**Deploying the Django Application using Gunicorn**

# **Objectives:**

* Ensure application is production ready

# **Prerequisites:**

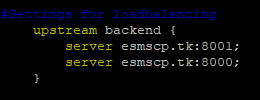
* Ensure virtual environment is created (Django\_Applications/virtual\_env)
* Install dependencies (sudo yum install python3-pip python3-dev libpq-dev)
* Ensure Django project contains the following files: \_\_init\_\_.py, settings.py, urls.py, wsgi.py, and manage.py.

# **Installation steps:**

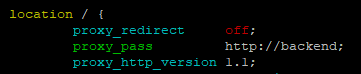
1. Access the virtual environment directory inside our Django Application (cd Django\_Application)
2. Activate the virtual environment (source virtual\_env/bin/activate). You should see “virtual\_env” in brackets.



1. Install Django and Gunicorn inside your virtual environment with the local instance of pip (pip install django gunicorn)
2. Access project directory (cd /home/ec2-user/Django\_Application/virtual\_env/ICA-1/ICA)
3. Load the project’s WSGI module using gunicorn, and set the number of workers to 3 and enable no hangup, so we are able to run the application in the background even after exiting the environment. (nohup gunicorn ICA.wsgi -b [0.0.0.0:800](http://0.0.0.0:8080/)1 -w 3 &)
4. Enter Ctrl-C and enter “**deactivate**” to exit the environment. Access domain/ip-address:8001 and it should still be running.
5. Edit the nginx.conf file to include the upstream module containing our ports (sudo vim /etc/nginx/nginx.conf). We named it ‘backend’ in this case.



1. Reference our ‘backend’ module to the reverse proxy under our 443 server’s “location” tab.



1. Save changes and exit the configuration file.
2. Restart nginx for changes to take effect (sudo service nginx restart)

Source:

<https://www.digitalocean.com/community/tutorials/how-to-set-up-django-with-postgres-nginx-and-gunicorn-on-ubuntu-18-04>

<http://nginx.org/en/docs/http/load_balancing.html>