

# Install PowerDNS on Ubuntu

By  
Ahmad Ropai

# Powerdns

---

## Mengapa Menggunakan PowerDNS?

PowerDNS menyediakan dua solusi server nama:

- Server Resmi, yang menggunakan database untuk menyelesaikan kueri tentang domain.
- Recursor, yang berkonsultasi dengan server otoritatif lain untuk menyelesaikan kueri.

Server nama lain menggabungkan kedua fungsi tersebut secara otomatis. PowerDNS menawarkannya secara terpisah, dan memungkinkan perpaduan dua solusi secara mulus untuk pengaturan modular.

## Install Powerdns di Ubuntu 18.04, 20.04, & 22.04

---

### Step A: Install and Configure MariaDB Server

#### 1. Update and upgrade system packages:

```
sudo apt update && sudo apt upgrade
```

#### 2. Install the MariaDB server and client with:

```
sudo apt install mariadb-server mariadb-client
```

#### 3. Connect to MariaDB with:

```
# login to mysql  
sudo mysql
```

#### 4. Create a database for the PowerDNS nameserver:

```
# create database with name powerdns  
create database powerdns;
```

#### 5. Grant all privileges to the pda user and provide the user password:

```
#create user  
CREATE USER 'user'@localhost IDENTIFIED BY 'password1';
```

```
# give grant access
grant all privileges on powerdns.* TO 'user'@'localhost' identified by
'YOUR_PASSWORD_HERE';
# commit
flush privileges;

#show all user
SELECT User FROM mysql.user;
```

 image mysql powerdns setup

## 6. Connect to the database:

```
# use database powerdns
use powerdns;
```

## 7. Use the following SQL queries to create tables for the pda database:

```
CREATE TABLE domains (
  id                INT AUTO_INCREMENT,
  name              VARCHAR(255) NOT NULL,
  master            VARCHAR(128) DEFAULT NULL,
  last_check        INT DEFAULT NULL,
  type              VARCHAR(6) NOT NULL,
  notified_serial    INT UNSIGNED DEFAULT NULL,
  account           VARCHAR(40) CHARACTER SET 'utf8' DEFAULT NULL,
  PRIMARY KEY (id)
) Engine=InnoDB CHARACTER SET 'latin1';
CREATE UNIQUE INDEX name_index ON domains(name);
CREATE TABLE records (
  id                BIGINT AUTO_INCREMENT,
  domain_id          INT DEFAULT NULL,
  name              VARCHAR(255) DEFAULT NULL,
  type              VARCHAR(10) DEFAULT NULL,
  content            VARCHAR(64000) DEFAULT NULL,
  ttl               INT DEFAULT NULL,
  prio              INT DEFAULT NULL,
  change_date        INT DEFAULT NULL,
  disabled           TINYINT(1) DEFAULT 0,
  ordername          VARCHAR(255) BINARY DEFAULT NULL,
  auth              TINYINT(1) DEFAULT 1,
  PRIMARY KEY (id)
) Engine=InnoDB CHARACTER SET 'latin1';
CREATE INDEX nametype_index ON records(name,type);
CREATE INDEX domain_id ON records(domain_id);
CREATE INDEX ordername ON records (ordername);
CREATE TABLE supermasters (
  ip                VARCHAR(64) NOT NULL,
  nameserver         VARCHAR(255) NOT NULL,
```

```
account          VARCHAR(40) CHARACTER SET 'utf8' NOT NULL,
PRIMARY KEY (ip, nameserver)
) Engine=InnoDB CHARACTER SET 'latin1';
CREATE TABLE comments (
  id              INT AUTO_INCREMENT,
  domain_id       INT NOT NULL,
  name            VARCHAR(255) NOT NULL,
  type            VARCHAR(10) NOT NULL,
  modified_at     INT NOT NULL,
  account         VARCHAR(40) CHARACTER SET 'utf8' DEFAULT NULL,
  comment         TEXT CHARACTER SET 'utf8' NOT NULL,
  PRIMARY KEY (id)
) Engine=InnoDB CHARACTER SET 'latin1';
CREATE INDEX comments_name_type_idx ON comments (name, type);
CREATE INDEX comments_order_idx ON comments (domain_id, modified_at);
CREATE TABLE domainmetadata (
  id              INT AUTO_INCREMENT,
  domain_id       INT NOT NULL,
  kind            VARCHAR(32),
  content         TEXT,
  PRIMARY KEY (id)
) Engine=InnoDB CHARACTER SET 'latin1';
CREATE INDEX domainmetadata_idx ON domainmetadata (domain_id, kind);
CREATE TABLE cryptokeys (
  id              INT AUTO_INCREMENT,
  domain_id       INT NOT NULL,
  flags           INT NOT NULL,
  active          BOOL,
  content         TEXT,
  PRIMARY KEY(id)
) Engine=InnoDB CHARACTER SET 'latin1';
CREATE INDEX domainidindex ON cryptokeys(domain_id);
CREATE TABLE tsigkeys (
  id              INT AUTO_INCREMENT,
  name            VARCHAR(255),
  algorithm       VARCHAR(50),
  secret         VARCHAR(255),
  PRIMARY KEY (id)
) Engine=InnoDB CHARACTER SET 'latin1';
CREATE UNIQUE INDEX namealgoindex ON tsigkeys(name, algorithm);
```

8. Confirm the tables have been created with:

```
show tables;

# And then exit mysql
exit;
```

 image mysql powerdns tables

## Step B: Install PowerDNS


1. Switch to the root user:

```
sudo su -
```

2. Disable the systemd-resolved service with:

Layanan yang diselesaikan dengan systemd menyediakan resolusi nama untuk aplikasi lokal. PowerDNS menggunakan layanannya sendiri untuk resolusi nama.

```
systemctl disable --now systemd-resolved
```

image disable systemd resolved

3. Delete the system service configuration file with:

```
# delete default resolver  
rm -rf /etc/resolv.conf
```

4. Create the new resolv.conf file:

```
# with google nameserver resolver  
echo "nameserver 8.8.8.8" | sudo tee /etc/resolv.conf
```

5. Install the PowerDNS server and database backend packages with:

```
apt-get install pdns-server pdns-backend-mysql -y
```

## Step C: Configure PowerDNS

Configure the local PowerDNS file to connect to the database:

1. Open the configuration file for editing:

```
nano /etc/powerdns/pdns.d/pdns.local.gmysql.conf
```

2. Add the following information to the file:

```
# MySQL Configuration
#
# Launch gmysql backend
launch+=gmysql

# gmysql parameters
gmysql-host=127.0.0.1
gmysql-port=3306
gmysql-dbname=pda
gmysql-user=pda
gmysql-password=YOUR_PASSWORD_HERE
gmysql-dnssec=yes
# gmysql-socket=
```

Notes: Exchange the database name, user, and password with the correct parameters if using different ones. Save and close the file.

### 3. Change the file permissions:

```
chmod 777 /etc/powerdns/pdns.d/pdns.local.gmysql.conf
```

### 4. Stop the pdns service:

```
systemctl stop pdns
```

### 5. Test the connection to the database:

```
pdns_server --daemon=no --guardian=no --loglevel=9
```

 image check connection to mysql

The output shows a successful connection. Press CTRL+C to exit the test.


### 6. Start the service:

```
systemctl start pdns
```

### 7. Check the connection with the ss command:

```
ss -alnp4 | grep pdns
```

Verify the TCP/UDP port 53 is open and in LISTEN/UCONN state.

 image check powerdns port

## Step D: Install PowerDNS Admin Dependencies

The PowerDNS Admin helps manage PowerDNS through a web interface. To install the dashboard locally, do the following:

1. Install the Python development package:

```
apt install python3-dev
```

2. Install dependencies:

```
apt install -y git libmysqlclient-dev libsasl2-dev libldap2-dev libssl-dev  
libxml2-dev libxslt1-dev libxmlsec1-dev libffi-dev pkg-config apt-  
transport-https python3-venv build-essential curl
```

3. Fetch the Node.js setup:

```
curl -sL https://deb.nodesource.com/setup_14.x | sudo bash -
```

4. Install Node.js with:

```
apt install -y nodejs
```

5. Next, install the Yarn package manager. Fetch the Yarn public key and add it to apt:

Yarn helps build the asset files.

```
curl -sS https://dl.yarnpkg.com/debian/pubkey.gpg | apt-key add -
```

6. Add Yarn to the list of sources:

```
echo "deb https://dl.yarnpkg.com/debian/ stable main" | tee  
/etc/apt/sources.list.d/yarn.list
```

7. Update the apt sources list:

```
apt update -y
```

8. Install Yarn with:

```
apt install yarn -y
```

9. Clone the PowerDNS Admin Git repository to /opt/web/powerdns-admin:

```
git clone https://github.com/ngoduykhanh/PowerDNS-Admin.git  
/opt/web/powerdns-admin
```

If using a different directory, exchange the destination directory in the command and in all subsequent appearances.

10. Navigate to the cloned Git directory:

```
cd /opt/web/powerdns-admin
```

11. Create a Python virtual environment:

```
python3 -mvenv ./venv
```

12. Activate the virtual environment with:

```
source ./venv/bin/activate
```

13. Upgrade pip to the latest version:

The pip package manager helps install additional Python requirements.



```
pip install --upgrade pip
```

14. Install the requirements from the requirements.txt file:

```
pip install -r requirements.txt
```

After installing all the requirements, the PowerDNS Admin requires additional configuration before running.

if error pg\_config executable not found add this

```
sudo apt install libpq-dev
```

## Step E: Configure and Run PowerDNS Admin

To configure and start PowerDNS Admin on a local instance, do the following:

1. Use the cp command to copy the example development.py Python file to production.py:

```
cp /opt/web/powerdns-admin/configs/development.py /opt/web/powerdns-admin/configs/production.py
```

2. Open the production.py file for editing:

```
nano /opt/web/powerdns-admin/configs/production.py
```

3. Edit the following lines:

```
#import urllib.parse

SECRET_KEY = 'e951e5a1f4b94151b360f47edf596dd2'

SQLA_DB_PASSWORD = 'changeme'
```

4. Uncomment the library import, provide a randomly generated secret key, and provide the correct database password.

```
import urllib.parse

SECRET_KEY = '\x19\xc7\xd8\xa7$\xb6P*\xc6\xb8\xa1E\x90P\x12\x95'

#change
### DATABASE CONFIG
SQLA_DB_USER = 'alfa'
SQLA_DB_PASSWORD = 'YOUR_PASSWORD_HERE'
SQLA_DB_HOST = '127.0.0.1'
SQLA_DB_NAME = 'powerdns'
SQLALCHEMY_TRACK_MODIFICATIONS = True
```

Save and close the file.

5. Export the production app configuration variable:

```
export FLASK_CONF=./configs/production.py
```

6. Export the flask application variable:

```
export FLASK_APP=powerdnsadmin/__init__.py
```

7. Upgrade the database schema:

```
flask db upgrade
```

8. Install project dependencies:

```
yarn install --pure-lockfile
```

9. Build flask app assets:

```
flask assets build
```

Wait for the build to complete.

10. Run the application with:

```
./run.py
```

Leave the application running.

11. The application currently runs on localhost on port 9191. Visit the following address:

```
# go to browser and access below  
http://localhost:9191
```

The login screen for PowerDNS Admin shows. Currently, there are no users, and the first user you register shall be the administrator account.

12. In the terminal, exit the virtual environment and log out of the root user with:

```
exit  
  
#or  
# exit from venv python  
deactivate
```

The terminal returns to a regular state.

## Step F: Create PowerDNS Admin Service

Configure PowerDNS Admin to run on startup:

1. Create a systemd service file for PowerDNS Admin:

```
sudo nano /etc/systemd/system/powerdns-admin.service
```

2. Add the following contents:

```
[Unit]  
Description=PowerDNS-Admin  
Requires=powerdns-admin.socket  
After=network.target  
[Service]  
User=root  
Group=root  
PIDFile=/run/powerdns-admin/pid  
WorkingDirectory=/opt/web/powerdns-admin  
ExecStartPre=/bin/bash -c '$$(mkdir -p /run/powerdns-admin/)'
```

```
ExecStart=/opt/web/powerdns-admin/venv/bin/gunicorn --pid /run/powerdns-admin/pid --bind unix:/run/powerdns-admin/socket  
'powerdnsadmin:create_app()'  
ExecReload=/bin/kill -s HUP $MAINPID  
ExecStop=/bin/kill -s TERM $MAINPID  
PrivateTmp=true  
[Install]  
WantedBy=multi-user.target
```

### 3. Create a unit file:

```
sudo systemctl edit --force powerdns-admin.service
```

### 4. Append the following:

```
[Service]  
Environment="FLASK_CONF=../configs/production.py"
```

### 5. Create a socket file:

```
sudo nano /etc/systemd/system/powerdns-admin.socket
```

### 6. Insert the following information:

```
[Unit]  
Description=PowerDNS-Admin socket  
  
[Socket]  
ListenStream=/run/powerdns-admin/socket  
  
[Install]  
WantedBy=sockets.target
```

### 7. Create an environment file:

```
sudo nano /etc/tmpfiles.d/powerdns-admin.conf
```

### 8. Add the following information:

```
d /run/powerdns-admin 0755 pdns pdns -
```

#### 9. Reload the daemon:

```
sudo systemctl daemon-reload
```

#### 10. Start and enable the service and socket:

```
sudo systemctl start powerdns-admin.service powerdns-admin.socket
```

```
sudo systemctl enable powerdns-admin.service powerdns-admin.socket
```

#### 11. Check the status with:

```
sudo systemctl status powerdns-admin.service powerdns-admin.socket
```

The services show as running without any errors.

 image check service

## Step G: Install and Configure Nginx

To configure PowerDNS Admin to run on Nginx, do the following:

#### 1. Install Nginx with:

```
sudo apt install nginx -y
```

#### 2. Edit the Nginx configuration file:

```
sudo nano /etc/nginx/conf.d/pdns-admin.conf
```

#### 3. Add the following information:

```
server {
    listen *:80;
    server_name                localhost;

    index                      index.html index.htm index.php;
    root                      /opt/web/powerdns-admin;
    access_log                 /var/log/nginx/powerdns-admin.local.access.log
combined;
    error_log                  /var/log/nginx/powerdns-admin.local.error.log;

    client_max_body_size       10m;
    client_body_buffer_size    128k;
    proxy_redirect             off;
    proxy_connect_timeout      90;
    proxy_send_timeout         90;
    proxy_read_timeout         90;
    proxy_buffers               32 4k;
    proxy_buffer_size          8k;
    proxy_set_header           Host $host;
    proxy_set_header           X-Real-IP $remote_addr;
    proxy_set_header           X-Forwarded-For
$proxy_add_x_forwarded_for;
    proxy_headers_hash_bucket_size 64;

    location ~ ^/static/ {
        include /etc/nginx/mime.types;
        root /opt/web/powerdns-admin/powerdnsadmin;

        location ~* \.(jpg|jpeg|png|gif)$ {
            expires 365d;
        }

        location ~* ^.+.(css|js)$ {
            expires 7d;
        }
    }

    location / {
        proxy_pass             http://unix:/run/powerdns-admin/socket;
        proxy_read_timeout     120;
        proxy_connect_timeout  120;
        proxy_redirect          off;
    }
}
```

If using a different server name, change localhost to your server address.

#### 4. Confirm the file has no syntax errors:

```
nginx -t
```

5. Change the ownership of powerdns-admin to www-data:

```
sudo chown -R www-data:www-data /opt/web/powerdns-admin
```

6. Restart the Nginx service:

```
sudo systemctl restart nginx
```

7. Access the PowerDNS admin page through the browser:

```
localhost
```

If linking to a different address, use the address provided in the Nginx configuration file.

## Step H: Configure PowerDNS API

To configure the PowerDNS API, do the following:

1. Log into the PowerDNS Admin via the browser. If running for the first time, create a new user first. The first user is automatically the administrator.
2. Open the API Keys tab on the left menu.  
  
PowerDNS Admin page menu API keys
3. Click the Add Key+ button.
4. The Role field defaults to the Administrator user. Add an optional description for the key.
5. Click Create Key to generate an API key.
6. A popup window prints the key. Copy the key and press Confirm to continue.
7. Navigate to the Dashboard page.
8. Enter the domain and API key. Save the changes.

image powerdns key

image powerdns admin

9. Enable the API in the PowerDNS configuration. Open the configuration file in the terminal:

```
nano /etc/powerdns/pdns.conf
```

10. Uncomment and change the following lines:

```
api=yes  
api-key=yoursecretekey  
webserver=yes
```