**Set Up Django with Postgres, Nginx, and Gunicorn**

<https://www.journaldev.com/28002/django-postgres-nginx-gunicorn-ubuntu>

<http://michal.karzynski.pl/blog/2013/06/09/django-nginx-gunicorn-virtualenv-supervisor/>

<https://rahmonov.me/posts/run-a-django-app-with-nginx-gunicorn-and-supervisor/>

With the virtual environment active, let’s install Gunicorn, Django, and Psycopg2 Postgres adapter using pip. Gunicorn can be installed system wide or for a particular environment. The installation inside the environment or system wide will cause a different location for the .sock file. If Gunicorn is installed in the virtual project environment the .sock file will be generated inside the project directory. If installed system wide, it will appear at

$ pip install django gunicorn psycopg2-binary

gunicorn --bind 0.0.0.0:8000 project.wsgi

This fires up Gunicorn on the same interface and port that the Django server was running on. You can go back and verify that the Django application is running on the web browser. Move to the directory where settings.py is located. Hello is the name of the project, it must match the application name entry at the line WSGI\_APPLICATION. It will bind at one of the allowed hosts, listed in settings.py at the line: ALLOWED\_HOSTS. The port can be any free port. Both of these gunicorn commands will start a socket listening for incoming requests:

**gunicorn hello.wsgi:application --bind example.com:8001**

This is the output of a successful gunicorn start:

**gunicorn RestDemo.wsgi:application --bind** [**www.restdemo.com:8001**](http://www.restdemo.com:8001)

[2020-03-15 14:12:40 -0600] [25794] [INFO] Starting gunicorn 20.0.4

[2020-03-15 14:12:40 -0600] [25794] [INFO] Listening at: http://127.0.0.1:8001 (25794)

[2020-03-15 14:12:40 -0600] [25794] [INFO] Using worker: sync

[2020-03-15 14:12:40 -0600] [25797] [INFO] Booting worker with pid: 25797

**gunicorn --workers 1 --bind unix:/home/monicarhvm/pythonprojects/envs/restdemo/bin/gunicorn.sock RestDemo.wsgi:application**

[2020-03-15 14:28:54 -0600] [27703] [INFO] Starting gunicorn 20.0.4

[2020-03-15 14:28:54 -0600] [27703] [INFO] Listening at: unix:/home/monicarhvm/pythonprojects/envs/restdemo/bin/gunicorn.sock (27703)

[2020-03-15 14:28:54 -0600] [27703] [INFO] Using worker: sync

[2020-03-15 14:28:54 -0600] [27706] [INFO] Booting worker with pid: 27706

Let’s have gunicorn run as a service daemon:

We are going to **create a systemd service** file with Gunicorn using the following command

$ sudo vim /etc/systemd/system/gunicorn.service

Starting with the [Unit] section, paste the following content

[Unit]

Description=gunicorn daemon

After=network.target

This section specifies dependencies and metadata

Next, create the [service] section.

In this section, we shall specify the user and group that the process should run under. In this case, the user is root and the group is www-data. The group is specified as www-data so that Nginx can seamlessly communicate with Gunicorn.

The full path to Gunicorn executable will then be indicated. Since Nginx is installed within the same server, we will bind it to a Unix socket.

Paste the following content

[Service]

User=james

Group=www-data

WorkingDirectory=/home/james/project

ExecStart=/home/james/project/projectenv/bin/gunicorn --access-logfile - --workers 3 --bind unix:/home/james/project/project.sock project.wsgi:application

Finally, create [Install] section and append the following lines

[Install]

WantedBy=multi-user.target

Great! Now our systemd service file is complete.

Save and close the text editor. Restart Gunicorn service and enable it to start on boot

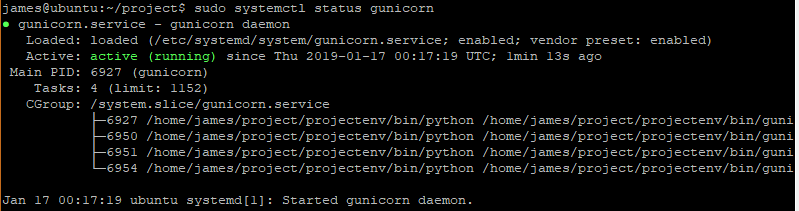
$ sudo systemctl start gunicorn

$ sudo systemctl enable gunicorn

To check the status of gunicorn run

$ systemctl status gunicorn

**Sample output**



## Verify for the presence of Gunicorn Socket file. In this case gunicorn is installed inside the restdemo virtual environment using pip install with the activated restdemo environment. The ExecStart block can either invoke the gunicorn service daemon or, in this case it executes a bash script that starts gunicorn service. The content of the bash script is pasted below. This is the content of the /etc/systemd/system/gunicorn.service file:

[Unit]

Description=gunicorn daemon

#After=network.target

[Service]

Type=simple

User=monicarhvm

#WorkingDirectory=/home/monicarhvm/pythonprojects/RestDemo

#ExecStart=/home/monicarhvm/pythonprojects/envs/restdemo/bin/gunicorn --workers 3 --bind unix:/home/monicarhvm/pythonprojects/RestDemo/RestDemo.sock RestDemo.wsgi:application

ExecStart=/home/monicarhvm/pythonprojects/RestDemo/gunicorn\_start.sh

[Install]

WantedBy=multi-user.target

**This is a bash script that will start gunicorn to service requests to the restdemo application. Gunicorn\_start.sh must be executable, change the bits to make it executable. The path to the script can be given as an argument of the ExecStart block in the gunicorn service daemon file,** /etc/systemd/system/gunicorn.service

#!/bin/bash

NAME="RestDemo"                              #Name of the application (\*)

DJANGODIR=/home/monicarhvm/pythonprojects/RestDemo             # Django project directory (\*)

SOCKFILE=/home/monicarhvm/pythonprojects/envs/restdemo/bin/gunicorn.sock        # we will communicate using this unix socket (\*)

USER=monicarhvm                                        # the user to run as (\*)

GROUP=monicarhvm                                     # the group to run as (\*)

NUM\_WORKERS=1                                     # how many worker processes should Gunicorn spawn (\*)

DJANGO\_SETTINGS\_MODULE=RestDemo.settings             # which settings file should Django use (\*)

DJANGO\_WSGI\_MODULE=RestDemo.wsgi                     # WSGI module name (\*)

echo "Starting $NAME as `whoami`"

# Activate the virtual environment

cd $DJANGODIR

source /home/monicarhvm/pythonprojects/envs/restdemo/bin/activate

export DJANGO\_SETTINGS\_MODULE=$DJANGO\_SETTINGS\_MODULE

export PYTHONPATH=$DJANGODIR:$PYTHONPATH

# Create the run directory if it doesn't exist

RUNDIR=$(dirname $SOCKFILE)

test -d $RUNDIR || mkdir -p $RUNDIR

# Start your Django Unicorn

# Programs meant to be run under supervisor should not daemonize themselves (do not use --daemon)

exec /home/monicarhvm/pythonprojects/envs/restdemo/bin/gunicorn ${DJANGO\_WSGI\_MODULE}:application \

  --name $NAME \

  --workers $NUM\_WORKERS \

  --user $USER \

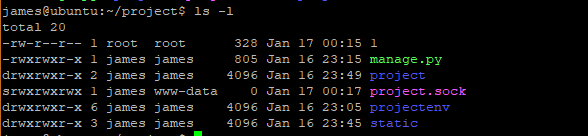
  --bind=unix:$SOCKFILE

**Verify for the presence of Gunicorn Socket file**

Now that we have verified that Nginx is up and running, let us verify the existence of the project.sock file in our project directory

$ ls -l /home/james/project

**Sample output**



**NOTE:**

If project.sock file is missing, this is an indicator that gunicorn was not able to start correctly.  
Additionally, you can check the gunicorn logs by executing the command below

$ sudo journalctl -u gunicorn

There are a number of reasons why Nginx could not have created the project.sock file. Some include

1. Project files being owned by root user instead of sudo user
2. The working directory within the /etc/systemd/system/gunicorn.service not pointing to the project directory.
3. Incorrect configurations in the ExecStart directive

If all configurations are okay, then you should not get any errors after running

$ sudo journalctl -u gunicorn

Once you make changes to the /etc/systemd/system/gunicorn.service file, ensure that you reload the daemon service for the changes to take effect

$ sudo systemctl daemon-reload

$ sudo systemctl restart gunicorn

**Create an Nginx virtual server configuration for Django**

Each Nginx virtual server should be described by a file in the /etc/nginx/sites-available directory. You select which sites you want to enable by making symbolic links to those in the /etc/nginx/sites-enabled directory.

Create a new nginx server configuration file for your Django application running on example.com in /etc/nginx/sites-available/hello. The file should contain something along the following lines. A more detailed example is available [from the folks who make Gunicorn](https://github.com/benoitc/gunicorn/blob/master/examples/nginx.conf).

|  |  |
| --- | --- |
|  | upstream hello\_app\_server { |
|  | # fail\_timeout=0 means we always retry an upstream even if it failed |
|  | # to return a good HTTP response (in case the Unicorn master nukes a |
|  | # single worker for timing out). |
|  |  |
|  | server unix:/webapps/hello\_django/run/gunicorn.sock fail\_timeout=0; |
|  | } |
|  |  |
|  | server { |
|  |  |
|  | listen 80; |
|  | server\_name example.com; |
|  |  |
|  | client\_max\_body\_size 4G; |
|  |  |
|  | access\_log /webapps/hello\_django/logs/nginx-access.log; |
|  | error\_log /webapps/hello\_django/logs/nginx-error.log; |
|  |  |
|  | location /static/ { |
|  | alias /webapps/hello\_django/static/; |
|  | } |
|  |  |
|  | location /media/ { |
|  | alias /webapps/hello\_django/media/; |
|  | } |
|  |  |
|  | location / { |
|  | # an HTTP header important enough to have its own Wikipedia entry: |
|  | # http://en.wikipedia.org/wiki/X-Forwarded-For |
|  | proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for; |
|  |  |
|  | # enable this if and only if you use HTTPS, this helps Rack |
|  | # set the proper protocol for doing redirects: |
|  | # proxy\_set\_header X-Forwarded-Proto https; |
|  |  |
|  | # pass the Host: header from the client right along so redirects |
|  | # can be set properly within the Rack application |
|  | proxy\_set\_header Host $http\_host; |
|  |  |
|  | # we don't want nginx trying to do something clever with |
|  | # redirects, we set the Host: header above already. |
|  | proxy\_redirect off; |
|  |  |
|  | # set "proxy\_buffering off" \*only\* for Rainbows! when doing |
|  | # Comet/long-poll stuff. It's also safe to set if you're |
|  | # using only serving fast clients with Unicorn + nginx. |
|  | # Otherwise you \_want\_ nginx to buffer responses to slow |
|  | # clients, really. |
|  | # proxy\_buffering off; |
|  |  |
|  | # Try to serve static files from nginx, no point in making an |
|  | # \*application\* server like Unicorn/Rainbows! serve static files. |
|  | if (!-f $request\_filename) { |
|  | proxy\_pass http://hello\_app\_server; |
|  | break; |
|  | } |
|  | } |
|  |  |
|  | # Error pages |
|  | error\_page 500 502 503 504 /500.html; |
|  | location = /500.html { |
|  | root /webapps/hello\_django/static/; |
|  | } |
|  | } |

[**view raw**](https://gist.github.com/postrational/5747293/raw/483545cb741318e5755e428bef1a3882f2063331/hello.nginxconf)[hello.nginxconf](https://gist.github.com/postrational/5747293#file-hello-nginxconf)hosted with ❤ by [**GitHub**](https://github.com/)

Create a symbolic link in the sites-enabled folder:

$ sudo ln -s /etc/nginx/sites-available/hello /etc/nginx/sites-enabled/hello

Restart Nginx:

$ sudo service nginx restart

If you navigate to your site, you should now see your Django welcome-page powered by Nginx and Gunicorn. Go ahead and develop to your heart’s content.

At this stage you may find that instead of the Django welcome-page, you encounter the default “*Welcome to nginx!*” page. This may be caused by the default configuration file, which is installed with Nginx and masks your new site’s configuration. If you don’t plan to use it, delete the symbolic link to this file from /etc/nginx/sites-enabled.

### Uninstalling the Django application

If time comes to remove the application, follow these steps.

Remove the virtual server from Nginx sites-enabled folder:

$ sudo rm /etc/nginx/sites-enabled/hello\_django

Restart Nginx:

$ sudo service nginx restart

If you never plan to use this application again, you can remove its config file also from the sites-available directory

$ sudo rm /etc/nginx/sites-available/hello\_django

Stop the application with Supervisor:

$ sudo supervisorctl stop hello

Remove the application from Supervisor’s control scripts directory:

$ sudo rm /etc/supervisor/conf.d/hello.conf

If you never plan to use this application again, you can now remove its entire directory from webapps:

$ sudo rm -r /webapps/hello\_django

# Run a Django app with Nginx, Gunicorn and Supervisor

This tutorial is the continuation of [this one](http://rahmonov.me/posts/run-a-django-app-with-nginx-and-gunicorn/) where we learned how to run a django app with nginx and gunicorn. Now we will add Supervisord into the mix.

## The reason we need Supervisord

Right now, we have our app running with Nginx and Gunicorn. However, every time our machine boots we have to start gunicorn and overall, controlling (stopping, restarting and etc) gunicorn is very difficult. What we want is an easy way of doing so.

Welcome [Supervisord](http://supervisord.org/) which allows us to monitor and control a number of processes on UNIX-like operating systems.

Let's remember how we used to start our app:

gunicorn --daemon --workers 3 --bind unix:/home/ubuntu/myproject/myproject.sock myproject.wsgi

I know right?! It is very long to type and those paths are very error prone. Now, try to stop that daemon (see what I mean? :) ). We will have to find all those gunicorn processes and kill them, which is at least cruel.

With supervisord at our disposal, it will be very easy and convenient to execute those commands:

supervisorctl start myproject

supervisorctl stop myproject

supervisorctl restart myproject

You see how easy it is now?! Good. Now, let's set up this beast.

## Installation and Setup

To install, type the following:

sudo apt-get install supervisor

Now, restart it:

sudo service supervisor restart

The main configuration file of supervisord is here /etc/supervisor/supervisord.conf. If we take a look, we will see that it contains these lines:

[include]

files = /etc/supervisor/conf.d/\*.conf

It means that config files of specific projects can be stored here /etc/supervisor/conf.d/ and they will be included in that main file.

So, let's create myproject.conf in /etc/supervisor/conf.d/ folder:

sudo vim /etc/supervisor/conf.d/myproject.conf

and configure our project:

[program:RestDemo]

environment = PYTHONUNBUFFERED=1

command = **/home/monicarhvm/pythonprojects/envs/restdemo/bin/gunicorn RestDemo.wsgi:application --bind www.restdemo.com:8002** ; Command to start app

directory=/home/monicarhvm/pythonprojects/RestDemo

user = monicarhvm

autostart=true

autorestart=true                                                        ; User to run as

stdout\_logfile = /home/monicarhvm/pythonprojects/RestDemo/logs/supervisor.log   ; Where to write log messages

redirect\_stderr = true

environment=LANG=en\_US.UTF-8,LC\_ALL=en\_US.UTF-8            ; Set UTF-8 as default encoding

Let's look at the significance of each line now:

[program:myproject]

Here, we are defining a program with the name myproject. This name will be used when we do such commands as:

sudo supervisorctl start myproject

Next:

command=/home/ubuntu/myprojenv/bin/gunicorn --workers 3 --bind unix:/home/ubuntu/myproject/myproject.sock myproject.wsgi

This line is used to define a command which is used when we start or restart our project.

directory=/home/ubuntu/myproject

This line indicates a path from which that command will be run.

autostart=true

autorestart=true

These lines define certain behavior of the script under different conditions. Autostart tells the script to start on system boot and autorestart tells it to restart when it exists for some reason.

stderr\_logfile=/var/log/myproject.err.log

stdout\_logfile=/var/log/myproject.out.log

And these final lines define two files where different kinds of logs are stored. Obviously, error logs will be stored in myproject.err.log and others in myproject.out.log.

Let's save the file and execute the following commands to bring these changes into effect:

sudo supervisorctl reread

sudo supervisorctl update

To verify that everything is working, type this:

ps ax | grep gunicorn

You should see several gunicorn processes running. Or, you can go to localhost:8000 and you will see your django app up and running.

Or, you can now use supervisor to check whether your app is running:

sudo supervisorctl status myproject

Now, go ahead and play with those supervisorctl commands to start, stop, restart and check the status of your app.

Let's do one more thing. Let's see the builtin supervisor web interface in action. Those who don't like command line to control processes will love this.

Open up /etc/supervisor/supervisor.conf and place these lines at the beginning of the file:

[inet\_http\_server]

port=0.0.0.0:9001

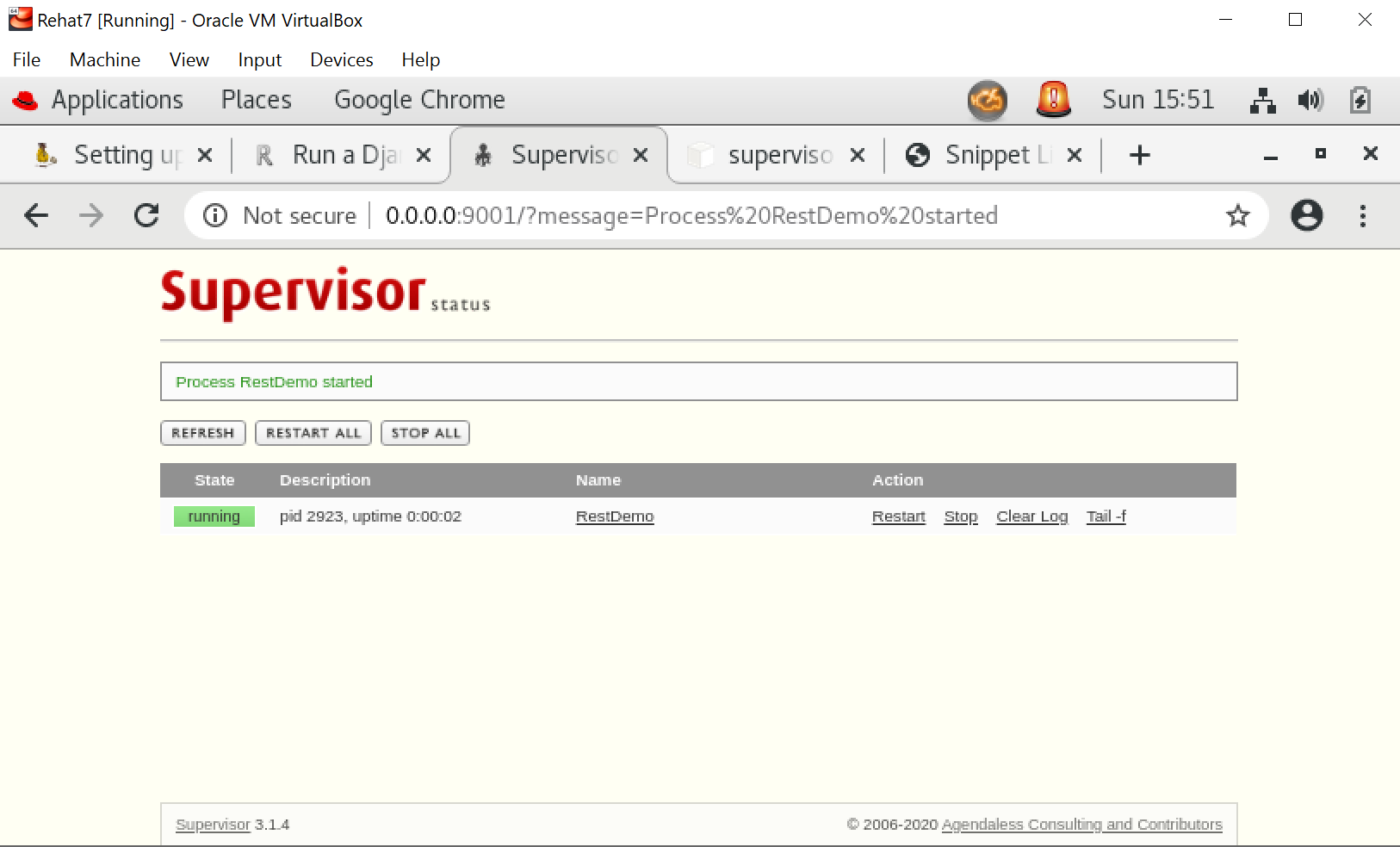
This will indicate that the supervisor web interface will run on 0.0.0.0:9001.

Save the file and reload supervisor:

sudo supervisorctl reload

Open up your browser and go to 0.0.0.0:9001. You will see something like this:

[Web interface of Supervisor](https://rahmonov.me/static/images/supervisor.jpg)

****

# Supervisor common commands:

760  sudo supervisorctl status RestDemo

  775  sudo gedit /etc/supervisor/conf.d/RestDemo.conf

  776  sudo supervisorctl reread

  790  sudo supervisorctl update

  791  sudo supervisorctl status RestDemo

# Nginx common commands:

  329  sudo systemctl stop nginx

  330  sudo systemctl start nginx

  331  sudo systemctl status nginx.service

  365  sudo nginx -t

  367  sudo systemctl enable nginx

  393  sudo gedit /etc/nginx/nginx.conf

  666  ps -ef | grep nginx

  707  sudo gedit /var/log/nginx/error.log

  734  sudo gedit /var/log/nginx/accesslog

  744  sudo journalctl -u nginx

  745  sudo tail -f  /var/log/nginx/access.log

   920  sudo ln -s /etc/nginx/sites-available/RestDemo.nginxcof /etc/nginx/sites-enabled/RestDemo.nginxconf

# Gunicorn common commands:

294  sudo nano /etc/systemd/system/gunicorn.service

  518  ps ax | grep gunicorn

  520  ps -ef | grep gunicorn

  523  ./gunicorn\_start.sh

  682  gunicorn --bind 0.0.0.0:8000 RestDemo.wsgi:application

  683  gunicorn --bind restdemo.com:8000 RestDemo.wsgi:application

  684  gunicorn --bind [www.restdemo.com](https://urldefense.proofpoint.com/v2/url?u=http-3A__www.restdemo.com&d=DwQFaQ&c=CwMyoO6GefVjerZuCCPcOQ&r=e41T37QojCvwZ9rxyEKGnQ22lNzG7wtgFhr_-7XpoJ8&m=UMu9HZVAq6hI7s26m7Au1LZ6aQtUUReZH6Kx82djaPk&s=CKvIxZ9w-GKbtohM21PAgtdf8MnEuTmRVOeb16Dko_I&e=):8000 RestDemo.wsgi:application

  685  gunicorn --bind localhost:8000 RestDemo.wsgi:application

  686  gunicorn --workers 1 --bind unix:/home/monicarhvm/envs/restdemo/bin/RestDemo.sock RestDemo.wsgi:application

  736  sudo systemctl status gunicorn

  743  sudo systemctl restart gunicorn

  772  gunicorn --workers 1 --bind unix:/home/monicarhvm/pythonprojects/envs/restdemo/bin/gunicorn.sock RestDemo.wsgi:application

  829  gunicorn --bind 0.0.0.0:8000 RestDemo.wsgi

  870  sudo gedit /etc/systemd/system/gunicorn.service

  988  sudo systemctl status -l gunicorn