DNStats

Matthew Burket

Mar 31, 2021

CONTENTS:

CHAPTER

ONE

INSTALL

1. **Install Dependencies** You will need the following packages:
    - Python 3.7 or later
    - **Python3 headers**
        - `python3-dev` on Debian/Ubuntu
        - `python-devel` on Fedora/Centos/
    - **pip**
        - python3-pip
    - **Libevent headers** `libevent-devel` on Fedora/Centos
    - **postgresql**
        - `postgresql-server`
        - `postgresql-contrib`
        - `postgresql`
    - **RabbitMQ**
        - See https://www.rabbitmq.com/download.html
2. Ensure Postgres and RabbitMQ are running

#. Create Database In a ``psql` shell run

create user dnstats with password 'changeme!'; create database dnstats owner dnstats;

See the Postgres doc for more details on how to configure Postgres.

1. Clone the repo:

```
git clone git@git.assignitapp.com:dnstats/dnstats.git
```

2. Change into the repo:

```
cd dnstats
```

3. Copy config:

```
cp dnstats.src.env dnstats.env
```

4. Edit dnstats.src.env

   Update `AMQP`, `` `DB` ``, and `CELERY_BACKEND`

5. Install virtualenv:

```
pip3 install virtualenv --user
```

6. Create virtualenv:

```
virtualenv -p python3 venv
```

7. Active venv:

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

8. Install Python Dependencies:

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

9. Load config:

```
source sendgrid.env
```

10. Seed Database:

```
python -m dnstats.db.seed
```

11. Start celery:

```
celery -A dnstats.celery worker
```

12. Run Task for seeding sites

CHAPTER

TWO

POSTGRES INSTALL

The default Postgres config is not really suitable for the DNStats work download. The follow are some changes that should help.

2.1 postgresql.conf

1. Setup max_connections Default 300, should be set to 500 or 1000 for DNStats.
2. shared_buffers Set to 128 MB

2.2 pg_hba.conf

See Postgresql docs.

CHAPTER THREE

GRADING

3.1 Methods

The grading methods shall be in a package under grading. For example, DMARC grading will be *dnstats.grading.dmarc*. The method shall be called *dnstats.grading.dmarc.grade*

CHAPTER FOUR

SPF GRADING

SPF grading starts getting a ceiling from the final qualifier. Then the grading method will use `dnstats.dnsvlidate.spf.validate()` to ensue the record is valid. If the record is invalid the grading method shall return 0, this is due to RFC 7208, Section 4.6.

4.1 Policy Grading

Policy	Grade
Fail	100
Soft-fail	75
Neutral	50
Default Neutral	40

4.2 Demerits

CHAPTER

FIVE

DNSTATS

5.1 dnstats.charts

5.1.1 dnstats.charts.colors

5.1.2 dnstats.charts.asset_utils

5.2 dnstats.utils

`dnstats.utils.`**`chunks`**(*array*, *size*)

Source: https://chrisalbon.com/python/data_wrangling/break_list_into_chunks_of_equal_size/

`dnstats.utils.`**`validate_fqdn`**(*fqdn: str*) → bool

Give a string checks if its a Fully qualified domain name (FQDN).

This regex is based Django's validator for URls.

Copyright (c) Django Software Foundation and individual contributors. All rights reserved.

Parameters **`fqdn`** –

Returns bool if input is a FQDN

`dnstats.utils.`**`validate_url`**(*url: str*) → bool

This regex is based Django's validator for URls.

Copyright (c) Django Software Foundation and individual contributors. All rights reserved.

Parameters **`url`** –

Returns if the input is a url

5.3 dnstats.dnsutils

5.3.1 dnstats.dnsutils.ns

5.4 dnstats.dnsvaildate

5.4.1 dnstats.dnsvalidate.caa

class `dnstats.dnsvalidate.caa.`**`CAAErrors`**(*value*)

An enumeration.

`INVALID_FLAG = 2`

`INVALID_PROPERTY_STRUCTURE = 0`

`INVALID_TAG = 3`

`INVALID_VALUE = 4`

`IODEF_INVALID_EMAIL = 8`

`IODEF_INVALID_URL = 9`

`IODEF_NO_SCHEME = 7`

`ISSUEWILD_DOMAIN_INVALID = 10`

`ISSUE_DOMAIN_INVALID = 11`

`NO_CAA_RECORDS = 1`

`TAG_TOO_LONG = 12`

`VALUE_NOT_QUOTED = 6`

`VALUE_QUOTE_ERROR = 5`

class `dnstats.dnsvalidate.caa.`**`Caa`**(*caa_records: list*, *domain: str*)

DNS validation for CAA

property errors

property iodef

property issue

property issuewild

`dnstats.dnsvalidate.caa.`**`validate`**(*caa_result_set: list*, *domain: str*) → dict

Validate a CAA record set based on RFC 8659

Parameters

- **`caa_result_set`** – a list of CAA records as str
- **`domain`** – the domain of the CAA records as str

Returns dict

5.4.2 dnstats.dnsvalidate.dmarc

class dnstats.dnsvalidate.dmarc.**Dmarc**(*dmarc_result_set*)

” Validates and represents a DMARC record

property adkim

property aspf

property errors

property fo

property is_valid

property p

Policy of the DMARC policy :return:

property pct

property rua

property ruf

property sp

class dnstats.dnsvalidate.dmarc.**DmarcErrors**(*value*)

An enumeration.

INVALID_ADKIM_VALUE = 1

INVALID_ASPF_VALUE = 2

INVALID_DMARC_RECORD = 0

INVALID_DMARC_RECORD_START = 10

INVALID_FAILURE_REPORTING_VALUE = 3

INVALID_PCT_VALUE = 9

INVALID_POLICY = 4

INVALID_RF_VALUE = 7

INVALID_RI_VALUE = 8

INVALID_SUBDOMAIN_POLICY = 5

MULTIPLE_DMARC_RECORDS = 6

NO_DMARC_RECORD = 11

dnstats.dnsvalidate.dmarc.**validate**(*dmarc_result_set: list*) → dict

5.4.3 dnstats.dnsvalidate.ns

class dnstats.dnsvalidate.ns.**Ns**(*ns_records: list*, *ns_ip_addresses: dict*, *ns_server_ns_results: dict*, *domain: str*)

Validate NS records for a given domain.

class dnstats.dnsvalidate.ns.**NsErrors**(*value*)

An enumeration.

NAMESERVER_HAS_INVALID_RESPONSE = 4

NAMESERVER_HAS_NO_A = 3

```
    NAMESERVER_IS_NOT_PUBLIC = 5

    NAME_SERVER_MISMATCH = 7

    NO_NAME_SERVERS_RETURNED = 6

    NO_NS_RECORDS = 0

    NULL_NS_RECORD = 2

    ONLY_ONE_NS_RECORD = 1
```

5.4.4 dnstats.dnsvalidate.soa

5.4.5 dnstats.dnsvalidate.spf

CHAPTER

SIX

ARCH

CHAPTER

SEVEN

SCANNING ARCH

The section will explain how the scan process works. It will setup through the process.

7.1 Kick off

The scan is kicked off by Celery Beat, exact time can be found in the beat setup method for celery.py. Beat calls `dnstats.celery.do_run()`. `dnstats.celery.do_run()` makes a run. If `DNSTATS_ENV` is `Development` the run will only run for top few hundred or so sites. If that is not set the scan will scan the whole one million sites. This method also sends the start of scan email. Once the email as been sent `dnstas.celery.launch_run()` is called with the id of the run just created.

7.2 Queuing

Now that `dnstas.celery.launch_run()` has been called the run is looked up. This method grabs the site needs for the scan. Next the list of sites is broken into lists of 10,000 sites. We loop over the list of lists, the outer loop. In a inner loop we celery chain of `dnstas.celery.site_stat()` into `dnstas.celery.process_result()`. Each group of 10,000 is in a celery group. Once the the outer loop is done the server ends a done with scanning message.

7.3 Scanning

All of the scanning is sent to the `gevent` queue. The name has made when I was tryng to use gevent for forking model for the scans, but that failed. The `gevent` queue should have lots of power. As of writing I 10 worker vms with 6 celery worker each that have 25 concurrency each. Since this task is mostly network bound this makes sense.

Only thing that should happen in this stage is do dns look ups. All analysis should happen in later stages. The `dnstas.celery.site_stat()` returns a `dict` and then celery passes this to `dnstas.celery.process_result()`.

7.4 Process Results

This stage uses the `celery` (default) queue. This queue is much less powershell than the `gevent` (scanning) queue. In production I have about 10 celery vms with 1 celery worker with about 20 concurrency each.

In this stage celery passes the `dict` from `dnstas.celery.site_stat()` into `dnstas.celery.process_result()`. The first stage is build the `dnstats.models.SiteRun` and then save it. Next, all the grading methods are called async, i.e. enqueues the grading jobs.

7.5 Grading

This stage uses the `celery` queue.

The grading method gets the `dnstats.models.SiteRun` based on the passed `dnstats.models.SiteRun` id passed. Then it gets the site. Does the grading logic and updates the `dnstats.models.SiteRun` with the grades. Once the grading is saved the remarks (or errors from the grading and validation process) are saved to the database.

CHAPTER

EIGHT

ABOUT DNSTATS

8.1 Glossary

DMARC DMARC or Domain-based Message Authentication, Reporting and Conformance

CAA DNS Certification Authority Authorization

SPF SPF or Sender Policy Framework is method to help detect fraudulent emails. See **RFC 7208**.

Certificate Authority Certificate Authority or CA

DNSSEC DNS SEC

DNS Domain Name Service

CHAPTER NINE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

INDEX