DNS-Lexicon

Release 3.8.1

Oct 15, 2021

Table of Contents

CHAPTER 1

Introduction

1.1 Why using Lexicon?

Lexicon provides a way to manipulate DNS records on multiple DNS providers in a standardized way.

Lexicon can be used as:

- a CLI tool:

```
# Create a TXT entry in domain.net zone hosted by CloudFlare
lexicon cloudflare create domain.net TXT --name foo --content bar
```

- or a Python library:

```
# Create a TXT entry in domain.net zone hosted by CloudFlare
from lexicon.client import Client
from lexicon.config import ConfigResolver

action = {
    "provider_name" : "cloudflare",
    "action": "create",
    "domain": "domain.net",
    "type": "TXT",
    "name": "foo",
    "content": "bar",
}
config = ConfigResolver().with_env().with_dict(action)
Client(config).execute()
```

Lexicon was designed to be used in automation, specifically letsencrypt.

- Generating Intranet & Private Network SSL Certificates using Lets Encrypt & Lexicon

1.2 Supported providers

Only DNS providers who have an API can be supported by *lexicon*.

The current supported providers are:

- Aliyun.com
- AuroraDNS
- AWS Route53
- Azure DNS
- Cloudflare
- ClouDNS
- CloudXNS
- ConoHa
- Constellix
- DigitalOcean
- Dinahosting
- DirectAdmin
- DNSimple v1, v2
- DnsMadeEasy
- DNSPark
- DNSPod
- Dreamhost
- Dynu
- EasyDNS
- Easyname
- EUserv
- ExoScale
- Gandi RPC (old) / LiveAPI
- Gehirn
- Glesys
- GoDaddy
- Google Cloud DNS
- Gransy (sites subreg.cz, regtons.com and regnames.eu)
- Hover
- Hurricane Electric DNS
- Hetzner
- Infoblox

- Infomaniak
- Internet.bs
- INWX
- Joker.com
- Linode
- Linode v4
- LuaDNS
- Memset
- Mythic Beasts (v2 API)
- Njalla
- Namecheap
- Namesilo
- Netcup
- NFSN (NearlyFreeSpeech)
- NS1
- OnApp
- Online
- OVH
- Plesk
- PointHQ
- PowerDNS
- Rackspace
- Rage4
- RcodeZero
- RFC2136
- Sakura Cloud by SAKURA Internet Inc.
- SafeDNS by UKFast
- SoftLayer
- Transip
- UltraDNS
- Vercel
- Vultr
- Yandex
- Zilore
- Zonomi

CHAPTER 2

User guide

2.1 Installation

Warning: It is strongly advised with pip to install Lexicon in a Python virtual environment, in order to avoid interference between Python modules preinstalled on your system as OS packages and modules installed by pip (see https://docs.python-guide.org/dev/virtualenvs/).

To use lexicon as a CLI application, do the following:

```
$ pip install dns-lexicon
```

Some providers (like Route53 and TransIP) require additional dependencies. You can install the provider specific dependencies separately:

```
$ pip install dns-lexicon[route53]
```

To install lexicon with the additional dependencies of every provider, do the following:

```
$ pip install dns-lexicon[full]
```

You can also install the latest version from the repository directly.

```
$ pip install git+https://github.com/AnalogJ/lexicon.git
```

and with Route 53 provider dependencies:

```
$ pip install git+https://github.com/AnalogJ/lexicon.git#egg=dns-lexicon[route53]
```

Note: As an alternative you can also install Lexicon using the OS packages available for major Linux distributions (see *lexicon* or *dns-lexicon* package in https://pkgs.org/download/lexicon).

2.2 Usage

```
$ lexicon -h
  usage: lexicon [-h] [--version] [--delegated DELEGATED]
                 {cloudflare,cloudxns,digitalocean,dnsimple,dnsmadeeasy,dnspark,
↪dnspod,easydns,luadns,namesilo,nsone,pointhq,rage4,route53,vultr,yandex,zonomi}
                 ...

  Create, Update, Delete, List DNS entries

  positional arguments:
    {cloudflare,cloudxns,digitalocean,dnsimple,dnsmadeeasy,dnspark,dnspod,easydns,
↪luadns,namesilo,nsone,pointhq,rage4,route53,vultr,yandex,zonomi}
                          specify the DNS provider to use
      cloudflare          cloudflare provider
      cloudxns            cloudxns provider
      digitalocean        digitalocean provider
    ...
      rage4               rage4 provider
      route53             route53 provider
      vultr               vultr provider
      yandex              yandex provider
      zonomi              zonomi provider

  optional arguments:
    -h, --help            show this help message and exit
    --version             show the current version of lexicon
    --delegated DELEGATED
                          specify the delegated domain


  $ lexicon cloudflare -h
  usage: lexicon cloudflare [-h] [--name NAME] [--content CONTENT] [--ttl TTL]
                            [--priority PRIORITY] [--identifier IDENTIFIER]
                            [--auth-username AUTH_USERNAME]
                            [--auth-token AUTH_TOKEN]
                            {create,list,update,delete} domain
                            {A,AAAA,CNAME,MX,NS,SPF,SOA,TXT,SRV,LOC}

  positional arguments:
    {create,list,update,delete}
                          specify the action to take
    domain                specify the domain, supports subdomains as well
    {A,AAAA,CNAME,MX,NS,SPF,SOA,TXT,SRV,LOC}
                          specify the entry type

  optional arguments:
    -h, --help            show this help message and exit
    --name NAME           specify the record name
    --content CONTENT     specify the record content
    --ttl TTL             specify the record time-to-live
    --priority PRIORITY   specify the record priority
    --identifier IDENTIFIER
                          specify the record for update or delete actions
    --auth-username AUTH_USERNAME
                          specify email address used to authenticate
    --auth-token AUTH_TOKEN
```

(continues on next page)

(continued from previous page)

```
                        specify token used authenticate

                        specify the entry type

  optional arguments:
    -h, --help          show this help message and exit
    --name NAME         specify the record name
    --content CONTENT   specify the record content
    --ttl TTL           specify the record time-to-live
    --priority PRIORITY specify the record priority
    --identifier IDENTIFIER
                        specify the record for update or delete actions
    --auth-username AUTH_USERNAME
                        specify email address used to authenticate
    --auth-token AUTH_TOKEN
                        specify token used authenticate
```

Using the lexicon CLI is pretty simple:

```
# setup provider environmental variables:
export LEXICON_CLOUDFLARE_USERNAME="myusername@example.com"
export LEXICON_CLOUDFLARE_TOKEN="cloudflare-api-token"

# list all TXT records on cloudflare
lexicon cloudflare list example.com TXT

# create a new TXT record on cloudflare
lexicon cloudflare create www.example.com TXT --name="_acme-challenge.www.example.com.
↪" --content="challenge token"

# delete a  TXT record on cloudflare
lexicon cloudflare delete www.example.com TXT --name="_acme-challenge.www.example.com.
↪" --content="challenge token"
lexicon cloudflare delete www.example.com TXT --identifier="cloudflare record id"
```

2.3 Configuration

2.3.1 Authentication

Most supported DNS services provide an API token, however each service implements authentication differently. Lexicon attempts to standardize authentication around the following CLI flags:

- `--auth-username` - For DNS services that require it, this is usually the account id or email address
- `--auth-password` - For DNS services that do not provide an API token, this is usually the account password
- `--auth-token` - This is the most common auth method, the API token provided by the DNS service

You can see all the `--auth-*` flags for a specific service by reading the DNS service specific help: `lexicon cloudflare -h`

2.3.2 Environmental variables

Instead of providing authentication information via the CLI, you can also specify them via environmental variables. Every DNS service and auth flag maps to an environmental variable as follows: `LEXICON_{DNS Provider Name}_{Auth Type}`

So instead of specifying `--auth-username` and `--auth-token` flags when calling `lexicon cloudflare ...`, you could instead set the `LEXICON_CLOUDFLARE_USERNAME` and `LEXICON_CLOUDFLARE_TOKEN` environmental variables.

If you've got a subdomain delegation configured and need records configured within that (eg, you're trying to set `test.foo.example.com` where `foo.example.com` is configured as a separate zone), set `LEXICON_DELEGATED` to the delegated domain.

```
LEXICON_DELEGATED=foo.example.com
```

2.3.3 TLD cache

The tldextract library is used by Lexicon to find the actual domain name from the provided FQDN (eg. `domain.net` is the actual domain in `www.domain.net`). Lexicon stores `tldextract` cache by default in `~/.lexicon_tld_set` where `~` is the current user's home directory. You can change this path using the `LEXICON_TLDEXTRACT_CACHE` environment variable.

For instance, to store `tldextract` cache in `/my/path/to/tld_cache`, you can invoke Lexicon like this from a Linux shell:

```
LEXICON_TLDEXTRACT_CACHE=/my/path/to/tld_cache lexicon myprovider create www.example.
↪net TXT ...
```

2.4 Integration

Lexicon can be integrated with various tools and process to help handling DNS records.

2.4.1 Let'sEncrypt instructions

Lexicon has an example dehydrated hook file that you can use for any supported provider. All you need to do is set the PROVIDER env variable.

```
PROVIDER=cloudflare dehydrated --cron --hook dehydrated.default.sh --challenge dns-01
```

Lexicon can also be used with Certbot and the included Certbot hook file (requires configuration).

2.4.2 Docker

There is an included example Dockerfile that can be used to automatically generate certificates for your website.

CHAPTER 3

Configuration reference

3.1 Providers options

3.1.1 Providers available

The following Lexicon providers are available:

aliyun	*aurora*	*azure*	*cloudflare*
cloudns	*cloudxns*	*conoha*	*constellix*
ddns	*digitalocean*	*dinahosting*	*directadmin*
dnsimple	*dnsmadeeasy*	*dnspark*	*dnspod*
dreamhost	*dynu*	*easydns*	*easyname*
euserv	*exoscale*	*gandi*	*gehirn*
glesys	*godaddy*	*googleclouddns*	*gransy*
gratisdns	*henet*	*hetzner*	*hostingde*
hover	*infoblox*	*infomaniak*	*internetbs*
inwx	*joker*	*linode*	*linode4*
localzone	*luadns*	*memset*	*mythicbeasts*
namecheap	*namesilo*	*netcup*	*nfsn*
njalla	*nsone*	*oci*	*onapp*
online	*ovh*	*plesk*	*pointhq*
powerdns	*rackspace*	*rage4*	*rcodezero*
route53	*safedns*	*sakuracloud*	*softlayer*
transip	*ultradns*	*vercel*	*vultr*
yandex	*zeit*	*zilore*	*zonomi*

3.1.2 List of options

aliyun

- `auth_key_id` Specify access key id for authentication
- `auth_secret` Specify access secret for authentication

aurora

- `auth_api_key` Specify api key for authentication
- `auth_secret_key` Specify the secret key for authentication

azure

- `auth_client_id` Specify the client id (aka application id) of the app registration
- `auth_client_secret` Specify the client secret of the app registration
- `auth_tenant_id` Specify the tenant id (aka directory id) of the app registration
- `auth_subscription_id` Specify the subscription id attached to the resource group
- `resource_group` Specify the resource group hosting the dns zone to edit

cloudflare

- `auth_username` Specify email address for authentication (for global api key only)
- `auth_token` Specify token for authentication (global api key or api token)
- `zone_id` Specify the zone id (if set, api token can be scoped to the target zone)

cloudns

- `auth_id` Specify user id for authentication
- `auth_subid` Specify subuser id for authentication
- `auth_subuser` Specify subuser name for authentication
- `auth_password` Specify password for authentication
- `weight` Specify the srv record weight
- `port` Specify the srv record port

cloudxns

- `auth_username` Specify api-key for authentication
- `auth_token` Specify secret-key for authentication

conoha

- `auth_region` Specify region. if empty, region 'tyo1' will be used.
- `auth_token` Specify token for authentication. if empty, the username and password will be used to create a token.
- `auth_username` Specify api username for authentication. only used if –auth-token is empty.
- `auth_password` Specify api user password for authentication. only used if –auth-token is empty.
- `auth_tenant_id` Specify tenand id for authentication. only used if –auth-token is empty.

constellix

- `auth_username` Specify the api key username for authentication
- `auth_token` Specify secret key for authenticate=

ddns

- `auth_token` Specify the key used in format <alg>:<key_id>:<secret>
- `ddns_server` Specify ip of the ddns server

digitalocean

- `auth_token` Specify token for authentication

dinahosting

- `auth_username` Specify username for authentication
- `auth_password` Specify password for authentication

directadmin

- `auth_password` Specify password for authentication (or login key for two-factor authentication)
- `auth_username` Specify username for authentication
- `endpoint` Specify the directadmin endpoint

dnsimple

- `auth_token` Specify api token for authentication
- `auth_username` Specify email address for authentication
- `auth_password` Specify password for authentication
- `auth_2fa` Specify two-factor auth token (otp) to use with email/password authentication

dnsmadeeasy

- `auth_username` Specify username for authentication
- `auth_token` Specify token for authentication

dnspark

- `auth_username` Specify api key for authentication
- `auth_token` Specify token for authentication

dnspod

- `auth_username` Specify api id for authentication
- `auth_token` Specify token for authentication

dreamhost

- `auth_token` Specify api key for authentication

dynu

- `auth_token` Specify api key for authentication

easydns

- `auth_username` Specify username for authentication
- `auth_token` Specify token for authentication

easyname

- `auth_username` Specify username used to authenticate
- `auth_password` Specify password used to authenticate

euserv

- `auth_username` Specify email address for authentication
- `auth_password` Specify password for authentication

exoscale

- `auth_key` Specify api key for authentication
- `auth_secret` Specify api secret for authentication

gandi

- `auth_token` Specify gandi api key
- `api_protocol` (optional) specify gandi api protocol to use: rpc (default) or rest

gehirn

- `auth_token` Specify access token for authentication
- `auth_secret` Specify access secret for authentication

glesys

- `auth_username` Specify username (cl12345)
- `auth_token` Specify api key

godaddy

- `auth_key` Specify the key to access the api
- `auth_secret` Specify the secret to access the api

googleclouddns

- **`auth_service_account_info`** specify the service account info in the google json format: can be either the path of a file prefixed by 'file::' (eg. file::/tmp/service_account_info.json) or the base64 encoded content of this file prefixed by 'base64::' (eg. base64::eyjhbgcioyj...)

gransy

- `auth_username` Specify username for authentication
- `auth_password` Specify password for authentication

gratisdns

- `auth_username` Specify email address for authentication
- `auth_password` Specify password for authentication

henet

- `auth_username` Specify username for authentication
- `auth_password` Specify password for authentication

hetzner

- `auth_token` Specify hetzner dns api token

hostingde

- `auth_token` Specify api key for authentication

hover

- `auth_username` Specify username for authentication
- `auth_password` Specify password for authentication

infoblox

- `auth_user` Specify the user to access the infoblox wapi
- `auth_psw` Specify the password to access the infoblox wapi
- `ib_view` Specify dns view to manage at the infoblox
- `ib_host` Specify infoblox host exposing the wapi

infomaniak

- `auth_token` Specify the token

internetbs

- `auth_key` Specify api key for authentication
- `auth_password` Specify password for authentication

inwx

- `auth_username` Specify username for authentication
- `auth_password` Specify password for authentication

joker

- `auth_token` Specify the api key to connect to the joker.com api

linode

- `auth_token` Specify api key for authentication

linode4

- `auth_token` Specify api key for authentication

localzone

- `filename` Specify location of zone master file

luadns

- `auth_username` Specify email address for authentication
- `auth_token` Specify token for authentication

memset

- `auth_token` Specify api key for authentication

mythicbeasts

- `auth_username` Specify api credentials username
- `auth_password` Specify api credentials password
- `auth_token` Specify api token for authentication

namecheap

- `auth_token` Specify api token for authentication
- `auth_username` Specify username for authentication
- `auth_client_ip` Client ip address to send to namecheap api calls
- `auth_sandbox` Whether to use the sandbox server

namesilo

- `auth_token` Specify key for authentication

netcup

- `auth_customer_id` Specify customer number for authentication
- `auth_api_key` Specify api key for authentication
- `auth_api_password` Specify api password for authentication

nfsn

- `auth_username` Specify username used to authenticate
- `auth_token` Specify token used to authenticate

njalla

- `auth_token` Specify api token for authentication

nsone

- `auth_token` Specify token for authentication

oci

- `auth_config_file` The full path including filename to an oci configuration file.
- `auth_user` The ocid of the user calling the api.
- `auth_tenancy` The ocid of your tenancy.
- `auth_fingerprint` The fingerprint for the public key that was added to the calling user.
- `auth_key_content` The full content of the calling user's private signing key in pem format.
- `auth_pass_phrase` If the private key is encrypted, the pass phrase must be provided.
- `auth_region` The home region of your tenancy.
- `auth_type` Valid options are 'api_key' (default) or 'instance_principal'.

onapp

- `auth_username` Specify email address of the onapp account
- `auth_token` Specify api key for the onapp account
- `auth_server` Specify url to the onapp control panel server

online

- `auth_token` Specify private api token

ovh

- `auth_entrypoint` Specify the ovh entrypoint
- `auth_application_key` Specify the application key
- `auth_application_secret` Specify the application secret
- `auth_consumer_key` Specify the consumer key

plesk

- `auth_username` Specify username for authentication
- `auth_password` Specify password for authentication
- `plesk_server` Specify url to the plesk web ui, including the port

pointhq

- `auth_username` Specify email address for authentication
- `auth_token` Specify token for authentication

powerdns

- `auth_token` Specify token for authentication
- `pdns_server` Uri for powerdns server
- `pdns_server_id` Server id to interact with
- `pdns_disable_notify` Disable slave notifications from master

rackspace

- `auth_account` Specify account number for authentication
- `auth_username` Specify username for authentication. only used if –auth-token is empty.
- `auth_api_key` Specify api key for authentication. only used if –auth-token is empty.
- `auth_token` Specify token for authentication. if empty, the username and api key will be used to create a token.
- `sleep_time` Number of seconds to wait between update requests.

rage4

- `auth_username` Specify email address for authentication
- `auth_token` Specify token for authentication

rcodezero

- `auth_token` Specify token for authentication

route53

- `auth_access_key` Specify access_key for authentication
- `auth_access_secret` Specify access_secret for authentication
- `private_zone` Indicates what kind of hosted zone to use. if true, use only private zones. if false, use only public zones
- `auth_username` Alternative way to specify the access_key for authentication
- `auth_token` Alternative way to specify the access_secret for authentication

safedns

- `auth_token` Specify the api key to authenticate with

sakuracloud

- `auth_token` Specify access token for authentication
- `auth_secret` Specify access secret for authentication

softlayer

- `auth_username` Specify username for authentication
- `auth_api_key` Specify api private key for authentication

transip

- `auth_username` Specify username for authentication

- `auth_api_key` Specify api private key for authentication

ultradns

- `auth_token` Specify token for authentication; if not set –auth-token, –auth-password are used
- `auth_username` Specify username for authentication
- `auth_password` Specify password for authentication

vercel

- `auth_token` Specify your api token

vultr

- `auth_token` Specify token for authentication

yandex

- `auth_token` Specify pdd token (https://tech.yandex.com/domain/doc/concepts/access-docpage/)

zeit

- `auth_token` Specify your api token

zilore

- `auth_key` Specify the zilore api key to use

zonomi

- `auth_token` Specify token for authentication
- `auth_entrypoint` Use zonomi or rimuhosting api

3.2 Passing provider options to Lexicon

There are three ways to pass a provider option to Lexicon (we suppose here that the provider option is named `auth_token`):

- by **CLI flag**: set the flag `--auth-token` to Lexicon while invoking it, for instance:

```
$ lexicon cloudflare create domain.net TXT --name foo --content bar --auth-token␣
↪YOUR_TOKEN
```

- by **environment variable**: set the environment variable `LEXICON_CLOUDFLARE_AUTH_TOKEN`, for instance:

```
$ LEXICON_CLOUDFLARE_AUTH_TOKEN=YOUR_TOKEN cloudflare create domain.net TXT --
↪name foo --content bar
```

- by **configuration file**: construct a configuration file containing the provider options, for instance:

```
$ cat /path/to/config/lexicon.yml
cloudflare:
  auth_token: YOUR_TOKEN
$ lexicon cloudflare create domain.net TXT --name foo --content bar --config-dir /
↪path/to/config
```

Note: Lexicon will look for two types of configuration files in the provided path to `--config-dir` (current workdir by default): a general configuration file named `lexicon.yml` and a provider-specific configuration file named `lexicon_[PROVIDER_NAME].yml`.

For a general configuration file, provider options need be set under a key named after the provider:

```
# /path/to/config/lexicon.yml
clouflare:
  auth_token: YOUR_TOKEN
```

For a provider-specific configuration file, provider options need to be set at the root:

```
# /path/to/config/lexicon_cloudflare.yml
auth_token: YOUR_TOKEN
```

3.3 Passing general options to Lexicon

General options are options not specific to a provider, like `delegated`. They can be passed like the provider options (by CLI, by environment variable or by configuration file). Please note that for configuration file, options will be set at the root, and cannot be set in provider-specific configuration files.

```
# /path/to/config/lexicon.yml
delegated: domain.net
cloudflare:
  ...
```

3.4 The `auto` provider

The `auto` provider is a special provider. It resolves dynamically the actual provider to use based on the domain provided to Lexicon. To do so, it resolves the nameservers that serve the DNS zone for this domain, and find the relevant DNS provider based on an internal map that associates each DNS provider to its known nameservers.

Basically if `domain.net` is served by CloudFlare, and a TXT entry needs to be inserted in this domain, you can use the following command:

```
lexicon auto create domain.net TXT --name foo --content bar
```

The options specific to the actual provider that will be used still need to be set, by CLI flags, environment variables or configuration files. However for CLI, each option name will be prefixed with `[ACTUAL_PROVIDER]-` when passed to `auto`. For instance, the `auth_token` option for `cloudflare` will be passed using `--cloudflare-auth-token`.

CHAPTER 4

Developer guide

Thanks! There are tons of different DNS services, and unfortunately a large portion of them require paid accounts, which makes it hard for us to develop `lexicon` providers on our own. We want to keep it as easy as possible to contribute to `lexicon`, so that you can automate your favorite DNS service. There are a few guidelines that we need contributors to follow so that we can keep on top of things.

4.1 Potential providers

Potential providers are as follows. If you would like to contribute one, please follow the current document instructions and open a pull request.

- AHNames
- DurableDNS (?? Can't set TXT records ??)
- cyon.ch
- Dyn ($$ requires paid account $$)
- EntryDNS ($$ requires paid account $$)
- FreeDNS
- Host Virtual DNS ($$ requires paid account $$)
- HostEurope
- Infoblox NIOS
- ironDNS ($$ requires paid account $$)
- ISPConfig
- InternetX autoDNS
- KingHost
- Liquidweb ($$ requires paid account $$)

- Loopia ($$ requires paid account $$)
- NFSN (NearlyFreeSpeech) ($$ requires paid account $$)
- Servercow
- selectel.com
- TELE3
- UltraDNS ($$ requires paid account $$)
- UnoEuro API
- VSCALE
- WorldWideDns ($$ requires paid account $$)
- Zerigo ($$ requires paid account $$)
- Zoneedit
- **Any others I missed**

4.2 Setup a development environment

Fork, then clone the repo:

```
$ git clone git@github.com:your-username/lexicon.git
```

Install Poetry if you not have it already:

```
$ curl -sSL https://raw.githubusercontent.com/python-poetry/poetry/master/get-poetry.
↪py | python
```

Configure the virtual environment with full providers support and activate it:

```
$ cd lexicon
$ poetry install -E full
$ source .venv/bin/activate
```

Make sure the tests pass:

```
$ tox -e py
```

You can test a specific provider using:

```
$ pytest lexicon/tests/providers/test_foo.py
```

Note: Please note that by default, tests are replayed from recordings located in `tests/fixtures/cassettes`, not against the real DNS provider APIs.

4.3 Adding a new DNS provider

Now that you have a working development environment, lets add a new provider. Internally lexicon does a bit of magic to wire everything together, so the only thing you'll really need to do is is create the following file.

- `lexicon/providers/foo.py`

Where `foo` should be replaced with the name of the DNS service in lowercase and without spaces or special characters (eg. `cloudflare`)

Your provider file should contain 3 things:

- a `NAMESERVER_DOMAINS` which contains the domain(s) used by the DNS provider nameservers FQDNs (eg. Google Cloud DNS uses nameservers that have the FQDN pattern `ns-cloud-cX-googledomains.com`, so `NAMESERVER_DOMAINS` will be `['googledomains.com']`).
- a `provider_parser` which is used to add provider specific commandline arguments. eg. If you define two cli arguments: `--auth-username` and `--auth-token`, those values will be available to your provider via `self._get_provider_option('auth_username')` or `self._get_provider_option('auth_token')` respectively
- a `Provider` class which inherits from BaseProvider, which is in the `base.py` file. The BaseProvider defines the following functions, which must be overridden in your provider implementation:
  - `_authenticate`
  - `_create_record`
  - `_list_records`
  - `_update_record`
  - `_delete_record`
  - `_request`

  It also provides a few helper functions which you can use to simplify your implementation. See the cloudflare.py file, or any provider in the lexicon/providers/ folder for examples

It's a good idea to review the provider specification to ensure that your interface follows the proper conventions.

Note: Please keep in mind the following:

- `lexicon` is designed to work with multiple versions of python. That means your code will be tested against python 3.6 and 3.8 on Windows, Linux and Mac OS X.
- any provider specific dependencies should be added to the `setup.py` file, under the `extra_requires` heading. The group name should be the name of the provider. eg:

```
extras_require={
    'route53': ['boto3']
}
```

4.4 Testing your provider

4.4.1 Test against the live API

First let's validate that your provider shows up in the CLI.

```
$ lexicon foo --help
```

If everything worked correctly, you should get a help page that's specific to your provider, including your custom optional arguments.

Now you can run some manual commands against your provider to verify that everything works as you expect.

```
$ lexicon foo list example.com TXT
$ lexicon foo create example.com TXT --name demo --content "fake content"
```

Once you're satisfied that your provider is working correctly, we'll run the integration test suite against it, and verify that your provider responds the same as all other `lexicon` providers. `lexicon` uses `vcrpy` to make recordings of actual HTTP requests against your DNS service's API, and then reuses those recordings during testing.

The only thing you need to do is create the following file:

- `lexicon/tests/providers/test_foo.py`

Then you'll need to populate it with the following template:

```
# Test for one implementation of the interface
from lexicon.tests.providers.integration_tests import IntegrationTestsV2
from unittest import TestCase

# Hook into testing framework by inheriting unittest.TestCase and reuse
# the tests which *each and every* implementation of the interface must
# pass, by inheritance from integration_tests.IntegrationTests
class FooProviderTests(TestCase, IntegrationTestsV2):
    """Integration tests for Foo provider"""
    provider_name = 'foo'
    domain = 'example.com'
    def _filter_post_data_parameters(self):
        return ['login_token']

    def _filter_headers(self):
        return ['Authorization']

    def _filter_query_parameters(self):
        return ['secret_key']

    def _filter_response(self, response):
        """See `IntegrationTests._filter_response` for more information on how
        to filter the provider response."""
        return response
```

Make sure to replace any instance of `foo` or `Foo` with your provider name. `domain` should be a real domain registered with your provider (some providers have a sandbox/test environment which doesn't require you to validate ownership).

The `_filter_*` methods ensure that your credentials are not included in the `vcrpy` recordings that are created. You can take a look at recordings for other providers, they are stored in the tests/fixtures/cassettes/ sub-folders.

Then you'll need to setup your environment variables for testing. Unlike running `lexicon` via the CLI, the test suite cannot take user input, so we'll need to provide any CLI arguments containing secrets (like `--auth-*`) using environmental variables prefixed with `LEXICON_FOO_`.

For instance, if you had a `--auth-token` CLI argument, you can populate it using the `LEXICON_FOO_AUTH_TOKEN` environmental variable.

Notice also that you should pass any required non-secrets arguments programmatically using the `_test_parameters_override()` method. See test_powerdns.py for an example.

4.4.2 Test recordings

Now you need to run the `py.test` suite again, but in a different mode: the live tests mode. In default test mode, tests are replayed from existing recordings. In live mode, tests are executed against the real DNS provider API, and recordings will automatically be generated for your provider.

To execute the `py.test` suite using the live tests mode, execute py.test with the environment variable `LEXICON_LIVE_TESTS` set to `true` like below:

```
LEXICON_LIVE_TESTS=true pytest lexicon/tests/providers/test_foo.py
```

If any of the integration tests fail on your provider, you'll need to delete the recordings that were created, make your changes and then try again.

```
rm -rf tests/fixtures/cassettes/foo/IntegrationTests
```

Once all your tests pass, you'll want to double check that there is no sensitive data in the `tests/fixtures/cassettes/foo/IntegrationTests` folder, and then `git add` the whole folder.

```
git add tests/fixtures/cassettes/foo/IntegrationTests
```

Finally, push your changes to your Github fork, and open a PR.

4.4.3 Skipping Tests/Suites

Neither of the snippets below should be used unless necessary. They are only included in the interest of documentation.

In your `lexicon/tests/providers/test_foo.py` file, you can use `@pytest.mark.skip` to skip any individual test that does not apply (and will never pass)

```
@pytest.mark.skip(reason="can not set ttl when creating/updating records")
def test_provider_when_calling_list_records_after_setting_ttl(self):
    return
```

You can also skip extended test suites by inheriting your provider test class from `IntegrationTestsV1` instead of `IntegrationTestsV2`:

```
from lexicon.tests.providers.integration_tests import IntegrationTestsV1
from unittest import TestCase

class FooProviderTests(TestCase, IntegrationTestsV1):
    """Integration tests for Foo provider"""
```

4.5 CODEOWNERS file

Finally you should add yourself to the CODEOWNERS file, in the root of the repo. It's my way of keeping track of who to ping when I need updated recordings as the test suites expand & change.

CHAPTER 5

Provider specification

5.1 General

- **name** Clients should provide FQDN. Providers should handle both FQDN and relative names.
- **ttl** Reasonable default is 6 hours since it's supported by most services. Any service that does not support this must be explicitly mentioned somewhere.
- **record** All provider/API records must be translated to the following format:

```
{
    'id': string, // optional, provider specified unique id. Clients to treat this as␣
↪opaque.
    'type': string, // upper case, valid record type. eg. A, CNAME, TXT
    'name': string, // lowercase, FQDN. eg. test.record.example.com
    'ttl': integer, // positive integer, in seconds. eg. 3600
    'content': string, //double quoted/escaped values should be unescaped. eg. "\"TXT␣
↪content\"" should become "TXT content"
    'options': {
        'mx': { // MX options
            'priority': integer
        }
    }
}
```

5.2 API Operations

5.2.1 create_record

- **Normal Behavior** Create a new DNS record. Return a boolean `True` if successful.
- **If Record Already Exists** Do nothing. **DO NOT** throw exception.

- **TTL** If not specified or set to `0`, use reasonable default.
- **Record Sets** If service supports record sets, create new record set or append value to existing record set as required.

5.2.2 list_record

- **Normal Behaviour** List all records. If filters are provided, send to the API if possible, else apply filter locally. Return value should be a list of records.
- **Record Sets** Ungroup record sets into individual records. Eg: If a record set contains 3 values, provider ungroup them into 3 different records.
- **Linked Records** For services that support some form of linked record, do not resolve, treat as CNAME.

5.2.3 update_record

- **Normal Behaviour** Update a record. Record to be updated can be specified by providing id OR name, type and content. Return a boolean `True` if successful.
- **Record Sets** If matched record is part of a record set, only update the record that matches. Update the record set so that records other than the matched one are unmodified.
- **TTL**
  - If not specified, do not modify ttl.
  - If set to `0`, reset to reasonable default.
- **No Match** Throw exception?

5.2.4 delete_record

- **Normal Behaviour** Remove a record. Record to be deleted can be specified by providing id OR name, type and content. Return a boolean `True` if successful.
- **Record sets** Remove only the record that matches all the filters.
  - If content is not specified, remove the record set.
  - If length of record set becomes 0 after removing record, remove the record set.
  - Otherwise, remove only the value that matches and leave other records as-is.
- **No Match** Do nothing. **DO NOT** throw exception