dino Documentation

uberspace.de

Mar 13, 2023

Contents

Dino is a modern DNS record editor for PowerDNS. It uses the PowerDNS-API, has a high test coverage, rich documentation and comes with batteries included!

Take a look at the documentation sections on the left side and choose something to your liking.

CHAPTER 1

Database

Before we can start installing dino the database needs to be prepared. There are no special requrements so any database supported by django may be used. Official support is given for SQLite, PostgreSQL and MySQL/MariaDB.

With the exception of SQLite these instructions assume that your database system was installed, intialized, correctly configured and started and only deals with aspects relevant to our setup.

1.1 SQLite

By default dino uses a SQLite database, which is just a file on your disk. In this case, no configuration is necessary. Please keep in mind that, while being a capable database, SQLite does not scale well and has weaker guarantees than server-based database solutions like PostgreSQL.

1.2 PostgreSQL

The Wikipedia describes PostgreSQL as:

> (...) an open source object-relational database management system with an emphasis on extensibility and standards compliance.

Create a new user and database for dino:

```
postgres@host:~$ createuser dino -P
Enter password for new role: ******
Enter it again: ******
postgres@host:~$ createdb dino --owner=dino
```

Note: If dino and PostgreSQL are running on the same host, your setup can be password-less using a UNIX socket connection and Peer Authentication. Skip the `-P` parameter in this case.

1.3 MySQL / MariaDB

MariaDB describes itself as:

> (...) one of the most popular database servers in the world. It's made by the original developers of MySQL and guaranteed to stay open source. Notable users include Wikipedia, WordPress.com and Google.

Create a new user and database for dino:

```
root@host:~# mysql
mysql> CREATE DATABASE dino DEFAULT CHARACTER SET utf8mb4 DEFAULT COLLATE utf8mb4_
↪general_ci;
mysql> GRANT ALL PRIVILEGES ON dino.* TO dino@'localhost' IDENTIFIED BY '*********';
mysql> FLUSH PRIVILEGES;
```

CHAPTER 2

Setup using pip

This guide explains how to install dino inside a fresh ubuntu 18.04 bionic box. It includes the installation of all on-host prerequisites and provides you with a fully functional Dino instance.

Most of the steps are the same for other distributions like Debian or CentOS, so you should be able to adpat this to your needs as you go. We'd be glad about full guides for other operating systems. So please feel free to send us tips or even open a Pull Request on GitHub.

2.1 Prerequisites

2.1.1 PowerDNS

To use dino, a running PowerDNS instance is required. In addition to the default config, the following options are needed:

```
# 1. enable the API and choose a sane API key
api=yes
api-key=SOME_LONG_API_KEY_CHANGE_ME_PLEASE

# 2. enable the webserver and give dino acces to it
#    either by whitelisting it here, in your reverse proxy or in your firewall.
webserver=yes
webserver-address=0.0.0.0
webserver-port=8080
webserver-allow-from=1.2.3.4/32
```

Remember (. . . or write down) server hostname, webserver port as well as the API key.

2.1.2 Python 3.6

Python 3.6 or newer is required. Ubuntu already comes with python 3.6, so we're good here. To complete the python setup, install the package manager pip, which is needed later.

```
root@ubuntu-bionic:~# python3 --version
Python 3.6.7
root@ubuntu-bionic:~# apt install -y python3-pip
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
(...)
```

Note: **Debian** users might be able to compile python 3.6 from source or get it from the Debian Sid repos.

Note: **CentOS** users can get python 3.6 from the IUS Project; install `python36u` as well as `python36u-pip`.

2.1.3 Other dependencies

We also need some further dependencies to install uWSGI via pip in the next step.

```
root@ubuntu-bionic:~# apt install -y python3-setuptools build-essential python3-dev
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
(...)
root@ubuntu-bionic:~# pip3 install wheel
Collecting wheel
  Downloading https://files.py..
(...)
```

2.1.4 uWSGI

uWSGI actually runs dino and will later provide it via HTTP on port 8080. It is installed globally using pythons dependency manager `pip`:

```
root@ubuntu-bionic:~# pip3 install uwsgi
Collecting uwsgi
  Downloading https://files.py..
(...)
```

2.1.5 System User

To make sure dino can only access what it needs to access, create a new user.

Note: Instead of using the default `www-dino`, you can freely choose any non-existing username. Just make sure to adapt the following steps and the systemd unit accordingly.

```
root@ubuntu-bionic:~# adduser --disabled-password --disabled-login \
  --system --home /opt/dino www-dino
```

2.2 Dino

```
root@ubuntu-bionic:~# sudo -Hu www-dino pip3 install --user \
  https://github.com/Uberspace/dino/archive/master.zip#subdirectory=src
Collecting https://github.com/Uberspace/dino/archive/master.zip#subdirectory=src
  Downloading https://github.com/Uberspace/dino/archive/master.zip
    / 1.3MB 295.2MB/s
(...)
Successfully installed (...) dino-0.1 (...)
```

Note: If you'd like to use a database other than SQLite, the corresponding python client library needs to be installed. Use one of the following urls instead of the one given above:

- MySQL/MariaDB: https://github.com/Uberspace/dino/archive/master.zip#egg=dino[mysql]&subdirectory=src
- PostgreSQL: https://github.com/Uberspace/dino/archive/master.zip#egg=dino[pgsql]&subdirectory=src

Additionally, the `libmariadbclient-dev` apt package is required for mysql.

2.2.1 Configuration

Create a file called `/etc/dino.cfg` with the following content. It provides a bare-bones configuration for django, which should be extended. Please go through the *list of config options* and set the appropriate values.

```
# a long (>64 chars) and random (alpha-numeric-ish) string of characters
DINO_SECRET_KEY=
# URL to your PowerDNS server API endpoint, e.g. https://yourpowerdns.com/api/v1
DINO_PDNS_APIURL=
# PowerDNS API key from /etc/pdns/pdns.conf
DINO_PDNS_APIKEY=
# comma-separated list of hostnames dino should be reachable under
DINO_ALLOWED_HOSTS=
# a place for dino to drop internal data; must be writeable by dino and
# not publicly acccessible
DINO_BASE_DIR=/opt/dino
# make use of the X-Forwarded-Host/Proto headers in nginx config
DINO_TRUST_PROXY=True
```

Note: By default, dino uses a SQLite database inside `DINO_BASE_DIR`. If you'd like to use a different system or database location, provide the respective URL as `DINO_DB_URL`:

- SQLite: `sqlite:////some/absolute/path/db.sqlite3`
- PostgreSQL (Password): `postgres://dino:PASSWORD@127.0.0.1:5432/dino`
- PostgreSQL (UNIX-Socket / Peer Auth): `postgres://%2Fpath%2Fto%2Fsocket/dino`
- MySQL: `mysql://dino:PASSWORD@127.0.0.1:3306/dino`

Further information can be found in the dj-database-url documentation.

2.2.2 Service

To start dino automatically when your server boots up, create a new systemd unit in `/etc/systemd/system/dino.service` and add the following content.

> **Warning:** The path to uwsgi (`/usr/local/bin/uwsgi`) may vary on other distributions. To be on the safe side, use the command `which uwsgi` to get the path for your installation.

```
[Unit]
Description=uWSGI dino
After=networking.target

[Service]
ExecStart=/usr/local/bin/uwsgi --http-socket :8080 --master --workers 8 --module dino.
↪wsgi
User=www-dino
Restart=always
KillSignal=SIGQUIT
Type=notify
StandardError=syslog
NotifyAccess=all

[Install]
WantedBy=multi-user.target
```

Finally, load the newly create service:

```
root@ubuntu-bionic:~# systemctl daemon-reload
```

2.2.3 Finishing up

Initialize the database

The following command creates all tables needed by dino. This will connect to the database server specified in `DINO_DB_URL`, which defaults to using a file-based SQLite database. If you'd like to use a different database, jump back to the *Configuration* section and adpat the setting.

```
root@ubuntu-bionic:~# sudo -Hu www-dino python3 -m dino migrate
DEBUG enabled, but django_extensions not installed. skipping app.
Operations to perform:
  Apply all migrations: account, admin, auth, contenttypes, sessions, sites,␣
↪socialaccount, synczones, tenants
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying account.0001_initial... OK
  Applying account.0002_email_max_length... OK
(...)
```

Create an admin user

The very first user has to be created using an interactive command. Additional users can be created in the web interface, once we're up and running.

```
root@ubuntu-bionic:~# sudo -Hu www-dino python3 -m dino createsuperuser
```

Start dino

```
root@ubuntu-bionic:~# systemctl enable dino --now
```

Congratulations, is now running! You can verify this by querying the port directly:

```
root@ubuntu-bionic:~# curl 127.0.0.1:8080
<h1>Bad Request (400)</h1>
```

You can now configure your webserver to route requests to dino, either by yourself or using the *Webserver* guide.

2.3 Updates

To update dino repeat the `pip3 install` and `dino migrate` steps from the installation guide. Afterwards, restart dino to load the new code.

```
root@ubuntu-bionic:~# sudo -Hu www-dino pip3 install --user \
  https://github.com/Uberspace/dino/archive/master.zip#subdirectory=src
(...)
root@ubuntu-bionic:~# sudo -Hu www-dino python3 -m dino migrate
(...)
root@ubuntu-bionic:~# systemctl restart dino
```

CHAPTER 3

Configuration

Dino provides a variety of configuration options. Have a quick list through the list to see which ones are relevant to you!

3.1 Location

There are multiple ways to set configuration options:

1. File `/etc/dino.cfg`, global config
2. File `~/.dino.cfg`, user config
3. File `./dino.cfg`, local config
4. Process Environment (e.g. `$DINO_BASE_DIR`)

They are loaded in the order above; later files overwrite earlier ones.

3.2 File Format

Configuration is stored in a simple, non-nested `key=value` format. Comments can be added using `#` on dedicated lines. Lines without = are ignored.

```
# this is a comment
DINO_BASE_DIR=/opt/dino
DINO_SECRET_KEY = verysecret
```

3.3 Options

3.3.1 DINO_BASE_DIR

Existing directory for dino to write internal data to. It must thus be created beforehand and be writeable by the user you use to run dino. It is currently only used to store the SQLite database (if used), but may contain other data in future releases. Note that this directory must **not** be accssible publicly.

- **type**: String
- **required**: no
- **default**: `.../lib/python3.x/site-packages/dino/`
- **example**: `/opt/dino`

3.3.2 DINO_SECRET_KEY

Long (>64 chars), random and ascii string of characters. It is used by django to derive crypto keys for cookies and other security-critical applications. Ensure that the string is private at all times. It can be changed without negative effect, if leaked accidentially.

- **type**: String
- **required**: yes
- **example**: `Aixa1ahs1euyo2oopii-Y:eex8sie~d5`

This is a stock django setting; refer to the "SECRET_KEY" django docs for more information.

3.3.3 DINO_DEBUG

Run dino in development mode. Do not enable this setting in production, as it might leak sensitive information to clients.

- **type**: Boolean
- **required**: no
- **default**: `False`

This is a stock django setting; refer to the "DEBUG" django docs for more information.

3.3.4 DINO_ALLOWED_HOSTS

List of hostnames under which dino should be accessible at. Accessing dino using a `Host:` header not in this list, yields an 400 Bad Request error.

- **type**: List
- **required**: yes
- **example**: `dino.company.com,dino.internal`

This is a stock django setting; refer to the "ALLOWED_HOSTS" django docs for more information.

3.3.5 DINO_PDNS_APIURL

Full URL to your PowerDNS server API endpoint, including the `/api/v1` path.

- **type**: String
- **required**: yes
- **example**: `https://yourpowerdns.com/api/v1`

3.3.6 DINO_PDNS_APIKEY

PowerDNS API key from pdns.conf.

- **type**: String
- **required**: yes
- **example**: `wooviex7ui0Eiy2Gohth4foovoob5Eip`

3.3.7 DINO_LOGIN_PROVIDERS

Social login providers to load and offer to users. Please refer to the django-allauth docs for a list of available providers.

- **type**: List
- **required**: no
- **example**: `google,soundcloud`

3.3.8 DINO_DB_URL

Database to connect to; refer to dj-database-url for information on the URL schema.

- **type**: String
- **required**: no
- **default**: `sqlite:////home/docs/checkouts/readthedocs.org/user_builds/dino/envs/latest/lib/python3.7/site-packages/db.sqlite3`
- **example**: `mysql://dino:password@host/dino`

3.3.9 DINO_TRUST_PROXY

Whether to trust the information given in the `X-Forwarded-Proto` and `X-Forwarded-Host` HTTP headers. If dino is behind a reverse proxy, set this to `True`. Ensure that your server software is a) setting these headers and b) discards any content provided by clients.

- **type**: Boolean
- **required**: no
- **default**: `False`

3.3.10 DINO_HTTPS_ONLY

Whether to enforce HTTPS, set HSTS and send cookies on HTTPS only. Recommended, if your setup exposes dino on HTTPS (which, again, is recommended).

- **type**: Boolean
- **required**: no
- **default**: `False`

3.3.11 DINO_TIMEZONE

Timezone to use for auditing and logging.

- **type**: String
- **required**: no
- **default**: `UTC`
- **example**: `Europ/Berlin`

This is a stock django setting; refer to the "TIMEZONE" django docs for more information.

3.3.12 DINO_ENABLE_EMAIL_SIGNUP

Whether to let users create permissionless accounts without any prior authentication using username/password. not recommended, refer to *User Signup* for details.

- **type**: Boolean
- **required**: no
- **default**: `False`

3.3.13 DINO_ENABLE_SOCIAL_SIGNUP

Whether to let users create permissionless accounts without any prior authentication social login (google, openid, . . .). not recommended, refer to *User Signup* for details.

- **type**: Boolean
- **required**: no
- **default**: `False`

3.3.14 DINO_VALID_SIGNUP_DOMAINS

If `DINO_ENABLE_EMAIL_SIGNUP` is enabled, restrict creation of new users to the given domains. Any user, who can receive mail at a whitelisted domain will then be able to create a permissionless account without any prior authentication. Accounts need to be activated by verifiying the email address, though.

- **type**: List
- **required**: no
- **default**: `[]`

- **example**: `company.com,company.internal`

3.3.15 DINO_ZONE_DEFAULT_KIND

PowerDNS kind to set for new zones, may be Native, Master or Slave. See PowerDNS Docs.

- **type**: String
- **required**: no
- **default**: `Native`
- **example**: `None`

3.3.16 DINO_ZONE_DEFAULT_NAMESERVERS

List of nameservers to set for new zones.

- **type**: List
- **required**: no
- **example**: `ns1.company.com,ns2.company.com`

3.3.17 DINO_ZONE_DEFAULT_MASTERS

List of masters to set for new zones.

- **type**: List
- **required**: no
- **example**: `1.3.3.7,1.3.3.8`

3.3.18 DINO_USE_DEFAULT_RECORD_TYPES

Whether to offer a selection of default record types (A, AAAA, MX, CAA, ...) in the GUI, or rely on `DINO_CUSTOM_RECORD_TYPES` only.

- **type**: Boolean
- **required**: no
- **default**: `True`

3.3.19 DINO_CUSTOM_RECORD_TYPES

Additional record types to offer in the GUI. Any record type can be used here, but PowerDNS or secondary DNS servers might not be able to handle them.

- **type**: List
- **required**: no
- **example**: `X25,SPF,DS`

CHAPTER 4

Webserver

Even though dino is running, it's currently not possible to reach it from the outside world (e.g. your browser). For that to happen we need a reverse proxy. In our example nginx will be used, but you can use any other software like Apache, Lighttpd or Traefik.

Note: Just like the pip setup guide, this guide assumes a fresh ubuntu 18.04 bionic box. Most of the steps are the same or can be easily adapted for other distributions.

4.1 Installation

```
root@ubuntu-bionic:~# apt install -y nginx
```

Warning: Some distributions do not allow nginx to make network connections. This can be changed by enabling the appropriate SELinux boolean, like so:

```
root@ubuntu-bionic:~# apt install -y policycoreutils
root@ubuntu-bionic:~# setsebool httpd_can_network_connect true -P
```

4.2 Configuration

First, remove the default site, if present:

```
root@ubuntu-bionic:~# rm -f /etc/nginx/sites-enabled/default
```

Then, create our new configuration in `/etc/nginx/sites-enabled/dino`:

```
server {
  # this configuration does not include any HTTP, which is a bad idea.
  # use the generator below or any other resource to add transport encryption
  # to your requests.
  # https://mozilla.github.io/server-side-tls/ssl-config-generator/
  listen 80 default_server;
  listen [::]:80 default_server;

  server_name _;

  location / {
    proxy_http_version 1.1;
    proxy_set_header Host $host;
    proxy_set_header X-Forwarded-Host $host;
    proxy_set_header X-Forwarded-Proto $scheme;
    proxy_set_header X-Forwarded-Port $server_port;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_pass http://localhost:8080;
  }
}
```

4.3 Static Files

Dino is able to serve all requests itself - this includes static files like CSS or JavaScript. For small setups, pinging python and django for each and every static file is quite fast enough; refer to the whitenoise documentation details. Bigger setups with a high volume of requests might benefit from caching those requests.

You can cache everything that is requested from the `/static/` directory. URLs look like `/static/style/zoneeditor.d7de63624ec9.css`, contain a hash, which changes when the content changes. This means that you can cache them without any kind of timeout or cache invalidation.

4.4 Start nginx

```
root@ubuntu-bionic:~# systemctl enable nginx
root@ubuntu-bionic:~# systemctl restart nginx
```

4.5 Finishing up

You're done! Fire up your browser and open the URL configured in your webserver. Then you can log in using the account created using the `createsuperuser` command.

CHAPTER 5

Backups

Dino does currently not store any data outside the database. Should there be a need to store data outside the database in future releases, it will be saved inside the `$DINO_BASE_DIR` directory (e.g. `/opt/dino`). If you want your setup to be futureproof, make backups of it.

For now, all you have to do is backup your database. Below are a couple of common options.

5.1 SQLite

By default dino uses a SQLite database, which is just a file on your disk. To back it up copy `$DINO_BASE_DIR/db.sqlite3` (e.g. `/opt/dino/db.sqlite3`) to somewhere safe outside the host.

5.2 PostgreSQL

For most setups a simple `pg_dump` backup will suffice. For write-heavy setups a more robust primary/secondary backup solution might be prefered.

```
postgres@host:~$ pg_dump --clean --create --file=dino_$(date -I).sql dino
```

Refer to the PostgreSQL manual for further instructions.

5.3 MySQL / MariaDB

For most setups a simple `mysqldump` backup will suffice. For write-heavy setups a more robust primary/secondary backup solution might be prefered.

```
root@host:~# mysqldump --single-transaction --add-drop-database --databases dino >␣
↪dino_$(date -I).sql
```

Refer to the MySQL manual for further instructions.

CHAPTER 6

Users & Tenants

Dino has a basic multi-tenancy feature, which allows you to create users and give them access to a subset of the available domains. You can also use admin accounts instead, if this functionality is not needed.

6.1 Users

Dino knows a couple kinds of users:

- admin users, has full access to all zones
- tenant users, has access to associated zones only

During the setup guide a single administrator user was created. You can use this account to log into dino. Further users should be created using the web interface. Visit `/admin/auth/user` and use the "Add User" button to create new users. They can later be associated with a tenant.

6.2 Tenants

By default, there are no tenants created. Visit `/admin/tenants/tenant` in the web interface to create your first one. During creation process you can add users and zones to the new tenant.

Users may have one of the following permission levels:

- tenant users:
  - cannot delete or create zones
  - can create/edit/delete records in associated zones
- tenant admins:
  - can create or delete zones
  - can create/edit/delete records in associated zones

Tenants admins can currently not add new users themselves.

CHAPTER 7

Zone Data

Dino tries to mirror as little data as possible from PowerDNS. To enable permission managment, dino stores a list of zone names within the database. However, there is no permanent storage of the zone records. The zone list is refreshed each time a user opens the zone list.

Modifications like creating zones or deleting records are always carried out live via the PowerDNS API, if and only if the appropriate UI features are used. There is no syncing happening in the background.

7.1 Modifcation outside dino

Since dino has to store some of the PowerDNS data localy, there are some currently some caveats when modifying data outside dino. We are working to resolve most of them in future releaes.

- deleting zones outside of dino is not supported and will result in errors
- zones can be created outside dino and will be synced
- records can be created, changed and deleted outside dino and will be shown
- concurrent editing of the same zone from within inside and outside dino is not recommended due to the nature of the PowerDNS API. Assuming a small enough time window, some records might get lost.

CHAPTER 8

User Signup

By default dino doesn't allow new users to sign up themselves. This means that users have to be created manually, using a username and password. The login data can then be passed to the user. Depending on your organisation, allowing the right users to create accounts themselves might be required.

8.1 Enabling Signup

To enable users to create accounts themselves, add the `DINO_ENABLE_EMAIL_SIGNUP` setting to your dino configuration (`/etc/dino.cfg`, or another location):

```
# enable signup via email/password
DINO_ENABLE_EMAIL_SIGNUP=True
# and/or enable social signup, see below
#DINO_ENABLE_SOCIAL_SIGNUP=True
```

Warning: Please be aware that this enables anyone with access to dino to create new, permissionless accounts. This is probably not what you want. Continue to the next section to restrict signups.

8.2 Restricting Domains

To combat the "everything can sign up"-siguration, you can restrict the mail domains, which can be used to create new accounts. This applies to both email/password accounts and social accounts. This can be used to only allow signups from your company domain.

In `/etc/dino.cfg` (or wherever you configure your dino), add the following and restart dino:

```
DINO_VALID_SIGNUP_DOMAINS=example.com
```

The above configuration allows anyone with a `...@example.com` mail address to create accounts.

8.3 Social Accounts

Dino utilizes django-allauth to enable login and signup using various providers. Most of them will not be of much use in typical dino deployments; you might find Google, GitLab or OpenID useful, if your organisation uses one of them for authentication interally.

We'll run through adding Google login to dino. The process is almost identical for most other providers; take a look at the allauth documentation for your provider for details.

First, we need to load the google provider. In `/etc/dino.cfg` (or wherever you configure your dino), add the following and restart dino:

```
DINO_LOGIN_PROVIDERS=google
# and optionally
#DINO_ENABLE_SOCIAL_SIGNUP=True
```

Now, continue with the django-allauth docs about app registration, to create a config at google and add it to dino. After those setps, the provider setup is finished.