PowerDNS Installation

PowerDNS Installation

Setting up the Basics

First we'll need to install some dependencies and set the PowerDNS Repositories. We'll also create a directory to put our generated credentials and files in.

```
# Install basic dependencies
apt-get install software-properties-common gnupg2 lsb-release curl -y

# Get the current key from the PowerDNS Repository, this could be
outdated
# Master Branch
wget -O- https://repo.powerdns.com/CBC8B383-pub.asc | gpg --dearmor >
/etc/apt/trusted.gpg.d/pdns_master.gpg
# Stable Branch
wget -O- https://repo.powerdns.com/FD380FBB-pub.asc | gpg --dearmor >
/etc/apt/trusted.gpg.d/pdns_stable.gpg

# Set some variables for repository setup
systemReleaseVersion=$(lsb_release -cs)
osLabel=$(lsb_release -sa 2>/dev/null|head -n 1|tr '[:upper:]'
'[:lower:]')

# Create a Working Directory for the installation
mkdir -p /opt/pdns_install
workpath="/opt/pdns_install"
```

Adding the Source Repositories

Once the basics have been installed we can set up the **PowerDNS Repositories**.

If you only wish to install the PDNS Authoritative server you can just use one repo, but I recommend adding all of them just in case.

```
# PowerDNS Authoritative Repository
repo_pdnsAuth="deb [arch=amd64] http://repo.powerdns.com/${osLabel}
${systemReleaseVersion}-auth-48 main"
```

```
echo $repo_pdnsAuth > "/etc/apt/sources.list.d/pdns-auth.list"

# PowerDNS Recursor Repository
repo_pdnsRec="deb [arch=amd64] http://repo.powerdns.com/${osLabel}
${systemReleaseVersion}-rec-48 main"
echo $repo_pdnsRec > "/etc/apt/sources.list.d/pdns-rec.list"

# PowerDNS DNS Dist Repository
repo_dnsdist="deb [arch=amd64] http://repo.powerdns.com/${osLabel}
${systemReleaseVersion}-dnsdist-18 main"
echo $repo_dnsdist > "/etc/apt/sources.list.d/dnsdist.list"

echo \
"Package: pdns-*
Pin: origin repo.powerdns.com
Pin-Priority: 600" > /etc/apt/preferences.d/pdns
```

Installing the PowerDNS Backend

Before installing PowerDNS you'll need to choose a suitable backend (Postgres or MariaDB). I recommend Postgres but MariaDB is also a good choice.

MariaDB / MySQL

```
# Add MariaDB Public Key
wget -O- https://mariadb.org/mariadb_release_signing_key.asc | gpg --
dearmor > /etc/apt/trusted.gpg.d/mariadb.gpg

# Add MariaDB Repository
echo "# MariaDB Repository List" >
"/etc/apt/sources.list.d/mariadb.list"
echo "deb [arch=amd64,arm64,ppc64el,s390x]
https://mirrors.ukfast.co.uk/sites/mariadb/repo/10.5/${osLabel}
${systemReleaseVersion} main" >>
"/etc/apt/sources.list.d/mariadb.list"
echo "deb-src
https://mirrors.ukfast.co.uk/sites/mariadb/repo/10.5/${osLabel}
${systemReleaseVersion} main" >>
"/etc/apt/sources.list.d/mariadb.list"

# Update the package lists:
apt update
```

```
# Install MariaDB
apt install mariadb-server
```

Postgresql (PGSQL)

```
# Add PostgreSQL Public Key
wget -O- https://www.postgresql.org/media/keys/ACCC4CF8.asc | gpg --
dearmor > /etc/apt/trusted.gpg.d/postgresql.gpg

# Add PostgreSQL Repository
echo "deb http://apt.postgresql.org/pub/repos/apt
${systemReleaseVersion}-pgdg main" >
/etc/apt/sources.list.d/pgdg.list

# Update the package lists:
apt-get update

# Install the latest version of PostgreSQL.
# If you want a specific version, use 'postgresql-12' or similar
instead of 'postgresql':
apt-get -y install postgresql

# Enable PostgreSQL Service
systemctl enable postgresql

# Start PostgreSQL Service
systemctl start postgresql
```

Don't forget to set your local unix socket connection in the **pg_hba.conf**file to *md5*or *scram-sha-256*instead of *peer*

Generating the Database

Once the DBMS is installed you'll need to generate credentials for PowerDNS and create it's DB Schema.

```
pdns_db="pdns"
pdns_db_user="pdnsadmin"
pdns_pwd="$(tr -dc A-Za-z0-9 </dev/urandom | head -c 13 ; echo '')"
pdnsadmin_salt="$(tr -dc A-Za-z0-9 </dev/urandom | head -c 32 ; echo
'')"
```

```
pdns_apikey="$(tr -dc A-Za-z0-9 </dev/urandom | head -c 32 ; echo
'')"

echo "pdns_db=$pdns_db" > "$workpath/db_credentials"
echo "pdns_db_user=$pdns_db_user" >> "$workpath/db_credentials"
echo "pdns_pwd=$pdns_pwd" >> "$workpath/db_credentials"
echo "pdnsadmin_salt=$pdnsadmin_salt" >> "$workpath/db_credentials"
echo "pdns_apikey=$pdns_apikey" >> "$workpath/db_credentials"
echo "workpath=$workpath" >> "$workpath/db_credentials"
echo 'systemReleaseVersion=$(lsb_release -cs)' >>
"$workpath/db_credentials"
echo 'osLabel=$(lsb_release -sa 2>/dev/null|head -n 1|tr '[:upper:]'
'[:lower:]')' >> "$workpath/db_credentials"
chown root:root "$workpath/db_credentials"
chmod 640 "$workpath/db_credentials"
cd "$workpath"
```

Once we've generated the credentials we can move on to create the database.

MariaDB / MySQL

```
# MySQL/MariaDB - Set this variable for stuff down the road
db_type="mysql"
echo "db_type=$db_type" >> "$workpath/db_credentials"

echo \
"-- PowerDNS MySQL/MariaDB Create DB File
CREATE DATABASE pdns;
GRANT ALL ON pdns.* TO pdnsadmin@localhost IDENTIFIED BY '$pdns_pwd';
FLUSH PRIVILEGES;" > "$workpath/pdns-createdb-my.sql"
mariadb -u root < "$workpath/pdns-createdb-my.sql"
```

Postgresql (PGSQL)

```
# PostgreSQL - Set this variable for stuff down the road
db_type="pgsql"
echo "db_type=$db_type" >> "$workpath/db_credentials"

echo \
"-- PowerDNS PGSQL Create DB File
CREATE USER $pdns_db_user WITH ENCRYPTED PASSWORD '$pdns_pwd';
CREATE DATABASE $pdns_db OWNER $pdns_db_user;
```

```
GRANT ALL PRIVILEGES ON DATABASE $pdns_db TO $pdns_db_user;" >
"$workpath/pdns-createdb-pg.sql"
sudo -u postgres psql < "$workpath/pdns-createdb-pg.sql"
```

Installing PowerDNS/DNSDist

```
# Update the package lists:
apt update

# For MariaDB
apt-get install pdns-backend-mysql -y
# For PostgreSQL
apt-get install pdns-backend-pgsql -y

# PowerDNS Authoritative Server
apt-get install pdns-server -y

# PowerDNS Recursor Server
apt install pdns-recursor

# DNS Dist Balancer
apt install dnsdist

# Disable systemd-resolved
systemctl disable systemd-resolved
systemctl stop systemd-resolved
```

> You'll want to set up an external DNS for resolution in the resolv.conffile if you do not set up a Recursor DNS.

Populating the Database

Once the PowerDNS Server has been installed you'll need to populate it's database with the provided Schema from it's backend.

```
db_config_path="/etc/powerdns/pdns.d"

# Define config file based on selected DB type
db_config_file="$db_config_path/g$db_type.conf"
```

```
if [[ $db_type == "mysql" ]]; then
mariadb -u pdnsadmin -p $pdns_db < "/usr/share/pdns-
backend-$db_type/schema/schema.$db_type.sql"
else
current_pgsql_ver=$(psql -V|awk -F " " '{print $NF}'|awk -F "."
'{print $1}')
export PGPASSWORD="$pdns_pwd"
psql -U $pdns_db_user -h 127.0.0.1 -d $pdns_db < "/usr/share/pdns-
backend-$db_type/schema/schema.$db_type.sql"
unset PGPASSWORD
fi

# Copy the example config file
cp /usr/share/doc/pdns-backend-$db_type/examples/g$db_type.conf
/etc/powerdns/pdns.d/

# Change the values in your PowerDNS Config file to match your saved
credentials
sed -i "s/\(^g$db_type-dbname=\).*/\1$pdns_db/" "$db_config_file"
sed -i "s/\(^g$db_type-host=\).*/\1127.0.0.1/" "$db_config_file"
if [[ $db_type == "mysql" ]]; then
sed -i "s/\(^g$db_type-port=\).*/\13306/" "$db_config_file"
else
sed -i "s/\(^g$db_type-port=\).*/\15432/" "$db_config_file"
fi
sed -i "s/\(^g$db_type-user=\).*/\1$pdns_db_user/" "$db_config_file"
sed -i "s/\(^g$db_type-password=\).*/\1$pdns_pwd/" "$db_config_file"
chmod 640 "$db_config_file"
chown root:pdns "$db_config_file"
```

Configuring PowerDNS (Authoritative)

Before starting up the service you'll need to set-up some basic parameters:

```
# Set your Local IP Address
local-address=127.0.0.1, $EXTERNAL_SERVER_IP

# If you only use the PowerDNS Authoritative Server
local-port=53

# PDNS Auth Port with DNSDist
local-port=5300
```

Configuring PowerDNS (Recursor)

Once you've successfully installed the PowerDNS Recursor service you can go ahead and add Forward Zones. Usually you should add these zones if they're yours and should be resolved by the PowerDNS Authoritative Server.

To do that edit the file /etc/powerdns/recursor.conf.

You can add zones in the following format:

```
# First Forward Zone
forward-zones=example.com=127.0.0.1:5300
# N Forward Zone
forward-zones+=example.com=127.0.0.1:5300
```

> This basically redirects your Recursor requests to the Authoritative Server. It's recommended to only allow Recursive Queries to your LAN and only expose your Authoritative DNS to external queries.

Setting up the API Key

Next up you'll need to set up your API Key for the Flask Front-end to be able to communicate with your PDNS Back-end.

```
# Set the API Key to your generated key and API Parameter to "Yes"
sed -i "s/.*api=\(yes\|no\)+$/api=yes/" "/etc/powerdns/pdns.conf"
sed -i "s/\(.*api-key=\).*/\1$pdns_apikey/" "/etc/powerdns/pdns.conf"
sed -i "s/.*webserver=\(yes\|no\)+$/webserver=yes/"
"/etc/powerdns/pdns.conf"

# Stop and start the service
systemctl stop pdns
systemctl start pdns
```

And add the API Webserver Port and Allowed CIDR.

```
# Webserver/API access is only allowed from these subnets
```

```
webserver-allow-from=127.0.0.1,$LAN_CIDR

# Webserver/API Port to listen on
webserver-port=8081
```

Installing the Front-end / Admin UI

Dependency Installation

Once the backend is installed, we'll install the admin front-end.

```
# Install PowerDNS Admin Dependencies
apt-get install nginx \
python3-dev \
python3-venv \
libsasl2-dev \
libldap2-dev \
libssl-dev \
libxml2-dev \
libxslt1-dev \
libxmlsec1-dev \
libffi-dev \
pkg-config \
apt-transport-https \
virtualenv \
build-essential \
libmariadb-dev \
git \
libpq-dev \
python3-flask -y
```

After installing the required dependencies you'll also need to install NodeJS and YARN

```
# NodeJS
curl -sL https://deb.nodesource.com/setup_20.x | bash -
apt-get update -y
apt-get install nodejs -y

# YARN
wget -O- https://dl.yarnpkg.com/debian/pubkey.gpg | gpg --dearmor >
/etc/apt/trusted.gpg.d/yarnpkg.gpg
```

```
echo "deb https://dl.yarnpkg.com/debian/ stable main" >
"/etc/apt/sources.list.d/yarn.list"
apt-get update -y
apt-get install yarn -y
```

Now we can clone the PowerDNS Admin git repository and compile it.

```
# Clone the Repository
git clone https://github.com/ngoduykhanh/PowerDNS-Admin.git
/var/www/html/pdns
# OR
git clone https://github.com/PowerDNS-Admin/PowerDNS-Admin.git
/var/www/html/pdns

# Go into the working directory and enable the virtual environment
cd "/var/www/html/pdns/"
virtualenv -p python3 flask
source "./flask/bin/activate"

# There's a bug with PyYAML 5.4 and Cython 3.0
sed -i "s/PyYAML==5.4/PyYAML==6.0/g" requirements.txt

pip install --upgrade pip

# Install the requirements
pip install -r requirements.txt

# Deactivate the virtualenv
deactivate
```

After this is done we can set up all the environment variables for the Front-end.

```
prod_config="/var/www/html/pdns/configs/production.py"
cp "/var/www/html/pdns/configs/development.py" $prod_config

# Set Filesystem_sessions
sed -i "s/\(^FILESYSTEM_SESSIONS_ENABLED = \).*/\1True/" $prod_config

# Set SALT on the corresponding Parameter
sed -i "s/\(^SALT = \).*/\1\'$pdnsadmin_salt\'/" $prod_config
# Set APIKEY on the corresponding Parameter
```

```
sed -i "s/\(^SECRET_KEY = \).*/\1\'$pdns_apikey\'/" $prod_config
# Set MySQL/MariaDB/PGSQL Password on the corresponding Parameter
sed -i "s/\(^SQLA_DB_PASSWORD = \).*/\1\'$pdns_pwd\'/" $prod_config
# Set DB Name
sed -i "s/\(^SQLA_DB_NAME = \).*/\1\'$pdns_db\'/" $prod_config
# Set DB User
sed -i "s/\(^SQLA_DB_USER = \).*/\1\'$pdns_db_user\'/" $prod_config

# If you're using PostgreSQL add the following statements to the
configuration file and install psycopg2
if [[ $db_type != "mysql" ]]; then
source "./flask/bin/activate"
pip install psycopg2
deactivate
fi

sed -i "s/#import urllib.parse/import urllib.parse/g" $prod_config

# Insert PORT after SQLA_DB_USER
if [[ $db_type == "mysql" ]]; then
db_port="3306"
else
db_port="5432"
fi

if ! [[ $(grep "SQLA_DB_PORT" $prod_config) ]]; then
sed -i "/^SQLA_DB_USER.*/a SQLA_DB_PORT = $db_port" $prod_config
else
sed -i "s/\(^SQLA_DB_PORT = \).*/\1$db_port/" $prod_config
fi

# Insert DB URI
cat >> $prod_config <<EOF
SQLALCHEMY_DATABASE_URI = '$db_type://{}:{}@{}/{}'.format(
urllib.parse.quote_plus(SQLA_DB_USER),
urllib.parse.quote_plus(SQLA_DB_PASSWORD),
SQLA_DB_HOST,
SQLA_DB_NAME
)
EOF

# Comment default SQLite DATABASE URI
sed -i "s/\(^SQLALCHEMY_DATABASE_URI = 'sqlite.*\)/#\1/" $prod_config
```

Compiling the Front-end / Admin UI

Once the dependencies and requirements are all installed we can proceed to compile and set up the Front-end

```
cd "/var/www/html/pdns/"
source "./flask/bin/activate"

export FLASK_APP=powerdnsadmin/__init__.py
export FLASK_CONF=../configs/production.py
flask db upgrade
yarn install --pure-lockfile
flask assets build
deactivate
```

Setting up the Front-end Services

Copy the following files onto your NGINX Directory or create them with the following text blocks.

/etc/systemd/system/pdnsadmin.service

```
[Unit]
Description=PowerDNS-Admin
Requires=pdnsadmin.socket
After=network.target

[Service]
PIDFile=/run/pdnsadmin/pid
User=pdns
Group=pdns
Environment="FLASK_CONF=../configs/production.py"
WorkingDirectory=/var/www/html/pdns
ExecStart=/var/www/html/pdns/flask/bin/gunicorn --pid
/run/pdnsadmin/pid --bind unix:/run/pdnsadmin/socket
'powerdnsadmin:create_app()'
ExecReload=/bin/kill -s HUP $MAINPID
ExecStop=/bin/kill -s TERM $MAINPID
PrivateTmp=true

[Install]
```

```
WantedBy=multi-user.target
```

/etc/systemd/system/pdnsadmin.socket

```
[Unit]
Description=PowerDNS-Admin socket

[Socket]
ListenStream=/run/pdnsadmin/socket

[Install]
WantedBy=sockets.target
```

/etc/nginx/sites-enabled/powerdns-admin.conf

```
server {
listen *:80;
server_name pdnsadmin.example.com;

index index.html index.htm index.php;
root /var/www/html/pdns;
access_log /var/log/nginx/pdnsadmin_access.log combined;
error_log /var/log/nginx/pdnsadmin_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 /var/www/html/pdns/powerdnsadmin;
```

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

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

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

```
chown -R pdns:www-data "/var/www/html/pdns"
nginx -t && systemctl restart nginx
```

After you've copied and created your NGINX Entry you'll want to enable the services so that PowerDNS-Admin starts automatically.

```
echo "d /run/pdnsadmin 0755 pdns pdns -" >>
"/etc/tmpfiles.d/pdnsadmin.conf"
mkdir "/run/pdnsadmin/"
chown -R pdns: "/run/pdnsadmin/"
chown -R pdns: "/var/www/html/pdns/powerdnsadmin/"

systemctl daemon-reload

systemctl enable --now pdnsadmin.service pdnsadmin.socket
```

Your database installation credentials are saved at **/opt/pdns_install/db_credentials**

To check the PowerDNS Admin status you can do:

```
systemctl status pdnsadmin.service pdnsadmin.socket
```

- Default API URL: http://localhost:8081

Configuring DNSDist

Finally, if you wish to use DNSDist to handle Authoritative and Recursive queries selectively based on CIDR, you'll have to make the following changes:

```
-- dnsdist configuration file, an example can be found in
/usr/share/doc/dnsdist/examples/

setLocal('{EXTERNAL_SERVER_IP}:53')
--addLocal('{ANOTHER_IP}:53')
setACL({'0.0.0.0/0', '::/0'}) -- Allow all IPs access

newServer({address='127.0.0.1:5300', pool='auth'})
newServer({address='127.0.0.1:5301', pool='recursor'})

recursive_ips = newNMG()
recursive_ips:addMask('127.0.0.0/8')
recursive_ips:addMask('{LAN_CIDR}') -- These network masks are the
ones from allow-recursion in the Authoritative Server
addAction(NetmaskGroupRule(recursive_ips), PoolAction('recursor'))
addAction(AllRule(), PoolAction('auth'))

-- disable security status polling via DNS
setSecurityPollSuffix("")
```

This will essentially filter out your LAN Hosts from External Queries

> You'll need to change your Auth and Recursor Server ports to match this configuration file (Auth → 5300 / Rec → 5301)

Unsetting all variables

Finally once the setup phase is done, you'll want to clear the variables.

```
unset systemReleaseVersion
unset osLabel
unset repo_pdnsAuth
unset repo_pdnsRec
unset repo_dnsdist
```

```
unset pdns_pwd
unset pdnsadmin_salt
unset pdns_apikey
unset workpath
```

Last modified at Aug 8, 2023