Installing and Configuring R.A.P.I.D Web App

2017

Table of Contents

[I. Launching an AWS EC2 Instance 2](#_Toc485989990)

[II. RAPID Web Application Installation 2](#_Toc485989991)

[III.Celery Tasks Troubleshooting Steps 5](#_Toc485989992)

[IV. Launching an AWS RDS Instance 7](#_Toc485989993)

[V. RAPID Database Installation 8](#_Toc485989994)

[VI. Attaching an AWS EBS Volume to an Instance: 11](#_Toc485989995)

[VII. Making an Amazon Volume Available for Use: 12](#_Toc485989996)

[VIII. Extending Live Sessions for SSH connections: 13](#_Toc485989997)

[IX. Enabling Log Rotation for Celery logs: 13](#_Toc485989998)

[X. Installing Miscellaneous Libraries: 14](#_Toc485989999)

**Installing and Configuring Rapid Web App**

# I. Launching an AWS EC2 Instance

Follow the instructions to install and configure an AWS EC2 Instance running Ubuntu.

1. In the AWS Console, go to Services 🡪 Compute 🡪 EC2
2. Launch an Instance
3. In the Amazon Machine Image (AMI) page, select Ubuntu Server 16.04 Volume type.
4. In the Instance Type page, select Type: t2.large
5. Review and Launch the instance. Select Security Group ‘rapid-sec-group’ and accept defaults for other settings.
6. Note the Availability Zone and copy it to the clipboard for later. This will be needed when configuring the RDS server.
7. Review the inbound rules for the Security group rapid-sec-group. Ensure that ports 22 and 443 are open to any IP.
8. Ensure that an Elastic IP address is attached to the new instance so if it’s restarted, you don’t lose the IP address.

# II. RAPID Web Application Installation

Follow the instructions to install and configure the RAPID Web Application on the web server

1. Log into the AWS Ubuntu Web Server via SSH.

* Ssh –i “new\_scrappy.pem” **ubuntu@ec2-54-71-4-244.us-west-2.compute.amazonaws.com**

1. Checkout the RAPID repo from github

* git clone <https://github.com/gdit-cnd/RAPID> RAPID

1. Checkout the develop or desired branch

* Git checkout <branch\_name>

1. Create a secrets.json file in the $HOME\RAPID directory and copy and paste the contents from the production version.
2. Update the correct path for secrets.json in $HOME\RAPID\settings\base.py file

Ie., with open(“**/<home>/ RAPID/secrets.json**”, ”r”)

1. Update the $HOME\RAPID\settings\local.py file with the correct environment info

BASE\_SITE\_URL = ‘http://0.0.0.0:8000”sudo

ALLOWED\_HOSTS = [’0.0.0.0’]

LOGGING = {

…

'file': {

…

'filename': '/<home>/ RAPID/RAPID.log',

}…

1. Update the $HOME\RAPID\settings\production.py file

BASE\_SITE\_URL = ‘http://0.0.0.0”

ALLOWED\_HOSTS = [’0.0.0.0’, ‘ec2-0-0-0-0.us-west-2.compute.amazonaws.com']

ADMINS = (('RAPID-Admin’, 'rapidpivot@gmail.com'),)

LOGGING = {

…

'file': {

…

'filename':'/<home>/RAPID/RAPID.log',

1. Update the APPLICATION\_DIR parameter in the install.sh file
2. Add permissions to $HOME\RAPID\install.sh

* **chmod +x install.sh**

1. Run install.sh as administrator

* **sudo ./install.sh**

1. Create symbolic links for libraries.
   1. Create the folder structure /var/www/rapid
   2. In the folder /var/www/rapid, create a symbolic link for the media and static physical folder structures using the sudo ln -s command

EXAMPLE:

sudo ln -s /<home>/RAPID/static/DataTables-1.10.5/media /var/www/rapid/media

sudo ln -s /<home>/RAPID/static /var/www/rapid/static

1. Generate and Configure SSL Certs

Copy the production SSL certs over to the following locations /etc/apache2/ssl

As an alternative, you can generate SSL Certs (self-signed certificate) using openssl

(Please reference link https://www.digitalocean.com/community/tutorials/how-to-create-a-ssl-certificate-on-apache-for-ubuntu-14-04)

1. Create the /etc/apache2/ssl directory

sudo mkdir /etc/apache2/ssl

1. Change directory to /etc/apache2/ssl and generate key and certificate using openssl

cd /etc/apache2/ssl

EXAMPLE:

sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/apache2/ssl/apache.key -out /etc/apache2/ssl/apache.crt

When you hit enter, you will be asked a number of questions.

Country Name (2 letter code) [AU]:US

State or Province Name (full name) [Some-State]:Virginia

Locality Name (eg, city) []:Reston

Organization Name (eg, company) [Internet Widgits Pty Ltd]:DIA

Organizational Unit Name (eg, section) []:CND

Common Name (e.g. server FQDN or YOUR name) []:<Webserver\_IP\_Address>

Email Address []:

verify following files are created in the /etc/apache2/ssl directory

apache.key

apache.crt

1. Configure Rapid configuration files
   1. Copy the following example templates located at: /external\_configs/apache2/ into the /etc/apache2/sites-available directory

rapid-ssl.conf

rapid.conf

* 1. Update the rapid.conf and rapid-ssl.conf templates located in the /etc/apache2/sites-available directory

WSGIDaemonProcess RAPID python-path=/<home>/RAPID

WSGIScriptAlias / /<home>/RAPID/RAPID/wsgi.py

RedirectPermanent / https://ec2-0-0-0-0.us-west-2.compute.amazonaws.com

ServerAdmin rapidpivot@gmail.com

ServerName ec2-0-0-0-0.us-west-2.compute.amazonaws.com

ServerAlias [www.RapidDev.com](http://www.RapidDev.com)

<Directory /<home>/RAPID/RAPID>

<Files wsgi.py>

Require all granted

</Files>

</Directory>

* 1. Enable the SSL module to active it

sudo a2enmod ssl

* 1. Activate the SSL Virtual Host to use custom templates

sudo a2ensite rapid-ssl.conf

sudo a2ensite rapid.conf

To make future updates to the configuration files, you need to disable and then renable the SSL Virtual Host to use the templates. Use the following commands to disable the templates.

sudo a2dissite rapid-ssl.conf

sudo a2dissite rapid.conf

* 1. Restart Apache to load the new virtual host files

sudo /etc/init.d apache2 reload

sudo /etc/init.d/apache2 restart

1. Configure Celery configuration files

Change the permissions on the following files (ie., sudo chmod 640)

/etc/default/celery\_beat

/etc/default/celery\_daemon

/etc/default/celery\_pivoteer

1. Edit the CHDIR parameter in the following files to point to the working space directory -CHDIR (/<home>/RAPID)

/etc/default/celery\_beat

/etc/default/celery\_daemon

/etc/default/celery\_pivoteer

1. Start the apache and celery worker processes

> sudo etc/init.d/apache2 start (if not already started)

> sudo etc/init.d/celery\_pivoteer start

> sudo etc/init.d/celery\_daemon start

> sudo etc/init.d/celery\_beat start

NOTE: To stop the celery processes, they have to be executed in the following order

> sudo etc/init.d/celery\_beat stop

> sudo etc/init.d/celery\_daemon stop

> sudo etc/init.d/celery\_pivoteer stop

> sudo etc/init.d/apache2 stop

1. Copy the GeoLite2-City.mmdb from the production server- /apps/RAPID/core folder to the new server /<home>/RAPID/core folder
2. Add a rule to the associated Security Group for the AWS server

Type-SSH Protocol -TCP Port-22 Source- 0.0.0.0/0

Type-HTTPS Protocol-TCP Port-443 Source-0.0.0.0/0

1. Add a rule to the associated Security Group for the AWS Database RDS server

Type-PostgreSQL Protocol-TCP Port-5432 Source-<web server’s IP>

# III.Celery Tasks Troubleshooting Steps

1. Check that the celery RabbitMQ server is running

* sudo rabbitmqctl status

1. If it’s not, then start the RabbitMQ server

* sudo /etc/init.d/rabbitmq-server start
* sudo systemctl start rabbitmq-server

1. If the Rabbit service is corrupted, deinstall and reinstall the program

* sudo apt-get remove rabbitmq-server
* sudo apt-get install rabbitmq-server

1. If Celery services are corrupted or log files not generated, deinstall and reinstall libraries by doing the following:
2. Close out of PyCharm and stop the apache and all celery services
3. Deinstall the following libraries in the /usr/local/lib/python3.5/dist-packages directory

NOTE: Some libraries may have multiple versions which can cause conflicts with dependencies. If this is the case, you need to deinstall all of the multiple versions one at a time because pip may not know which version to remove. Use root to install libraries so files will be removed from the correct location path

* pip3 uninstall celery==<version\_no>

example: pip3 uninstall celery==3.1.16

* pip3 uninstall kombu==<version\_no>
* pip3 uninstall amqp==<version\_no>
* pip3 uninstall billiard==<version\_no>
* pip3 uninstall djcelery==<version\_no>
* pip3 uninstall Django==1.8.2
* pip3 uninstall django\_braces==1.4.0
* pip3 uninstall django\_pgjson==0.3.1
* pip3 uninstall django\_widget\_tweaks==1.3
* pip3 uninstall deepdiff==3.2.1

1. Deinstall rabbitmq-service

* sudo apt-get remove rabbitmq-server (try this if celery does not remove)

1. Confirm files are deleted in the following paths before you reinstall:

/usr/local/lib/python3.5/dist-packages/celery

/usr/local/lib/python3.5/dist-packages/celery-3.1.16.dist-info

/etc/default/celery\*

/etc/init.d/celery\*

/var/log/celery

/var/run/celery

/usr/local/bin/celery\*

1. Reinstall libraries by running install.sh (reinstall all libraries and dependencies including the libraries listed in the requirements.txt file) using the root account.

* ./RAPID/install.sh

Reinstall celery (manually)

* sudo pip3 install celery==3.1.16

Verify following files are created in /usr/local/bin directory

Celery

Celerybeat

Celeryd

Celery-multi

IF you’re getting an ImportError: No module named ‘vine’ with celery 4.0.2, upgrade the following libraries

* sudo pip3 install kombu –upgrade
* sudo pip3 install celery –upgrade
* sudo pip3 install django-celery --upgrade
* sudo pip3 install pip-check

You may get a message that says

Warning: celery\_pivoteer.service changed on disk. Run ‘systemctl daemon-reload’ to reload units

>systemctl daemon-reload

If you are getting the following errors in the app, then you need to deinstall all Django related libraries and reinstall using pip3. Confirm libraries have correctly been deleted from //usr/local/lib/python3.5/dist-packages directory before reinstalling. Also reinstall libraries as **root** so it will install to the correct location path.

From Django.utils.module\_loading import import\_by\_path as import\_string

ImportError: cannot import name ‘import\_by\_path’

* sudo pip3 uninstall Django==<version>
* sudo pip3 uninstall Django\_pgson==<version>
* sudo pip3 uninstall Django\_braces==<version>
* sudo pip3 uninstall Django\_widget\_tweaks==<version>

1. Configure Celery configuration files (follow configuration steps in the previous section)
2. Restart the celery worker processes (follow steps in the previous section)
3. If log files are still not generated, run the celery processes in verbose mode

* sudo sh –x /etc/init.d/celery\_pivoteer start
* sudo sh –x /etc/init.d/celery\_daemon start
* sudo sh –x /etc/init.d/celery\_beat start

If there’s no evidence in the log files, skip the daemonization step to see the errors:

* sudo C\_FAKEFORK=1 sh –x /etc/init.d/celery\_pivoteer start
* sudo C\_FAKEFORK=1 sh –x /etc/init.d/celery\_daemon start
* sudo C\_FAKEFORK=1 sh –x /etc/init.d/celery\_beat start

For online references, go to <http://docs.celeryproject.org/en/latest/userguide/daemonizing.html>

1. Install DeepDiff=3.0.0 version and jsonpickle==0.9.4

* Pip3 install DeepDiff

# IV. Launching an AWS RDS Instance

**Follow the Instructions to install and configure an AWS RDS Instance running PostGreSQL.**

1. In the AWS Console, go to Services 🡪 Database 🡪 RDS
2. Launch a DB Instance
3. In the Select Engine page, select PostgreSQL database and select Dev/Test option for production type.
4. In the Specify DB Details page, select the following
   1. DB Instance Class – db.t2.large
   2. Multi-AZ Deployment – No
   3. Storage Type – General Purpose (SSD)
   4. Allocated Storage – 20GB
   5. DB Instance Identifier – rapid-prod (This is the DB Instance name)
   6. Master Username – rapid (This is the username you will use to log into the database)
   7. Master Password – rapidrapid
   8. Confirm Password - rapidrapid
5. In the Configure Advance Settings page, specify the following items (and accept defaults for other settings)
   1. Availability Zone – same zone as AWS EC2 instance launched (see previous section)
   2. VPC Security Group – Create new Security Group
   3. Database Name – rapid
   4. Database Port - 5432
6. Launch DB Instance (wait a few seconds until you see ‘Your DB Instance is being created’)
7. View Your DB Instance
8. In the RDS Dashboard, you will see the DB instance listed. Wait until the status changes from ‘creating’ to ‘available’.
9. When the status is available, select the DB instance.
10. Confirm the **Endpoint** is displayed and in blue text. Copy the entire Endpoint to a clipboard. This will be your hostname to connect to the PostgreSQL database instance.

# V. RAPID Database Installation

**Follow the Instructions to install and configure the RAPID database.**

1. Log into the AWS EC2 production web server and create a backup of the database

* **sudo su – postgres**
* **Pg\_dump rapid > rapid\_prod.dump (rapid\_prod\_18feb.dump)**

Store the backup file - rapid\_prod.dump off to an accessible location to be referenced later.

You can also use winscp to download/upload the backup file. The destination folder needs to be pointing to the $Home folder (/home/ubuntu). The transfer mode needs to be in Binary with Ignore permission errors checked.

1. In the AWS EC2 web instance, update the following properties in the /apps/RAPID/secrets.json file:
   1. SQL\_NAME: Database Name from the previous section
   2. SQL\_HOST: Endpoint from the previous section
   3. SQL\_USER: Master Username from the previous section
   4. SQL\_PASS: Master Password from the previous section
2. If logging into a local instance of the PostgreSQL db, create a new PostgreSQL superuser account and set password (should be assigned to SQL\_USER and SQL\_PASS respectively in the secrets.json file):

* **sudo su - postgres**
* **createuser rapid - -pwprompt**
* **psql**
* **ALTER USER rapid CREATEDB**

1. If logging into a local PostgreSQL db, use the follow command to log in as the postgres user

* **psql**

Note: To log in as the rapid user. The host is the **Endpoint** for the database instance from the above section.

* **psql --host=localhost ---username=rapid –dbname=<database\_name>**
* **Enter the password for the username:**

1. If logging into an external AWS RDS server, use the follow command to log in as the postgres user. The host is the **Endpoint** for the database instance from the above section. The Endpoint should also be saved the secrets.json file

* **psql --host=database.server.rds.amazonaws.com --port=5432 --username postgres**

Note: To log in as the rapid user ….

* **psql --host= database.server.rds.amazonaws.com --port=5432 –username=rapid -dbname=postgres**
* **Enter the password for the username:**

Note: To test the connection to the external AWS RDS server

* **nc -zq database.server.rds.amazonaws.com 5432**

1. Create the database rapid as the postgres user

* **CREATE DATABASE rapid**

Confirm the database is created by typing the following at the command prompt

* **\l**

A listing of the databases will be displayed as follows…

Name | Owner | Encoding | Collate | Ctype | Access privileges

-----------+----------+----------+-------------+-------------+-----------------------

postgres | rapid | UTF8 | en\_US.UTF-8 | en\_US.UTF-8 |

rapid | rapid | UTF8 | en\_US.UTF-8 | en\_US.UTF-8 |

1. Connect to the rapid database as the owner

* **\connect rapid**

1. Restore the production database backup to the rapid database

* **\i /home/Ubuntu/data/rapid\_prod.dump**

The screen will display with a list of DML commands and counts. It will take about 5 minutes.\q

1. Verify the data has been uploaded by typing the following at the command prompt

* **\dt**

A listing of the database relations will be displayed as follows.

1. You can also query on the data to confirm that the tables are populated. The counts should match those of production.

* **Select count(\*) from monitors\_certificatemonitor;**

**Count should be >=110**

* **Select count(\*) from profiles\_profile;**

**Count should be >= 361 records**

To exit the out of the psql prompt. This will close the connection to the database

* **\q**

1. Open a new SSH or terminal session and connect to the AWS web server.
2. Navigate to the $HOME/Rapid folder.
3. Update the secrets.json file with the correct database connection string and authentication info:

* SQL\_NAME: Database Name from the previous section
* SQL\_HOST: Endpoint from the previous section
* SQL\_USER: Master Username from the previous section
* SQL\_PASS: Master Password from the previous section

Example:

"SQL\_NAME": "rapid",

"SQL\_HOST": "rapid-t2large.cjuelvk2x7js.us-west-2.rds.amazonaws.com",

"SQL\_USER": "rapid",

"SQL\_PASS": "rapidrapid",

1. Verify the following files are in the \apps\RAPID\monitors\migrations folder

**0003\_create\_subscriptions.py**

**0004\_migrate\_subscriptions.py**

1. Synchronize the database with the code updates:

* **sudo python3 manage.py syncdb**

**Confirm the following messages are displayed:**

Applying monitors.0003\_create\_subscriptions

Applying monitors.0004\_migrate\_subscriptions

Populating certificate subscriptions

Populating domain subscriptions

Populating IP subscription

OK

**If you are getting error messages, then you need to generate the migration scripts and migrate them (Follow steps 16-18). If the database is correctly updated, then skip to step 19.**

1. Create migrations for the new code:

* **sudo python3 manage.py makemigrations**

1. Confirm the new migration files are created and also created in <home>/RAPID/monitors/migrations folder

* **sudo python3 manage.py showmigrations**

1. Execute the new migration files:

* **sudo python3 manage.py migrate**

1. Confirm the database has been updated by returning to the database server and entering the following command at the postgres psql prompt

* **\dt**

A listing of the database relations will be displayed as follows. Confirm the following new tables are created

Schema | Name | Type | Owner | Size | Description

--------+----------------------------------+-------+-------+------------+-------------

public | monitors\_certificatesubscription| table | rapid | 0 bytes |

public | monitors\_domainsubscription| table | rapid | 0 bytes |

public | monitors\_ipsubscription| table | rapid | 0 bytes |

Run the following query at the psql prompt to confirm the tables are populated

* **Select count(\*) from monitors\_certificatesubscription;**
* **Select count(\*) from monitors\_domainsubscription;**
* **Select count(\*) from monitors\_ipsubscription;**

The rowcount should be greater or equal to the count from the respective tables beow:

* **Select count(\*) from monitors\_certificatemonitor;**
* **Select count(\*) from monitors\_domainmonitor;**
* **Select count(\*) from monitors\_ipmonitor;**
* **Select \* from Django\_migrations**

**Confirm that the following migrations have been added listed:**

**0003\_create\_subscriptions**

**0004\_migration\_subscriptions**

1. Run the following query to check for bad data in the tag table

* **select \* from monitors\_certificatemonitor\_tags**

**where certificatemonitor\_id not in**

**(select certificate\_id from monitors\_certificatesubscription) order by certificatemonitor\_id;**

Clean up any bad certificate values in the tag table

* **update monitors\_certificatemonitor\_tags**

**set certificatemonitor\_id = replace(replace(certificatemonitor\_id,'',''),' ','');**

1. Run the following query to check for bad data that did not get migrated over

* select certificate\_value from monitors\_certificatemonitor

where certificate\_value not in

(select certificate\_id from monitors\_certificatesubscription)

And certificate\_value not in

(select certificatemonitor\_id from monitors\_certificatemonitor\_tags)

order by certificate\_value;

Execute the following query to delete any bad data that did not get migrated over

* delete from monitors\_certificatemonitor

where certificate\_value not in

(select certificate\_id from monitors\_certificatesubscription)

And certificate\_value not in

(select certificatemonitor\_id from monitors\_certificatemonitor\_tags)

1. Restart the apache and celery services on the webserver. This needs to be done on the webserver in the following order:

(stopall.sh)

* **sudo /etc/init.d/celery\_beat stop**
* **sudo /etc/init.d/celery\_daemon stop**
* **sudo /etc/init.d/celery\_pivoteer stop**
* **sudo /etc/init.d/apache2 stop**

(startall.sh)

* **sudo /etc/init.d/apache2 start**
* **sudo /etc/init.d/celery\_pivoteer restart**
* **sudo /etc/init.d/celery\_daemon restart**
* **sudo /etc/init.d/celery\_beat restart**

# VI. Attaching an AWS EBS Volume to an Instance:

**Follow these instructions if hard drive space becomes an issue and you need to add more storage volume to an AWS instance**

1. Log into the Amazon EC2 console.
2. In the navigation panel, choose Volumes.
3. Select a volume and choose Actions, Create Volume.
4. In the **Create Volume** dialog box, select/enter the following information and accept defaults for the remaining fields:

* Volume Type: General Purpose SSD (GP2)
* Size: size of volume
* Availability Zone: Select same Availability Zone as the instance

1. Select the volume and choose Actions, Attach Volume.
2. In the **Attach Volume** dialog box, start typing the name or ID of the instance to attach the volume to for **Instance**, and select it from the list of suggestion options (only instances that are in the same Availability Zone as the volume are displayed).
3. You can keep the suggested device name, or enter a different supported device name.
4. Choose **Attach**.
5. Follow the instructions to Make an Amazon Volume Available for Use.

# VII. Making an Amazon Volume Available for Use:

**Follow these instructions to configure the storage volume and make it available to an AWS instance**

1. Connect to your instance via SSH.
2. Use the **df -h --total** command to see the total of disk space of an instance and the percentage in use
   * df -h --total
3. Use the **lsblk** command to view your available disk devices and their mount points.
   * lsblk
4. Confirm the new volume (/dev/xvdf) is attached. Volumes are prefixed with a “/dev” although it’s not displayed.
   * xvda 8G
     + xvda1 8G
   * xvdf 30G
5. Determine the new volume type: sudo file –s <device\_name>
   * sudo file –s /dev/xvdf

Confirm the following displays:

* + /dev/xvdf: data

1. If the volume type is data, then a file system needs to be created for this volume: sudo mkfs –t ext4 <device\_name>
   * sudo mkfs –t ext4 /dev/xvdf
2. Create a mount point directory for the volume: sudo mkdir <mount\_point\_directory>

If the directory exists, skip this step.

Example:

* + sudo mkdir /var/log/celery

1. Mount the volume to the directory in step 6: sudo mount <device\_name> <mount\_point\_directory>

Example:

* + sudo mount /dev/xvdf /var/log/celery

1. To mount this volume on every system reboot, add an entry for the mount to the /etc/fstab file. Back up the fstab file before doing anything.
2. Add a new line to the end of the fstab file for your volume using the following format:

NOTE: You can also use the UUID as a robust way to name devices. Use the ‘blkid’ to print the UUID for devices

Format*: <device\_name> <mount\_point\_directory> <file\_system\_type> <fs\_mntops> <fs\_freq> <fs\_passno>*

Examples*:*

*/dev/rapid\_vg/app\_vol /apps ext4 defaults,discard 0 0*

*/dev/xvdf /var/log ext4 defaults,discard 0 0*

*UUID=djfaksjffkjasjfjsdf / ext4 errors=remount-ro 0 1*

1. After you’ve added the new entry to the /etc/fstab, you need to check that your entry works by typing the following command. If the command does not produce an error, then the file is OK. If errors are produced, refer to <http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ebs-using-volumes.html> Errors in the /etc/fstab can render a system unbootable. Do not shut down a system that has errors in the /etc/fstab file.
   * sudo mount -a

# VIII. Extending Live Sessions for SSH connections:

**Follow these instructions to enable SSH sessions to be active longer. This is useful because the default putty sessions will often timeout after 15 minutes of idleness.**

1. Add the following files to the bottom of the /etc/sshd\_config.

ClientAliveInterval 60

ClientAliveCountMax 30

1. Save the file and restart the ssh services

sudo service ssh restart

sudo service ssh reload

# IX. Enabling Log Rotation for Celery logs:

**Follow these instructions to rotate log files on a scheduled basis. This is useful to prevent log files from reaching the maximum file size.**

1. Create a celery config file in /etc/logrotate.d/ folder as save it as ‘celery’
2. Add the following info to this celery file

/var/log/celery/beat.log {

rotate 14

daily

compress

delaycompress

missingok

notifempty

create 640 celery root

}

In the same file, add the same entries for the other log files: p1.log, p2.log, w1.log

For references, refer to https://serversforhackers.com/managing-logs-with-logrotate

1. Run the logrotate manually

sudo logrotate –v –f /etc/logrotate.d/celery

1. Navigate to the /var/log/celery directory and confirm the log files have been rotated.
2. To view the status of the logrotate, review file /var/lib/logrotate/status. You should see a timestamp of the celery tasks that reflects the time the logs were rotated.

# X. Installing Miscellaneous Libraries:

**Geckodriver v0.16.1**

1. At the command prompt on the web server, type in the following

* **Wget https://github.com/mozilla/geckodriver/releases/download/v0.16.1/geckodriver-v0.16.1-linux64.tar.gz**
* **cp geckodriver-v0.16.1-linux64.tar.gz /usr/local/lib/python3.5/dist-packages/**
* **cd to /usr/local/lib/python3.5/dist-packages/**
* **tar -xvzf geckodriver-v0.16.1-linux64.tar.gz**
* **confirm geckodriver is created in /usr/local/lib/python3.5/dist-packages**

**Selenium v3.4.2**

1. At the command prompt on the web server, type in the following

* **sudo apt-get install selenium**
* **confirm selenium-3.4.2.dist-info is created in /usr/local/lib/python3.5/dist-packages**