Adrián Deccico

just some random thoughts about coding

Setting up Django 1.3 + NGinx 1.0.5 + Green Unicorn 0.13 in an Ubuntu 11.10 EC2 instance

December 27th, 2011 Leave a comment Go to comments

Django has become the de-facto web framework for Python. Although, since Django just specializes in dynamic content, you have to combine it (at least in production) with an HTTP server to serve static content such as css, javascript files and images files. In the past, the communication protocol between Python web applications was CGI, FastCGI or mod_python. But after PEP-333 was accepted the faster and more efficient WSGI became the standard.

Green Unicorn is a Python WSGI HTTP Server for UNIX. Its combination with the high performance HTTP server NGinx is gaining lot of momentum in the Python community.

From “man gunicorn”...

> Green Unicorn (gunicorn) is an HTTP/WSGI server designed to serve fast clients or sleepy applications. That is to say; behind a buffering front-end server such as nginx or lighttpd.

We are describing here how to combine a Django application with Green Unicorn and Nginx within a pristine EC2 Ubuntu 11.10 image.

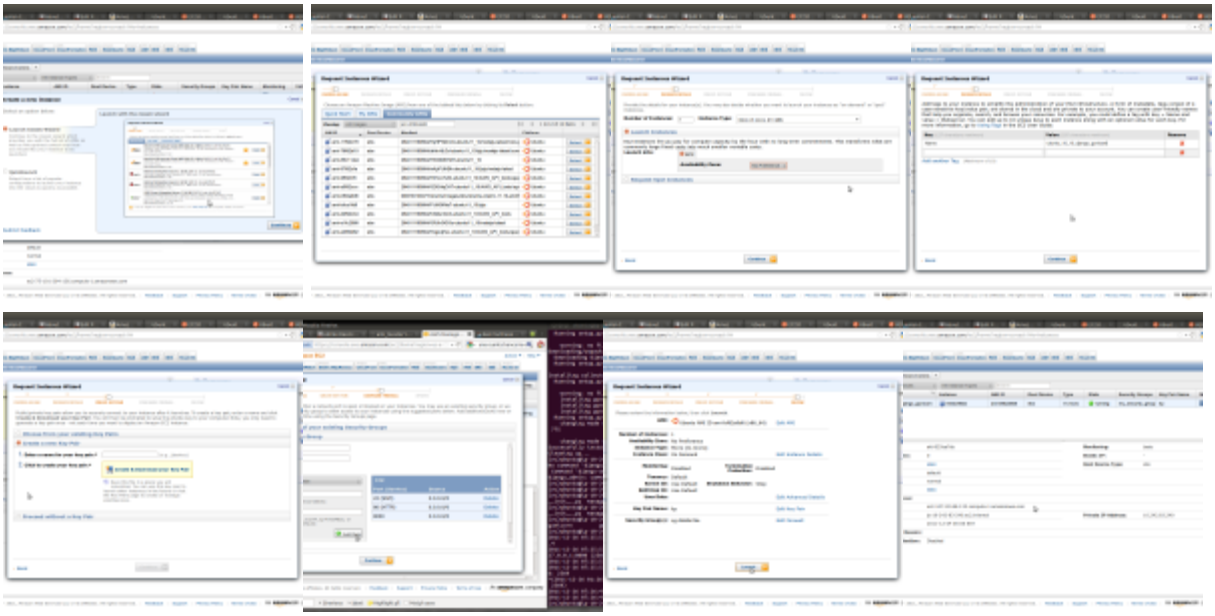
Getting an Ubuntu instance

First of all we are going to http://cloud.ubuntu.com/ami/ to check the last released ami. As described in the image, we are looking for the last 64 bits, Ubuntu 11.10 EB2 image in the US-East region, which so far is **ami-bf62a9d6**.

- Why Ubuntu 11.10 ? Because is the last stable release.
- Why EBS? Because is faster and with more options like on the fly snapshots or easy image resizing than instance-store.
- Why US-EAST? Because is cheaper and usually with more features than the other regions.
- We also choose 64 bits so we can eventually expand to a bigger instance.

Starting your free EC2 instance

So far there is a nice offer for new customers where you can get a free EC2 Linux Micro Instance usage (613 MB of memory and 32-bit and 64-bit platform support) for one year. Please open your account and then launch your instance as described in the following screen-shots:



Launching and configuring your Ubuntu box

At this point you should have your EC2 or your own Ubuntu box up and running. If you are using EC2 you should ssh into it with this command:

```
ssh -i you_private_key.pem ubuntu@DNS_NAME
```

Then, please follow this steps to start setting your environment:

```
#install Python Package Installer
sudo apt-get install python-pip
#upgrade PIP itself
sudo pip install pip --upgrade
#install Green Unicorn
#install Virtualenv to generate our own isolated environment
sudo pip install virtualenv
#Installing NGINX
sudo apt-get install nginx
#Creating our Virtualenv environment
virtualenv --no-site-packages django_app
cd django_app
#activating the environment
source bin/activate
#installing Green Unicorn and Django
```

```
pip install django gunicorn
#Creating a Django project
django-admin.py startproject app
cd app
#start Django application with Green Unicorn
gunicorn_django -b 0.0.0.0:8000
```

Now go to your browser and check if your application is working as expected:

Finally, just Ctrl+C and “deactivate” so you can continue with Nginx...

Setting up Nginx

Now we will set up Nginx in order to have it listening in the port 80, serving static files and working with Gunicorn and Django in order to serve dynamic content.

```
#prepare directories
sudo mkdir -p /opt/django/logs/nginx/
#Create directories and softlinks for static content and templates
mkdir $HOME/django_app/static
mkdir $HOME/django_app/templates
sudo ln -s $HOME/django_app/static /opt/django
#download and set up Nginx configuration. Basically it will listen
sudo mv /etc/nginx/sites-available/default /etc/nginx/sites-availab
wget https://bitbucket.org/deccico/django_gunicorn/raw/tip/server/e
sudo cp default /etc/nginx/sites-available/default
```

Testing Static resources

Configuring settings.py

You need to tell Django where your templates are, for that reason, add your absolute "templates" directory to your /home/ubuntu/django_app/app/settings.py file

Now look for the TEMPLATE_DIRS section and add your template directory

```
TEMPLATE_DIRS = (
# Put strings here, like "/home/html/django_templates" or "C:/www/dj
# Always use forward slashes, even on Windows.
# Don't forget to use absolute paths, not relative paths.
'/home/ubuntu/django_app/templates',
)
```

Configuring urls.py

/home/ubuntu/django_app/app/urls.py will help us telling Django to return our template page when we introduce the /test_static request.

To make things easier we are using a dynamic view:

```
from django.conf.urls.defaults import patterns
from django.views.generic.simple import direct_to_template

urlpatterns = patterns('',
(r'^test_static/$',             direct_to_template, {'template': 'te
)
```

Preparing test template

We need to prepare our template. Please create this file /home/ubuntu/django_app /templates/test_static.html with the following content:

```
<!--Load static will recover the static prefix variable and saving

{% load static %}

{% get_static_prefix as STATIC_PREFIX %}

<!--Here we are just showing how easy is to recover static resources
```

```
8 Test image
9 <img src="{{ STATIC_PREFIX }}django.png" />
```

After creating the template file you will need an actual image 🙂 You can download one from the same repo.

```
1 cd /home/ubuntu/django_app/static
2
3 wget https://bitbucket.org/deccico/django_gunicorn/raw/tip/static/dj
```

Restarting Nginx and voilà

Finally, we just need to restart Nginx and test everything is working.

```
1 #First we restart Nginx
2
3 sudo service nginx restart
```

Now we will start our Django application again:

```
1 cd /home/ubuntu/django_app/app
2 #activating the environment
3 source ../bin/activate
4 #start Django application with Green Unicorn
5 gunicorn_django -b 0.0.0.0:8000
```

Then we just go to our browser and request test_static in port 80. In my case the full address is: http://ec2-107-20-88-133.compute-1.amazonaws.com/test_static

As you can see above, we got Nginx serving the image while Django give us the content.

Type “Ctrl + C” and then “deactivate” again so we can be ready for the next step...

Setting Green Unicorn automatic start

We will use **Upstart** so we don't have to worry about starting our application whenever we need to restart the server. One more configuration file and a bootstrap script will do the job here:

In /home/ubuntu/django_app/run.sh we create:

```
#!/bin/bash
set -e
LOGFILE=/var/log/gunicorn/django_app.log
LOGDIR=$(dirname $LOGFILE)
NUM_WORKERS=3  #recommended formula here is 1 + 2 * NUM_CORES

#we don't want to run this as root..
USER=www-data
GROUP=www-data

cd /home/ubuntu/django_app
source bin/activate
cd app
test -d $LOGDIR || mkdir -p $LOGDIR
exec gunicorn_django -w $NUM_WORKERS \
  --log-level=debug \
  --log-file=$LOGFILE 2>>$LOGFILE \
  --user=$USER --group=$GROUP
```

After creating the file, please remember to execute chmod a+x /home/ubuntu/django_app /run.sh

On the other hand, Upstart configuration file have to be located in: */etc/init/django_app.conf* with the following content:

```
description "Django Application"

start on runlevel [2345]

stop on runlevel [06]

respawn

respawn limit 10 5

exec /home/ubuntu/django_app/run.sh
```

Now we can control our application with

```
sudo start django_app

sudo stop django_app
```

To sum up

The puropose of this post is to provide a path to make the installation and configuration of this stack easier. While some things could be changed or improved the focus was to stress how easily we can configure everything from scratch rather than exploring or discussing different options.

The code and configuration files of this post can be found here: https://bitbucket.org/deccico/django_gunicorn Have fun!

References

- https://docs.djangoproject.com/en/1.3/ref/generic-views/
- http://upstart.ubuntu.com/getting-started.html
- http://senko.net/en/django-nginx-gunicorn/
- https://help.ubuntu.com/community/EC2StartersGuide
- https://help.ubuntu.com/community/UEC/Images
- https://help.ubuntu.com/community/UbuntuCloudInfrastructure
- http://cloud.ubuntu.com/ami/

Tags: django, ec2, gunicorn, nginx, python, snippet, sysadmin, ubuntu
Comments (23) Trackbacks (1) Leave a comment Trackback

1. 
   Harold Spencer, Jr.
   December 27th, 2011 at 17:04 | #1
   Reply | Quote

   Really good article. I am curious, did you try launching this on Eucalyptus? Did you run into any issues? Thanks again for the article. Its really insightful.

2. 
   Jonas
   December 27th, 2011 at 17:10 | #2
   Reply | Quote

Why is Green Unicorn preferrable to running WSGI or uwsgi directly with Nginx (there are modules for it, and uwsgi shows very impressive performance numbers)?

3. 
Sonny
December 27th, 2011 at 19:30 | #3
Reply | Quote

Great post! Saved me a lot of time, thanks Ádrian.

4. 
suvash
December 28th, 2011 at 01:09 | #4
Reply | Quote

hi there,

definitely a great post. but i ran into an issue(which i also kinda fixed but not happily)

In the section “Launching and configuring your Ubuntu box”
> gunicorn_django -b 0.0.0.0:8000
fails with the following exception
“ImportError: No module named django.core.handlers.wsgi”

after a while of though, i figured gunicorn_django wasn’t able to see the django installed in the virtualenv, so I deactivated the virtualenv and then installed django globally(without virtualenv)

i then enterted the virtualenv and ran the gunicorn_django again, this time it didn’t complain.

would you know a better way than using this ugly hack ?

5. 
Austen
December 28th, 2011 at 18:44 | #5
Reply | Quote

I think the issue could be that we install gunicorn in the global scale of python, but then make a virtualenv without site packages. I tried installing gunicorn inside the virtualenv this error disappeared. @suvash

6. 
arek
December 28th, 2011 at 20:56 | #6
Reply | Quote

@suvash
hi – what worked for me was uninstalling gunicorn (sudo pip uninstall gunicorn) after deactivating virtualenv (deactivate), and then installing gunicorn in my virtualenv – so source bin/activate, and then pip install gunicorn

let me know if this worked for you

Arek

7. 
arek
December 28th, 2011 at 20:58 | #7
Reply | Quote

oh, I'm sorry i haven't noticed Austen already solved suvash's issue (I haven't refreshed this page for some time). Great article btw, thanks.

8. 
admin
December 28th, 2011 at 22:10 | #8
Reply | Quote

@Harold Spencer, Jr.
I didn't try in Eucalyptus but it should just work since there is nothing EC2 specific here.

9. 
admin
December 28th, 2011 at 22:15 | #9
Reply | Quote

@Jonas
Doesn't look like there is a big performance difference between uwsgi and Gunicorn while the second is easier to set up and administer and consumes less cpu/memory.

Anyway I think both options are valid.

10. 
admin
December 28th, 2011 at 22:16 | #10
Reply | Quote

@Sonny
cheers mate!

11. 

admin
December 28th, 2011 at 22:36 | #11
Reply | Quote

@suvash @arek @austen
Thanks for the correction, I updated the text as you mention.

As you mention, the idea is to install everything inside our Virtualenv. At some point I tried using gunicorn.d which replace the Upstart utility but doesn't work with Virtualenv, that's why I messed up that line.

12. 
suvash
December 29th, 2011 at 13:18 | #12
Reply | Quote

Thanks a lot for the solution, why didn't I think of that eh ? guess my mind doesn't work good enough at 4am @Austen

13. 
suvash
December 29th, 2011 at 13:24 | #13
Reply | Quote

@admin

Thanks a lot there. great article btw !

Maybe you'd also want to look into pythonbrew(manage multiple python installs + virtualenv management). I like the fact that pythonbrew(almost similiar to rvm in ruby) makes it simple to manage both pythons and venvs in one folder(home directory) instead of spreading the virtualenvs around in different directories. (available in pip as well as https://github.com/utahta/pythonbrew)

I couldn't manage to install pythons using pythonbrew on the aws image above, however would be great if you could do so.

14. 
suvash
December 29th, 2011 at 13:46 | #14
Reply | Quote

@admin

Further along, in the "Setting up Nginx" section, I'm not sure if "sudo" is required for creating "static" and "template" directory (line 4,5). This will only let root to create/edit files in that directory.

15. 
admin
December 29th, 2011 at 13:56 | #15
Reply | Quote

@suvash
hi, definitely sudo is not required for directories under $HOME. I updated the text.

Regarding Pythonbrew, it definitely looks interesting.

Cheers

16. 
Daniel
January 7th, 2012 at 15:13 | #16
Reply | Quote

Thanks so much for this great post. Have been trying to get Django running on an EC2 for the entire last week. This time it took 17 Minutes. Can't thank you enough.

17. 
Daniel
January 7th, 2012 at 18:24 | #17
Reply | Quote

Only problem is that I don't get start django_app up and running. I tried to reboot machine, but whenever I try to execute django_app I get:

start: Rejected send message, 1 matched rules; type="method_call", sender=":1.11" (uid=1000 pid=1610 comm="start django_app ") interface="com.ubuntu.Upstart0_6.Job" member="Start" error name="(unset)" requested_reply="0" destination="com.ubuntu.Upstart" (uid=0 pid=1 comm="/sbin /init")

Do you have any suggestions where I could have went wrong? Thanks.

18. 
Daniel
January 7th, 2012 at 18:37 | #18
Reply | Quote

Do I need to install Upstart in order to use start / stop django_app ?

19. 
Daniel
January 7th, 2012 at 19:03 | #19

Reply | Quote

OK, I solved, I had to use sudo start django_app — now it works like a charm. Thanks so much. I still don't understand the whole virtualenv thing, only that I need it for Gunicorn. Right?

20. 
admin
January 8th, 2012 at 11:40 | #20
Reply | Quote

@Daniel
Hi Daniel, I just updated the text. Thanks.

As you surely noticed Upstart is already installed in your Ubuntu system.

>>I still don't understand the whole virtualenv thing, only that I need it for Gunicorn. Right?

Virtualenv is useful for keeping your whole environment isolated, making possible to have two Python/Django applications running in the same system with a different set of (maybe) contradictory dependencies.

For example you could easily setup another Django application that uses another Gunicorn version different than 0.13.4. For our tutorial purposes, Gunicorn could be installed globally.

All in all Virtualenv and PIP is the recommended option for any Python development beyond very simple scripts.

More info:

http://en.wikipedia.org/wiki/Dependency_hell

http://pypi.python.org/pypi/virtualenv

21. 
Daniel
January 9th, 2012 at 11:54 | #21
Reply | Quote

@admin Thanks a lot. I ended up not using VirtualEnv as I could simply not install psycopg2 to get PostgreSQL running. I may try again some day, but for now I have enough other problems setting up Django/Database etc.

22. 
Alexis Bellido
January 9th, 2012 at 21:34 | #22

Reply | Quote

This is a good opportunity to wrap all these steps as a Fabric file, that way the next time you need to setup a server you just run one or two commands from the command line.

I wrote a Fabric file for a similar setup (Django 1.3, Ubuntu, Nginx, gunicorn and upstart) and released it as the Django gunicorn fabfile project on GitHub: https://github.com/alexisbellido/The-Django-gunicorn-fabfile-project

23. 
adrian
January 10th, 2012 at 01:09 | #23
Reply | Quote

@Alexis Bellido
Hi Alexis, I absolutely agree with you. Having the manual steps in this tutorial is just the first step. I am working in a tool (Bellatrix, already in the Cheese shop) to help with the automation of this.

Probably will take a look at your scripts to combine it with Fabric.

1. December 28th, 2011 at 07:42 | #1
Adrián Deccico: Setting up Django 1.3 + NGinx... | Python | Syngu

Name (required)

E-Mail (will not be published) (required)

Website

Subscribe to comments feed

Submit Comment

Getting reTweets from Python
RSS

Twitter

Recent Posts

- Setting up Django 1.3 + NGinx 1.0.5 + Green Unicorn 0.13 in an Ubuntu 11.10 EC2 instance
- Getting reTweets from Python
- Concurrency in Hudson

Tag Cloud

about agile c# c++ c++ builder continuous integration django ec2 geometry google app engine groovy gunicorn hudson jenkins nginx open-id programming python regular expressions scrum snippet snippet.jenkins sysadmin twitter ubuntu windows api

Links

- El Mundo Corregido

Top WordPress

Theme by NeoEase. Valid XHTML 1.1 and CSS 3.