Stephen Smoogen <ssmoogen@redhat.com>

The Great Backyard Computer Count

Better estimates of systems in the wild.



# 

# Background

In order for a community to know which products it should invest in, it needs to have an understanding of which ones are being used and where. The Fedora Project gathers its usage statistics using IP address hits from the update tool. This masks when multiple servers are behind a proxy or when a host moves from ip address to another due to network changes. The data is then placed in flat files which can take days to reanalyze old data in order to make new queries of “what country is growing”, “what architecture and release are shrinking”, or “what are the combined trends of that data”. Finally the data is massaged further to draw graphs of usage via shell scripts.

The Great Backyard Computer Count (GBCC) is meant to address these concerns and some others. First by adding a UUID plugin to the system update utility, users can be better counted. Second by moving most of the data into an SQL database, better queries can be done ‘on the fly’ as they are needed. Also new architectures can be found and added to the database as needed without needing to reprocess old data. Finally by automating some of the graphing and data collection, the data can be shown in a web interface to the people needing access.

The document is current as of the 0.5 version as tagged in the main github repository

<https://github.com/smooge/GreatBackyardComputerCount>

# Great Backyard Computer Count Training Guide

System Overview

The software is run inside of Fedora Infrastructure and related organizations to allow for more accurate counting of systems getting updates for Fedora, EPEL, and CentOS operating systems. The code is currently written in Python 2.7 and runs in a CentOS 7 environment. Future versions will be rewritten to work with Python 3.x as needed. In order to ensure that the installation and usage is able to be replicated, all minimum requirements for libraries are included in the requirements.txt file.

The current software author and point of contact is Stephen Smoogen < [smooge@fedoraproject.org](mailto:smooge@fedoraproject.org) > and tickets can be opened for feature requests or problems in the upstream source code repository.

## Document References

* Great Backyard Computer Count (GBCC) <https://github.com/smooge/GreatBackyardComputerCount/>
* Fedora Project <https://getfedora.org/>
* CentOS Linux <https://centos.org/>
* Fedora Infrastructure <https://pagure.io/fedora-infrastructure/>
* Mirror manager <https://github.com/fedora-infra/mirrormanager2/>

# Training Prerequisites

For installation and maintenance of the software, a moderate knowledge of Python, databases, and how the site uses mirror manager or similar software to store existing files. For usage of the command line tool, all that is needed is to know where the related log files are. For viewing current data counts, the ability to use a web browser, a knowledge of where the GBCC server is running, and having proper access (accepted login/password) to that website.

## Installation Training

We will go over how to install the software in a standalone testing mode. First you will need to have a Linux operating system with Python 2.7 installed on it. All testing was done on CentOS 7.5 so that will be the version used in the documentation.

1. Install OS and set up basic user.
2. Install the software packages git and set up python virtual environment.

$ sudo -i

Password: <>

# yum install epel-release

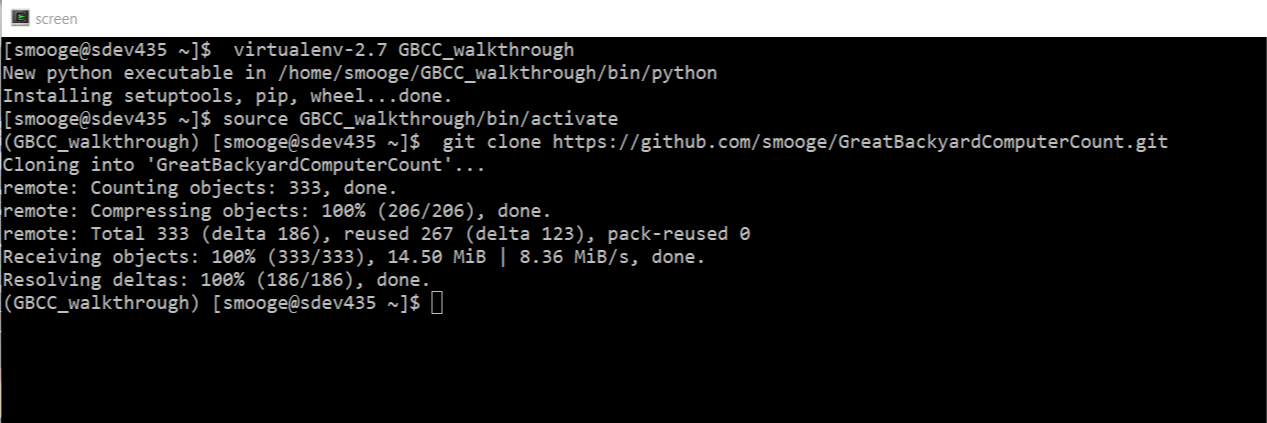
# yum install git python-virtualenv python2-geoip2

# exit

$ virtualenv-2.7 GBCC\_walkthrough

$ source GBCC\_walkthrough/bin/activate

$ git clone <https://github.com/smooge/GreatBackyardComputerCount.git>



3. Download a copy of the MaxmindDB geolocation database you want to use. Testing was done with a paid copy but the free version should also work: <https://dev.maxmind.com/geoip/geoip2/geolite2/>

4. Edit the default configs to the appropriate directories for your site.

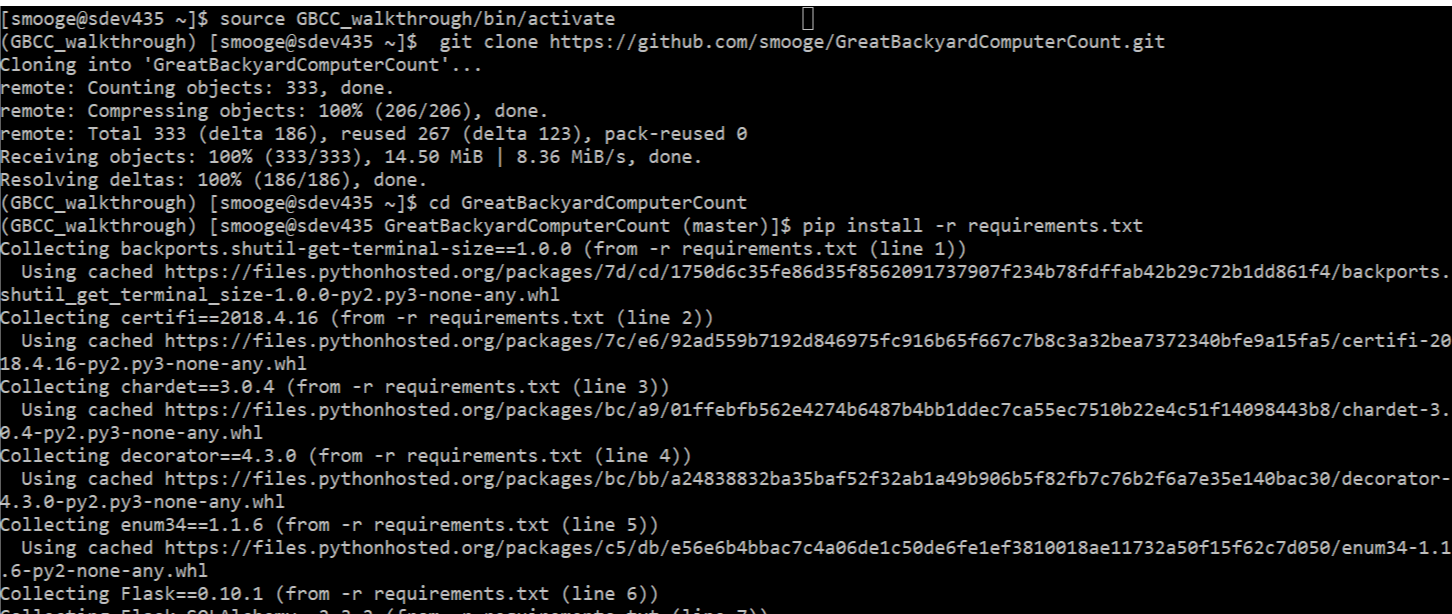
(GBCC\_walkthrough) $ cd GreatBackyardComputerCount

(GBCC\_walkthrough) $ vi GreatBackyardComputerCount/config.py

change the values of basedir and any other variables needed. It defaults to using the basedir as the tree for log-files and such.

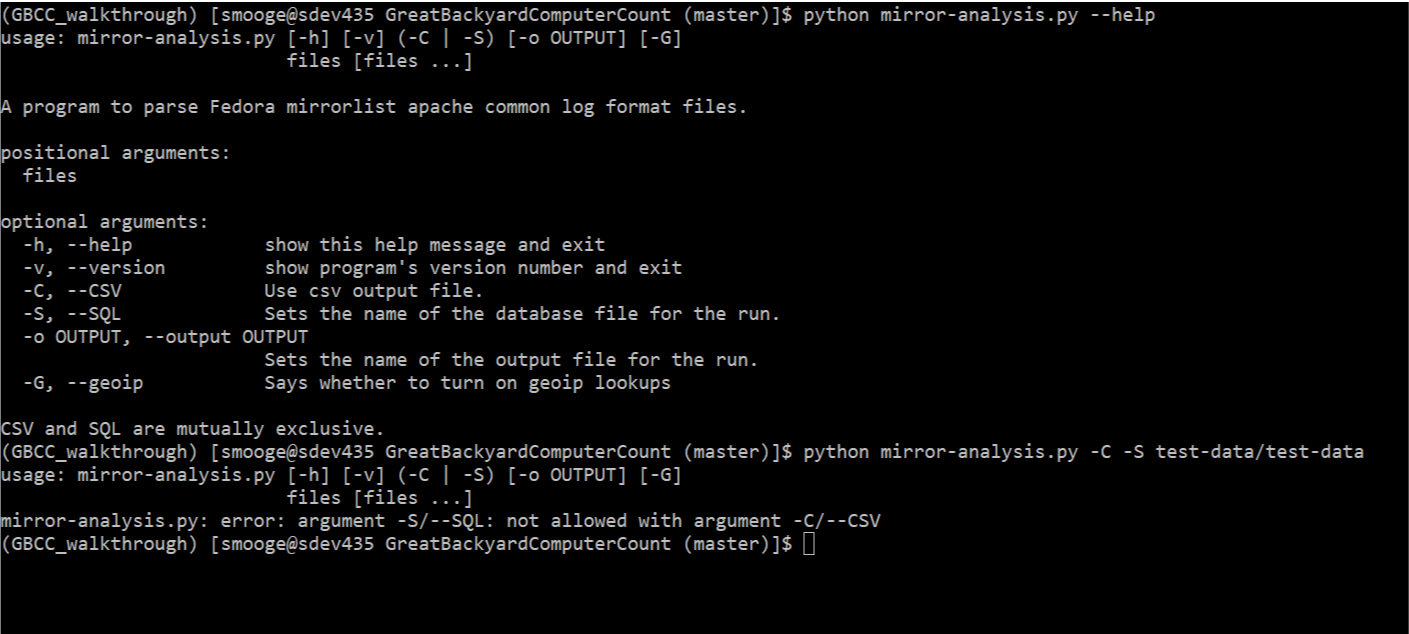
1. You can now set the working environment with pip

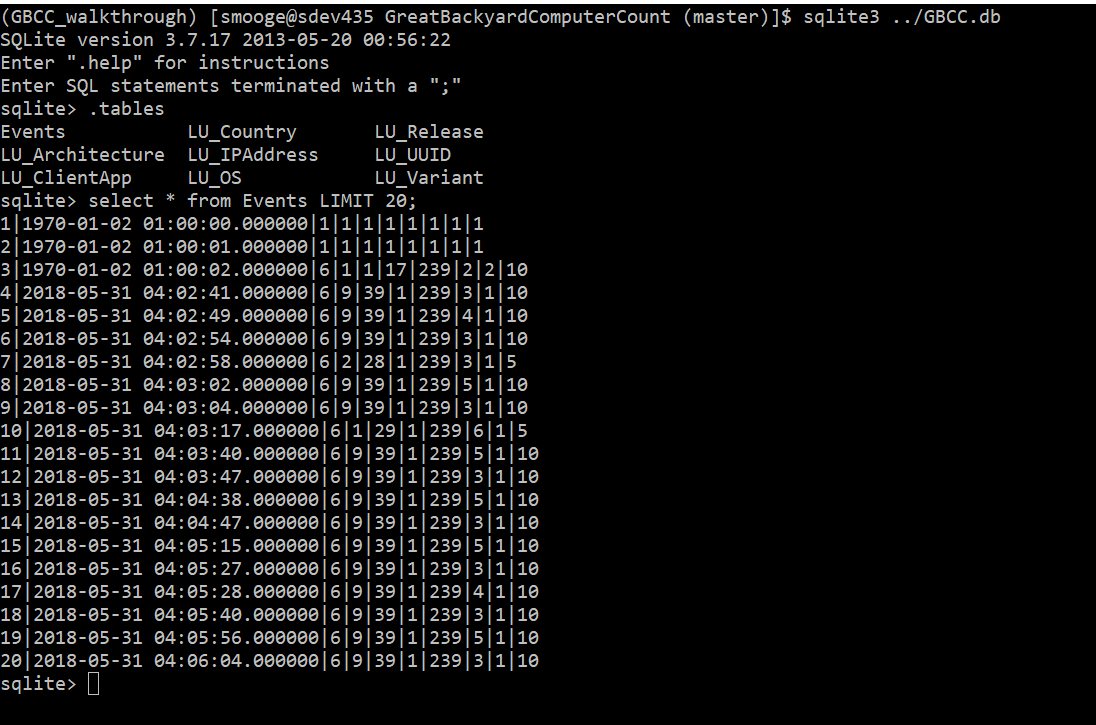
(GBCC\_walkthrough) $ pip install -r requirements.txt



1. To test that the program will work, you can now test the

operation of the basic log analysis tool:





7. Next we initialize the databases. By default the system uses sqlite for its usage. Further work would be needed to make it works with postgres or mysql.

(GBCC\_walkthrough)$ python ./initialize\_db.py

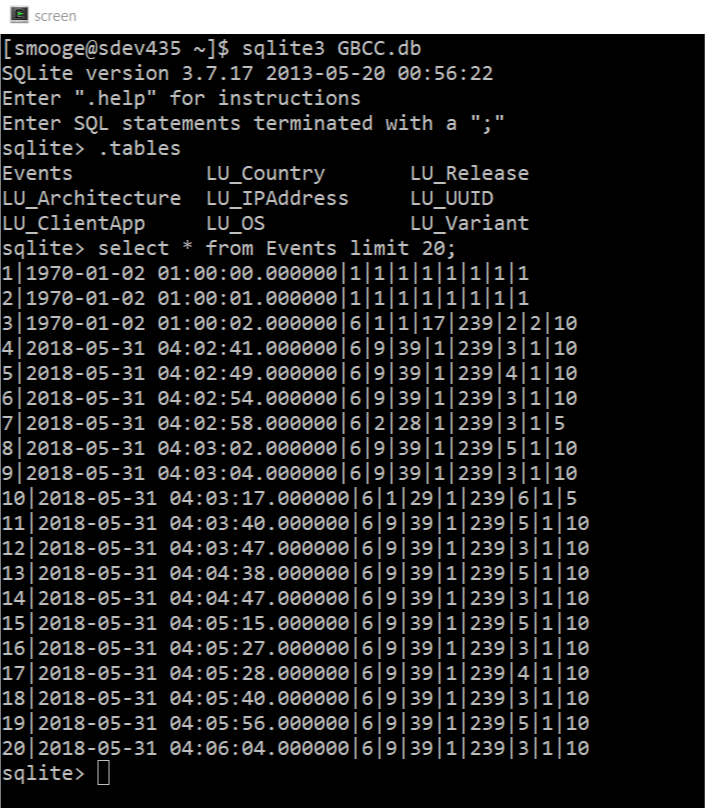
8. Log data can now be imported into the database. Data usually has the form the apache common log format with the important data being on entry 7: /mirrorlist?repo=epel-7&arch=x86\_64

152.19.134.142 - - [31/May/2018:04:02:41 +0000] "GET /mirrorlist?repo=epel-7&arch=x86\_64 HTTP/1.1" 200 2701 "-" "urlgrabber/3.10 yum/3.4.3"

This says what kind of request was made (asked for mirrorlist versus metalink) What the repository was (repo=epel-7) and what the architecture of the data was (arch=x86\_64). Other data is taken from the ip address, and the last area which is the string data of which client was used.

9. Data can now be imported into the database using the mirror-analysis script

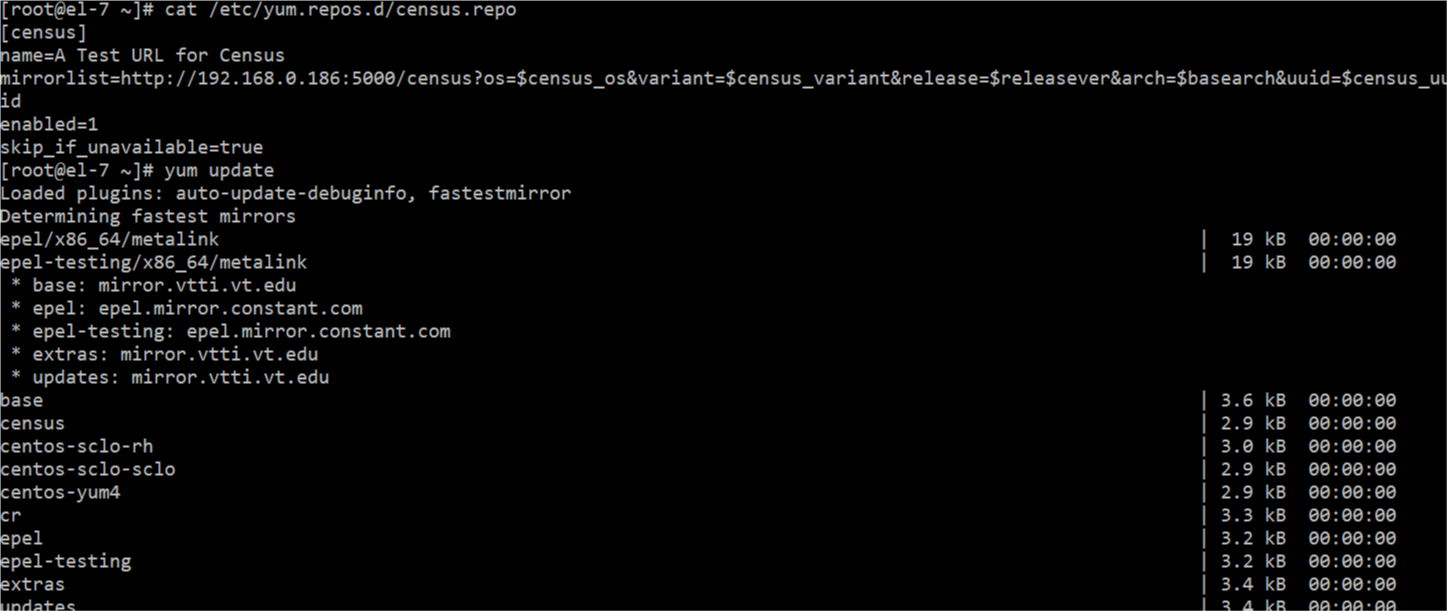
(GBCC\_walkthrough)$ python ./mirror-analysis.py -S -G test-data/test-data



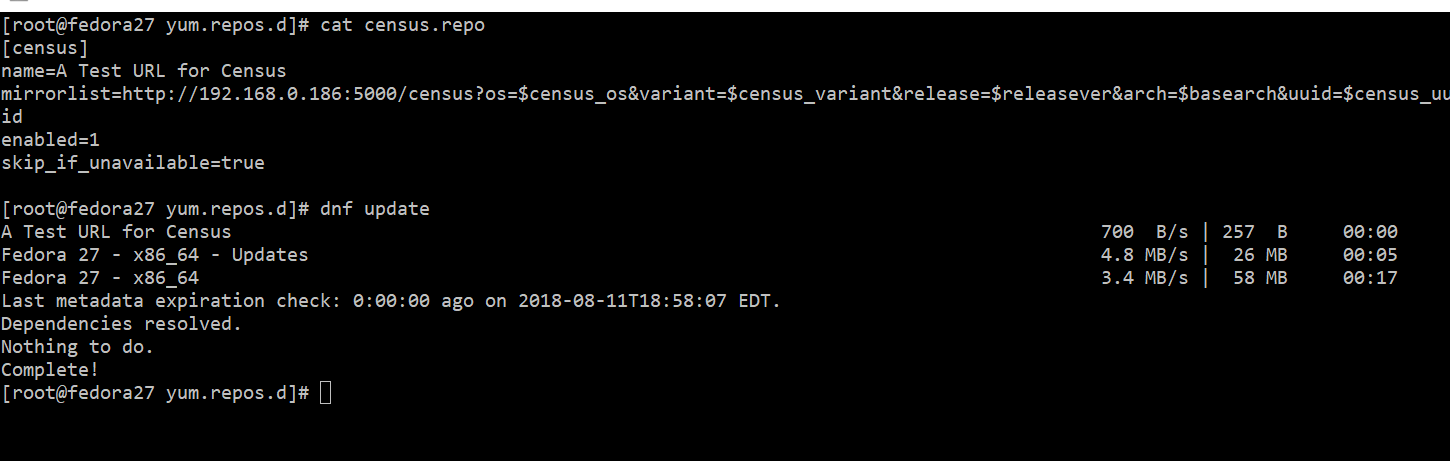
Currently this can take a long time because of the combined slowness of sql lookup/inserts in SQL-lite and the GEOIP lookups. A 13 MB file can take 40 minutes to load.

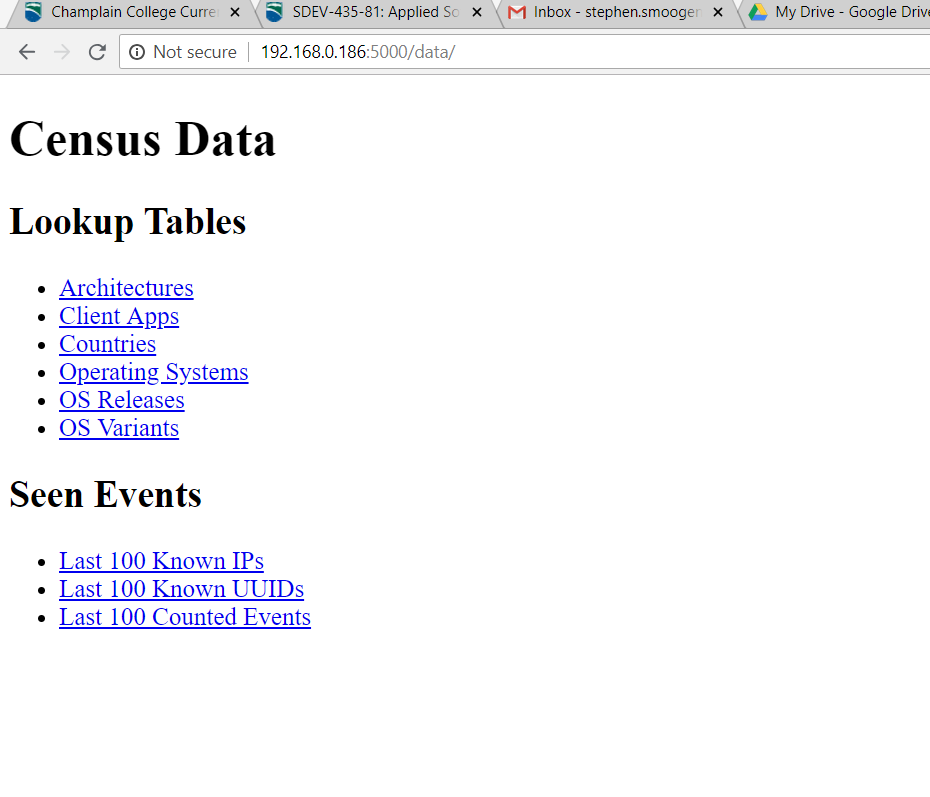
## Application Running

Once the application has been setup it can be run as a webserver using the supplied command: runserver.py. This will use the python built in webserver and look for requests on port 5000. This can now be checked with a browser going to http://<ipaddress>:5000/data/ which will give pages for what can be currently queried from the database. Checkins can now also be done with clients using a yum repository file:

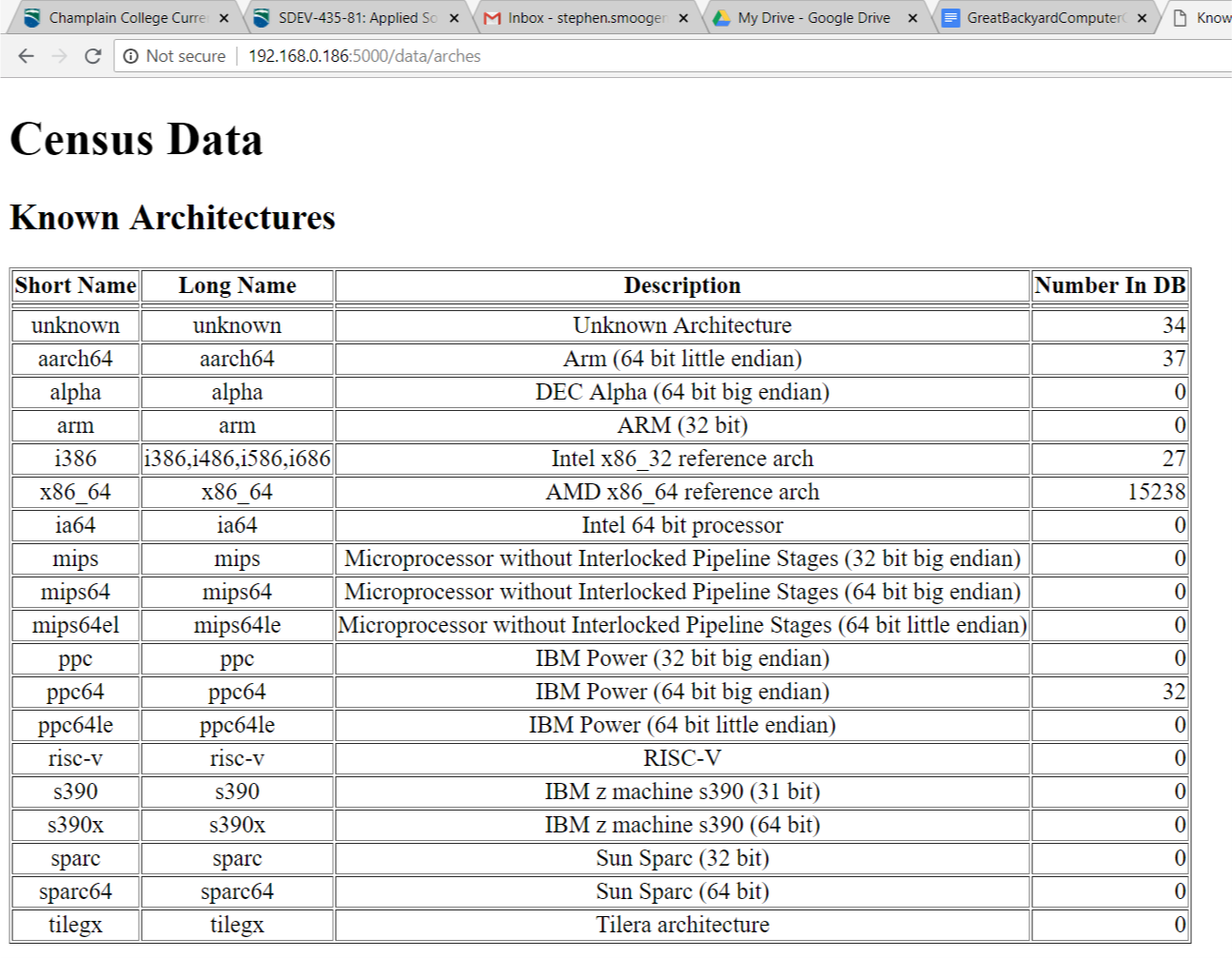


Examples of running on Fedora 27 and CentOS-7

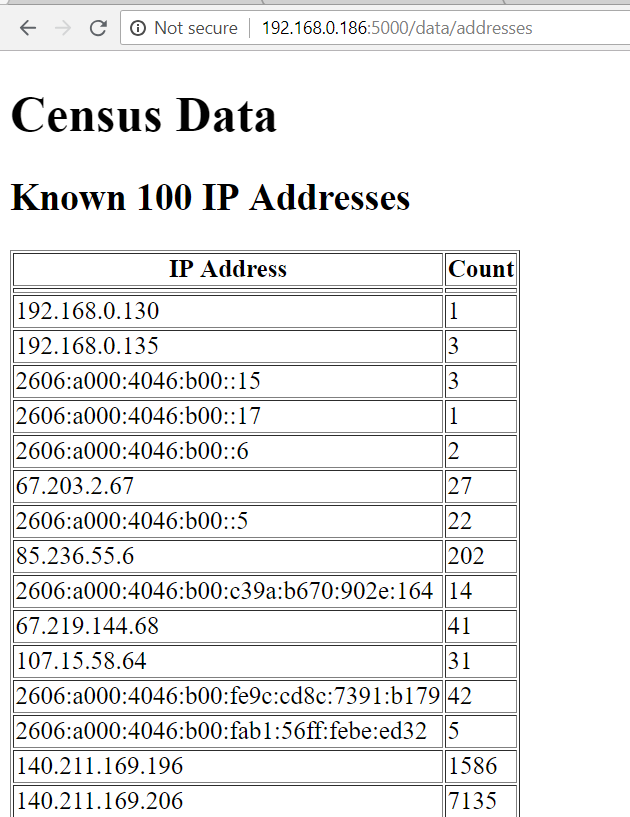




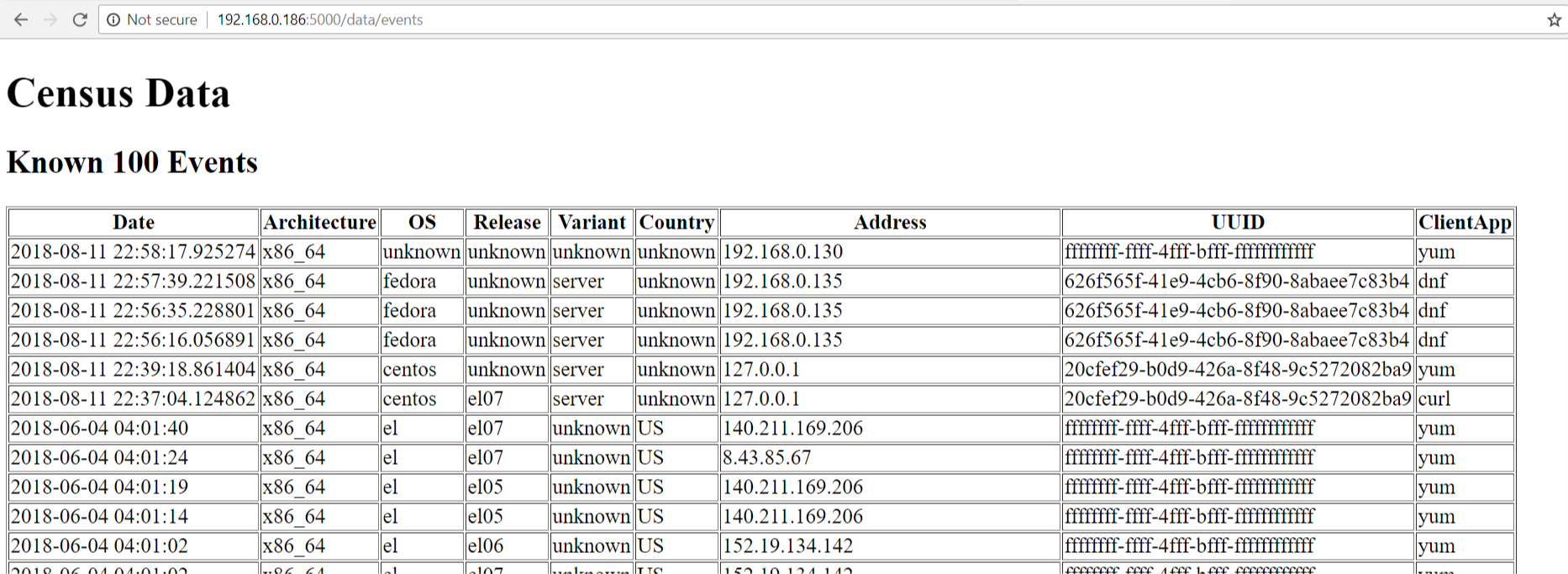
Main lookup page of the application



Example of lookup data



Example ip addresses loaded into database



Example of main event data to show that events are being recorded by the web application. The data is in reverse data order. This version of the server is a prototype and not meant for full load testing yet. It will need improvements as listed in the maintenance section to finish.

# Maintenance Outline

This section covers items which must be completed for the product to go from 0.5 to 1.0 release.

1. SQL speedups. Currently the software is very slow on inserts due to the multiple lookups which are being done before this. The easiest solution will be to load each of the lookup pages into hash table with the name as the key and the pkid as the datum. This would cut down the number of lookups to the DB down by 5 (uuid and ip address will need to still be looked up or created). Another solution would be to put the insert into a stored procedure which then requires that it is linked to the type of database used (aka mysql vs postgres vs...) Other speedups may be possible due to the data layout.
2. SQL Table updates (maintenance). Currently the database has no idea of which version it is running against. It will need to be refactored to use alembic versions so that future updates and changes to the database can be automated with outside scripts.
3. Adding lookups. Currently the SQL lookup tables require additional entries when new releases, architectures or other regular expressions are added. This will be something that needs to be added every several releases.
4. Database size maintenance. Currently we store every event in a log file as a separate event. This means that our entire database would have around 146 billion events stored in the main table. That is incredibly unwieldy and will need a way to either archive off or store only the data needed per day. Originally this was going to be a 'count' stat for the tuple (date, ip, uuid, arch, repo, variant, client) so that if that showed up 20 times it would only have 1 record with a 20 in it. [This would cover the majority of duplicate records] However the speed to lookup and then update the record was slower than adding another record so we are doing a straight dump and then will do queries to cut out noise.
5. Graphing. Currently the data reports that are given are incredibly weak. Built in reports using specific queries and feeding to a javascript UI in the client fell out of scope for the initial timeline and will need to be implemented in the next section.
6. Testing. Currently all testing is done manually with limited failure modes. The next task will be to reimplement all code with a wrapper set of test-units on each routine.

Date: 2018-08-11

Proposed 1.0 release date: 2018-12-11.