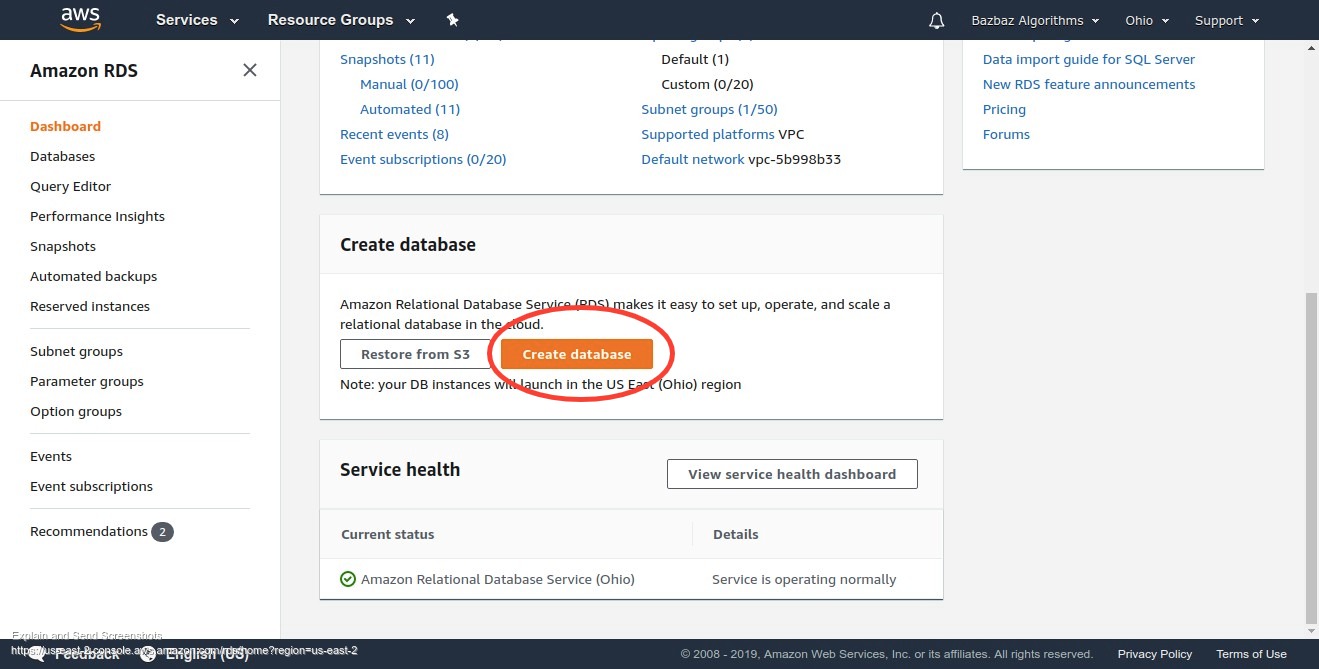
**Deploying Flask app on EC2 instance and connecting RDS.**

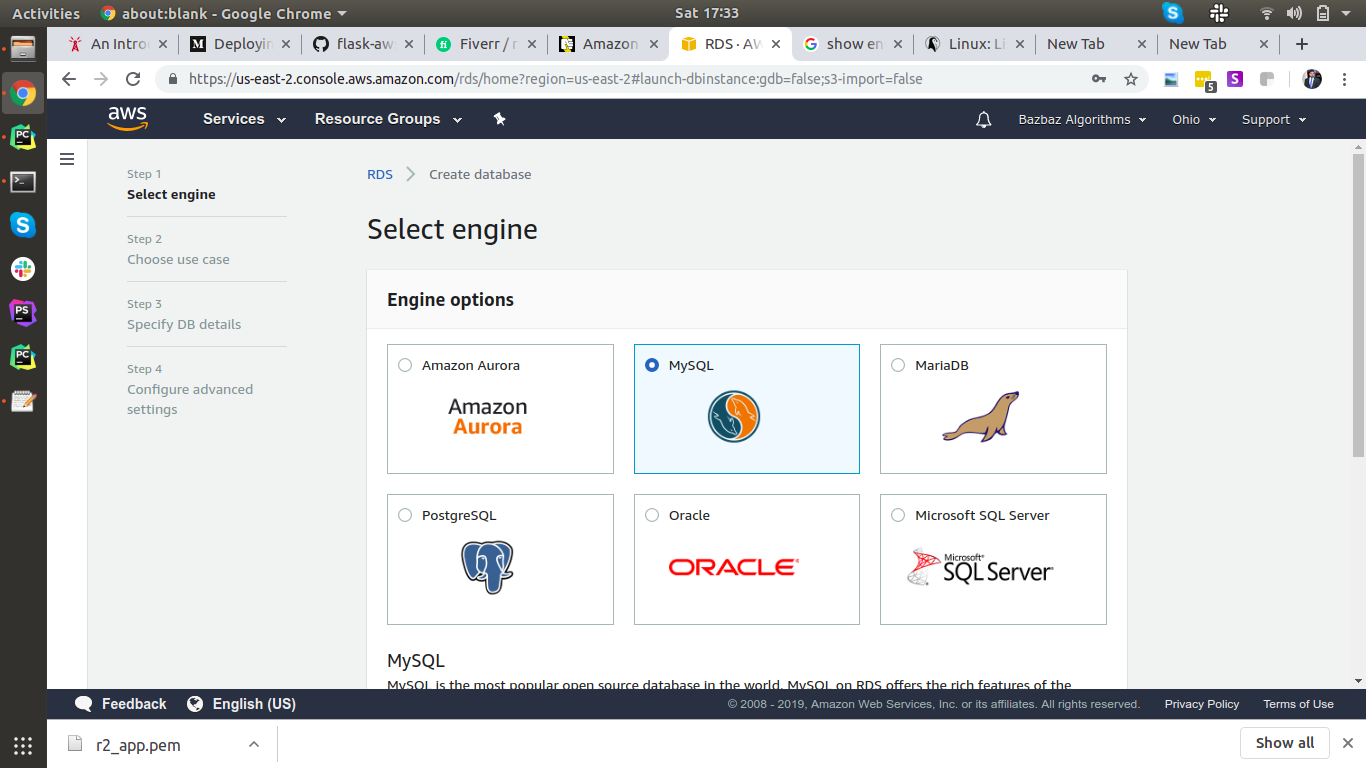
1. Connecting RDS

* Creating RDS.

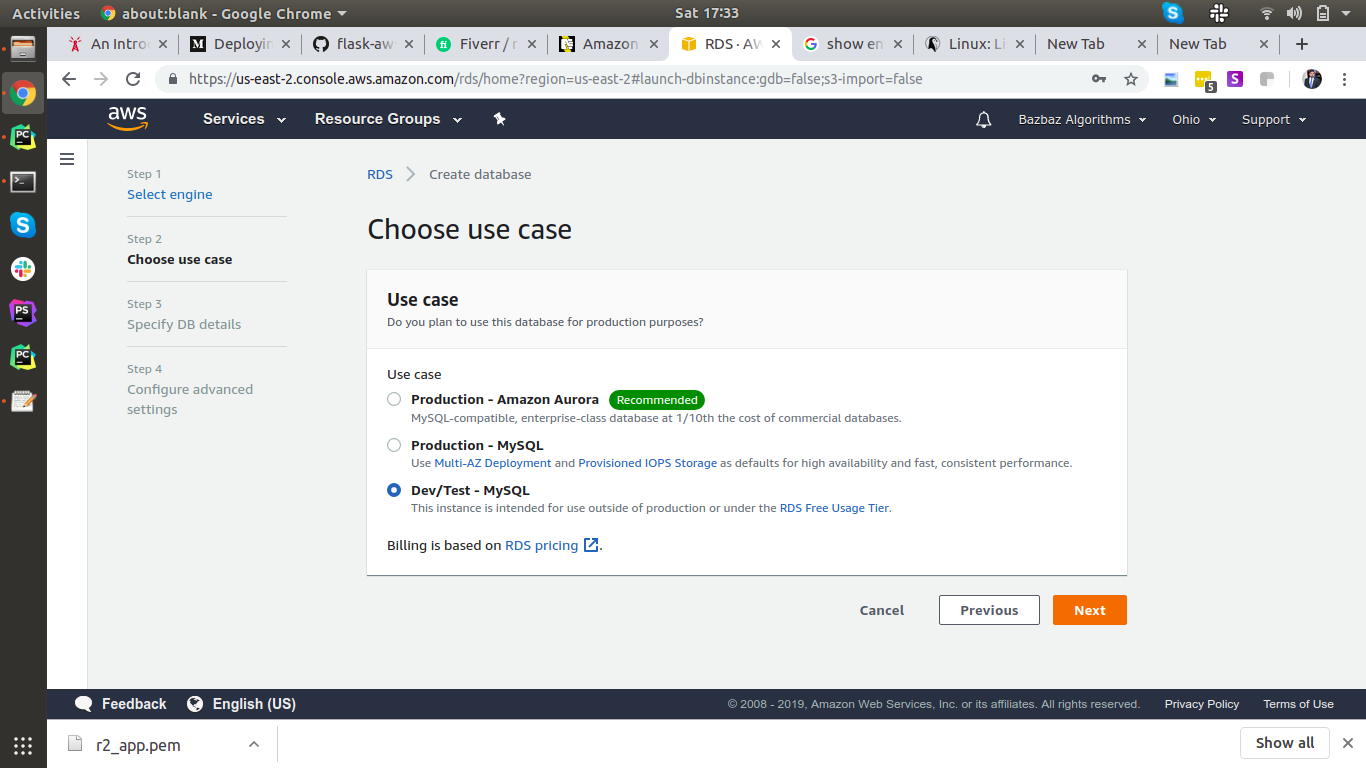
Login to your AWS account and go to RDS [Console](https://us-east-2.console.aws.amazon.com/rds/home?region=us-east-2).

Click on create database.



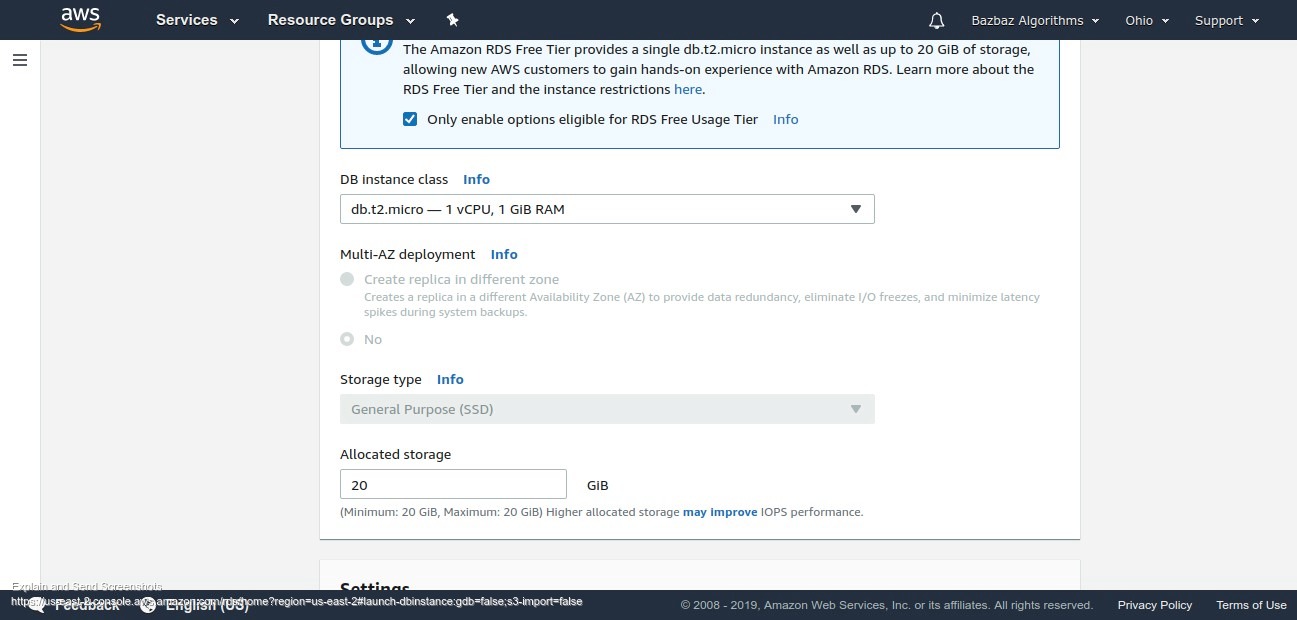
Select MySQL(in our Case) or which ever Database you need and click next.  


For Free Tier click on 3rd option “Dev/Test - MySQL”.



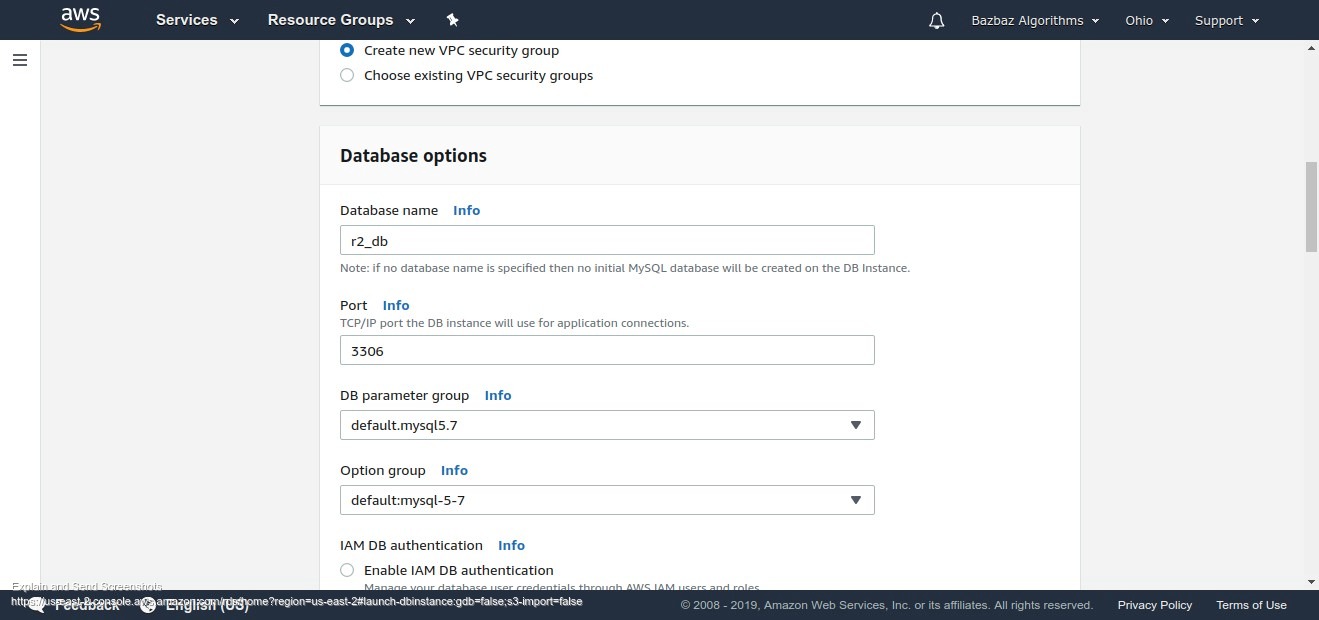
Select version and activate ‘Only enable options eligible for RDS Free Usage Tier’

Select master username e.g; root and master password click next.



Next page make sure your instance on which you are going to connect this database and this RDS instance lies in same VPC. and give name of your database. You can create database later as well by connecting to rds instance from your EC2 instance by running

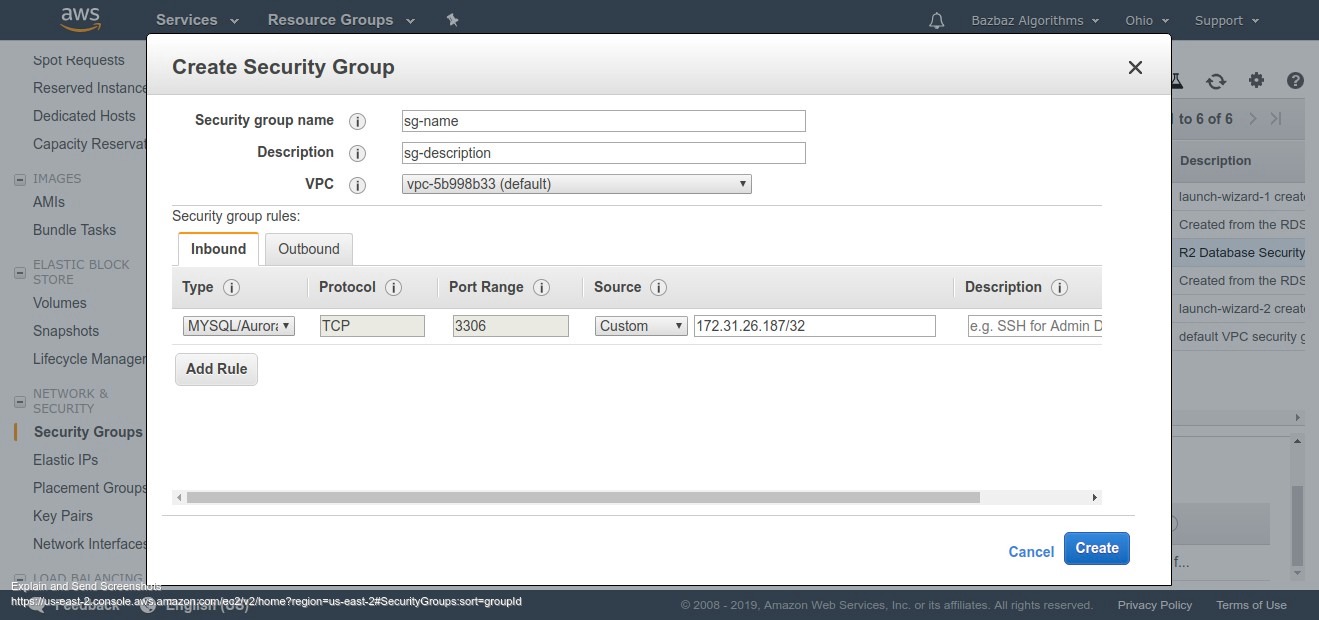
|  |
| --- |
| mysql -h <endpoint> -u <master username> -p Password: <Master Password> mysql> create DATABASE <database name>; |



Click Create Database.

Next Need to assign [Security Group](https://us-east-2.console.aws.amazon.com/ec2/v2/home?region=us-east-2#SecurityGroups:sort=groupId).

In Inbound tab Select MySQL/Aurora in type and in source option add Internal Ip of your instance.



Attach this security Group with RDS instance. Select RDS instance and click “Modify” in “Network & Security” Section add this security group. And click continue.

After this confirm the connection by ssh into your instance and try to connect with this data base by running

|  |
| --- |
| mysql -h <endpoint> -u <master username> -p |

* Connecting to Flask

Set string as

SQLALCHEMY\_DATABASE\_URI =**"mysql+pymysql://<db user>:<db pass>@<endpoint>/<dbname>"**

1. Setting Up Server
2. Install nginx
3. Install python
4. Install supervisor
5. Setup virtual Environment in Envs/ Directory
6. Install gunicorn
7. Created gunicron.conf.py file you can see in R2/ directory.

Started server using command

|  |
| --- |
| gunicorn -c /path/to/gunicorn.conf.py R2 |

If everything working fine then we can make a supervisor process of this

1. Making supervisor process involves creating r2.conf file in /etc/supervisor/conf.d/ directory. Please have a look at the configurations there.

Then run following commands

|  |
| --- |
| sudo supervisorctl reread sudo supervisorctl update sudo supervisorctl restart <process name> |

It would start gunicorn process in background.

1. Next we need to configure nginx.

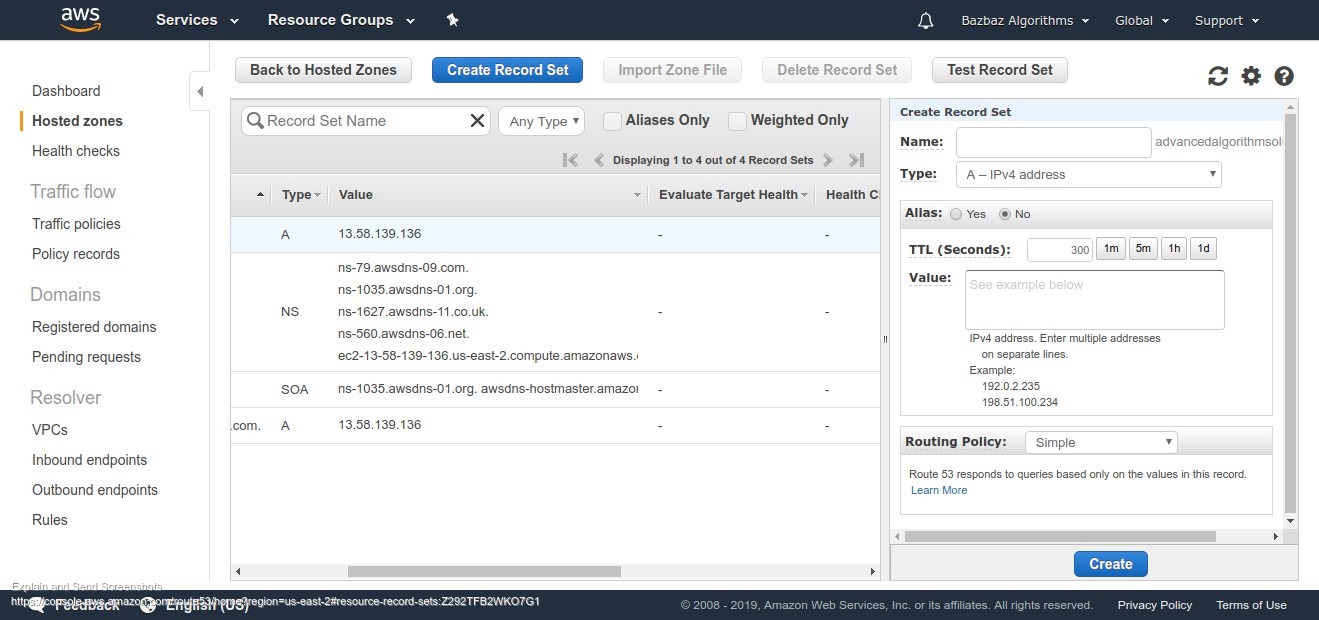
Create file in /etc/nginx/sites-available/<domain-name>.conf or without any extension. At first keep the configuration simple by just putting.

|  |
| --- |
| server {  listen 80;  server\_name your\_public\_dnsname\_here;   location / {  proxy\_pass http://<address in gunicorn file used in bind>;  } } |

1. Now we need to point domain to our server from [route53](https://console.aws.amazon.com/route53/home?region=us-east-2).

Click on “Hosted Zones” in sidebar and open domain detail page.

On top click on “Create Record set”.



“Name” is the subdomain leave empty at first and in “Value” enter your instance’s public ip.

And click create now again create a new record set and and in “Name” enter ‘www’ and value again public ip of instance. Click create.

1. Now update nginx conf to redirect all traffic from <http://domain.com> to <http://www.domain.com>

|  |
| --- |
| server {  listen 80;  server\_name advancedalgorithmsolutions.com;  return 301 http://www.advancedalgorithmsolutions.com$request\_uri;  }  server {  listen 80;  server\_name advancedalgorithmsolutions.com;   location / {  proxy\_pass http://<address in gunicorn file used in bind>;  } } |

1. We need to create symlinks of nginx conf file in /etc/nginx/sites-enabled directory

|  |
| --- |
| sudo ln -s /etc/nginx/sites-available/r2 /etc/nginx/sites-enabled/r2 |

Make sure file in sites-enabled directory contains the same content as the file in sites-available directory

1. Test Nginx Configurations are error free by running

|  |
| --- |
| sudo nginx -t |

If successful run

|  |
| --- |
| sudo service nginx reload |

1. Ideally the site should be up by now on http. Now we need to make if https.

For making site https. Follow instructions on [CertBot](https://certbot.eff.org/). This is an excellent free tool to make sites https. License expires after a month but certbot installs a software which checks for expiring licenses and keep updating them.

3. Take Database Dump and Execute

* Login to instance and move to directory where you want to keep sql.gz file, i used /home/ubuntu/db\_backup/
* Run command

|  |
| --- |
| Mysqldump -h <rds endpoint> -u root -p r2\_db | gzip -9 > r2\_db\_backup\_<date>.sql.gz |

* To execute Database Dump from directory where our sql.gz file is run

|  |
| --- |
| pv r2\_db\_backup\_<date>.sql.gz | gunzip | mysql -h <rds endpoint> -u root -p <database name> |

4. Code Deployment Strategies

* Always work on beta branch on local.
* After finishing an update or functionality login to Beta server

|  |
| --- |
| ssh -i /path/to/r2\_app.pem ubuntu@18.218.145.144 |

* Activate Virtual Environment

|  |
| --- |
| source Envs/env\_r2/bin/activate |

* Move to code directory

|  |
| --- |
| cd webapps/R2\_beta |

* Then take pull of beta branch

|  |
| --- |
| git pull origin beta |

* Then restart supervisor process by running,

|  |
| --- |
| sudo supervisorctl restart r2\_beta |

* If the Changes are working fine close terminal and move to local code.  
  (lets assume that live branch is master) Switch to master branch

|  |
| --- |
| git checkout master |

* Merge beta branch into master

|  |
| --- |
| git merge beta |

* Push master branch

|  |
| --- |
| git push origin master |

* ssh into live server

|  |
| --- |
| ssh -i /path/to/r2\_app.pem ubuntu@13.58.139.136 |

* Activate Virtual Environment

|  |
| --- |
| source Envs/env\_r2/bin/activate |

* Move to code directory

|  |
| --- |
| cd R2 |

* Take pull of master branch

|  |
| --- |
| git pull origin master |

* Run any project related commands or install any dependencies etc and restart supervisor gunicorn process by running

|  |
| --- |
| sudo supervisorctl restart r2 |

* Verify changes.

5. Log File locations

All logs can be found in /home/ubuntu/logs/ directory.

gunicorn.log r2-nginx-error.log r2\_gunicorn.log r2\_gunicorn\_stdout.log

* gunicorn.log - file contains all the gunicorn access related errors
* r2\_gunicorn\_stdout.log - file contains basically all your Flask related outputs and errors
* R2-nginx-error.log - file will have nginx related error logs like if some user access url which doesn’t exists and returned a 404 error that error would get log here.

Any system wide logs (nginx service logs, supervisor service logs) are in /var/log/ directory.