10-Week Internship at Lawrence Berkeley National Laboratory

2018 :: Edsel Norwood II

Shreyas Cholia :: Lawrence Berkeley National Laboratory :: Data Science & Technology

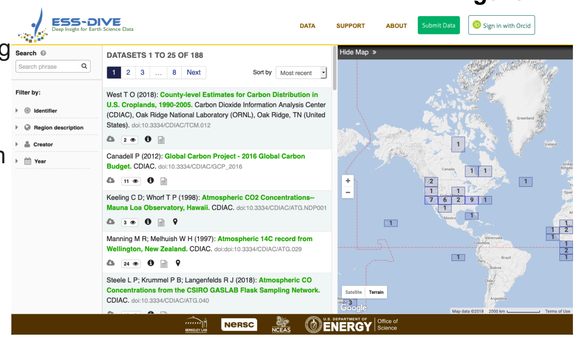
Abstract

During the summer of 2018 I took part in a 10-week undergraduate internship at Lawrence Berkeley National Laboratory (LBL). The department in which I was working under was the Data Science and Technology department, directed by Deborah Agarwal. During the course of the internship I was tasked with the development of user tools for a service by the name of Environmental Systems Science Data Infrastructure for a Virtual Ecosystem (ESS-DIVE). This included working on multiple different projects during the 10 weeks at LBL.

What is ESS-DIVE

ESS-DIVE *(shown in figure 1)* is a data archiving service that is funded by the U.S. Department of Energy (DOE). The purpose of ESS-DIVE is to archive and publicly share data obtained from observational, experimental, and modeling research that is funded by the DOE's Office of Science.

Figure 1



Virus Total Command Line Tool

The internship began with the task of using a REST API from a virus scanning service called VirusTotal to create a command line tool *(shown in figure 2)*. REST API stands for representational state transfer application program interface. The purpose of this project was to have a working prototype that could later be implemented into a larger project. The command line tool was written in Python and made use of a library called argparser. This library allowed the use of several different"flags" within a terminal, all of which allowed for easy access to the different services and functions within the VirusTotal API. Along with learning REST API, this project introduced me to SQLite3 database management and command line argument parsing. The end-goal is to use the VirusTotal API to automatically scan files that are uploaded to ESS-Dive. This is important because the ESS-Dive archive is hosted on LBL's National Energy Research Scientific Computing Center (NERSC) and allowing users to upload potentially harmful files is a major security risk.

Figure 2

```
usage: vt.py [-h] [-db] [-clean] [-version] [-post POST] [-update] [-get GET]
             [-item ITEM]

optional arguments:
  -h, --help  show this help message and exit
  -db         Will print the current state of the database that houses
              informationon each file passed through the api.
  -clean      Will delete all records in the current database.
  -version    Program version
  -post POST  Used to send file to virustotal api to be scanned. | python
              vt.py -post <filepath> |
  -update     Will update current file status or create new record in existing
              database
  -get GET    Value should contain either the keyword <sha> when sending a
              file's full path or the keyword <resource> when sending a file's
              resource id. | python vt.py -get sha -item <filepath> | python
              vt.py -get resource -item <file resource id> |
  -item ITEM  Value should contain either resource id for the desired file or
              the file's full path.
```

ESS-DIVE Team Members

Shreyas Cholia - Group Lead
Val Hendrix – Senior Software Engineer
Fianna O'Brien – Software Engineer
Jon Hays – Student Assistant

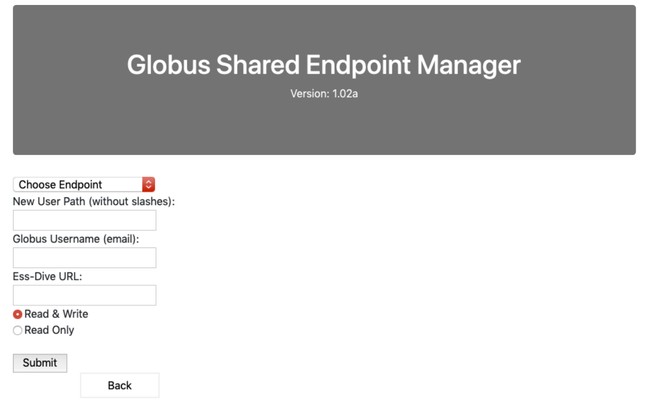
Virus Total Web Application

Following the first project I was tasked to create a web application that implemented the core functionality of the previous project. The purpose of this web app was to create a more user friendly GUI (graphical user interface) using the VirusTotal API *(code snippet shown in figure 3)*. This was achieved using the Flask, which is a micro framework for Python, and other HTML based web development tools such as AJAX and JavaScript. This web app used a lot of the old functionality from the command line tool while also adding extended functionally. Each file that is sent to the VirusTotal API has four possible statuses that it can be in and they are: "file sent", "safe", "threat", and "queued". The status (along with additional information) of every file sent through this web app is stored in a SQLite3 database. "Safe", "threat", and "queued" are fairly self explanatory but "file sent" is slightly different. The status "file sent" the user has successfully sent a file to the API but a status has not yet been retrieved from the API. The user can also choose to update all files that hold a status of "file sent" or "queued", as an alternative the user can also update each file individually on the database viewing page.

Figure 3

```python
def post_file(filename, key, base='https://www.virustotal.com/vtapi/v2/file/scan'):
    """
        This function receives an api key, and file path. It then sends a post request to the VirusTotal api
        and parses through the response for either a new value in the varible resource or the default value.
        :param filename: holds the path for a given file
        :param key: holds the user's unique api key
        :param base: holds the url that hosts virustotal's file scan
        :return: will return either the resource id a file or a None value
        """
    try:
        # initialize resource as zero to later check for a change in value
        resource = 0

        # set parameters and read in file
        params = {'apikey': key}
        files = {'file': (filename, open(filename, 'rb'))}

        # post file to virus total api
        response = requests.post(base, files=files, params=params)

        # check response status
        if response.status_code == 204:
            logging.error('ERROR: API Resonded "Too Many Requests"')

        # parse json for response content
        parsed_json = json.loads(response.content)

        # return resource id for report
        resource = (parsed_json['resource'])
        return resource

    except IOError:

        logging.info('File: ' + filename)
        logging.error('ERROR: File Could Not Be Found')
```

Figure 4



Globus Shared Endpoint Form

The final project that worked on for LBL involved using the Globus Transfer API to create a second web application that allowed a user to create a shared folder on the ESS-Dive shared endpoint *(shown in figure 4)*. The issue that this web application intends to solve is once the VirusTotal API is implemented into ESS-Dive any datasets that are larger than 35 megabytes won't be able to be sent to the VirusTotal API. This is due to a restriction on VirusTotal's side. With this functionality a user will be able to fill out a form with their Globus information and create a shared folder on the ESS-Dive shared endpoint, from there they can upload their dataset to the endpoint and notify the ESS-Dive team once they're done. Once the ESS-Dive team has been notified that the user is done upload their data they can then manually submit the files to be scanned with VirusTotal.

Conclusion

In conclusion, the experience gained from my internship at LBL will come to benefit me in the upcoming years of not only my college career, but also my professional development. I am extremely grateful for the opportunity to work with such an amazing team on a project that has such a significant impact.