Quick Links

Online Demo

Download

FAQ

Latest Docs

Ask a Question

Report a Bug

Screenshots

This page in other versions: Latest (4.19) | 4.18 | 4.17 | Development

pgAdmin 4 4.19 documentation » Getting Started » Deployment » previous | next | index

Warning: This documentation is for a pre-release version of pgAdmin 4

Contents

The config.py File ¶

There are multiple configuration files that are read at startup by pgAdmin. These are as follows:

- `config.py`: This is the main configuration file, and should not be modified. It can be used as a reference for configuration settings, that may be overridden in one of the following files.
- `config_distro.py`: This file is read after `config.py` and is intended for packagers to change any settings that are required for their pgAdmin distribution. This may typically include certain paths and file locations. This file is optional, and may be created by packagers in the same directory as `config.py` if needed.
- `config_local.py`: This file is read after `config_distro.py` and is intended for end users to change any default or packaging specific settings that they may wish to adjust to meet local preferences or standards.This file is optional, and may be created by users in the same directory as `config.py` if needed.

Note

If the SERVER_MODE setting is changed in `config_distro.py` or `config_local.py`, you will most likely need to re-set the LOG_FILE, SQLITE_PATH, SESSION_DB_PATH and STORAGE_DIR values as well as they will have been set based on the default configuration or overridden by the runtime.

The default `config.py` file is shown below for reference:

```
# -*- coding: utf-8 -*-

##########################################################

#
# pgAdmin 4 - PostgreSQL Tools
#
# Copyright (C) 2013 - 2020, The pgAdmin Development Team
# This software is released under the PostgreSQL Licence
#
# config.py - Core application configuration settings
#
##########################################################


import logging
import os
import sys
import json

if sys.version_info[0] >= 3:
    import builtins
else:
    import __builtin__ as builtins

# We need to include the root directory in sys.path to
ensure that we can
# find everything we need when running in the standalone
runtime.
root = os.path.dirname(os.path.realpath(__file__))
if sys.path[0] != root:
    sys.path.insert(0, root)

from pgadmin.utils import env, IS_PY2, IS_WIN,
fs_short_path

##########################################################

# Application settings
##########################################################


# Name of the application to display in the UI
APP_NAME = 'pgAdmin 4'
APP_ICON = 'pg-icon'

##########################################################

# Application settings
##########################################################


# NOTE!!!
# If you change any of APP_RELEASE, APP_REVISION or
APP_SUFFIX, then you
# must also change APP_VERSION_INT to match.
#
```

```
# Any changes made here must also be made in runtime/pgAdmin4.pro and
# runtime/Info.plist
#

# Application version number components
APP_RELEASE = 4
APP_REVISION = 19

# Application version suffix, e.g. 'beta1', 'dev'. Usually an empty string
# for GA releases.
APP_SUFFIX = ''

# Numeric application version for upgrade checks. Should be in the format:
# [X]XYYZZ, where X is the release version, Y is the revision, with a leading
# zero if needed, and Z represents the suffix, with a leading zero if needed
APP_VERSION_INT = 41900

# DO NOT CHANGE!
# The application version string, constructed from the components
if not APP_SUFFIX:
    APP_VERSION = '%s.%s' % (APP_RELEASE, APP_REVISION)
else:
    APP_VERSION = '%s.%s-%s' % (APP_RELEASE, APP_REVISION, APP_SUFFIX)

# Copyright string for display in the app
# Any changes made here must also be made in runtime/pgAdmin4.pro
APP_COPYRIGHT = 'Copyright (C) 2013 - 2020, The pgAdmin Development Team'

##########################################################

# Misc stuff
##########################################################


# Path to the online help.
HELP_PATH = '../../../docs/en_US/_build/html/'

# Languages we support in the UI
LANGUAGES = {
    'en': 'English',
    'zh': 'Chinese (Simplified)',
    'cs': 'Czech',
    'fr': 'French',
    'de': 'German',
    'it': 'Italian',
    'ja': 'Japanese',
    'ko': 'Korean',
    'pl': 'Polish',
    'ru': 'Russian',
    'es': 'Spanish',
}

# DO NOT CHANGE UNLESS YOU KNOW WHAT YOU ARE DOING!
# List of modules to skip when dynamically loading
MODULE_BLACKLIST = ['test']

# DO NOT CHANGE UNLESS YOU KNOW WHAT YOU ARE DOING!
# List of treeview browser nodes to skip when dynamically loading
NODE_BLACKLIST = []

#########################################################

# Server settings
#########################################################


# The server mode determines whether or not we're running on a web server
# requiring user authentication, or desktop mode which uses an automatic
# default login.
#
# DO NOT DISABLE SERVER MODE IF RUNNING ON A WEBSERVER!!
#
# We only set SERVER_MODE if it's not already set. That's to allow the
# runtime to force it to False.
#
# NOTE: If you change the value of SERVER_MODE in an included config file,
#       you may also need to redefine any values below that are derived
#       from it, notably various paths such as LOG_FILE and anything
#       using DATA_DIR.

if (not hasattr(builtins, 'SERVER_MODE')) or builtins.SERVER_MODE is None:
    SERVER_MODE = True
else:
    SERVER_MODE = builtins.SERVER_MODE

# HTTP headers to search for CSRF token when it is not provided in the form.
# Default is ['X-CSRFToken', 'X-CSRF-Token']
WTF_CSRF_HEADERS = ['X-pgA-CSRFToken']

# User ID (email address) to use for the default user in desktop mode.
# The default should be fine here, as it's not exposed in the app.
DESKTOP_USER = 'pgadmin4@pgadmin.org'

# This option allows the user to host the application on a LAN
# Default hosting is on localhost
```

```
(DEFAULT_SERVER='localhost').
# To host pgAdmin4 over LAN set DEFAULT_SERVER='0.0.0.0'
(or a specific
# adaptor address.
#
# NOTE: This is NOT recommended for production use, only
for debugging
# or testing. Production installations should be run as a
WSGI application
# behind Apache HTTPD.
DEFAULT_SERVER = '127.0.0.1'

# The default port on which the app server will listen if
not set in the
# environment by the runtime
DEFAULT_SERVER_PORT = 5050

# Enable X-Frame-Option protection.
# Set to one of "SAMEORIGIN", "ALLOW-FROM origin" or ""
to disable.
# Note that "DENY" is NOT supported (and will be silently
ignored).
# See https://tools.ietf.org/html/rfc7034 for more info.
X_FRAME_OPTIONS = "SAMEORIGIN"

# Hashing algorithm used for password storage
SECURITY_PASSWORD_HASH = 'pbkdf2_sha512'

# Reverse Proxy parameters
# You must tell the middleware how many proxies set each
header
# so it knows what values to trust.
# See https://tinyurl.com/yyg7r9av
# for more information.

# 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 = 1

# 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

# NOTE: CSRF_SESSION_KEY, SECRET_KEY and
SECURITY_PASSWORD_SALT are no
#       longer part of the main configuration, but are
stored in the
#       configuration databases 'keys' table and are
auto-generated.

# COMPRESSION
```

```
COMPRESS_MIMETYPES = [
    'text/html', 'text/css', 'text/xml',
'application/json',
    'application/javascript'
]
COMPRESS_LEVEL = 9
COMPRESS_MIN_SIZE = 500

# Set the cache control max age for static files in flask
to 1 year
SEND_FILE_MAX_AGE_DEFAULT = 31556952

# This will be added to static urls as url parameter with
value as
# APP_VERSION_INT for cache busting on version upgrade.
If the value is set as
# None or empty string then it will not be added.
# eg - http:localhost:5050/pgadmin.css?intver=3.13
APP_VERSION_PARAM = 'ver'

# Add the internal version param to below extensions only
APP_VERSION_EXTN = ('.css', '.js', '.html', '.svg',
'.png', '.gif', '.ico')

# Data directory for storage of config settings etc. This
shouldn't normally
# need to be changed - it's here as various other
settings depend on it.
# On Windows, we always store data in %APPDATA%\pgAdmin.
On other platforms,
# if we're in server mode we use /var/lib/pgadmin,
otherwise ~/.pgadmin
if IS_WIN:
    # Use the short path on windows
    DATA_DIR = os.path.realpath(
        os.path.join(fs_short_path(env('APPDATA')),
u"pgAdmin")
    )
else:
    if SERVER_MODE:
        DATA_DIR = '/var/lib/pgadmin'
    else:
        DATA_DIR =
os.path.realpath(os.path.expanduser(u'~/.pgadmin/'))

# An optional login banner to show security
warnings/disclaimers etc. at
# login and password recovery etc. HTML may be included
for basic formatting,
# For example:
# LOGIN_BANNER = "<h4>Authorised Users Only!</h4>" \
#                  "Unauthorised use is strictly
forbidden."
LOGIN_BANNER = ""

##########################################################

# Log settings
```

```
##########################################################


# Debug mode?
DEBUG = False

# Application log level - one of:
#   CRITICAL 50
#   ERROR    40
#   WARNING  30
#   SQL      25
#   INFO     20
#   DEBUG    10
#   NOTSET    0
CONSOLE_LOG_LEVEL = logging.WARNING
FILE_LOG_LEVEL = logging.WARNING

# Log format.
CONSOLE_LOG_FORMAT = '%(asctime)s: %(levelname)s\t%
(name)s:\t%(message)s'
FILE_LOG_FORMAT = '%(asctime)s: %(levelname)s\t%
(name)s:\t%(message)s'

# Log file name. This goes in the data directory, except
on non-Windows
# platforms in server mode.
if SERVER_MODE and not IS_WIN:
    LOG_FILE = '/var/log/pgadmin/pgadmin4.log'
else:
    LOG_FILE = os.path.join(DATA_DIR, 'pgadmin4.log')

##########################################################

# Server Connection Driver Settings
##########################################################


# The default driver used for making connection with
PostgreSQL
PG_DEFAULT_DRIVER = 'psycopg2'

# Maximum allowed idle time in minutes before which
releasing the connection
# for the particular session. (in minutes)
MAX_SESSION_IDLE_TIME = 60

##########################################################

# User account and settings storage
##########################################################


# The default path to the SQLite database used to store
user accounts and
# settings. This default places the file in the same
directory as this
# config file, but generates an absolute path for use
througout the app.
```

```
SQLITE_PATH = env('SQLITE_PATH') or
os.path.join(DATA_DIR, 'pgadmin4.db')

# SQLITE_TIMEOUT will define how long to wait before
throwing the error -
# OperationError due to database lock. On slower system,
you may need to change
# this to some higher value.
# (Default: 500 milliseconds)
SQLITE_TIMEOUT = 500

# Allow database connection passwords to be saved if the
user chooses.
# Set to False to disable password saving.
ALLOW_SAVE_PASSWORD = True

# Maximum number of history queries stored per
user/server/database
MAX_QUERY_HIST_STORED = 20

##########################################################

# Server-side session storage path
#
# SESSION_DB_PATH (Default: $HOME/.pgadmin4/sessions)
##########################################################

#
# We use SQLite for server-side session storage. There
will be one
# SQLite database object per session created.
#
# Specify the path used to store your session objects.
#
# If the specified directory does not exist, the setup
script will create
# it with permission mode 700 to keep the session
database secure.
#
# On certain systems, you can use shared memory (tmpfs)
for maximum
# scalability, for example, on Ubuntu:
#
# SESSION_DB_PATH = '/run/shm/pgAdmin4_session'
#
##########################################################

SESSION_DB_PATH = os.path.join(DATA_DIR, 'sessions')

SESSION_COOKIE_NAME = 'pga4_session'

##########################################################

# Mail server settings
##########################################################


# These settings are used when running in web server mode
```

```
for confirming
# and resetting passwords etc.
# See: http://pythonhosted.org/Flask-Mail/ for more info
MAIL_SERVER = 'localhost'
MAIL_PORT = 25
MAIL_USE_SSL = False
MAIL_USE_TLS = False
MAIL_USERNAME = ''
MAIL_PASSWORD = ''
MAIL_DEBUG = False

# Flask-Security overrides Flask-Mail's
MAIL_DEFAULT_SENDER setting, so
# that should be set as such:
SECURITY_EMAIL_SENDER = 'no-reply@localhost'

##########################################################

# Mail content settings
##########################################################


# These settings define the content of password reset
emails
SECURITY_EMAIL_SUBJECT_PASSWORD_RESET = "Password reset
instructions for %s" \
                                        % APP_NAME
SECURITY_EMAIL_SUBJECT_PASSWORD_NOTICE = "Your %s
password has been reset" \
                                         % APP_NAME
SECURITY_EMAIL_SUBJECT_PASSWORD_CHANGE_NOTICE = \
    "Your password for %s has been changed" % APP_NAME

##########################################################

# Upgrade checks
##########################################################


# Check for new versions of the application?
UPGRADE_CHECK_ENABLED = True

# Where should we get the data from?
UPGRADE_CHECK_URL =
'https://www.pgadmin.org/versions.json'

# What key should we look at in the upgrade data file?
UPGRADE_CHECK_KEY = 'pgadmin4'

# Which CA file should we use?
# Default to cacert.pem in the same directory as
config.py et al.
CA_FILE =
os.path.join(os.path.dirname(os.path.realpath(__file__)),
                       "cacert.pem")

##########################################################

```

```
# Storage Manager storage url config settings
# If user sets STORAGE_DIR to empty it will show all
volumes if platform
# is Windows, '/' if it is Linux, Mac or any other unix
type system.

# For example:
# 1. STORAGE_DIR = get_drive("C") or get_drive() # return
C:/ by default
# where C can be any drive character such as "D", "E",
"G" etc
# 2. Set path manually like
# STORAGE_DIR = "/path/to/directory/"
#########################################################

STORAGE_DIR = os.path.join(DATA_DIR, 'storage')

#########################################################

# Default locations for binary utilities (pg_dump,
pg_restore etc)
#
# These are intentionally left empty in the main config
file, but are
# expected to be overridden by packagers in
config_distro.py.
#
# A default location can be specified for each database
driver ID, in
# a dictionary. Either an absolute or relative path can
be specified.
# In cases where it may be difficult to know what the
working directory
# is, "$DIR" can be specified. This will be replaced with
the path to the
# top-level pgAdmin4.py file. For example, on macOS we
might use:
#
# $DIR/../../SharedSupport
#
#########################################################

DEFAULT_BINARY_PATHS = {
    "pg": "",
    "ppas": "",
    "gpdb": ""
}

#########################################################

# Test settings - used primarily by the regression suite,
not for users
#########################################################


# The default path for SQLite database for testing
TEST_SQLITE_PATH = os.path.join(DATA_DIR,
'test_pgadmin4.db')
```

```
#########################################################

# Allows flask application to response to the each
request asynchronously
#########################################################

THREADED_MODE = True

#########################################################

# Do not allow SQLALCHEMY to track modification as it is
going to be
# deprecated in future
#########################################################

SQLALCHEMY_TRACK_MODIFICATIONS = False

#########################################################

# Number of records to fetch in one batch in query tool
when query result
# set is large.
#########################################################

ON_DEMAND_RECORD_COUNT = 1000

#########################################################

# Allow users to display Gravatar image for their
username in Server mode
#########################################################

SHOW_GRAVATAR_IMAGE = True

#########################################################

# Set cookie path
#########################################################

COOKIE_DEFAULT_PATH = '/'
COOKIE_DEFAULT_DOMAIN = None
SESSION_COOKIE_DOMAIN = None
SESSION_COOKIE_SAMESITE = 'Lax'

#########################################################

# Skip storing session in files and cache for specific
paths
#########################################################

SESSION_SKIP_PATHS = [
    '/misc/ping'
]

#########################################################

# Session expiration support
```

```
##########################################################

# SESSION_EXPIRATION_TIME is the interval in Days. Session will be
# expire after the specified number of *days*.
SESSION_EXPIRATION_TIME = 1

# CHECK_SESSION_FILES_INTERVAL is interval in Hours. Application will check
# the session files for cleanup after specified number of *hours*.
CHECK_SESSION_FILES_INTERVAL = 24

# USER_INACTIVITY_TIMEOUT is interval in Seconds. If the pgAdmin screen is left
# unattended for <USER_INACTIVITY_TIMEOUT> seconds then the user will
# be logged out. When set to 0, the timeout will be disabled.
# If pgAdmin doesn't detect any activity in the time specified (in seconds),
# the user will be forcibly logged out from pgAdmin. Set to zero to disable
# the timeout.
# Note: This is applicable only for SERVER_MODE=True.
USER_INACTIVITY_TIMEOUT = 0

# OVERRIDE_USER_INACTIVITY_TIMEOUT when set to True will override
# USER_INACTIVITY_TIMEOUT when long running queries in the Query Tool
# or Debugger are running. When the queries complete, the inactivity timer
# will restart in this case. If set to False, user inactivity may cause
# transactions or in-process debugging sessions to be aborted.
OVERRIDE_USER_INACTIVITY_TIMEOUT = True

##########################################################

# SSH Tunneling supports only for Python 2.7 and 3.4+
##########################################################

SUPPORT_SSH_TUNNEL = True
# Allow SSH Tunnel passwords to be saved if the user chooses.
# Set to False to disable password saving.
ALLOW_SAVE_TUNNEL_PASSWORD = False

##########################################################

# Master password is used to encrypt/decrypt saved server passwords
# Applicable for desktop mode only
##########################################################

MASTER_PASSWORD_REQUIRED = True
```

```
##########################################################

# Allows pgAdmin4 to create session cookies based on IP address, so even
# if a cookie is stolen, the attacker will not be able to connect to the
# server using that stolen cookie.
# Note: This can cause problems when the server is deployed in dynamic IP
# address hosting environments, such as Kubernetes or behind load
# balancers. In such cases, this option should be set to False.
##########################################################

ENHANCED_COOKIE_PROTECTION = True

##########################################################

# Local config settings
##########################################################


# Load distribution-specific config overrides
try:
    from config_distro import *
except ImportError:
    pass

# Load local config overrides
try:
    from config_local import *
except ImportError:
    pass


# SUPPORT_SSH_TUNNEL can be override in local config file and if that
# setting is False in local config then we should not check the Python version.
if (SUPPORT_SSH_TUNNEL is True and
    ((sys.version_info[0] == 2 and sys.version_info[1] < 7) or
     (sys.version_info[0] == 3 and sys.version_info[1] < 4))):
    SUPPORT_SSH_TUNNEL = False
    ALLOW_SAVE_TUNNEL_PASSWORD = False

# Disable USER_INACTIVITY_TIMEOUT when SERVER_MODE=False
if not SERVER_MODE:
    USER_INACTIVITY_TIMEOUT = 0
```