Authentication

dozzle.dev/guide/authentication

11 de diciembre de 2025

Dozzle supports two configurations for authentication. In the first configuration, you bring your own authentication method by protecting Dozzle through a proxy. Dozzle can read appropriate headers out of the box.

If you do not have an authentication solution, then Dozzle has a simple file-based user management solution. Authentication providers are set up using the `--auth-provider` flag. In both configurations, Dozzle will try to save user settings to disk. This data is written to `/data`.

File-Based User Management

Dozzle supports multi-user authentication by setting `--auth-provider` to `simple`. In this mode, Dozzle will attempt to read the users file from `/data/`, prioritizing `users.yml` over `users.yaml` if both files are present. If only one of the files exists, it will be used. The log will indicate which file is being read (e.g., `Reading users.yml file`).

Example file paths:

- `/data/users.yml`
- `/data/users.yaml`

The content of the file looks like:

yaml

```
users:
  # "admin" here is username
  admin:
    email: me@email.net
    name: Admin
    # Generate with docker run -it --rm amir20/dozzle generate --name Admin --email
me@email.net --password secret admin
    password: $2a$11$9ho4vY2LdJ/WBopFcsAS0uORC0x2vuFHQgT/yBqZyzclhHsoaIkzK
    filter:
    roles:
```

Dozzle uses `email` to generate avatars using Gravatar. It is optional. The password is hashed using `bcrypt` which can be generated using `docker run amir20/dozzle generate`.

WARNING

In previous versions of Dozzle, SHA-256 was used to hash passwords. Bcrypt is now more secure and is recommended for future use. Dozzle will revert to SHA-256 if it does not find a bcrypt hash. It is advisable to update the password hash to bcrypt using `generate`. For more details, see this issue.
You will need to mount this file for Dozzle to find it. Here is an example:

clidocker-compose.ymlusers.yml

sh

```
$ docker run -v /var/run/docker.sock:/var/run/docker.sock -v /path/to/dozzle/data:/data -p
8080:8080 amir20/dozzle --auth-provider simple
```

Or using Docker secrets:

yaml

```
services:
  dozzle:
    image: amir20/dozzle:latest
    environment:
      - DOZZLE_AUTH_PROVIDER=simple
    secrets:
      - source: users
        target: /data/users.yml
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
      - dozzle:/data
secrets:
  users:
    file: users.yml
volumes:
  dozzle:
```

Extending Authentication Cookie Lifetime

By default, Dozzle uses session cookies which expire when the browser is closed. You can extend the lifetime of the cookie by setting `--auth-ttl` to a duration. Here is an example:

clidocker-compose.yml

sh

```
$ docker run -v /var/run/docker.sock:/var/run/docker.sock -v /path/to/dozzle/data:/data -p
8080:8080 amir20/dozzle --auth-provider simple --auth-ttl 48h
```

Note that only duration is supported. You can only use `s`, `m`, `h` for seconds, minutes and hours respectively.

Setting specific filters for users

Dozzle supports setting filters for users. Filters are used to restrict the containers that a user can see. Filters are set in the `users.yml` file. Here is an example:

yaml

```
users:
  admin:
    email:
    name: Admin
    password: $2a$11$9ho4vY2LdJ/WBopFcsAS0uORC0x2vuFHQgT/yBqZyzclhHsoaIkzK
    filter:

  guest:
    email:
    name: Guest
    password: $2a$11$9ho4vY2LdJ/WBopFcsAS0uORC0x2vuFHQgT/yBqZyzclhHsoaIkzK
    filter: "label=com.example.app"
```

In this example, the `admin` user has no filter, so they can see all containers. The `guest` user can only see containers with the label `com.example.app`. This is useful for restricting access to specific containers.

NOTE

Filters can also be set globally with the `--filter` flag. This flag is applied to all users. If a user has a filter set, it will override the global filter.

Setting specific roles for users

Dozzle allows assigning roles to users. Roles define what actions a user can perform on containers. Roles are configured in the users.yml file.

yaml

```
users:
  admin:
    email:
    name: Admin
    password: $2a$11$9ho4vY2LdJ/WBopFcsAS0uORC0x2vuFHQgT/yBqZyzclhHsoaIkzK
    roles:

  guest:
    email:
    name: Guest
    password: $2a$11$9ho4vY2LdJ/WBopFcsAS0uORC0x2vuFHQgT/yBqZyzclhHsoaIkzK
    roles: shell
```

In this example, the `admin` user has no roles specified, so they have full access to all container actions. The `guest` user has the shell role, meaning they can only open a shell in the containers. Roles make it easy to control and restrict what users can do in Dozzle.

Dozzle supports the following roles:

- **shell** - allows attach and exec in the container
- **actions** - allows performing container actions (start, stop, restart)
- **download** - allows downloading container logs
- **none** - denies all actions
- **all** - allows all actions (default)

Generating users.yml

Dozzle has a built-in `generate` command to generate `users.yml`. Here is an example:

sh

```
docker run -it --rm amir20/dozzle generate admin --password password --email
test@email.net --name "John Doe" --user-filter name=foo --user-roles shell > users.yml
```

In this example, `admin` is the username. Email and name are optional but recommended to display accurate avatars. `docker run -it --rm amir20/dozzle generate --help` displays all options. The `--user-filter` flag is a comma-separated list of filters. The `--user-roles` flag is a comma-separated list of roles.

Forward Proxy

Dozzle can be configured to read proxy headers by setting `--auth-provider` to `forward-proxy`.

clidocker-compose.yml

sh

```
$ docker run -v /var/run/docker.sock:/var/run/docker.sock -p 8080:8080 amir20/dozzle --
auth-provider forward-proxy
```

In this mode, Dozzle expects the following headers:

- `Remote-User` to map to the username e.g. `johndoe`
- `Remote-Email` to map to the user's email address. This email is also used to find the right Gravatar for the user.
- `Remote-Name` to be a display name like `John Doe`
- `Remote-Filter` to be a comma-separated list of filters allowed for user.
- `Remote-Roles` to be a comma-separated list of roles allowed for user.

Additionally, you can configure a logout URL with:

yaml

```
DOZZLE_AUTH_LOGOUT_URL: http://oauth2.example.ru/oauth2/sign_out
```

Setting up Dozzle with Authelia

Authelia is an open-source authentication and authorization server and portal fulfilling the identity and access management. While setting up Authelia is out of scope for this section, the configuration can be shared as an example for setting up Dozzle with Authelia.

▶ ➡️ Click to expand Authelia example

Setting up Dozzle with Cloudflare Zero Trust

Cloudflare Zero Trust is a service for authenticated access to self-hosted software. This section defines how Dozzle can be set up to use Cloudflare Zero Trust for authentication.

yaml

```
services:
  dozzle:
    image: amir20/dozzle:latest
    networks:
      - net
    environment:
      DOZZLE_AUTH_PROVIDER: forward-proxy
      DOZZLE_AUTH_HEADER_USER: Cf-Access-Authenticated-User-Email
      DOZZLE_AUTH_HEADER_EMAIL: Cf-Access-Authenticated-User-Email
      DOZZLE_AUTH_HEADER_NAME: Cf-Access-Authenticated-User-Email
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
    expose:
      - 8080
    restart: unless-stopped
```

After running the Dozzle container, configure the Application in Cloudflare Zero Trust dashboard by following the guide.

Setting up Dozzle with Pocket ID

You must first setup a container to pass OpenID Connect authentication through your reverse proxy.

Below is an example using oauth2-proxy.

▶ ➡️ Click to expand oauth2-proxy example