Canonical

# authd

Canonical

# Contents

authd is an authentication service for Ubuntu that integrates with multiple cloud identity providers. It offers a secure interface for system authentication, enabling cloud-based identity management for Ubuntu Desktop and Server.

authd has a modular design, comprising an authentication daemon and various identity brokers. This enables authd to support a growing list of identity providers. Currently, authd supports authentication with both MS Entra ID[1] and Google IAM[2]. An example broker is also provided to help developers create new brokers for additional identity services.

If an organization is pursuing cloud-based authentication of Ubuntu workstations and servers, authd is a secure and versatile service to support a full transition to the cloud.

[1] https://learn.microsoft.com/en-us/entra/fundamentals/whatis
[2] https://cloud.google.com/iam/docs/overview

# 1. Supported identity providers

### Google IAM

- Install authd and the Google IAM broker
- Configure the Google IAM broker

### Microsoft Entra ID

- Install authd and the Microsoft Entra ID broker
- Configure the Microsoft Entra ID broker

### Keycloak

- Install authd and the generic OIDC broker
- Configure the generic OIDC broker for Keycloak

# 2. In this documentation

- **Setup**: *Installing authd* (page 5) • *Configuring authd* (page 7) • *Changing authd versions* (page 36)
- **User login**: *Logging in with GDM* (page 19) • *Logging in with SSH* (page 22)
- **Groups and privileges (sudo, docker)**: *Configure user groups* (page 16) • *Group management reference* (page 47)
- **Deployment**: *Deploying with Landscape* (page 48) • *Deploying with cloud-init* (page 51)
- **Network file systems**: *Using with NFS* (page 23) • *Using with Samba* (page 29)
- **authd design**: *Architecture* (page 58) • *Security overview* (page 60)
- **Troubleshooting**: *Accessing logs* (page 32) • *Entering recovery mode on failed login* (page 35)
- **Documentation**: *How this documentation is structured* (page 65)

# 3. Project and community

authd is a member of the Ubuntu family. It's an open source project that warmly welcomes community projects, contributions, suggestions, fixes and constructive feedback.

- Code of conduct[3]
- *Contribute* (page 38)

Thinking about using authd for your next project? Get in touch!

## 3.1. How-to guides

These guides walk you through key operations you can perform with authd.

### 3.1.1. Installation and configuration

Installation of the authd daemon and an identity broker is required to support authentication of Ubuntu devices, with various options available for configuring authentication behavior and user management:

#### Install authd and brokers for identity providers

This project consists of two components:

- **authd**: The authentication daemon responsible for managing access to the authentication mechanism.
- **identity broker**: The services that handle the interface with an identity provider. There can be several identity brokers installed and enabled on the system.

authd is delivered as a Debian package for Ubuntu Desktop and Ubuntu Server.

**(Optional) Switching from stable to edge**

This guide describes how to install the stable version of authd and its brokers, which is recommended for production use.
If you want to try the **edge** version, read our guide on *changing authd versions* (page 36).

#### System requirements

- Ubuntu: Desktop or Server editions
- Release: 24.04 LTS or later
- Architectures: amd64, arm64

[3] https://ubuntu.com/community/ethos/code-of-conduct

### Install authd

On Ubuntu 26.04 LTS, `authd` is available directly from the Ubuntu archive.

**Add PPA before installing on Ubuntu 24.04**

On Ubuntu 24.04 LTS, `authd` must be installed from the stable PPA[4]. Add the PPA before proceeding:

```
sudo add-apt-repository ppa:ubuntu-enterprise-desktop/authd
```

[4] https://launchpad.net/~ubuntu-enterprise-desktop/+archive/ubuntu/authd

Install authd and any additional Debian packages needed for your system of choice:

### Ubuntu Desktop

```
sudo apt install authd gnome-shell yaru-theme-gnome-shell
```

### Ubuntu Server

```
sudo apt install authd
```

### Install brokers

The brokers are provided as snap packages and are available from the Snap Store. Install the broker corresponding to the identity provider that you want to use:

### Google IAM

To install the Google IAM broker, run the following command:

```
sudo snap install authd-google
```

At this stage, you have installed the main service and an identity broker to authenticate against Google IAM.

### Microsoft Entra ID

To install the Microsoft Entra ID broker, run the following command:

```
sudo snap install authd-msentraid
```

At this stage, you have installed the main service and an identity broker to authenticate against Microsoft Entra ID.

### Keycloak

Keycloak can be used with the generic OIDC broker. Install the broker with the following command:

```
sudo snap install authd-oidc
```

At this stage, you have installed the main service and an identity broker to authenticate against Keycloak or any other OIDC provider.

## Configure authd for identity providers

This guide shows how to configure identity brokers to support authentication of Ubuntu devices with authd and your chosen identity provider.

**Logging in multiple users with authd**

By default, the first user to authenticate and log in to a machine becomes the “owner” and only they are allowed to log in.
For other authenticated users to log in, they must first be added as an “allowed user”.
The steps you need to follow when allowing more users are outlined in *configure allowed users* (page 13) on the current page.

### Broker discovery

Copy the `.conf` file from the broker snap to the directory used to declare which brokers are available on the system:

### Google IAM

```
sudo cp /snap/authd-google/current/conf/authd/google.conf /etc/authd/brokers.d/
```

### Microsoft Entra ID

```
sudo cp /snap/authd-msentraid/current/conf/authd/msentraid.conf /etc/authd/brokers.d/
```

### Keycloak

```
sudo cp /snap/authd-oidc/current/conf/authd/oidc.conf /etc/authd/brokers.d/
```

**Note:**

Several brokers can be enabled at the same time.

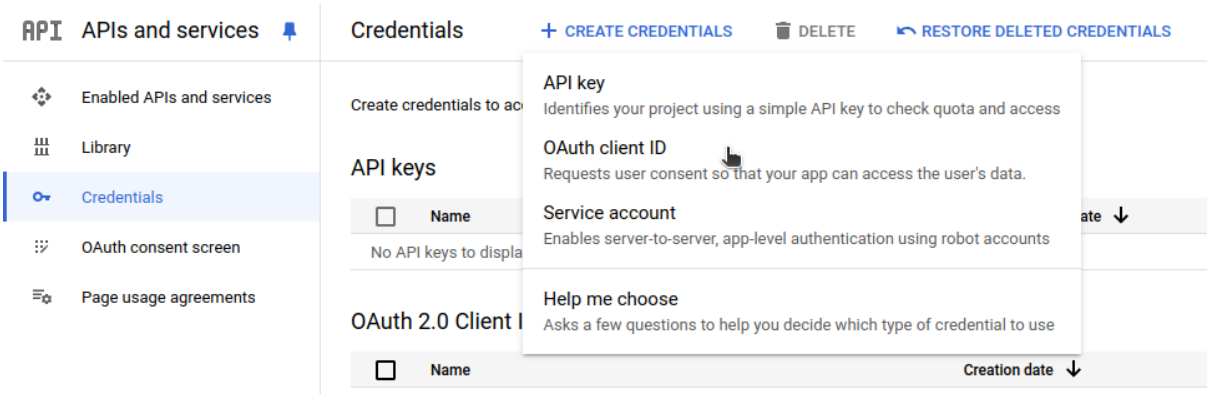
## Application registration

This section demonstrates registering an OAuth 2.0 application that your chosen broker can then use to authenticate users.

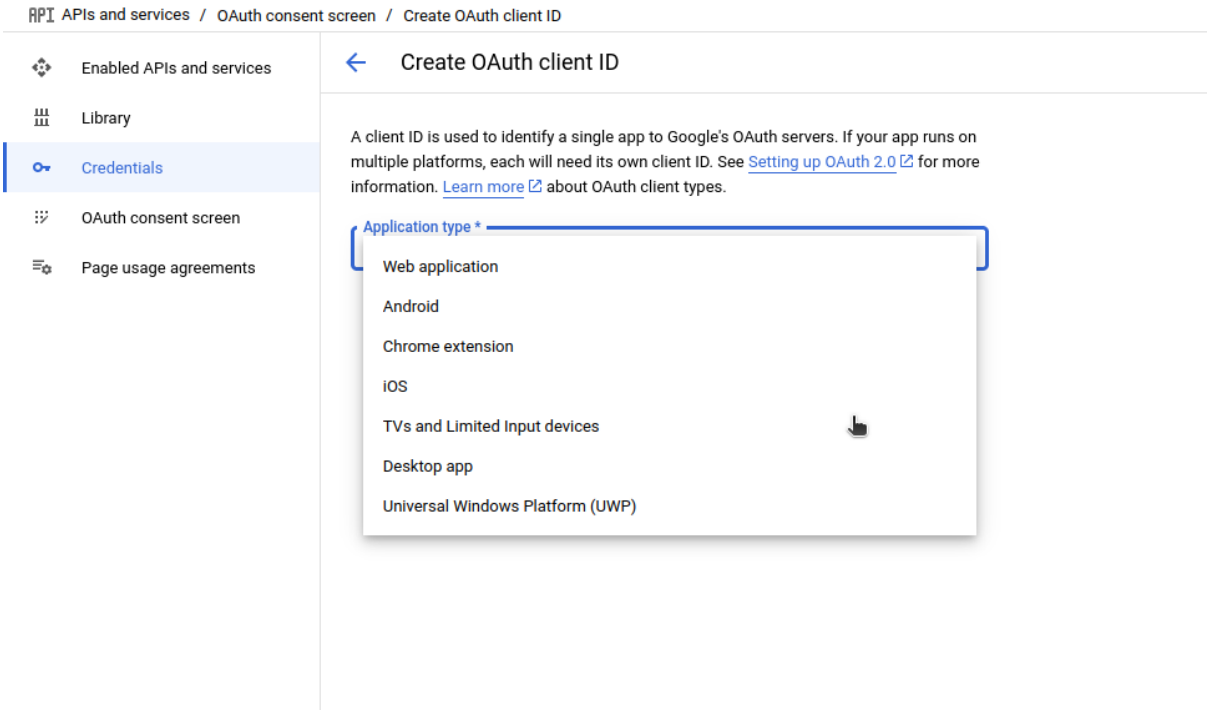
### Google IAM

To register a new application in Google IAM, go to the Credentials page[5].

Click *Create credentials*  *OAuth client ID*.



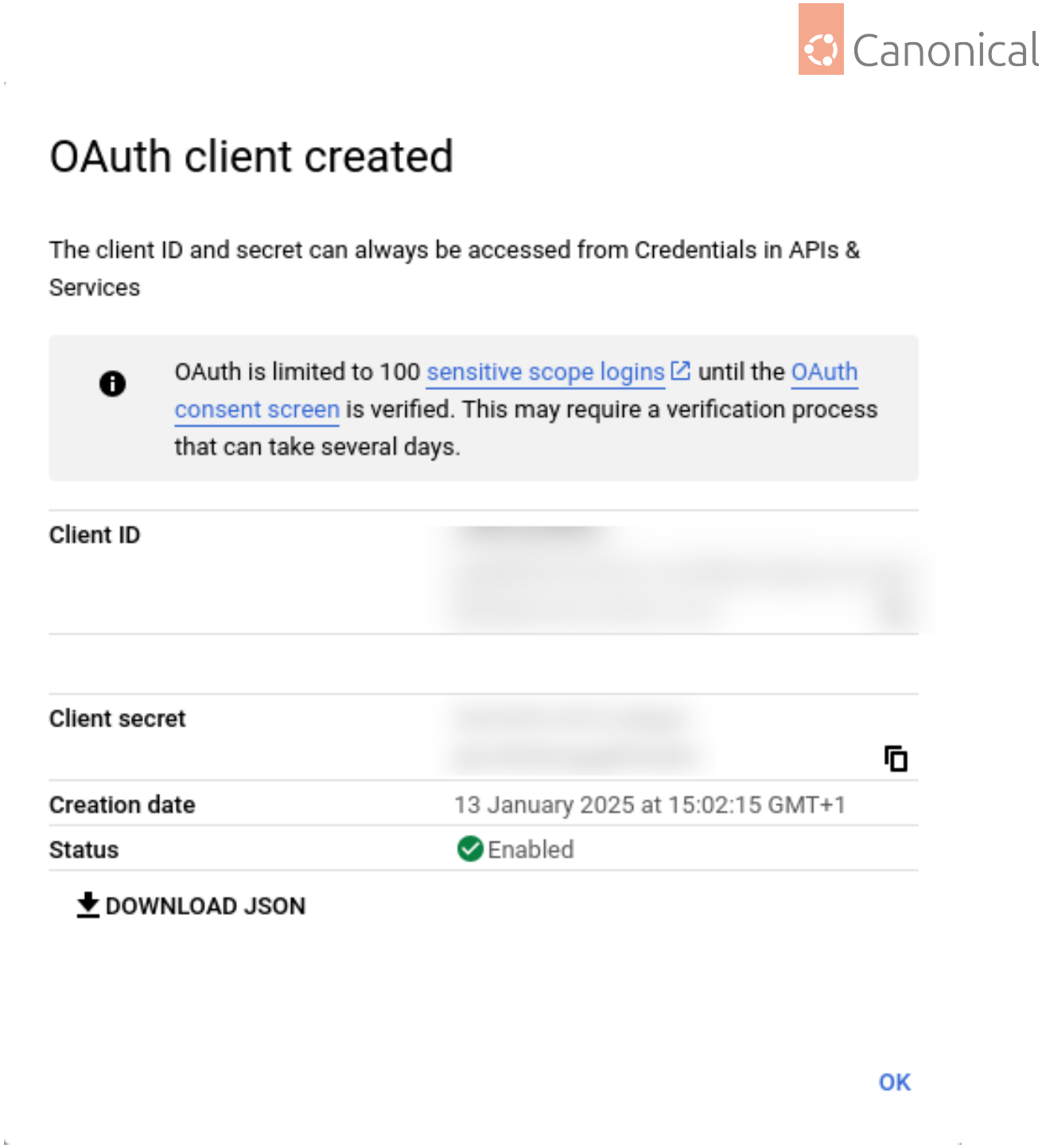

Select the *TVs and Limited Input devices* application type.




Name your OAuth 2.0 client and click *Create*.

Your app's `Client ID` and `Client secret` will be shown on the page, store them somewhere as you will need them in the next step.

[5] https://console.cloud.google.com/apis/credentials

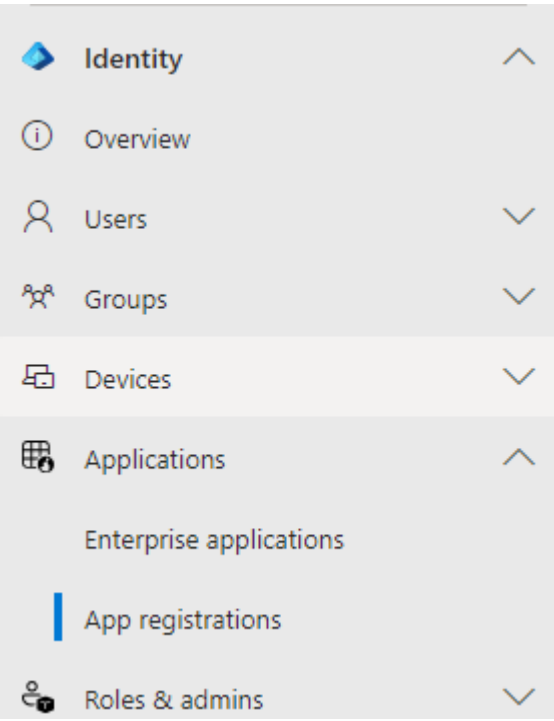



For more detailed information please refer to the OAuth 2.0 for TV and Limited-Input Device Applications documentation[6].

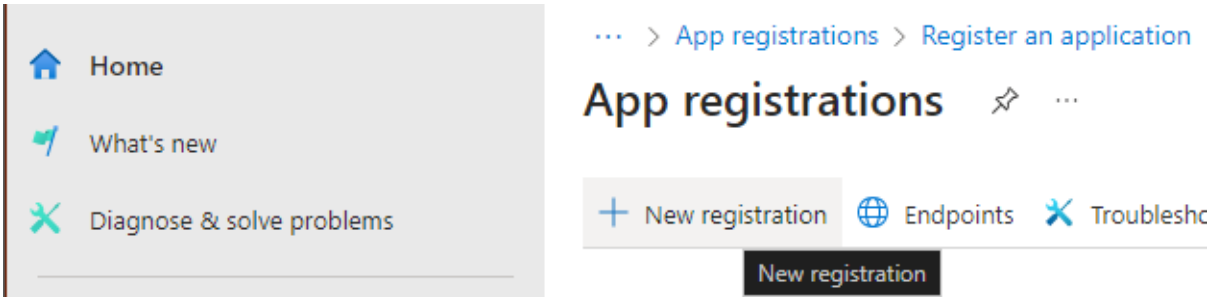
[6] https://developers.google.com/identity/protocols/oauth2/limited-input-device

### Microsoft Entra ID

Register a new application in the Microsoft Entra admin center. To register a new application, select the menu *Entra ID* → *App registrations*:



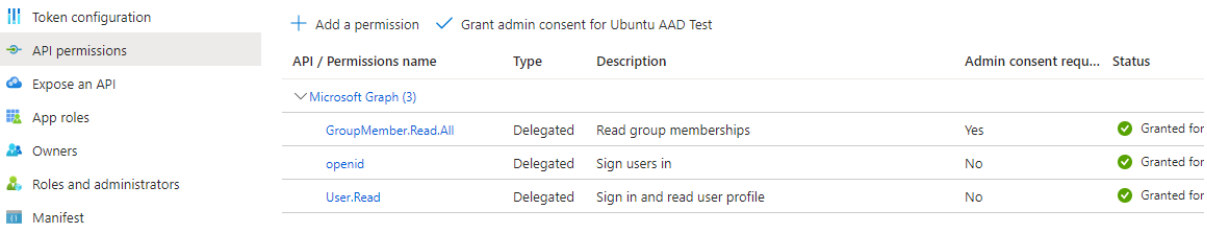

Then *New registration*:




Choose a name for the application, for example “Ubuntu authd”, and the appropriate account type for your use case. Then click on *Register* to create the application.

Once registered, note the *Application (client) ID* and the *Directory (tenant) ID*. These IDs correspond to the `<CLIENT_ID>` and `<ISSUER_ID>`, respectively, which are used in the next section.

In *Manage* → *API permissions*, set the following **Microsoft Graph** permissions:




Ensure the API permission type is set to **Delegated** for each permission.

The *GroupMember.Read.All* permission needs admin consent. Click on *Grant admin consent for <TENANT_NAME>* to provide this consent.

Finally, as the supported authentication mechanism is the device workflow, you need to allow the public client workflows. In *Manage* → *Authentication (Preview)* → *Settings*, ensure that *Allow public client flows* is set to **Enabled**.

The Microsoft documentation[7] provides detailed instructions for registering an application with the Microsoft identity platform.

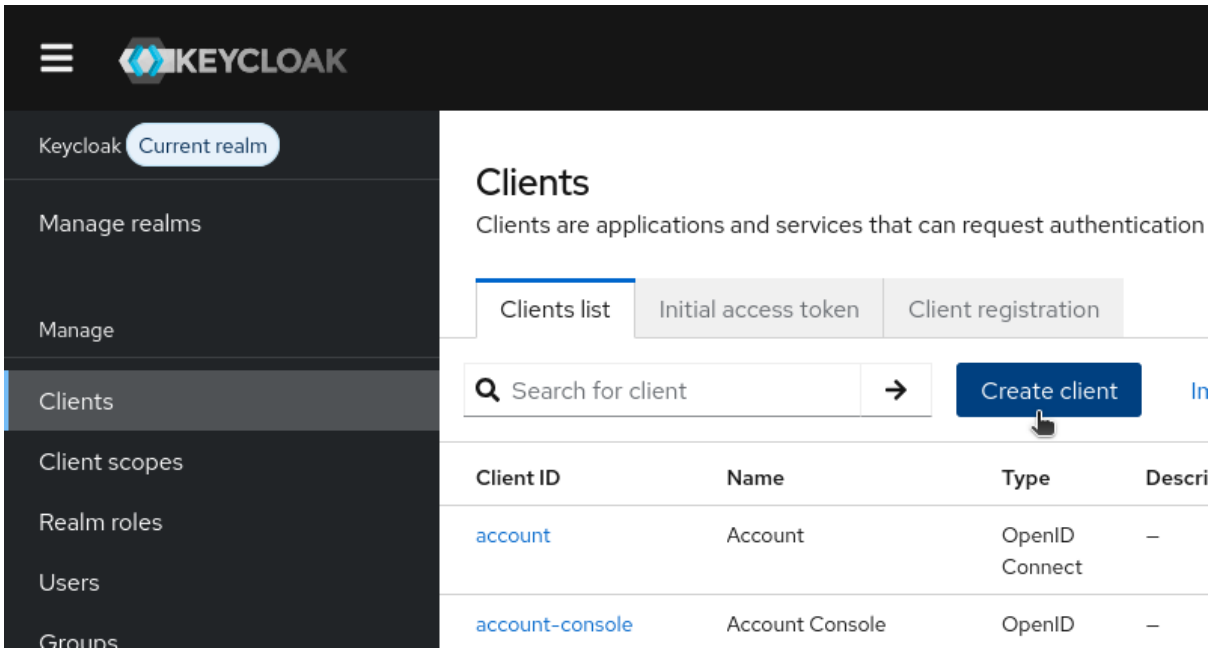
### Redirect URI

If you plan to use the device registration feature (see *configure device registration* (page 16)), you need to configure a redirect URI for the application. Go to *Manage* → *Authentication (Preview)*, click on *Add Redirect URI*, then choose *Mobile and desktop applications* and select the following URI:

```
https://login.microsoftonline.com/common/oauth2/nativeclient
```

### Keycloak

Register a new client in Keycloak. Go to *Manage* → *Clients* and create the client.




Configure the client as follows:

1. General settings
   - Set the client type to *OpenID Connect*.
   - Pick a valid client ID, for example, `ubuntu-authd`. This corresponds to the `<CLIENT_ID>` which is used in the next section.
2. Capability config

[7] https://learn.microsoft.com/en-us/entra/identity-platform/quickstart-register-app

- (Optional) Enable client authentication
- Enable the *OAuth 2.0 Device Authorization Grant* authentication flow.

3. Login settings
   - No need to change anything here.

Finally, click on *Save* to create the client.

If you enabled client authentication, find the client secret in the *Credentials* tab. This corresponds to the `<CLIENT_SECRET>` in the next section.

**Set email addresses for users**

Make sure that all users who should be able to log in through this broker have an email address configured in Keycloak.

### Broker configuration

Now we can configure the broker. Note that different brokers can require different configuration data.

### Google IAM

To configure Google IAM, edit `/var/snap/authd-google/current/broker.conf`:

```
[oidc]
issuer = https://accounts.google.com
client_id = <CLIENT_ID>
client_secret = <CLIENT_SECRET>
```

### Microsoft Entra ID

To configure Entra ID, edit `/var/snap/authd-msentraid/current/broker.conf`:

```
[oidc]
issuer = https://login.microsoftonline.com/<ISSUER_ID>/v2.0
client_id = <CLIENT_ID>
```

### Keycloak

To configure the authd-oidc broker for Keycloak, edit `/var/snap/authd-oidc/current/broker.conf`:

```
[oidc]
issuer = https://<host>/realms/<realm-name>
client_id = <CLIENT_ID>
```

If you enabled client authentication, you also need to add the client secret:

```
client_secret = <CLIENT_SECRET>
```

### Force remote authentication with the identity provider

By default, remote authentication with the identity provider only happens if there is a working internet connection and the provider is reachable during login.

If you want to force remote authentication, even when the provider is unreachable, enable it as follows:

```
[oidc]
...
force_provider_authentication = true
```

**Warning:**

In some cases, this may prevent login, such as when there are network issues.

### Configure extra scopes

Some identity providers require additional OIDC scopes beyond the default ones to function correctly. For example, Okta requires the `offline_access` scope to return a refresh token in the authentication response.

You can specify extra scopes in the `oidc` section of the broker configuration file:

```
[oidc]
...
## Comma-separated list of extra OIDC scopes to request
extra_scopes = offline_access
```

### Configure allowed users

The users who are allowed to log in (after successfully authenticating via the identity provider) are configured in the `users` section of the `/var/snap/authd-<broker_name>/current/broker.conf` file:

```
[users]
## 'allowed_users' specifies the users who are permitted to log in after
## successfully authenticating with the Identity Provider.
## Values are separated by commas. Supported values:
## - 'OWNER': Grants access to the user specified in the 'owner' option
##            (see below). This is the default.
## - 'ALL': Grants access to all users who successfully authenticate
##          with the Identity Provider.
## - <username>: Grants access to specific additional users
##               (e.g. user1@example.com).
## Example: allowed_users = OWNER,user1@example.com,admin@example.com
#allowed_users = OWNER
```

(continues on next page)

(continued from previous page)

```
## 'owner' specifies the user assigned the owner role. This user is
## permitted to log in if 'OWNER' is included in the 'allowed_users'
## option.
##
## If this option is left unset, the first user to successfully log in
## via this broker will automatically be assigned the owner role. A
## drop-in configuration file will be created in broker.conf.d/ to set
## the 'owner' option.
##
## To disable automatic assignment, you can either:
## 1. Explicitly set this option to an empty value (e.g. owner = "")
## 2. Remove 'OWNER' from the 'allowed_users' option
##
## Example: owner = user2@example.com
#owner =
```

By default, the first person to log in to the machine is automatically registered as the owner. If you wish to override this behavior then specify a list of allowed users with the `allowed_users` option, while omitting the `OWNER` keyword:

```
allowed_users = person1@email.com,person2@email.com
```

Alternatively, you can directly register someone as the owner by using the `owner` option:

```
owner = your@email.com
```

Explicitly setting an empty owner, has the same effect as omitting the `OWNER` keyword in `allowed_users`:

```
owner = ""
```

**Only my first logged-in user has access to the machine?**

By default, the first logged-in user is defined as the “owner” and only the owner can log in. To allow more users to log in, update the list of allowed users.
If an administrator is the first to log in to a machine and becomes the owner, they can ensure that the next user to log in becomes the owner by removing the `20-owner-autoregistration.conf` file:

#### Google IAM

```
sudo rm /var/snap/authd-google/current/broker.conf.d/20-owner-autoregistration.conf
```

#### Microsoft Entra ID

```
sudo rm /var/snap/authd-msentraid/current/broker.conf.d/20-owner-autoregistration.conf
```

#### Keycloak

```
sudo rm /var/snap/authd-oidc/current/broker.conf.d/20-owner-autoregistration.conf
```

This file is generated when a user logs in and becomes the owner. If it is removed, it will be regenerated on the next successful login.

### Configure the home directory location

By default, home directories for new authd users are created under `/home`, in the format `/home/<username>`.

You can change the base directory for new users’ home directories with the `home_base_dir` option in the `users` section of the broker configuration file:

```
[users]
## The directory where home directories are created when users log in for the first time.
## Paths are created in the format <home_base_dir>/<username>
home_base_dir = /home
```

**Note:**

Changing the base directory only affects users logging in for the first time.

### Configure user groups

Some brokers support adding users to groups that are configured in the identity provider. Group membership can be used to manage user privileges, including **sudo** and **docker** rights.

See the *group management reference* (page 47) for more details.

In addition, you can configure extra groups for authd users. On login, the users are added to these groups automatically. Specify any extra groups in the `users` section of the broker configuration file:

```
[users]
## A comma-separated list of local groups which authd users will be
## added to upon login.
## Example: extra_groups = users
#extra_groups =
```

There is also an `owner_extra_groups` option for specifying additional local groups, to which only the user with the *owner role* (page 13) is added:

```
## Like 'extra_groups', but only the user assigned the owner role
## will be added to these groups.
## Example: owner_extra_groups = sudo,lpadmin
#owner_extra_groups =
```

### Configure device registration

#### Google IAM

The Google IAM broker does not support device registration.

#### Microsoft Entra ID

When using the Microsoft Entra ID broker, you can enable automatic device registration, which allows administrators to manage registered devices in the Microsoft Entra admin center.

Automatic device registration can be enabled with the `register_device` option in the `msentraid` section of the broker configuration file:

```
[msentraid]
## Enable automatic device registration with Microsoft Entra ID
## when a user logs in through this broker.
##
## If set to true, authd will attempt to register the local machine
## as a device in Entra ID upon successful login.
##
## If set to false (the default), device registration will be skipped.
#register_device = false
```

**Changing this option forces re-authentication**

When changing this option, users are forced to re-authenticate via device authentication on the next login.

**Set the redirect URI**

Make sure that the application in the Microsoft Entra admin center has a redirect URI configured as described in *Redirect URI* (page **??**).

#### Keycloak

The authd-oidc broker does not support device registration.

### Restart the broker

When a configuration file is added you have to restart authd:

```
sudo systemctl restart authd
```

When the configuration of a broker is updated, you also have to restart the broker:

#### Google IAM

```
sudo snap restart authd-google
```

#### Microsoft Entra ID

```
sudo snap restart authd-msentraid
```

#### Keycloak

```
sudo snap restart authd-oidc
```

### Configure login timeout

By default on Ubuntu, the login timeout is 60s.

This may be too brief for a device code flow authentication.

It can be modified by changing the value of `LOGIN_TIMEOUT` in `/etc/login.defs`.

### Configure the authd service

The authd service is configured in `/etc/authd/authd.yaml`.

This provides configuration options for logging verbosity and UID/GID ranges.

### Configure password quality

You can change authd's local password policy to ensure that users always set strong passwords.

If your mobile device management (MDM) solution includes a compliance check for the passwords of authd users, you may also need to configure authd's password policy so that it matches that of the MDM.

authd depends on the libpwquality library, which supports configuring password quality.

To configure the local password policy for authd, create a drop file in `/etc/security/pwquality.conf.d/`. This will override the default values in `/etc/security/pwquality.conf`.

First, create a directory for the custom configuration file:

```
sudo mkdir -p /etc/security/pwquality.conf.d/
```

Then edit the file to add your settings.

```
sudoedit /etc/security/pwquality.conf.d/99_custom-options.conf
```

For example, if your policy requires that passwords have a 14 character minimum, edit the drop file to set the value for `minlen`:

Listing 1: /etc/security/pwquality.conf.d/99_custom-options.conf

```
# This file overrides the defaults set in /etc/security/pwquality.conf
# Refer to the pwquality file for additional configuration options.
# Minimum acceptable size for the new password (plus one if
# credits are not disabled which is the default). (See pam_cracklib manual.)
# Cannot be set to lower value than 6 (default is 8).
minlen = 14
```

The man pages[8] provide a full list of configuration options for the libpwquality library.

[8] https://manpages.ubuntu.com/manpages/noble/man5/pwquality.conf.5.html

### 3.1.2. Login and authentication

Users can log in and authenticate on Ubuntu Desktop or Ubuntu Server, with authd supporting both GDM and SSH:

#### Log in with GDM

#### Logging in with a remote provider

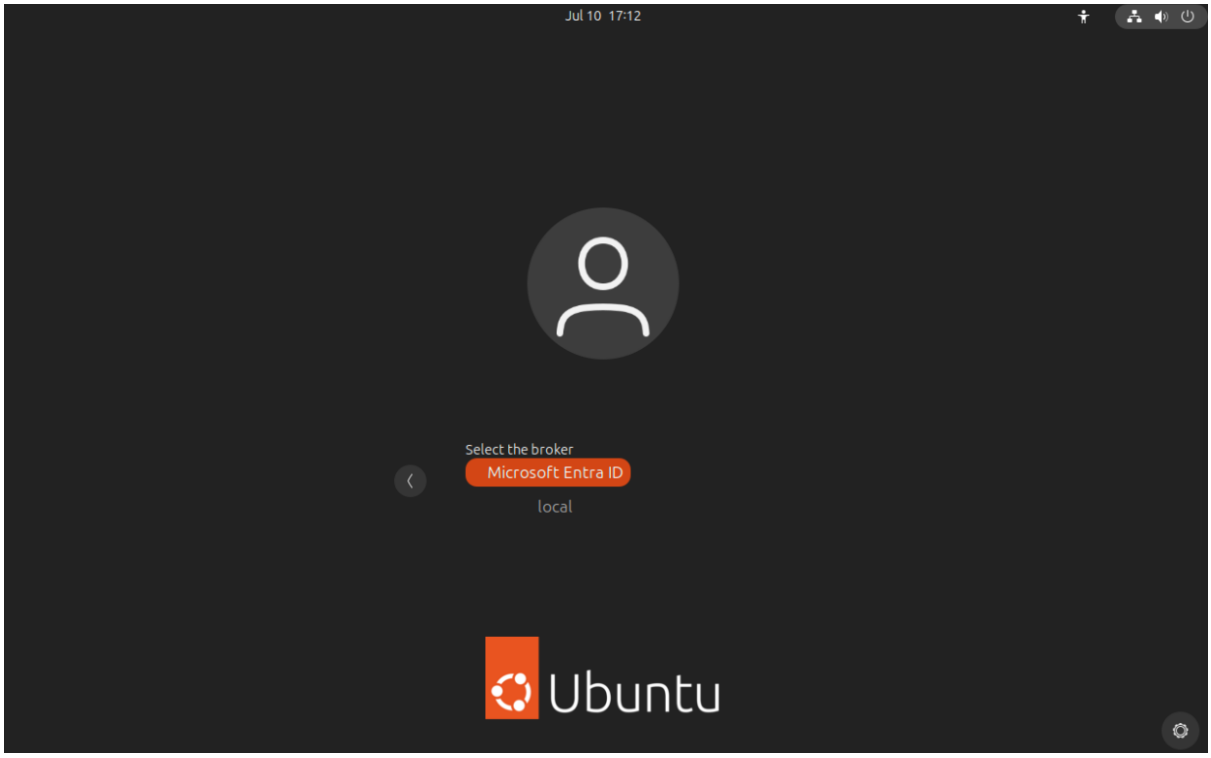
Once the system is configured you can log into your system using your remote provider credentials and the device code flow. In this example, we are going to use Microsoft Entra ID as the remote provider but the process is equivalent for other providers.

See all the available providers: *Install brokers* (page 6)

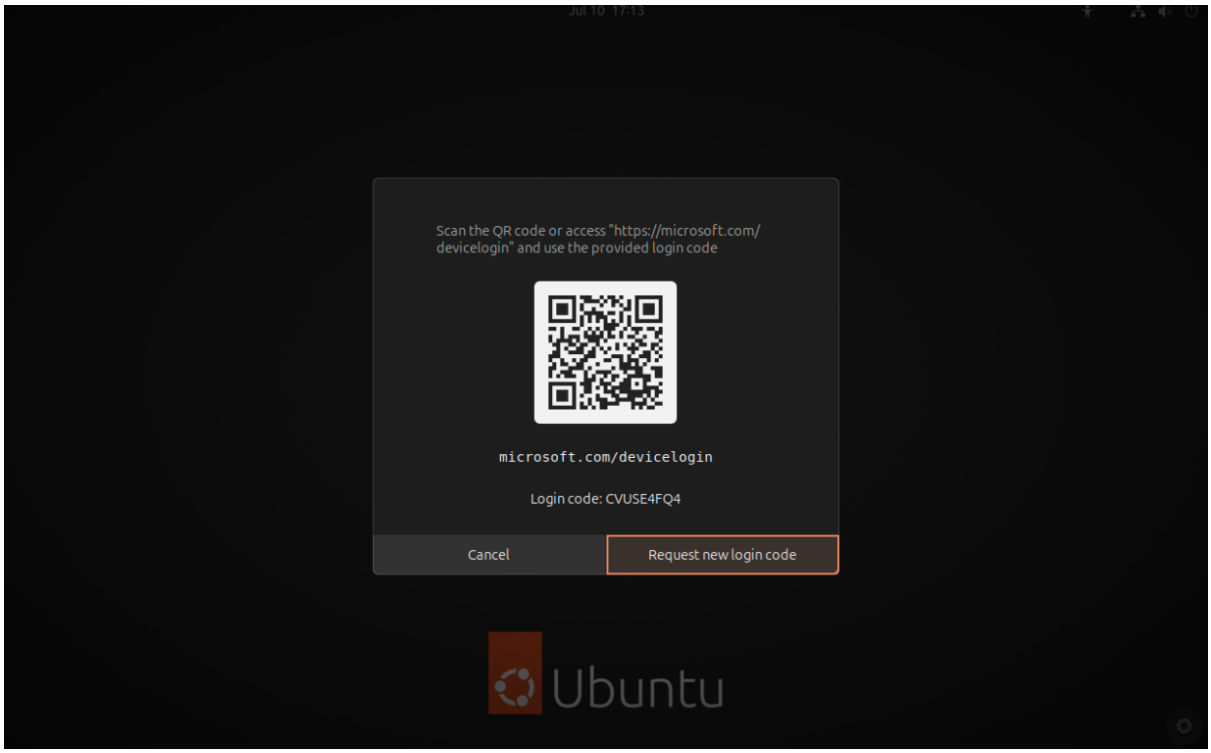
In the login screen (greeter), select `not listed` below the user name field.

Type your remote provider user name. The format is `user@domain.name`

Select the broker `Microsoft Entra ID`

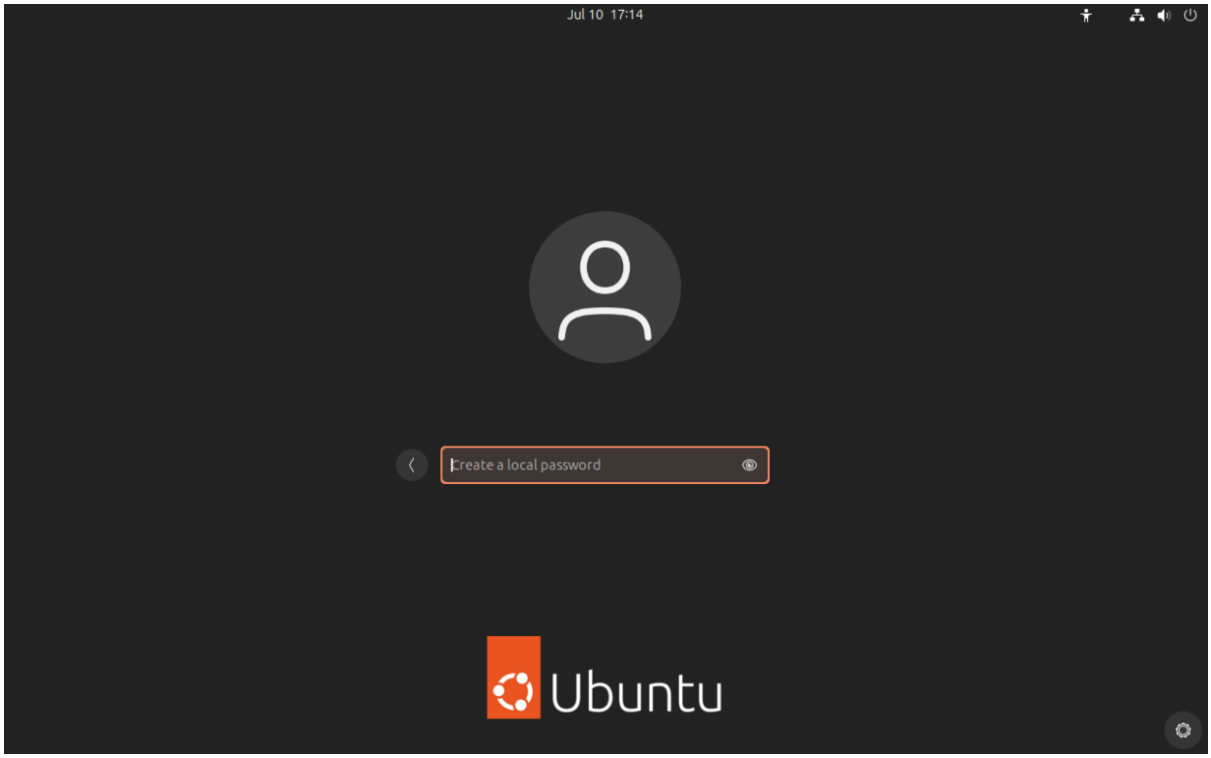



If MFA is enabled, a QR code and a login code are displayed.




From a second device, flash the QR code or type the URL in a web browser, then follow your provider's authentication process.

Upon successful authentication, the user is prompted to enter a local password. This password can be used for offline authentication.




## Commands

### authd

`authd` is socket-activated. It means that the service starts on-demand when it receives a request on a socket.

If you want to restart the service, you can stop it with `sudo systemctl stop authd` and it will restart automatically on the next message it receives.

Run `/usr/libexec/authd --help` to display the entire help.

### Broker management

The broker is managed through the `snap` command.

The main operation is to restart the broker to reload the configuration when it has changed. You can reload the broker with the command:

```
sudo snap restart authd-msentraid
```

> If you are using a different broker to `msentraid`, make sure to change the snap name when running this command.

## Log in with SSH

### Server configuration

To enable SSH access with `authd` you must configure `sshd` and the broker.

### SSH configuration

To configure SSH, create a file `/etc/ssh/sshd_config.d/authd.conf` with the following content:

```
UsePAM yes
KbdInteractiveAuthentication yes
```

Alternatively, you can directly set the keys in the sshd configuration file `/etc/ssh/sshd_config`.

Then restart the SSH server:

```
sudo systemctl restart ssh
```

**Managing SSH securely for authd deployments**

More detail on securely managing SSH authentication is provided in the *SSH section of the security overview* (page 62).

### Broker configuration

To configure the broker edit the file `/var/snap/authd-<broker_name>/current/broker.conf` and set the key `ssh_allowed_suffixes_first_auth` with the list of domains that you want to allow.

```
...

[users]
# The username suffixes that are allowed to log in via ssh without existing previously in
the system.
# The suffixes must be separated by commas.
ssh_allowed_suffixes_first_auth = <ALLOWED DOMAINS>
```

You can set several domains separated by a comma. For instance:

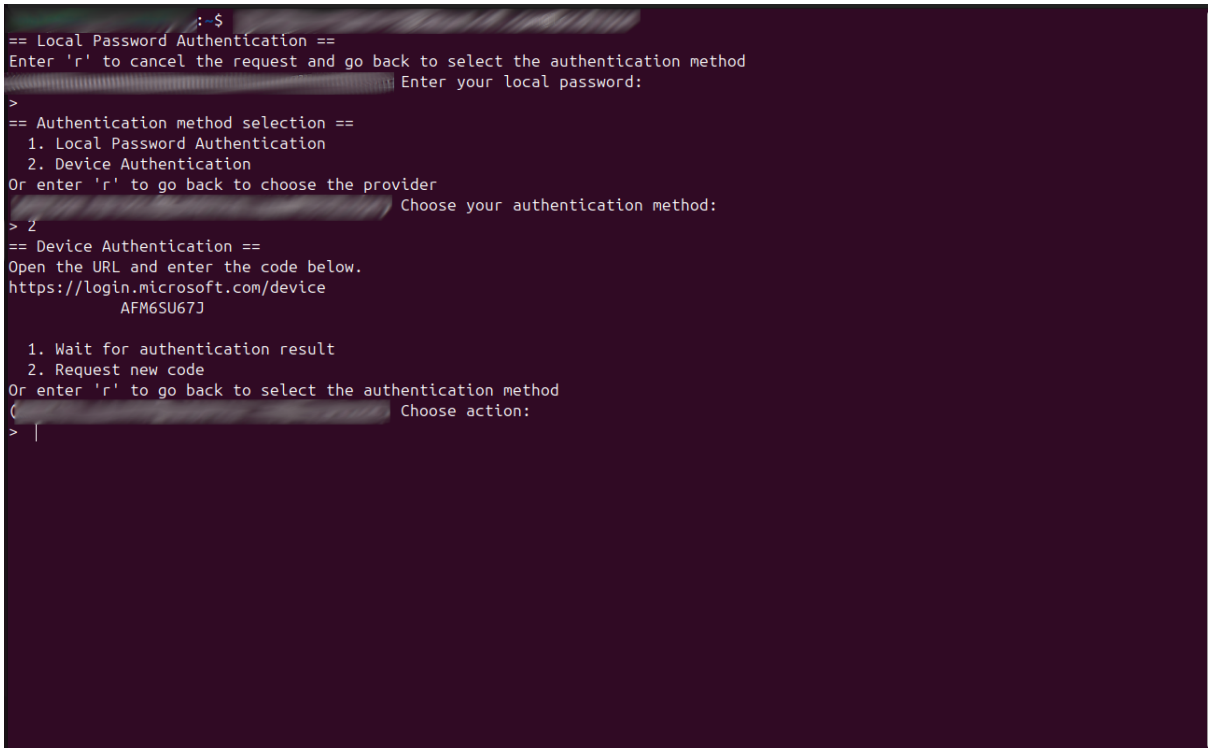
```
ssh_allowed_suffixes_first_auth = @example.com,@ubuntu.com
```

### Usage

Once this is all set up, you can ssh to the server in the same way that you would do with any server: `ssh <username>@<host>`. The format of `<username>` is the user handle on the provider, such as `user@domain.tld`.

For instance, here is an example using Microsoft Entra ID as a provider:

```
ssh user@domain.tld@remote.host
```




## 3.1.3. Network file systems

If using a network file system to access shared directories from authd-enabled machines, you can use ID mapping:

### Using authd with NFS

The user identifiers (UIDs) and group identifiers (GIDs) assigned by authd are unique to each machine. This means that when using authd with NFS, the UIDs and GIDs of users and groups on the NFS server will not match those on the client machines, which leads to permission issues.

To avoid these issues, you can use NFS with ID mapping and Kerberos. This ensures that the UIDs and GIDs are mapped correctly across all machines.

## Setting up NFS with IDMAP and Kerberos

This guide will walk you through setting up an NFS server with ID mapping and Kerberos authentication. After following the steps outlined below, the user `alice` will be able to access a shared directory on the server from a client machine.

---

## Steps for the server

### Step 1: Install required packages

1. **Install packages:** On the NFS server, run:

   ```
   sudo apt install -y nfs-kernel-server nfs-common rpcbind krb5-user krb5-admin-server krb5-kdc
   ```

2. **Handle Kerberos configuration prompts:** During the installation of `krb5-user`, you will be prompted to provide configuration details for Kerberos. Here's what to enter:

   - **Default Kerberos version 5 realm:** Enter the Kerberos realm name, which is the uppercase version of your domain. For example:

     ```
     EXAMPLE.COM
     ```

   - **Kerberos servers for your realm:** Enter the hostname of the Key Distribution Center (KDC). Assuming the KDC is on the same host as the NFS server:

     ```
     server.example.com
     ```

   - **Administrative server for your Kerberos realm:** Enter the hostname of the Kerberos admin server, which is also the same as the NFS server in this case:

     ```
     server.example.com
     ```

---

### Step 2: Configure Kerberos

1. **Create the Realm:**

   ```
   sudo krb5_newrealm
   ```

   Follow the prompts to set up the Kerberos realm.

2. **Add principals:** In Kerberos, a principal is a unique identity that is used for authentication.

   - Add a principal for the NFS server: This principal is used by the NFS client to authenticate when mounting an NFS directory.

     ```
     sudo kadmin.local addprinc -randkey nfs/server.example.com
     ```

- Add a principal for the user `alice`: This principal is used for authentication when the user accesses the mounted NFS directory.

```
sudo kadmin.local addprinc alice
```

When prompted, set a password for the user `alice`.

3. **Generate Keytabs:**

A *keytab* is a file that contains Kerberos principals and their associated secret keys. It allows services (such as NFS) to authenticate without needing to input a password each time.

- Export the keytab for the NFS server and the user `alice`:

```
sudo kadmin.local ktadd -k /etc/krb5.keytab nfs/server.example.com
```

---

### Step 3: Configure the NFS server

1. **Create and configure the shared directory:**

You'll need to create the directory to share via NFS and configure the shared directory in the `/etc/exports` file.

- **Create a directory owned by `alice`:**

```
sudo mkdir -p /srv/nfs/shared/alice
sudo chown alice:alice /srv/nfs/shared/alice
```

- **Configure exports:** Edit the `/etc/exports` file to define the shared directory:

```
sudo editor /etc/exports
```

Add this line:

```
/srv/nfs/shared *(rw,sync,no_subtree_check,sec=krb5)
```

2. **Configure IDMAP:** Edit the IDMAP configuration:

```
sudo editor /etc/idmapd.conf
```

Ensure the following is set:

```
[General]
Domain = example.com
```

3. **Restart services:**

```
sudo systemctl restart nfs-kernel-server rpcbind rpc-svcgssd
```

4. **Verify running services:** Check the status of the relevant services:

```
sudo systemctl status nfs-kernel-server rpcbind rpc-svcgssd
```

---

### Steps for the client

#### Step 1: Install required packages

1. **Install packages:** On the NFS client, run:

```
sudo apt install -y nfs-common krb5-user rpcbind
```

2. **Handle Kerberos configuration prompts:** During the installation of `krb5-user`, you will be prompted to provide configuration details for Kerberos again. Enter the same details as before:
   - **Default Kerberos version 5 realm:**

     ```
     EXAMPLE.COM
     ```

   - **Kerberos servers for your realm:**

     ```
     server.example.com
     ```

   - **Administrative server for your Kerberos realm:**

     ```
     server.example.com
     ```

---

#### Step 2: Copy the Kerberos keytab file

1. **Copy keytab file:** Securely copy the keytab from the server to the client and set the correct permissions:

```
scp root@server.example.com:/etc/krb5.keytab /tmp/krb5.keytab && \
sudo mv /tmp/krb5.keytab /etc/krb5.keytab && \
sudo chown root:root /etc/krb5.keytab && \
sudo chmod 600 /etc/krb5.keytab
```

---

#### Step 3: Configure NFS client

1. **Configure IDMAP:** Edit the IDMAP configuration:

```
sudo editor /etc/idmapd.conf
```

   Ensure the following is set:

```
[General]
Domain = example.com
```

2. **Restart services:**

```
sudo systemctl restart nfs-client.target rpc-gssd.service rpcbind.service
```

3. **Verify running services:** Check the status of the relevant services:

```
sudo systemctl status nfs-client.target rpc-gssd.service auth-rpcgss-module.service
rpcbind.service
```

---

### Step 4: Mount the NFS share

Mount the shared directory with Kerberos security:

```
sudo -u alice mkdir /home/alice/nfs
sudo mount -t nfs4 -o sec=krb5 server.example.com:/srv/nfs/shared/alice /home/alice/nfs
```

---

### Step 5: Obtain Kerberos ticket

Log in as the user `alice` and authenticate:

```
kinit alice
```

Verify the ticket:

```
klist
```

---

### Step 6: Test and debug

1. **Test access to the share:** As the user `alice`, try accessing the share:

   ```
   ls -la /home/alice/nfs
   ```

   Create a test file to verify write access:

   ```
   touch /home/alice/nfs/test
   ```

2. **Check logs if issues arise:**
   - On the server:

     ```
     sudo journalctl -u nfs-kernel-server -u rpcbind -u rpc-svcgssd
     ```

   - On the client:

     ```
     sudo journalctl -u rpcbind -u rpc-gssd
     ```

---

### Cleanup

If you no longer need the NFS share or want to clean up the configuration, follow these steps:

### On the server

1. **Purge installed packages:**

```
sudo apt purge "krb*" "nfs-*"
```

2. **Remove Kerberos configuration and data:**

```
sudo sh -c "rm -rf /etc/krb5* /var/lib/krb5kdc/* /tmp/krb5*"
```

3. **Remove the shared directory:**

```
sudo rm -rf /srv/nfs/shared
sudo rmdir /srv/nfs
```

### On the client

1. **Unmount the shared directory and delete the mountpoint:**

```
sudo umount /home/alice/nfs
sudo rmdir /home/alice/nfs
```

2. **Purge installed packages:**

```
sudo apt purge nfs-common krb5-* rpcbind
```

3. **Remove Kerberos data:**

```
sudo rm -f /etc/krb5.keytab /tmp/krb5*
```

---

### Additional resources

For a complete guide to setting up NFS on your client and server, see Network File System (NFS)[9] in the Ubuntu Server documentation.

[9] https://documentation.ubuntu.com/server/how-to/networking/install-nfs/

### Using authd with Samba

The user identifiers (UIDs) and group identifiers (GIDs) assigned by authd are unique to each machine. This means that when using authd with Samba, the UIDs and GIDs of users and groups on the Samba server will not match those on the client machines, which leads to permission issues.

To avoid these issues, you can use Samba with ID mapping. This ensures that the UIDs and GIDs are mapped correctly across all machines.

### Setting up Samba with ID mapping

This guide will walk you through setting up a Samba server with ID mapping. By following the steps outlined below, a user `alice` will be able to access a shared directory on the server from a client machine.

---

### Steps for the server

1. **Install Samba:** Install the Samba server package:

   ```
   sudo apt update
   sudo apt install samba
   ```

2. **Create the shared directory:** Create the directory to be shared and set ownership to the `alice` user:

   ```
   sudo mkdir -p /srv/samba/alice
   sudo chown alice:alice /srv/samba/alice
   ```

3. **Edit Samba configuration:** Open the Samba configuration file:

   ```
   sudo editor /etc/samba/smb.conf
   ```

   Add the following section at the end of the file:

   ```
   [alice]
   path = /srv/samba/alice
   browsable = yes
   writable = yes
   valid users = alice
   ```

   **Explanation**

   This section defines a Samba share named `alice` located at `/srv/samba/alice`. It is visible to users on the network (`browsable`), allows writing (`writable`), and restricts access to the `alice` user (`valid users`).

4. **Create a Samba user for `alice`:** Add the `alice` user to the Samba database and set a password:

```
sudo smbpasswd -a alice
```

Follow the prompts to set the Samba password for the user.

5. **Restart Samba service:** Restart the Samba service to apply the changes:

```
sudo systemctl restart smbd
```

---

### Steps for the client

1. **Install Samba client:** Install the required packages for connecting to Samba shares:

```
sudo apt update
sudo apt install smbclient cifs-utils
```

2. **Test access to the share:** Test connectivity using `smbclient`, making sure to replace `$SERVER` with the Samba server's hostname or IP address:

```
smbclient //$SERVER/alice -U alice
```

Enter the Samba password for `alice` when prompted. If successful, a `smb: \>` prompt appears.

3. **Mount the share:** Create a mount point for the share:

```
mkdir -p /home/alice/samba
```

Mount the share using the `cifs` filesystem type:

```
sudo mount -t cifs //$SERVER/alice /home/alice/samba -o user=alice,uid=$(id -u
alice),gid=$(id -g alice)
```

Enter the Samba password for `alice` when prompted.

4. **Optional: Add the share to `/etc/fstab` for persistent mounting:** To automatically mount the share at boot, use a credentials file:

   - Create a credentials file:

```
sudo editor /etc/samba/credentials
```

Add the following lines:

```
username=alice
password=YOUR_PASSWORD
```

   - Secure the credentials file:

```
sudo chmod 600 /etc/samba/credentials
```

   - Update `/etc/fstab`:

```
//$SERVER/alice /home/alice/samba cifs credentials=/etc/samba/credentials,
uid=alice,gid=alice 0 0
```

5. **Verify the mount:** As the user `alice`, try accessing the shared directory:

```
ls -la /home/alice/samba
```

Verify write access by creating a test file:

```
touch /home/alice/samba/test
```

6. **Test enforced access control (optional):**

> **Security note**
>
> **Security Note:** Files and directories in the share may appear as owned by `alice` on the client, but access control is enforced by the server.
> For example, if `alice` does not have permission on the server, access will be denied even if ownership appears correct on the client.

To test this, you can create a restricted directory on the server and attempt to access it on the client:

- Create a restricted directory on the server:

```
sudo mkdir /srv/samba/alice/secrets
sudo chmod 700 /srv/samba/alice/secrets
```

- Attempt to access it on the client:

```
ls /home/alice/samba/secrets
```

The terminal output will indicate that the user does not have access to the restricted directory:

```
ls: cannot open directory '/home/alice/samba/secrets': Permission denied
```

---

## Cleanup

### On the server

1. **Delete the shared directory:** Remove the directory used for the Samba share:

```
sudo rm -rf /srv/samba/alice
```

2. **Purge installed Samba packages:** If Samba is no longer needed, uninstall it completely:

```
sudo apt purge samba samba-common
sudo apt autoremove
```

---

### On the client

1. **Unmount the shared directory:**

   ```
   sudo umount /home/alice/samba
   ```

2. **Delete the mount point:**

   ```
   rmdir /home/alice/samba
   ```

3. **Remove fstab entry:** If you added the share to `/etc/fstab`, remove its entry:

   ```
   sudo editor /etc/fstab
   ```

   Locate and delete the line referencing the Samba share, then save and close.

4. **Delete credentials file:** If a credentials file was used, remove it:

   ```
   sudo rm /etc/samba/credentials
   ```

5. **Purge installed Samba client packages:** If Samba client tools are no longer needed, uninstall them:

   ```
   sudo apt purge samba-common smbclient cifs-utils
   sudo apt autoremove
   ```

---

### Additional resources

For a complete guide to setting up Samba on your client and server, see Samba[10] in the Ubuntu Server documentation.

## 3.1.4. Debugging and troubleshooting

When troubleshooting authd, you may need to work with logs or enter recovery mode:

### Accessing and configuring logs

Logs are generated for authd and its brokers, which can help when troubleshooting and reporting bugs.

[10] https://documentation.ubuntu.com/server/how-to/samba/

### authd

`authd` logs to the system journal.

For `authd` entries, run:

```
sudo journalctl -u authd.service
```

If you want logs for authd and all brokers on the system, run:

```
sudo journalctl -u authd.service -u "snap.authd-*.service"
```

For specific broker entries run the command for your chosen broker:

#### Google IAM

```
sudo journalctl -u snap.authd-google.authd-google.service
```

#### Microsoft Entra ID

```
sudo journalctl -u snap.authd-msentraid.authd-msentraid.service
```

For the GDM integration:

```
sudo journalctl /usr/bin/gnome-shell
```

For anything else or more broader investigation, use `sudo journalctl`.

### Configure logging verbosity

You can increase the verbosity of the logs in different ways.

#### PAM module

To increase the verbosity of the PAM messages, append `debug=true` to all lines that include `pam_authd_exec.so` or `pam_authd.so` in the PAM configuration files in `/etc/pam.d/` :

```
sudo sed -i '/pam_authd_exec\.so\|pam_authd\.so/ s/$/ debug=true/' /etc/pam.d/*
```

#### NSS module

To get more info on NSS requests to authd, export the `AUTHD_NSS_INFO=stderr` environment variable on any program using the authd NSS module.

### authd service

To increase the verbosity of the service itself, edit the service file:

```
sudo systemctl edit authd.service
```

Add the following lines to the override file and make sure to add `-vv` at the end of the `authd` command:

```
[Service]
ExecStart=
ExecStart=/usr/libexec/authd -vv
```

Then you need to restart the service with `sudo systemctl restart authd`.

### GDM

Ensure the lines in `/etc/gdm3/custom.conf` are not commented:

```
[debug]
# Uncomment the line below to turn on debugging
# More verbose logs
# Additionally lets the X server dump core if it crashes
Enable=true
```

Then you need to restart the service with `sudo systemctl restart gdm`.

### authd broker service

To increase the verbosity of the broker service, edit the service file:

### Google IAM

```
sudo systemctl edit snap.authd-google.authd-google.service
```

### Microsoft Entra ID

```
sudo systemctl edit snap.authd-msentraid.authd-msentraid.service
```

Add the following lines to the override file and make sure to add `-vv` to the exec command:

### Google IAM

```
[Service]
ExecStart=
ExecStart=/usr/bin/snap run authd-google -vv
```

### Microsoft Entra ID

```
[Service]
ExecStart=
ExecStart=/usr/bin/snap run authd-msentraid -vv
```

You will then need to restart the service with:

### Google IAM

`sudo snap restart authd-google`.

### Microsoft Entra ID

`sudo snap restart authd-msentraid`.

## How to enter recovery mode after failed login

If authd and/or the broker are missing, corrupted, or broken in any way, a user may be prevented from logging in.

To get access to the system for modifying configurations and installations in such cases, there are two main options:

1. Log in as root user or another local user with administrator privileges
2. Boot into recovery mode to get root access

The steps required for entering recovery mode are included below.

### Boot into recovery mode

If it is not possible to log in with the root user account or another local user account with administrator privileges, the user can boot into recovery mode:

1. Reboot the device
2. During the reboot, press and hold the right SHIFT key
3. When the Grub menu appears, select `advanced options for Ubuntu`
4. Choose `recovery mode` for the correct kernel version
5. Select `drop to root shell prompt`

The user then has access to the root filesystem and can proceed with debugging.

## 3.1.5. Updating and upgrading

Use the stable version of authd for production use, or switch to the edge version to try new features:

### Changing authd versions

To test new features or check if a bug has been fixed in a new version, you can switch to edge releases of authd and its brokers.

**Warning:**

Do not use the edge PPA in a production system, because it may apply changes to the authd database in a non-reversible way, which can make it difficult to roll back to the stable version of authd.

**Note:**

The default installation source depends on your Ubuntu release:

- **Ubuntu 26.04 LTS and later**: `authd` is available directly from the Ubuntu archive.
- **Ubuntu 24.04 LTS**: `authd` must be installed from the stable PPA[11].

Using the edge PPA lets you test pre-release deb packages on either release.

[11] https://launchpad.net/~ubuntu-enterprise-desktop/+archive/ubuntu/authd

### Switch authd to the edge PPA

The edge PPA[12] contains the latest fixes and features for authd.

```
sudo add-apt-repository ppa:ubuntu-enterprise-desktop/authd-edge
sudo apt install authd
```

Keep in mind that this version is not tested and may be incompatible with the current released version of the brokers.

To switch back to the stable version of authd:

```
sudo apt install ppa-purge
sudo ppa-purge ppa:ubuntu-enterprise-desktop/authd-edge
```

[12] https://launchpad.net/~ubuntu-enterprise-desktop/+archive/ubuntu/authd-edge

#### Switch broker snap to the edge channel

You can also switch to the edge channel of the broker snap:

#### Google IAM

```
sudo snap switch authd-google --edge
sudo snap refresh authd-google
```

#### Microsoft Entra ID

```
sudo snap switch authd-msentraid --edge
sudo snap refresh authd-msentraid
```

Keep in mind that this version is not tested and may be incompatible with the current released version of authd.

To switch back to stable after trying the edge channel:

#### Google IAM

```
sudo snap switch authd-google --stable
sudo snap refresh authd-google
```

#### Microsoft Entra ID

```
sudo snap switch authd-msentraid --stable
sudo snap refresh authd-msentraid
```

**Note:**

If using an edge release, you can read the edge version of the documentation[13]

[13] https://documentation.ubuntu.com/authd/edge-docs/

## 3.1.6. Contributing to authd

Contribute to the development of authd and its brokers, in addition to the authd documentation:

## Contributing to authd

A big welcome and thank you for considering making a contribution to authd and Ubuntu! It's people like you that help make these products a reality for users in our community.

By agreeing to follow these guidelines the contribution process should be easy and effective for everyone involved. This also communicates that you agree to respect the time of the developers working on this project. In return, we will reciprocate that respect by addressing your issues, assessing proposed changes and helping you finalize your pull requests.

These are mostly guidelines, not rules. Use your best judgment and feel free to propose changes to this document in a pull request.

## Code of conduct

We take our community seriously, holding ourselves and other contributors to high standards of communication. By contributing to this project you agree to uphold the Ubuntu community Code of Conduct[14].

## Getting Started

Contributions are made to this project via Issues and Pull Requests (PRs). These are some general guidelines that cover both:

- To report security vulnerabilities, use the advisories page of the repository and not a public bug report. Please use launchpad private bugs[15], which is monitored by our security team. On an Ubuntu machine, it's best to use `ubuntu-bug authd` to collect relevant information.
- General issues or feature requests should be reported to the GitHub Project[16]
- If you've never contributed before, see this post on ubuntu.com[17] for resources and tips on how to get started.
- Existing Issues and PRs should be searched for on the project's repository[18] before creating your own.
- While we work hard to ensure that issues are handled in a timely manner, it can take time to investigate the root cause. A friendly ping in the comment thread to the submitter or a contributor can help draw attention if your issue is blocking.

[14] https://ubuntu.com/community/ethos/code-of-conduct
[15] https://bugs.launchpad.net/ubuntu/+source/authd/+filebug
[16] https://github.com/canonical/authd/issues
[17] https://ubuntu.com/community/contribute
[18] https://github.com/canonical/authd

### Issues

Issues can be used to report problems with the software, request a new feature or discuss potential changes before a PR is created. When you create a new Issue[19], a template will be loaded that will guide you through collecting and providing the information that we need to investigate.

If you find an Issue that addresses the problem you're having, please add your own reproduction information to the existing issue rather than creating a new one. Adding a reaction[20] can also help by indicating to our maintainers that a particular problem is affecting more than just the reporter.

### Pull Requests

PRs to our project are always welcome and can be a quick way to get your fix or improvement slated for the next release. In general, PRs should:

- Only fix/add the functionality in question **OR** address wide-spread whitespace/style issues, not both.
- Add unit or integration tests for fixed or changed functionality.
- Address a single concern in the least possible number of changed lines.
- Include documentation in the repo or on our docs site[21].
- Be accompanied by a complete Pull Request template (loaded automatically when a PR is created).

For changes that address core functionality or that would require breaking changes (e.g. a major release), it's best to open an Issue to discuss your proposal first. This is not required but can save time when creating and reviewing changes.

In general, we follow the "fork-and-pull" Git workflow[22]:

1. Fork the repository to your own Github account.
2. Clone the fork to your machine.
3. Create a branch locally with a succinct but descriptive name.
4. Commit changes to that branch.
5. Follow any formatting and testing guidelines specific to this repo.
6. Push changes to your fork.
7. Open a PR in our repository and follow the PR template so that we can efficiently review the changes.

PRs will trigger unit and integration tests with and without race detection, linting and formatting validations, static and security checks, and freshness of generated files verification. All these tests must pass before any merge into the main branch.

[19] https://github.com/canonical/authd/issues
[20] https://github.blog/2016-03-10-add-reactions-to-pull-requests-issues-and-comments/
[21] https://documentation.ubuntu.com/authd/stable/
[22] https://github.com/susam/gitpr

Once merged into the main branch, `po` files and any documentation change will be automatically updated. Updates to these files are therefore not necessary in the pull request itself, which helps minimize diff review.

The authd documentation is published in **edge-docs** and **stable-docs** versions. Only the edge version is updated when documentation changes are merged into the main branch. If a documentation change should be applied to the stable documentation *before* the next release, create a separate PR against the `stable-docs` branch after your main PR has been merged, with the changes to the documentation cherry-picked from your main PR.

## Contributing to the code

### Required dependencies

This project has several build dependencies. You can install these dependencies from the top of the source tree using the `apt` command as follows:

```
sudo apt update
sudo apt build-dep .
sudo apt install devscripts
```

### Building and running the binaries

The project consists of the following binaries:

- `authd`: The main authentication service.
- `pam_authd.so`: A PAM native module (used by GDM).
- `pam_authd_exec.so`, `authd-pam`: A PAM module and its helper executable (used by other PAM applications).
- `libnss_authd.so`: An NSS module.

The project can be built as a Debian package. This process will compile all the binaries, run the test suite and produce the Debian packages.

Alternatively, for development purposes, each binary can be built manually and separately.

### Building the Debian package from source

Building the Debian package from source is the most straightforward and standard method for compiling the binaries and running the test suite. To do this, run the following commands from the top of the source tree:

> This is required to vendorize the Rust crates and must be done only once.

```
sudo apt install libssl-dev
cargo install cargo-vendor-filterer
```

Then build the Debian package:

```
debuild --prepend-path=${HOME}/.cargo/bin
```

The Debian packages are available in the parent directory.

### Building authd only

To build `authd` only, run the following command from the top of the source tree:

```
go build ./cmd/authd
```

The built binary will be found in the current directory. The daemon can be run directly from this binary without installing it on the system.

### Building the PAM module only

To build the PAM module, you first need to install the tooling to hook up the Go gRPC modules to protoc. From the top of the source tree run the following commands:

```
cd tools/
grep -o '_ ".*"' *.go | cut -d '"' -f 2 | xargs go install
cd ..
```

Then build the PAM module:

```
go generate ./pam/
go build -tags pam_binary_exec -o ./pam/authd-pam ./pam
```

This last command will produce two libraries (`./pam/pam_authd.so` and `./pam/go-exec/pam_authd_exec.so`) and an executable (`./pam/authd-pam`).

These modules must be copied to `/usr/lib/$(gcc -dumpmachine)/security/` while the executable must be copied to `/usr/libexec/authd-pam`.

For further information about the PAM module architecture and testing see the PAM Hacking[23] page.

### Building the NSS module only

To build the NSS module, from the top of the source tree run the command:

```
cargo build
```

This will build a debug release of the NSS module.

The library resulting from the build is located in `./target/debug/libnss_authd.so`. This module must be copied to `/usr/lib/$(gcc -dumpmachine)/libnss_authd.so.2`.

[23] https://github.com/canonical/authd/blob/main/pam/Hacking.md

### Building the broker

The authd brokers are packaged as separate snaps that are built and released independently from authd. The source code for the brokers is located in `./authd-oidc-brokers` and the snap packaging files are located in `./snap`.

To build the broker snap for a specific broker variant, follow these steps from the top of the source tree:

1. Ensure that the submodules are checked out:

   ```
   git submodule update --init --recursive
   ```

2. Prepare the `snapcraft.yaml` for the desired broker variant:

   ```
   ./snap/scripts/prepare-variant --broker <broker>
   ```

   where `<broker>` is one of `oidc`, `msentraid`, or `google`.

3. Build the broker snap:

   ```
   snapcraft pack
   ```

When the build succeeds, the resulting `.snap` file is created in the current working directory. You can install the locally built broker snap for development with a command such as:

```
snap install --dangerous ./path/to/broker.snap
```

### About the test suite

The project includes a comprehensive test suite made of unit and integration tests. All the tests must pass before the review is considered. If you have troubles with the test suite, feel free to mention it in your PR description.

You can run all tests with: `go test ./...` (add the `-race` flag for race detection).

Every package has a suite of at least package-level tests. They may integrate more granular unit tests for complex functionalities. Integration tests are located in `./pam/integration-tests` for the PAM module and `./nss/integration-tests` for the NSS module.

The test suite must pass before merging the PR to our main branch. Any new feature, change or fix must be covered by corresponding tests.

### Tests with dependencies

Some tests, such as the PAM CLI tests[24], use external tools such as VHS[25] to record and run the tape files needed for the tests. Those tools are not included in the project dependencies and must be installed manually.

Information about these tools and their usage will be linked below:

[24] https://github.com/canonical/authd/blob/5ba54c0a573f34e99782fe624b090ab229798fc3/pam/integration-tests/integration_test.go#L21

[25] https://github.com/charmbracelet/vhs

- VHS[26]: tutorial on using VHS as a CLI-based video recorder

### Code style

This project follow the Go code-style. For more detailed information about the code style in use, please check https://google.github.io/styleguide/go/.

### Contributing to the documentation

You can contribute to the documentation in various ways.

At the top of each page in the documentation, there is a **Give feedback** button. If you find an issue in the documentation, clicking this button will open an Issue submission on GitHub for the specific page.

For minor changes, such as fixing a single typo, you can click the **pencil** icon at the top right of any page. This will open up the source file in GitHub so that you can make edits directly.

For more significant changes to the content or organization of the documentation, you should create your own fork and follow the steps outlined in the section on *pull requests* (page 39).

### Building the documentation

After cloning your fork, change into the `/docs/` directory. The documentation is written in markdown files grouped under Diátaxis[27] categories.

A makefile is used to preview and test the documentation locally. To view all the possible commands, run `make` without arguments.

The command `make run` will serve the documentation at port `8000` on `localhost`. You can then preview the documentation in your browser and the preview will automatically update with each change that you make.

To clean the build environment at any point, run `make clean`.

When you submit a PR, there are automated checks for typos and broken links. Please run the tests locally before submitting the PR to save yourself and your reviewers time.

### Testing the documentation

Automatic checks will be run on any PR relating to documentation to verify spelling and the validity of links. Before submitting a PR, you can check for any issues locally:

- Check the spelling: `make spelling`
- Check the validity of links: `make linkcheck`

[26] https://github.com/charmbracelet/vhs?tab=readme-ov-file#tutorial
[27] https://diataxis.fr/

Doing these checks locally is good practice. You are less likely to run into failed CI checks after your PR is submitted and the reviewer of your PR can more quickly focus on the substance of your contribution.

If the documentation builds, your PR will generate a preview of the documentation on Read the Docs. This preview appears as a check in the CI. Click on the check to open the preview and confirm that your changes have been applied successfully.

### Open Documentation Academy

authd is a proud member of the Canonical Open Documentation Academy[28] (CODA).

CODA is an initiative to encourage open source contributions from the community, and to provide help, advice and mentorship to people making their first contributions.

A key aim of the initiative is to lower the barrier to successful open-source software contributions by making documentation into the gateway, and it's a great way to make your first open source contributions to projects like authd.

The best way to get started is to take a look at our project-related documentation tasks[29] and read our Getting started guide[30]. Tasks typically include testing and fixing documentation pages, updating outdated content, and restructuring large documents. We'll help you see those tasks through to completion.

You can get involved the with the CODA community through:

- The discussion forum[31] on the Ubuntu Community Hub
- The Matrix channel[32] for interactive chat
- Fosstodon[33] for the latest updates and events

### Contributor License Agreement

It is a requirement that you sign the Contributor License Agreement[34] in order to contribute to this project. You only need to sign this once and if you have previously signed the agreement when contributing to other Canonical projects you will not need to sign it again.

An automated test is executed on PRs to check if this agreement has been accepted.

[28] https://github.com/canonical/open-documentation-academy
[29] https://github.com/canonical/open-documentation-academy/issues
[30] https://discourse.ubuntu.com/t/getting-started/42769
[31] https://discourse.ubuntu.com/c/community/open-documentation-academy/166
[32] https://matrix.to/#/#documentation:ubuntu.com
[33] https://fosstodon.org/@CanonicalDocumentation
[34] https://ubuntu.com/legal/contributors

### Getting help

Join us in the Ubuntu Community[35] and post your question there with a descriptive tag.

# 3.2. Reference

These guides provide technical information about authd.

## 3.2.1. Providers

Multiple identity providers and brokers are supported by authd:

### Identity providers that authd supports

authd supports identity providers through its identity brokers. Each broker is available as a snap. Several brokers can be installed and enabled on a system.

| Provider | Broker snap | Install as a snap | Configure | Provider docs |
|---|---|---|---|---|
| Google IAM | authd-google[36] | `snap install authd-google` | Google IAM guide | Google[37] |
| Microsoft Entra ID | authd-msentraid[38] | `snap install authd-msentraid` | Microsoft Entra ID guide | Microsoft[39] |
| Keycloak | authd-oidc[40] | `snap install authd-oidc` | Keycloak guide | Keycloak[41] |

**Note:**

Support for multiple additional providers is planned for future releases of authd.

## 3.2.2. Troubleshooting

The documentation includes several pages that are helpful when troubleshooting authd:

[35] https://discourse.ubuntu.com/c/desktop/8
[36] https://snapcraft.io/authd-google
[37] https://cloud.google.com/iam/docs/overview
[38] https://snapcraft.io/authd-msentraid
[39] https://learn.microsoft.com/en-us/entra/fundamentals/whatis
[40] https://snapcraft.io/authd-oidc
[41] https://www.keycloak.org/documentation

## Troubleshooting

This page includes links to authd documentation that may be helpful when troubleshooting.

### Logging

Logs are generated for authd and its brokers, which can be useful when troubleshooting and reporting bugs.

To learn how to get logs and configure logging behavior, read the dedicated *guide on logging* (page 32).

### Changing versions

To test new features or check if a bug has been fixed in a new version, you can switch to edge releases for authd and its brokers.

This is described in the *guide on changing versions* (page 36).

### Only the first logged-in user can get access

This is the expected behavior, as the first logged-in user becomes the owner and only the owner has access by default.

To change access you can make the next logged-in user the owner or add more allowed users.

Guidelines on *configuring allowed users* (page 13) are outlined in the *configuring authd guide* (page 7).

### File ownership on shared network resources (NFS, Samba)

The user identifiers (UIDs) and group identifiers (GIDs) assigned by authd are unique to each machine. This means that when using authd with NFS or Samba, the UIDs and GIDs of users and groups on the server will not match those on the client machines, which leads to permission issues.

To avoid these issues, you can use ID mapping. For more information, see the dedicated guides on NFS and Samba:

- *Using authd with NFS* (page 23)
- *Using authd with Samba* (page 29)

#### Recovery mode for failed login

If authd and/or the broker are missing, corrupted, or broken in any way, a user may be prevented from logging in.

When this occurs, you can *boot into recovery mode* (page 35) to access to the system for modifying configurations and installations.

### 3.2.3. Groups

Managing groups of users that need the same access and permissions to resources is supported by the Microsoft Entra ID broker for authd:

#### Group and privilege management

Groups are used to manage users that all need the same access and permissions to resources. For example, you can manage **sudo** and **docker** rights of users based on group membership.

Groups from the identity provider can be mapped into local Linux groups for the user. You can also configure extra groups in the broker configuration file, as described in the *configuration guide* (page 16).

| Broker support for group management |
| --- |
| Groups are currently supported for the `msentraid` broker. |

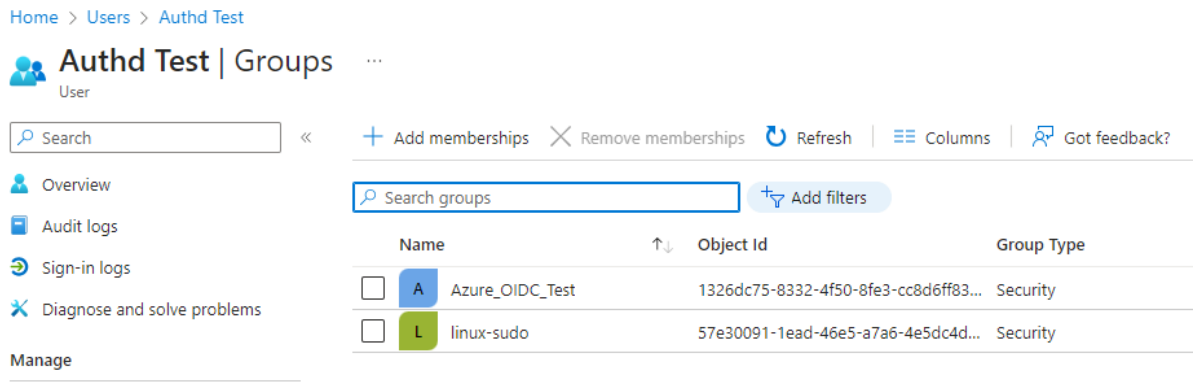
#### Microsoft Entra ID

Microsoft Entra ID supports creating groups and adding users to them.

See Manage Microsoft Entra groups and group membership[42]

[42] https://learn.microsoft.com/en-us/entra/fundamentals/how-to-manage-groups

For example, the user `authd test` is a member of the Entra ID groups `Azure_OIDC_Test` and `linux-sudo`:




This translates to the following Linux groups on the local machine:

```
~$ groups
aadtest-testauthd@uaadtest.onmicrosoft.com sudo azure_oidc_test
```

There are three types of groups:

1. **Primary group**: Created automatically based on the user name
2. **Local group**: Group local to the machine prefixed with `linux-`. For example, if the user is a member of the Azure group `linux-sudo`, they will be a member of the `sudo` group locally.
3. **Remote group**: All the other Azure groups the user is a member of.

## 3.2.4. Deployment

Deploying authd at scale can be achieved with Landscape or cloud-init. The documentation includes snippets to get you started.

### Reference snippets for Landscape deployment script

Landscape[43] is a systems management tool for the remote provisioning and management of Ubuntu machines.

Landscape can be used to remotely execute scripts[44] on client machines.

This page provides example snippets that can be used in your own deployment scripts to install and configure authd on Ubuntu machines at scale.

**Note:**

When you deploy the final script using Landscape, ensure that the script is run as the `root` user.

### Setup

Define the following environmental variables:

```
ISSUER_ID=<ISSUER_ID>
CLIENT_ID=<CLIENT_ID>
```

[43] https://documentation.ubuntu.com/landscape/
[44] https://documentation.ubuntu.com/landscape/how-to-guides/web-portal/web-portal-24-04-or-later/use-script-profiles/

### Installation

For Ubuntu 26.04 LTS, install the authd deb and the broker snap from the archive:

**Add PPA before installing on Ubuntu 24.04**

On Ubuntu 24.04 LTS, add the following line at the start of your script, before `apt-get install authd`, to add the stable PPA:

```
add-apt-repository -y ppa:ubuntu-enterprise-desktop/authd
```

#### Google IAM

```
apt-get upgrade -y
apt-get install -y authd
snap install authd-google
```

#### Microsoft Entra ID

```
apt-get upgrade -y
apt-get install -y authd
snap install authd-msentraid
```

**Tip:**

For more information on installing authd and its brokers, read the *installation guide* (page 5).

### Configuration

Configure authd and the broker:

#### Google IAM

```
sed -i "s|<CLIENT_ID>|$CLIENT_ID|g; s|<ISSUER_ID>|$ISSUER_ID|g" /var/snap/authd-google/
current/broker.conf
echo "ssh_allowed_suffixes_first_auth = @example.com" >> /var/snap/authd-google/current/
broker.conf
mkdir -p /etc/authd/brokers.d/
cp /snap/authd-google/current/conf/authd/google.conf /etc/authd/brokers.d/
cat <<EOF >> /etc/ssh/sshd_config.d/authd.conf
UsePAM yes
Match User *@example.com
    KbdInteractiveAuthentication yes
EOF
```

### Microsoft Entra ID

```
sed -i "s|<CLIENT_ID>|$CLIENT_ID|g; s|<ISSUER_ID>|$ISSUER_ID|g" /var/snap/authd-msentraid/
current/broker.conf
echo "ssh_allowed_suffixes_first_auth = @example.onmicrosoft.com" >> /var/snap/authd-
msentraid/current/broker.conf
mkdir -p /etc/authd/brokers.d/
cp /snap/authd-msentraid/current/conf/authd/msentraid.conf /etc/authd/brokers.d/
cat <<EOF >> /etc/ssh/sshd_config.d/authd.conf
UsePAM yes
Match User *@example.onmicrosoft.com
    KbdInteractiveAuthentication yes
EOF
```

**Tip:**

For more information on configuring authd, read the *configuration guide* (page 7).

### Restart the services

Restart the authd daemon, the broker snap, and the SSH service:

### Google IAM

```
systemctl restart authd ssh
snap restart authd-google
```

### Microsoft Entra ID

```
systemctl restart authd ssh
snap restart authd-msentraid
```

When you have a complete script, add it to the Landscape dashboard to run as the `root` user before executing on the target machines.

### Authentication

Once the script is deployed, user login should be possible with authd.

For example, *using SSH* (page 22):

```
ssh <username>@<host>
```

### Additional information

- Blog on Entra ID authentication on Ubuntu at scale[45]
- Video on Entra ID authentication on Ubuntu Desktop at scale[46]

## Reference snippets for cloud-init provisioning

Cloud-init[47] is an industry-standard method for cloud instance initialization. It can also be used to provision client machines during Ubuntu installation.

This page provides example snippets, which can be used in your own cloud config YAML files to deploy and configure authd on Ubuntu at scale.

### Setup

If using these snippets as part of a cloud config[48] file, set the appropriate header, identifying the file to cloud-init with `#cloud-config` and enabling Jinja templating with `## template: jinja`.

```
## template: jinja
#cloud-config
```

### Variables

Define the necessary environmental variables:

```
{% set ISSUER_ID = '<your_issuer_id>' %}
{% set CLIENT_ID = '<your_client_id>' %}
```

### Install authd

`authd` is available directly from the Ubuntu archive:

```
packages:
  - authd
```

[45] https://ubuntu.com/blog/entra-id-authentication-on-ubuntu-at-scale-with-landscape
[46] https://www.youtube.com/watch?v=1tYNEby5-hw
[47] https://cloudinit.readthedocs.io/en/latest/
[48] https://docs.cloud-init.io/en/latest/explanation/format/cloud-config.html#user-data-formats-cloud-config

**Add PPA before installing on Ubuntu 24.04**

On Ubuntu 24.04 LTS, `authd` must be installed from the stable PPA. Add the following to your cloud config, before the `packages` block:

```
apt:
  sources:
      source1:
          source: 'ppa:ubuntu-enterprise-desktop/authd'
```

### Install broker

Install the broker as a snap:

### Google IAM

```
snap:
 commands:
   - ['install', 'authd-google']
```

### Microsoft Entra ID

```
snap:
 commands:
   - ['install', 'authd-msentraid']
```

**Tip:**

For more information on installing authd and its brokers, read the *installation guide* (page 5).

### Install authd and apply configurations

To complete the setup:

- Configure SSH for user login
- Upgrade packages
- Configure authd and the broker
- Restart the services for the changes to take effect

**Important:**

Edit the allowed suffixes as appropriate.

### Google IAM

```
write_files:
  - path: /etc/ssh/sshd_config.d/authd.conf
    content: |
      UsePAM yes
      Match User *@example.com
          KbdInteractiveAuthentication yes

runcmd:
  - apt-get upgrade -y
  - sed -i 's|<CLIENT_ID>|{{ CLIENT_ID }}|g; s|<ISSUER_ID>|{{ ISSUER_ID }}|g' /var/snap/
authd-google/current/broker.conf
  - echo 'ssh_allowed_suffixes_first_auth = @example.com' >> /var/snap/authd-google/
current/broker.conf
  - sed -i 's/^\(LOGIN_TIMEOUT\t\t\)[0-9]\+/\1360/' /etc/login.defs
  - cp /snap/authd-google/current/conf/authd/google.conf /etc/authd/brokers.d/
  - snap restart authd-google
  - systemctl restart authd ssh
```

### Microsoft Entra ID

```
write_files:
  - path: /etc/ssh/sshd_config.d/authd.conf
    content: |
      UsePAM yes
      Match User *@example.onmicrosoft.com
          KbdInteractiveAuthentication yes

runcmd:
  - apt-get upgrade -y
  - sed -i 's|<CLIENT_ID>|{{ CLIENT_ID }}|g; s|<ISSUER_ID>|{{ ISSUER_ID }}|g' /var/snap/
authd-msentraid/current/broker.conf
  - echo 'ssh_allowed_suffixes_first_auth = @example.onmicrosoft.com' >> /var/snap/authd-
msentraid/current/broker.conf
  - sed -i 's/^\(LOGIN_TIMEOUT\t\t\)[0-9]\+/\1360/' /etc/login.defs
  - mkdir -p /etc/authd/brokers.d/
  - cp /snap/authd-msentraid/current/conf/authd/msentraid.conf /etc/authd/brokers.d/
  - snap restart authd-msentraid
  - systemctl restart authd ssh
```

**Tip:**

For more information on configuring authd, read the *configuration guide* (page 7).

#### Authentication

Once the script is deployed, user login should be possible with authd.

For example, *using SSH* (page 22):

```
ssh <username>@<host>
```

#### Additional information

- Blog on Entra ID authentication on Ubuntu at scale[49]
- Video on Entra ID authentication on Ubuntu Desktop at scale[50]

### 3.2.5. Command line tool (authctl)

The `authctl` command line tool is used to manage authd users and groups.

Available commands:

#### authctl reference

The `authctl` command line tool is used to manage authd users and groups.

Available commands:

#### authctl

Manage authd users and groups

#### Synopsis

authctl is a command-line tool for managing users and groups handled by authd.

```
authctl [flags]
```

#### Options

```
  -h, --help   help for authctl
```

[49] https://ubuntu.com/blog/entra-id-authentication-on-ubuntu-at-scale-with-landscape
[50] https://www.youtube.com/watch?v=1tYNEby5-hw

### SEE ALSO

- *authctl group* (page 57) - Commands related to groups
- *authctl user* (page 55) - Commands related to users

### authctl user

Commands related to users

```
authctl user [flags]
```

### Options

```
  -h, --help   help for user
```

### SEE ALSO

- *authctl* (page 54) - Manage authd users and groups
- *authctl user lock* (page 55) - Lock (disable) a user managed by authd
- *authctl user set-uid* (page 56) - Set the UID of a user managed by authd
- *authctl user unlock* (page 56) - Unlock (enable) a user managed by authd

### authctl user lock

Lock (disable) a user managed by authd

### Synopsis

Lock a user so that they cannot log in.

```
authctl user lock <user> [flags]
```

### Options

```
  -h, --help   help for lock
```

### SEE ALSO

- *authctl user* (page 55) - Commands related to users

### authctl user unlock

Unlock (enable) a user managed by authd

### Synopsis

Unlock a locked user so that they can log in again.

```
authctl user unlock <user> [flags]
```

### Options

```
  -h, --help   help for unlock
```

### SEE ALSO

- *authctl user* (page 55) - Commands related to users

### authctl user set-uid

Set the UID of a user managed by authd

### Synopsis

Set the UID of a user managed by authd to the specified value.

The new UID must be unique and non-negative. The command must be run as root.

The ownership of the user's home directory, and any files within the directory that the user owns, will automatically be updated to the new UID.

Files outside the user's home directory are not updated and must be changed manually. Note that changing a UID can be unsafe if files on the system are still owned by the original UID: those files may become accessible to a different account that is later assigned that UID.

```
authctl user set-uid <user> <uid> [flags]
```

### Examples

```
# Set the UID of user "alice" to 15000
authctl user set-uid alice 15000
```

### Options

```
-h, --help   help for set-uid
```

### SEE ALSO

- *authctl user* (page 55) - Commands related to users

## authctl group

Commands related to groups

```
authctl group [flags]
```

### Options

```
-h, --help   help for group
```

### SEE ALSO

- *authctl* (page 54) - Manage authd users and groups
- *authctl group set-gid* (page 57) - Set the GID of a group managed by authd

## authctl group set-gid

Set the GID of a group managed by authd

### Synopsis

Set the GID of a group managed by authd to the specified value.

The new GID must be unique and non-negative. The command must be run as root.

When a group's GID is changed, any users whose primary group is set to this group will have the GID of their primary group updated. The home directories of these users, and any files within these directories that are owned by the group, will be updated to the new GID.

Files outside users' home directories are not updated and must be changed manually. Note that changing a GID can be unsafe if files on the system are still owned by the original GID: those files may become accessible to a different group that is later assigned that GID.

```
authctl group set-gid <group> <gid> [flags]
```

#### Examples

```
  # Set the GID of group "staff" to 30000
  authctl group set-gid staff 30000
```

#### Options

```
  -h, --help   help for set-gid
```

#### SEE ALSO

- *authctl group* (page 57) - Commands related to groups

# 3.3. Explanation

These guides explain how authd works.

## 3.3.1. Architecture

authd has a modular design and can interface with multiple identity providers through different identity brokers.

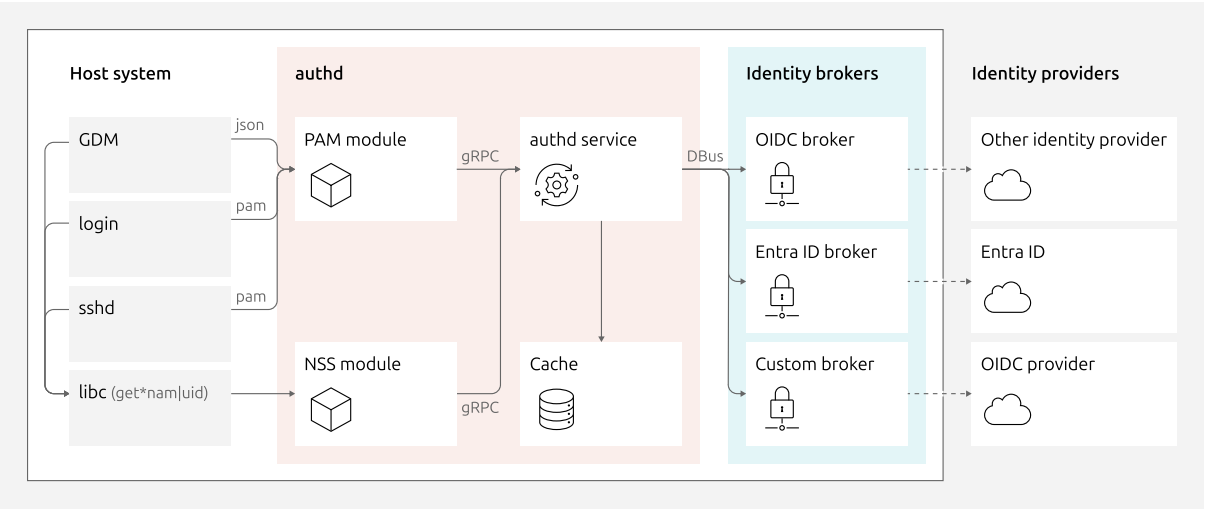
#### authd architecture

authd can help organizations ensure secure identity and access management by enabling seamless cloud-based authentication of Ubuntu machines. Here we explain the architecture of authd and some of its design decisions. Links are provided at the end to support further reading.

#### Architecture components

authd acts as an interface between the host system and external identity providers. Remote information is cached when authenticating with authd, which improves performance while also facilitating offline access.

The diagram below illustrates the components of authd and their communication methods:




The architecture of authd consists of the following components:

- The **authentication daemon** (authd): a daemon that runs on the host and manages access to the authentication service.
- An **identity broker**: a trusted component running on the host and serving as an interface with the identity provider.
- The **identity provider**: the remote service that manages digital identities, such as Microsoft Entra ID.
- A **PAM module**: the library that handles the authentication tasks of applications on the system. The authentication tasks that are currently handled by PAM include GDM, login, ssh and sudo. Support for webview is being developed.
- An **NSS module**: a module that queries authd to retrieve user information from the cache.

**Note:**

One or multiple identity brokers and identity providers can be enabled at the same time.

### Capabilities negotiation

The platform (whether Ubuntu Desktop, Ubuntu Server, or Ubuntu cloud instance) describes its capabilities and current state, including the availability of a graphical interface, online status, and other general features such as TPM, smartcard, or biometric support.

The broker, acting as the decision-maker, evaluates its capabilities and current state to select the appropriate workflow. It can also consider specific decision factors, such as temporarily bypassing full MFA or permitting offline cached authentication for a specified duration.

The broker is a trusted component within the system and serves as the interface with the identity provider.

### Inter-process communication

Internal components of authd, such as PAM and NSS modules, communicate over gRPC. The advantage of gRPC is that it delivers high performance and native streaming, with built-in security for efficient and secure communication in distributed systems.

The communication between authd and the brokers is done over D-Bus. D-Bus supports message broadcasting and enables efficient resource sharing. The communication only goes from the authentication daemon to the broker, which responds to requests. The transactions are encrypted, ensuring that communications between the broker and authd are secure.

### Links

- Microsoft Entra fundamentals documentation[51]
- OpenID Connect (OIDC) on the Microsoft identity platform[52]
- What is OpenID Connect and what do you use it for? - Auth0[53]
- What Is OpenID Connect (OIDC)? | Microsoft Security[54]

## 3.3.2. Security

Learn about key design decisions, security practices, and configuration options that affect the security of authd.

### Security overview for authd

authd lets Ubuntu systems verify user identities through trusted cloud identity providers and integrate those identities into local logins securely.

This overview outlines the key design decisions, security practices, and configuration options that affect the security of authd, and provides guidance for administrators on deploying authd securely.

### Deploying authd securely as an administrator

The first section of this overview provides guidance for administrators on deploying authd securely, and the second section explains design decisions that were made to enhance the security of authd.

[51] https://learn.microsoft.com/en-us/entra/fundamentals/
[52] https://learn.microsoft.com/en-us/entra/identity-platform/v2-protocols-oidc
[53] https://auth0.com/intro-to-iam/what-is-openid-connect-oidc
[54] https://www.microsoft.com/en-us/security/business/security-101/what-is-openid-connect-oidc

### Package updates

Keeping authd up to date is essential to ensure that security fixes are applied as soon as they are available.

A full authd installation consists of both the authd Debian package and at least one broker snap. Snaps are updated automatically. Further information is provided in the snap documentation on managing updates[55].

Update Debian packages regularly, either manually or by enabling automatic updates[56].

### Security of the identity provider

As described in the *authd architecture* (page 58) explanation, users authenticate with an identity provider, such as Microsoft Entra ID or Google IAM. The security of any system using authd therefore depends on the security of the configured identity provider.

Ensure that only authorized administrators have access to the identity provider. Administrators can add or remove users or change user credentials, which allows them to grant, revoke, or modify access to systems using authd.

### Allowed users

In addition to access control at the identity provider's side, allowed users can be restricted locally. See the *Configure allowed users* (page 13) section for details.

Ensure that only users who should have system access are configured as allowed users.

### Local password

To improve usability, authd allows users to create a local password after successfully authenticating with the identity provider. This password can then be used for subsequent logins without repeating the full identity provider authentication flow.

### Password strength

Strong passwords are critical to prevent unauthorized access.

authd uses libpwquality to enforce password complexity requirements. See the *Configure password quality* (page 18) section for details.

[55] https://snapcraft.io/docs/managing-updates
[56] https://documentation.ubuntu.com/server/how-to/software/automatic-updates/

### Force provider authentication

If the identity provider is reachable during login, authd verifies that the user is still allowed to authenticate with the identity provider. If the user's account has been disabled or removed, login is denied.

By default, if the identity provider cannot be reached (for example, due to network issues), users can still log in with their local password. This is to prevent accidental lockouts, but it also allows users whose access has been revoked at the identity provider to log in while the provider is unreachable.

To enforce verification with the identity provider even when it is unreachable, enable the *force_provider_authentication* (page 13) setting.

## Login via SSH

### SSH public key authentication

If SSH public key authentication is enabled, users whose access has been revoked at the identity provider can still log in using their SSH keys. This is because SSH key authentication does not involve authd.

To prevent users with revoked access from logging in with SSH, disable public key authentication for users managed by authd, by adding the following to `/etc/ssh/sshd_config.d/authd.conf` or directly to `/etc/ssh/sshd_config`:

```
Match User *@example.com
    PubkeyAuthentication no
```

**Note:**

Replace `@example.com` with the domain of your identity provider.

### SSH password authentication

As described in the *SSH configuration* (page 22) section, PAM integration must be enabled in sshd:

```
UsePAM yes
KbdInteractiveAuthentication yes
```

This setup allows users managed by authd to log in through PAM. It also enables PAM-based login for local Unix accounts, even when `PasswordAuthentication no` is set.

Administrators who want to deviate from the default SSH configuration and disallow password authentication for non-authd users can do so safely by using a match block that re-enables keyboard-interactive authentication only for authd users:

```
KbdInteractiveAuthentication no
PasswordAuthentication no
```

(continues on next page)

(continued from previous page)

```
Match User *@example.com
    KbdInteractiveAuthentication yes
```

**Note:**

Replace `@example.com` with the domain of your identity provider.

### UID and GID conflicts

When a new user logs in for the first time, or when a user is added to a new group in the identity provider (for providers that support group management, see Group management[57]), authd automatically assigns a unique user ID (UID) and group ID (GID).

Before assigning a UID or GID, authd checks that there are no collisions with existing users or groups on the system. However, if a user or group is later removed, or if the entire authd database (`/var/lib/authd/authd.sqlite3`) is deleted, previously assigned IDs can be reused for new users or groups.

This reuse can allow unintended access to files or directories owned by the old user, as the new user would inherit their numeric UID or GID.

To avoid this risk:

- Do not remove the authd database.
- Remove all files and directories owned by any users that you delete, especially if they contain sensitive data.

**Important:**

A tool for removing authd users along with their home directories will be provided in the future.

### How authd is designed for security

This section describes how authd is built to protect stored data and limit system exposure.

[57] https://documentation.ubuntu.com/authd/stable-docs/reference/group-management/

### Stored secrets

authd stores user secrets under `/var/snap/authd-<broker>/current/<issuer>/<user>/`.

That directory is created with mode `0700`, ensuring that only root can access it.

The secrets that authd stores are described below.

### Local password

A salted Argon2id hash of the local password is stored for verification. Hashing parameters:

- Memory: 64 KB
- Iterations: 1
- Parallelism: 1

### Tokens and user information

Tokens and user data retrieved during authentication — including the OAuth 2.0 refresh token and OpenID Connect `UserInfo` response — are cached to support login with the local password. These values are currently stored in cleartext.

We recommend enabling full disk encryption[58] to protect these secrets in case of device theft or loss.

### Sandboxing

authd uses sandboxing to limit system exposure:

- The authd brokers run as strictly confined[59] snaps. The only snap interface granted to them is `network`, which is required to communicate with the identity provider.
- The authd service uses systemd sandboxing options[60] to restrict access to system resources.

Because authd acts as an authentication service, a vulnerability in authd could still be exploited to gain full root privileges.

[58] https://documentation.ubuntu.com/security/docs/security-features/storage/encryption-full-disk/
[59] https://snapcraft.io/docs/snap-confinement
[60] https://manpages.ubuntu.com/manpages/noble/en/man5/systemd.exec.5.html#sandboxing

#### Reporting a vulnerability

See the authd security policy[61] for details on how to report security vulnerabilities in authd.

### 3.3.3. Documentation

Information about the authd documentation itself.

#### Structure of authd documentation

The authd documentation uses the Diátaxis structure[62].

Currently, the documentation consists of how-to guides, references, and explanations:

- A *how-to guide* (page 5) provides you with the steps necessary for completing specific tasks.
- A *reference* (page 45) gives you concise and factual information to support your understanding.
- An *explanation* (page 58) includes topic overviews and additional context on the software.

[61] https://github.com/canonical/authd?tab=security-ov-file#security-ov-file
[62] https://diataxis.fr/