

Quick Links



[Online Demo](#) 



[Download](#)



[FAQ](#)



[Latest Docs](#)



[Get Help](#)



[Screenshots](#)

This page in other versions: [Latest \(6.7\)](#) | [6.6](#) | [6.5](#) | [Development](#)

This document in other formats: [PDF](#) | [ePub](#) | [Tarball](#)

[pgAdmin 4 6.7 documentation](#) » [Getting Started](#) » [Deployment](#) » [Container Deployment](#)

[previous](#) | [next](#) | [index](#)

Contents

- [Container Deployment](#)
 - [PostgreSQL Utilities](#)
 - [Environment Variables](#)
 - [Mapped Files and Directories](#)
 - [Examples](#)
 - [Reverse Proxying](#)
 - [pgAdmin X-Forwarded-* Configuration](#)
 - [HTTP via Nginx](#)
 - [HTTPS via Nginx](#)
 - [Traefik](#)
- [Getting Started](#)
 - [Deployment](#)
 - [Login Dialog](#)
 - [Enabling two-factor authentication \(2FA\)](#)
 - [User Management Dialog](#)
 - [Change User Password Dialog](#)
 - [Lock/Restore Account](#)
 - [Enabling LDAP Authentication](#)
 - [Enabling Kerberos Authentication](#)
 - [Enabling OAUTH2 Authentication](#)
 - [Enabling Webserver Authentication](#)
 - [User Interface](#)
 - [Menu Bar](#)

- [Toolbar](#)
- [Tabbed Browser](#)
- [Tree Control](#)
- [Preferences Dialog](#)
- [Keyboard Shortcuts](#)
- [Search objects](#)
- [Connecting To A Server](#)
- [Managing Cluster Objects](#)
- [Managing Database Objects](#)
- [Creating or Modifying a Table](#)
- [Management Basics](#)
- [Backup and Restore](#)
- [Developer Tools](#)
- [pgAgent](#)
- [pgAdmin Project Contributions](#)
- [Release Notes](#)
- [Licence](#)

Container Deployment ¶

pgAdmin can be deployed in a container using the image at:

<https://hub.docker.com/r/dpage/pgadmin4/>

PostgreSQL Utilities ¶

The PostgreSQL utilities *pg_dump*, *pg_dumpall*, *pg_restore* and *psql* are included in the container to allow backups to be created and restored and other maintenance functions to be executed. Multiple versions are included in the following directories to allow use with different versions of the database server:

- PostgreSQL 10: */usr/local/pgsql-10*
- PostgreSQL 11: */usr/local/pgsql-11*
- PostgreSQL 12: */usr/local/pgsql-12*
- PostgreSQL 13: */usr/local/pgsql-13*
- PostgreSQL 14: */usr/local/pgsql-14*

The default binary paths set in the container are as follows:

```
DEFAULT_BINARY_PATHS = {  
    'pg-14': '/usr/local/pgsql-14',  
    'pg-13': '/usr/local/pgsql-13',  
    'pg-12': '/usr/local/pgsql-12',  
    'pg-11': '/usr/local/pgsql-11',  
    'pg-10': '/usr/local/pgsql-10'  
}
```

this may be changed in the [Preferences Dialog](#).

Environment Variables ¶

The container will accept the following variables at startup:

PGADMIN_DEFAULT_EMAIL

This is the email address used when setting up the initial administrator account to login to pgAdmin. This variable is required and must be set at launch time.

PGADMIN_DEFAULT_PASSWORD

This is the password used when setting up the initial administrator account to login to pgAdmin. This variable is required and must be set at launch time.

PGADMIN_DISABLE_POSTFIX

Default: <null>

If left unset, a Postfix server will be started to deliver password reset emails.

If set to any value, the Postfix server will not be started, and pgAdmin will need to be configured to use an external mail server using the `PGADMIN_CONFIG_` options below.

This option is useful if you're running in an environment that prevents the use of sudo to start Postfix, or if you wish to use an external mail server.

PGADMIN_ENABLE_TLS

Default: <null>

If left un-set, the container will listen on port 80 for connections in plain text. If set to any value, the container will listen on port 443 for TLS connections.

When TLS is enabled, a certificate and key must be provided. Typically these should be stored on the host file system and mounted from the container. The expected paths are `/certs/server.cert` and `/certs/server.key`

PGADMIN_LISTEN_ADDRESS

Default: [::]

Specify the local address that the servers listens on. The default should work for most users - in IPv4-only environments, this may need to be set to 0.0.0.0.

PGADMIN_LISTEN_PORT

Default: 80 or 443 (if TLS is enabled)

Allows the port that the server listens on to be set to a specific value rather than using the default.

PGADMIN_SERVER_JSON_FILE

Default: /pgadmin4/servers.json

Override the default file path for the server definition list. See the `/pgadmin4/servers.json` mapped file below for more information.

GUNICORN_ACCESS_LOGFILE

Default: - (stdout)

Specify an output file in which to store the Gunicorn access logs, instead of sending them to stdout.

GUNICORN_THREADS

Default: 25

Adjust the number of threads the Gunicorn server uses to handle incoming requests. This should typically be left as-is, except in highly loaded systems where it may be increased.

PGADMIN_CONFIG_*

This is a variable prefix that can be used to override any of the configuration options in pgAdmin's `config.py` file. Add the `PGADMIN_CONFIG_` prefix to any variable name from `config.py` and give the value in the format 'string value' for strings, True/False for booleans or 123 for numbers. See below for an example.

Settings are written to `/pgadmin4/config_distro.py` within the container, which is read after `/pgadmin4/config.py` and before `/pgadmin4/config_local.py`. Any settings given will therefore override anything in `config.py`, but can be overridden by settings in `config_local.py`.

Settings are only written to `/pgadmin4/config_distro.py` once, typically on first launch of the container. If `/pgadmin4/config_distro.py` contains one or more lines, then no changes are made; for example, if the container instance is restarted, or `/pgadmin4/config_distro.py` is mapped to a file on persistent storage (not recommended - use `/pgadmin4/config_local.py` instead)!

See [The config.py File](#) for more information on the available configuration settings.

Mapped Files and Directories ¶

The following files or directories can be mapped from the container onto the host machine to allow configuration to be customised and shared between instances.

Warning

Warning: pgAdmin runs as the *pgadmin* user (UID: 5050) in the *pgadmin* group (GID: 5050) in the container. You must ensure that all files are readable, and where necessary (e.g. the working/session directory) writeable for this user on the host machine. For example:

```
sudo chown -R 5050:5050 <host_directory>
```

On some filesystems that do not support extended attributes, it may not be possible to run pgAdmin without specifying a value for *PGADMIN_LISTEN_PORT* that is greater than 1024. In such cases, specify an alternate port when launching the container by adding the environment variable, for example:

```
-e 'PGADMIN_LISTEN_PORT=5050'
```

Don't forget to adjust any host-container port mapping accordingly.

/var/lib/pgadmin

This is the working directory in which pgAdmin stores session data, user files, configuration files, and its configuration database. Mapping this directory onto the host machine gives you an easy way to maintain configuration between invocations of the container.

/pgadmin4/config_local.py

This file can be used to override configuration settings in pgAdmin. Settings found in *config.py* can be overridden with deployment specific values if required. Settings in *config_local.py* will also override anything specified in the container environment through *PGADMIN_CONFIG_* prefixed variables.

/pgadmin4/servers.json

If this file is mapped, server definitions found in it will be loaded at launch time. This allows connection information to be pre-loaded into the instance of pgAdmin in the container. Note that server definitions are only loaded on first launch, i.e. when the configuration database is created, and not on subsequent launches using the same configuration database.

/certs/server.cert

If TLS is enabled, this file will be used as the servers TLS certificate.

/certs/server.key

If TLS is enabled, this file will be used as the key file for the servers TLS certificate.

Examples ¶

Run a simple container over port 80:

```
docker pull dpape/pgadmin4
docker run -p 80:80 \
  -e 'PGADMIN_DEFAULT_EMAIL=user@domain.com' \
  -e 'PGADMIN_DEFAULT_PASSWORD=SuperSecret' \
  -d dpape/pgadmin4
```

Run a simple container over port 80, setting some configuration options:

```
docker pull dpape/pgadmin4
docker run -p 80:80 \
  -e 'PGADMIN_DEFAULT_EMAIL=user@domain.com' \
  -e 'PGADMIN_DEFAULT_PASSWORD=SuperSecret' \
  -e 'PGADMIN_CONFIG_ENHANCED_COOKIE_PROTECTION=True' \
  -e 'PGADMIN_CONFIG_LOGIN_BANNER="Authorised users only!"' \
  -e 'PGADMIN_CONFIG_CONSOLE_LOG_LEVEL=10' \
  -d dpape/pgadmin4
```

Run a TLS secured container using a shared config/storage directory in */private/var/lib/pgadmin* on the host, and servers pre-loaded from */tmp/servers.json* on the host:

```
docker pull dpape/pgadmin4
docker run -p 443:443 \
  -v /private/var/lib/pgadmin:/var/lib/pgadmin \
  -v /path/to/certificate.cert:/certs/server.cert \
  -v /path/to/certificate.key:/certs/server.key \
  -v /tmp/servers.json:/pgadmin4/servers.json \
  -e 'PGADMIN_DEFAULT_EMAIL=user@domain.com' \
  -e 'PGADMIN_DEFAULT_PASSWORD=SuperSecret' \
  -e 'PGADMIN_ENABLE_TLS=True' \
  -d dpape/pgadmin4
```

Reverse Proxying ¶

Sometimes it's desirable to have users connect to pgAdmin through a reverse proxy rather than directly to the container it's running in. The following examples show how this can be achieved. With traditional reverse proxy servers such as [Nginx](#), pgAdmin is running in a container on the same host, with port 5050 on the host mapped to port 80 on the container, for example:

```
docker pull dpape/pgadmin4
docker run -p 5050:80 \
  -e "PGADMIN_DEFAULT_EMAIL=user@domain.com" \
  -e "PGADMIN_DEFAULT_PASSWORD=SuperSecret" \
  -d dpape/pgadmin4
```

pgAdmin X-Forwarded-* Configuration ¶

pgAdmin needs to understand how many proxies set each header so it knows what values to trust. The configuration parameters for the X-Forwarded-* options which are used for this purpose are shown below, along with their default values.

pgAdmin is configured by default to be able to run behind a reverse proxy even on a non-standard port and these config options don't normally need to be changed. If you're running an unusual configuration (such as multiple reverse proxies) you can adjust the configuration to suit.

```
# Number of values to trust for X-Forwarded-For
PROXY_X_FOR_COUNT = 1

# Number of values to trust for X-Forwarded-Proto.
PROXY_X_PROTO_COUNT = 0

# Number of values to trust for X-Forwarded-Host.
PROXY_X_HOST_COUNT = 0

# Number of values to trust for X-Forwarded-Port.
PROXY_X_PORT_COUNT = 1

# Number of values to trust for X-Forwarded-Prefix.
PROXY_X_PREFIX_COUNT = 0
```

HTTP via Nginx ¶

A configuration similar to the following can be used to create a simple HTTP reverse proxy listening for all hostnames with [Nginx](#):

```
server {
    listen 80;
    server_name _;

    location / {
        proxy_set_header Host $host;
        proxy_pass http://localhost:5050/;
        proxy_redirect off;
    }
}
```

If you wish to host pgAdmin under a subdirectory rather than on the root of the server, you must specify the location and set the *X-Script-Name* header which tells the pgAdmin container how to rewrite paths:

```
server {
    listen 80;
    server_name _;

    location /pgadmin4/ {
        proxy_set_header X-Script-Name /pgadmin4;
        proxy_set_header Host $host;
        proxy_pass http://localhost:5050/;
        proxy_redirect off;
    }
}
```

If Nginx is also running in a container, there is no need to map the pgAdmin port to the host, provided the two containers are running in the same Docker network. In such a configuration, the `proxy_pass` option would be changed to point to the pgAdmin container within the Docker network.

HTTPS via Nginx ¶

The following configuration can be used to serve pgAdmin over HTTPS to the user whilst the backend container is serving plain HTTP to the proxy server. In this configuration we not only set `X-Script-Name`, but also `X-Scheme` to tell the pgAdmin server to generate any URLs using the correct scheme. A redirect from HTTP to HTTPS is also included. The certificate and key paths may need to be adjusted as appropriate to the specific deployment:

```
server {
    listen 80;
    return 301 https://$host$request_uri;
}

server {
    listen 443;
    server_name _;

    ssl_certificate /etc/nginx/server.cert;
    ssl_certificate_key /etc/nginx/server.key;

    ssl on;
    ssl_session_cache builtin:1000 shared:SSL:10m;
    ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
    ssl_ciphers HIGH:!aNULL:!eNULL:!EXPORT:!CAMELLIA:!DES:!MD5:!PSK:!RC4;
    ssl_prefer_server_ciphers on;

    location /pgadmin4/ {
        proxy_set_header X-Script-Name /pgadmin4;
        proxy_set_header X-Scheme $scheme;
        proxy_set_header Host $host;
        proxy_pass http://localhost:5050/;
        proxy_redirect off;
    }
}
```

Traefik ¶

Configuring [Traefik](#) is straightforward for either HTTP or HTTPS when running pgAdmin in a container as it will automatically configure itself to serve content from containers that are running on the local machine, virtual hosting them at `<container_name>.<domain_name>`, where the domain name is that specified in the Traefik configuration. The container is typically launched per the example below:

```
docker pull dpape/pgadmin4
docker run --name "pgadmin4" \
    -e "PGADMIN_DEFAULT_EMAIL=user@domain.com" \
    -e "PGADMIN_DEFAULT_PASSWORD=SuperSecret" \
    -d dpape/pgadmin4
```

Note that the TCP/IP port has not been mapped to the host as it was in the Nginx example, and the container name has been set to a known value as it will be used as the hostname and may need to be added to the DNS zone file.

The following configuration will listen on ports 80 and 443, redirecting 80 to 443, using the default certificate shipped with Traefik. See the Traefik documentation for options to use certificates from LetsEncrypt or other issuers.

```
defaultEntryPoints = ["http", "https"]
```

```
[entryPoints]
[entryPoints.http]
  address = ":80"
  [entryPoints.http.redirect]
    entryPoint = "https"
[entryPoints.https]
  address = ":443"
  [entryPoints.https.tls]
```

```
[docker]
domain = "domain_name"
watch = true
```

If you wish to host pgAdmin under a subdirectory using Traefik, the configuration changes are typically made to the way the container is launched and not to Traefik itself. For example, to host pgAdmin under */pgadmin4/* instead of at the root directory, the Traefik configuration above may be used if the container is launched like this:

```
docker pull dpape/pgadmin4
docker run --name "pgadmin4" \
  -e "PGADMIN_DEFAULT_EMAIL=user@domain.com" \
  -e "PGADMIN_DEFAULT_PASSWORD=SuperSecret" \
  -e "SCRIPT_NAME=/pgadmin4" \
  -l "traefik.frontend.rule=PathPrefix:/pgadmin4" \
  -d dpape/pgadmin4
```

The *SCRIPT_NAME* environment variable has been set to tell the container it is being hosted under a subdirectory (in the same way as the *X-Script-Name* header is used with Nginx), and a label has been added to tell Traefik to route requests under the subdirectory to this container.