change at a minimum are:

PG_PASS - This value can be any long, strong password you want.

AUTHENTIK_SECRET_KEY - This values should be a long string of numbers, letters (upper and lower case), and symbols at least 64 characters in length.

You should additionally setup the SMTP email options as this is a major part of user management and self-service in any authentication provider, such as when a user needs to reset his / her password, MFA device, register (if allowed). It's also how the Authentik install will send you information about events, updates, etc.

Finally, for the COMPOSE_PORT_HTTP and COMPOSE_PORT_HTTPS you may want to change these to be ports that aren't quite so common. If your host is also running NGINX Proxy Manager you should definitely change these to less common ports, as 80 and 443 are already in use by NPM.

Once you've updated the values appropriately, you can save and close the file with CTRL + O, then Enter to confirm, and then CTRL + X.

Now you are ready to bring up your Authentik application. Use the commands:

```
docker compose up -d && docker compose logs -f
```

The first part tells docker to get the necessary images and start the containers running in the background, and the second part tells docker once the containers are up, show us the logs.

You can discontinue the logs with CTRL + C.

Now navigate to the IP address of your host machine, and the port you set for the non SSL (HTTP) access in the .env file. You should see the Authentik login page. If so, then you are doing well. Next, setup the reverse proxy.

Reverse Proxy Setup

Login to NGinX Proxy Manager (NPM) and click into the Proxy Hosts section. Select ‘Add Proxy Host’ from the upper right, and int he modal (pop-up) window that opens, we'll begin adding the information needed to get our domain name to resolve to our new server.

In the domain name field, enter your desired domain name. I used ‘authentik.routemehome.org’.

I own the domain ‘routemehome.org’, and have setup an A-record in DNS on the domain that will point the domain name to my public IP address.

Next, enter the private IP of your Authentik server. You can use ‘localhost’ if the Authentik server is running on the machine you are running NPM on. Next, enter the port number you entered in the .env file. If you didn't change it, it will be 80.

Enable the options for ‘Block Common Exploits’ and ‘Websocket Support’.

Now move to the SSL tab, and select “Request a New Certificate” from the drop-down box.

Enable the options for ‘Force SSL’, ‘HTTP/2 Support’, and both ‘HSTS’ options.

Enter your email address in the Email field, and enable the ‘Agree to Terms of Service’ option.

Click ‘Save’. If all is setup properly, the modal window will simply go away after a few seconds (maybe 30). Then you'll have a CA Certified LetsEncrypt certificate for your Authentik server.

You should now be able to get to your Authentik install using the FQDN (fully qualified domain name).

Configure

To create your first (admin) user, you need to go to your domain at a special address.

https://authentik.yourdomain.com/if/flow/initial-setup/

Of course, use your actual domain name.

Enter your preferred email, and learn from my mistake and make sure it's correct before moving on. Next enter a long, strong password, then enter it again to confirm it. I cannot stress enough, you should absolutely use a password manager for keeping and helping you create long, strong passwords. I highly recommend Bitwarden, and Vaultwarden if you prefer to self host.

Click Create Account, and you should be logged in as an administrative user.

Proxy Login

Make sure you have an outpost setup, and setup properly. We can use the authentik embedded outpost in this case.

Make Sure Outpost is set Correctly

Click on Outposts on the left side menu, then click the edit icon in the list of outposts. if this is a new install you likely only have one outpost at this point.

In the pop-up window that opens, verify that your ‘authentik_host’ has the same URL as your Authentik site. If you are using https://auth.my-great-domain.com to reach your Authentik install, then you want this value to be the same.

You can now dismiss the pop-up if everything looks good. Additionally, in the outposts list, you should see a green check mark under health and version.

Add a Provider

Next we need to add a provider for our setup.

Click on ‘Providers’ in the left side menu.

In this view, click on ‘Create’ at the top.

Again, when the pop-up window opens, select ‘Proxy Provider’, then click Next.

On the next page, enter a name that identifies the application you are creating a proxy authenticator for. The Proxy authentication is useful for applications and services you run that don't provide authentication in the app, but that you might like to expose to the internet. For instance, a dashboard, speedtest, etc.

Choose ‘Authorize Application’ from the Authentication Flow drop down.

Next, what you select will depend on how you are proxying your service traffic to your applications and service. If you are using a reverse proxy like NGinX Proxy Manager, Caddy, Traefik, etc, then you'll want to choose ‘Forward Auth (single application)’. If, however, you don't yet have a reverse proxy setup, then you'll potentially want to use the ‘Proxy’ option, which then turns Authentik into a reverse proxy for the site as well.

In our case we'll go with ‘Forward Auth (single application)’.

Now fill in your sites externally available FQDN (fully qualified domain name), for instance https://mydash.example.com

You can change Token Validity to any timeframe you wish, but leaving it at 24 hours should be fine.

Click ‘Finish’.

You should be returned to your ‘Provider’ list, and you should see the provider you just created. You'll likely have a warning in the list that the provider is not associated to any applications. No worries, we'll fix this right now.

Add an Application Entry for the Provider

In the left side menu click on ‘Applications’.

On the ‘Applications’ page, click the ‘Create’ button at the top, and you'll be presented with a pop-up window (modal window) where you need to fill in your application information.

Give your application a name that helps you easily identify the application. If you are creating an authentication for your dashboard, then call it ‘Dashboard’ for instance. As you fill in the ‘Name’ field, you'll notice that the ‘slug’ field is automatically filled in. The slug should not be changed unless you have a reason to do so, and the slug should always be lowercase, and not have spaces in it.

Now move down to the ‘Provider’ field, and select the provider you just created.

When creating these types of authentication flows, you always want to create a new provider, and a matching application entry for that provider, then select the provider for that application.

Finally, click the ‘Create’ button at the bottom of the modal window.

Ensure the Provider is now Satisfied with the Application Entry

Navigate back to your Providers list on the left, and notice that our provider no longer shows the warning about not being associated to an application.

Set the Application in Outposts

We are almost there. Navigate back to ‘Outposts’, and click the edit icon next to our outpost. In the modal window, next to the ‘Applications’ label, you should now see your application listed. You will click on this line in the field to select the application.

Click the ‘Update’ button at the bottom of the modal window. You should now see the application name listed under the ‘Providers’ column for the outpost. As you add more applications and providers, you'll see them added for any outpost you add them to. In this case, you can use this same outpost for multiple applications and providers needing a simple Forward Auth setup.

Get your Reverse Proxy Snippet

Finally, navigate back to ‘Providers’ on the left side menu, and click the Name of your provider. This will show the Provider details in a new page. Scroll down, and notice various reverse proxy options are shown as tabs. Select the reverse proxy you are using, and you'll see a snippet of code that you'll use to setup your reverse proxy entry so it will start using Authentik before the application will load.

In our case, we've used NGinX Proxy Manager, so I'll click that tab, and copy that bit of code.

I'll open NGinX Proxy Manager and select the 3-dot icon on the row for my application, then select ‘Edit’.

Move to the ‘Advanced’ tab in the modal window, and paste in the code snippet.

In the pasted snippet, scroll down to the line for ‘proxy_pass’, and change the FQDN for your Authentik instance to its internal IP address and port number.

NOTE: This is only done if you are running NGinX Proxy Manager, Authentik, and your Application on the same local area network.

In my case I change authentik.routemehome.org to 192.168.10.42.

Click ‘Save’.

If you now try to load your application, you should be prompted to login with Authentik. Verify you can login using your credentials, and you'd now setup.

OIDC OAuth2

Create an OIDC Provider

Creating an OIDC provider is really pretty straight forward. The more daunting task for me is figuring out what information each application may need for the OIDC connection. Even that is pretty easy though.

Again, for each application that you want to access, you need to create a provider, and an application entry that goes with the provider.

Let's tackle creating the Provider first.

1. Click Providers on the left side menu.
2. Click ‘Create’ at the top.
3. Select ‘OAuth2/OpenID Provider’
4. Click ‘Next’
5. Name your Provider with the name of the application you'll use it with (e.g. 'NextCloud OAuth')
6. Select the default Authentication Flow.
7. Select the default Authorization flow for explicit consent.
8. Leave Client Type as Confidential.
9. If you know you need a special redirect URI / Origin, fill it in, otherwise leave it blank and click ‘Finish’.

A couple of things to note. You may have noticed the fields with Client ID and Client Secret. You'll need these on pretty much every application with OAuth. You can always get back to these values in the Providers list, then click the ‘Edit’ icon to the right of the Provider you need them for.

Create the OIDC Application Entry

Next, we'll need to add an entry for our Application. Remember, each Provider must have an associated Application entry before it can be used for Authentication / Authorization.

1. In the left side menu, navigate to Applications, and click ‘Create’ at the top of the window.
2. Fill in a ‘Name’ for the application (e.g. “Headscale UI”). Notice the ‘Slug’ field will auto-fill. There's no need to change this field.
3. Select the Provider you just created from the drop list.
4. Set ‘Policy engine mode’ to ‘Any’ and click ‘Create’.

Note: If you'd like to have the application represented by it's Logo, then you can upload an image (I recommend png or svg for transparency) under the expandable ‘UI settings’ section before clicking ‘Create’, or anytime in the future by simply editing the Application entry.

If you now navigate back to your Providers list, you'll see that you Provider no longer shows a warning since you've now associated an Application entry to it.

Adding Your OIDC Credentials to the actual Application

This step will greatly depend on the application you are setting up with OIDC. For Headscale WebUI it is very straight-forward. For Nextcloud you need to add a bit more information, but the Nextcloud documentation on doing this is very good.

Generally you'll need 3 or 4 of the values available from your Provider entry.

1. Client ID (sometimes listed as Client Key) - You get this by clicking the edit icon to the right of the provider entry.
2. Client Secret - You get this by clicking the edit icon to the right of the provider entry.
3. OpenID Configuration URL - You get this by clicking on the Provider Name link in the Providers list.
4. OpenID Configuration Issuer - You get this by clicking on the Provider Name link in the Providers list.

There are other URLs listed that may be required for some applications, so familiarize yourself with those labels,, and know they are there in case you need them.

Four our purpose in setting up OIDC for Headscale WebUI we need to use just the Client ID, Client Secret, and Configuration URL.

In your docker-compose.yml file for Headscale / Headscale WebUI, set the Web UI AUTH_TYPE to “oidc”.

Comment out the entries for BASIC_AUTH_USER and BASIC_AUTH_PASS with a hashtag (#).

Uncomment the values for OIDC_AUTH_URL, OIDC_CLIENT_ID, and OIDC_CLIENT_SECRET.

Copy and paste the appropriate values from your Authentik provider, into the proper place next to your entries.

Keep in mind that yaml code is space dependent, so make sure you align everything properly using spaces as you go.

Save, and exit the docker-compose.yml file, and restart your docker-compose. I recommend doing a complete `docker-compose down` and then `docker compose up -d` to make sure the new changes take effect. You can check the compose logs to make sure no errors came up.

Now, navigate to your Headscale WebUI address, and you shud be directed to your Authentik site for login first.

Note: In Firefox, I occasionally get into a login loop. This is something to do with Firefox's caching system. You can usually get around this by logging in through a Private Browsing Window, or using another browser the first time. I have noticed that the loop does stop after completely closing firefox for a little bit.

Finally, you may need to go back into your Provider entry, and adjust the 'Redirect URI / origins" and make sure it has “https” in the address. Getting a Redirect URI error in your Authentik site is an indicator this may not be set properly. My production setup did this without issue, but my testing system put http only for some reason, and once I changed it everything worked perfectly.

Support My Channel and Content

Support my Channel and ongoing efforts through Patreon:

https://www.patreon.com/bePatron?u=234177