Install PowerDNS on Ubuntu 18.04, 20.04, & 22.04

Introduction

PowerDNS is an open-source DNSserver solution that helps resolve namespaces. PowerDNS supports high availability, data redundancy, and various backends, which makes it a flexible and robust solution.

This guide shows how to install PowerDNS and the PowerDNS Admin interface on Ubuntu.

Untitled Attachment

Prerequisites

- Access to the terminal.
- Access to the root user.
- A text editor, such as nano.
- A web browser to access PowerDNS Admin.

Why Use PowerDNS?

PowerDNS provides two nameserver solutions:

- The **Authoritative Server**, which uses the database to resolve queries about domains.
- The **Recursor**, which consults with other authoritative servers to resolve queries.

Other nameservers combine the two functions automatically. PowerDNS offers them separately, and allows the mix of the two solutions seamlessly for a modular setup. Additionally, PowerDNS is open source, works equally well for small and large query volumes, and offers many possibilities for backend solutions.

Installing PowerDNS on Ubuntu 18.04, 20.04, & 22.04

Follow the steps below to install and configure PowerDNS with the MariaDB server as a backend database. Additionally, the steps guide users through the setup of the PowerDNS Admin web interface and API.

Step 1: Install and Configure MariaDB Server

To install and configure MariaDB, do the following:

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
```

Note: Other possible database serversinclude PostgreSQL, MySQL, and other relational databases.

Wait for the installation to finish before continuing.
3. Connect to MariaDB with:

```
sudo mysql
```

Untitled Attachment

The terminal connects to a database session.
4. Create a database for the PowerDNS nameserver:

```
create database pda;
```

Note: If using a different database name, change all consequent commands accordingly.

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

```
grant all privileges on pda.* TO 'pda'@'localhost' identified by
'YOUR_PASSWORD_HERE';


flush privileges;
```

6. Connect to the database:

```
use pda;
```

Untitled Attachment

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';




REATE UNIQUE INDEX name_index ON domains(name);







EATE 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';




REATE INDEX nametype_index ON records(name,type);

CREATE INDEX domain_id ON records(domain_id);
```

```
CREATE INDEX ordername ON records (ordername);
```

```
EATE 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';
```

```
EATE 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';




REATE INDEX comments_name_type_idx ON comments (name, type);

CREATE INDEX comments_order_idx ON comments (domain_id, modified_at);







EATE TABLE domainmetadata (
```

```
id INT AUTO_INCREMENT,

domain_id INT NOT NULL,

kind VARCHAR(32),

content TEXT,

PRIMARY KEY (id)

) Engine=InnoDB CHARACTER SET 'latin1';




REATE INDEX domainmetadata_idx ON domainmetadata (domain_id, kind);









EATE 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';




REATE INDEX domainidindex ON cryptokeys(domain_id);









EATE TABLE tsigkeys (

id INT AUTO_INCREMENT,

name VARCHAR(255),

algorithm VARCHAR(50),

secret VARCHAR(255),

PRIMARY KEY (id)
```

```
) Engine=InnoDB CHARACTER SET 'latin1';
```

```
REATE UNIQUE INDEX namealgoindex ON tsigkeys(name, algorithm);
```

8. Confirm the tables have been created with:

```
show tables;
```

Untitled Attachment

The output lists the available tables.

9. Exit the database connection:

```
exit;
```

Untitled Attachment

The command returns the session to the terminal.

Step 2: Install PowerDNS

To install PowerDNS on Ubuntu, do the following:

1. Switch to the root user:

```
sudo su -
```

Untitled Attachment

The terminal session changes to the root user.

Note: Learn about the difference between sudo and su.

2. The **systemd-resolved**service provides the name resolutions to local applications. PowerDNS uses its own service for name resolutions.

Disable the **systemd-resolved**service with:

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

Untitled Attachment

The output confirms the service removal.

3. Delete the system service configuration file with:

```
rm -rf /etc/resolv.conf
```

4. Create the new *resolv.conf*file:

```
echo "nameserver 8.8.8.8" | sudo tee /etc/resolv.conf
```

Untitled Attachment

Appending the Google nameserver ensures DNS resolution.

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

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

Wait for the installation to complete before continuing.

Step 3: 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=
```

Untitled Attachment

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
```

Untitled Attachment

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
```

Untitled Attachment

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

Step 4: 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 -
```

Untitled Attachment

4. Install Node.jswith:

```
apt install -y nodejs
```

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

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

Untitled Attachment

Yarn helps build the asset files.

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 Yarnwith:

```
apt install yarn -y
```

9. Clone the PowerDNS Admin Git repositoryto */opt/web/powerdns-admin*:

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

Untitled Attachment

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
```

Untitled Attachment

13. Upgrade pip to the latest version:

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

The pippackage manager helps install additional Python requirements.

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

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

Untitled Attachment

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

Step 5: Configure and Run PowerDNS Admin

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

1. Use the cp commandto 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
```

```
ECRET_KEY = 'e951e5a1f4b94151b360f47edf596dd2'
```

```
QLA_DB_PASSWORD = 'changeme'
```

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

```
import urllib.parse
```

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

```
QLA_DB_PASSWORD = 'YOUR_PASSWORD_HERE'
```

Untitled Attachment

Note: Generate a random key using Python:

```
python3 -c "import os; print(os.urandom(16))"
```

Copy and paste the output into the **SECRET_KEY**value.
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
```

Untitled Attachment

8. Install project dependencies:

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

Untitled Attachment

9. Build flask app assets:

```
flask assets build
```

Untitled Attachment

Wait for the build to complete.
10. Run the application with:

```
./run.py
```

Untitled Attachment

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

```
http://localhost:9191
```

Untitled Attachment

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
```

The terminal returns to a regular state.

Step 6: 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
```

Untitled Attachment

3. Create a unit file:

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

4. Append the following:

```
[Service]

Environment="FLASK_CONF=../configs/production.py"
```

Untitled Attachment

5. Create a socketfile:

```
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
```

Untitled Attachment

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
```

Untitled Attachment

11. Check the status with:

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

Untitled Attachment

The services show as running without any errors.

Step 7: Install and Configure Nginx

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

1. Install Nginxwith:

```
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
```

Untitled Attachment

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 8: 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 Key**s tab on the left menu.

Untitled Attachment

3. Click the **Add Key+**button.

Untitled Attachment

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.

Untitled Attachment

6. A popup window prints the key. **Copy the key**and press **Confirm**to continue.

Untitled Attachment

7. Navigate to the **Dashboard**page.
8. Enter the domain and API key. Save the changes.

Untitled Attachment

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
```

11. Save the changes and close nano. The API is set up and ready to use.

Conclusion

After going through the steps in this guide, you've set up PowerDNS, the PowerDNS Admin web interface on Nginx, and connected the PowerDNS API.

Next, learn about the different DNS record typesor DNS security best practices.