

[Subscribe](#)

# How To Serve Flask Applications with Gunicorn and Nginx on Ubuntu 16.04



39

Posted May 19, 2016  117.7k

PYTHON FRAMEWORKS

PYTHON

NGINX

UBUNTU

UBUNTU 16.04

By: Justin Ellingwood

Not using **Ubuntu 16.04**? Choose a different version:

# Introduction

In this guide, we will be setting up a simple Python application using the Flask micro-framework on Ubuntu 16.04. The bulk of this article will be about how to set up the Gunicorn application server to launch the application and Nginx to act as a front end reverse proxy.

## Prerequisites

Before starting on this guide, you should have a non-root user configured on your server. This user needs to have `sudo` privileges so that it can perform administrative functions. To learn how to set this up, follow our [initial server setup guide](#).

To learn more about the WSGI specification that our application server will use to communicate with our Flask app, you can read the linked section of [this guide](#). Understanding these concepts will make this guide easier to follow.

When you are ready to continue, read on.

## Install the Components from the Ubuntu Repositories

Our first step will be to install all of the pieces that we need from the repositories. We will install `pip`, the Python package manager, in order to install and manage our Python components. We will also get the Python development files needed to build some of the Gunicorn components. We'll install Nginx now as well.

Update your local package index and then install the packages. The specific packages you need will depend on the version of Python you are using for your project.

If you are using **Python 2**, type:

```
$ sudo apt-get update  
$ sudo apt-get install python-pip python-dev nginx
```

If, instead, you are using **Python 3**, type:

```
$ sudo apt-get update  
$ sudo apt-get install python3-pip python3-dev nginx
```

# Create a Python Virtual Environment

Next, we'll set up a virtual environment in order to isolate our Flask application from the other Python files on the system.

Start by installing the `virtualenv` package using `pip`.

If you are using **Python 2**, type:

```
$ sudo pip install virtualenv
```

If you are using **Python 3**, type:

```
$ sudo pip3 install virtualenv
```

Now, we can make a parent directory for our Flask project. Move into the directory after you create it:

```
$ mkdir ~/myproject  
$ cd ~/myproject
```

We can create a virtual environment to store our Flask project's Python requirements by typing:

```
$ virtualenv myprojectenv
```

This will install a local copy of Python and `pip` into a directory called `myprojectenv` within your project directory.

Before we install applications within the virtual environment, we need to activate it. You can do so by typing:

```
$ source myprojectenv/bin/activate
```

Your prompt will change to indicate that you are now operating within the virtual environment. It will look something like this `(myprojectenv)user@host:~/myproject$`.

# Set Up a Flask Application

Now that you are in your virtual environment, we can install Flask and Gunicorn and get started on designing our application:

## Install Flask and Gunicorn

We can use the local instance of `pip` to install Flask and Gunicorn. Type the following commands to get these two components:

Note

Regardless of which version of Python you are using, when the virtual environment is activated, you should use the `pip` command (not `pip3`).

```
(myprojectenv) $ pip install gunicorn flask
```

## Create a Sample App

Now that we have Flask available, we can create a simple application. Flask is a micro-framework. It does not include many of the tools that more full-featured frameworks might, and exists mainly as a module that you can import into your projects to assist you in initializing a web application.

While your application might be more complex, we'll create our Flask app in a single file, which we will call `myproject.py`:

```
(myprojectenv) $ nano ~/myproject/myproject.py
```

Within this file, we'll place our application code. Basically, we need to import flask and instantiate a Flask object. We can use this to define the functions that should be run when a specific route is requested:

```
~/myproject/myproject.py
```

```
from flask import Flask  
app = Flask(__name__)
```

```
@app.route("/")
```

```
def hello():  
    return "<h1 style='color:blue'>Hello There!</h1>"  
  
if __name__ == "__main__":  
    app.run(host='0.0.0.0')
```

This basically defines what content to present when the root domain is accessed. Save and close the file when you're finished.

If you followed the initial server setup guide, you should have a UFW firewall enabled. In order to test our application, we need to allow access to port 5000.

Open up port 5000 by typing:

```
(myprojectenv) $ sudo ufw allow 5000
```

Now, you can test your Flask app by typing:

```
(myprojectenv) $ python myproject.py
```



Visit your server's domain name or IP address followed by `:5000` in your web browser:

`http://server_domain_or_IP:5000`

You should see something like this:



**Hello There!**

When you are finished, hit CTRL-C in your terminal window a few times to stop the Flask development server.

## Create the WSGI Entry Point

Next, we'll create a file that will serve as the entry point for our application. This will tell our Gunicorn server how to interact with the application.

We will call the file `wsgi.py`:

```
(myprojectenv) $ nano ~/myproject/wsgi.py
```

The file is incredibly simple, we can simply import the Flask instance from our application and then run it:

```
~/myproject/wsgi.py
```

```
from myproject import app
```

```
if __name__ == "__main__":  
    app.run()
```

Save and close the file when you are finished.

## Testing Gunicorn's Ability to Serve the Project

Before moving on, we should check that Gunicorn can correctly.

We can do this by simply passing it the name of our entry point. This is constructed by the name of the module (minus the `.py` extension, as usual) plus the name of the callable within the

application. In our case, this would be `wsgi:app`.

We'll also specify the interface and port to bind to so that it will be started on a publicly available interface:

```
(myprojectenv) $ cd ~/myproject  
(myprojectenv) $ gunicorn --bind 0.0.0.0:5000 wsgi:app
```

Visit your server's domain name or IP address with `:5000` appended to the end in your web browser again:

`http://server_domain_or_IP:5000`

You should see your application's output again:



**Hello There!**

When you have confirmed that it's functioning properly, press CTRL-C in your terminal window.

We're now done with our virtual environment, so we can deactivate it:

```
(myprojectenv) $ deactivate
```

Any Python commands will now use the system's Python environment again.

## Create a systemd Unit File

The next piece we need to take care of is the systemd service unit file. Creating a systemd unit file will allow Ubuntu's init system to automatically start Gunicorn and serve our Flask application whenever the server boots.

Create a unit file ending in `.service` within the `/etc/systemd/system` directory to begin:

```
$ sudo nano /etc/systemd/system/myproject.service
```

Inside, we'll start with the `[Unit]` section, which is used to specify metadata and dependencies. We'll put a description of our service here and tell the init system to only start this after the networking target has been reached:

```
/etc/systemd/system/myproject.service
```

```
[Unit]
Description=Gunicorn instance to serve myproject
After=network.target
```

Next, we'll open up the `[Service]` section. We'll specify the user and group that we want the process to run under. We will give our regular user account ownership of the process since it owns all of the relevant files. We'll give group ownership to the `www-data` group so that Nginx can communicate easily with the Gunicorn processes.

We'll then map out the working directory and set the `PATH` environmental variable so that the init system knows where our the executables for the process are located (within our virtual environment). We'll then specify the command to start the service. Systemd requires that we give the full path to the Gunicorn executable, which is installed within our virtual environment.

We will tell it to start 3 worker processes (adjust this as necessary). We will also tell it to create and bind to a Unix socket file within our project directory called `myproject.sock`. We'll set a umask value of `007` so that the socket file is created giving access to the owner and group, while restricting other access. Finally, we need to pass in the WSGI entry point file name and the Python callable within:

```
/etc/systemd/system/myproject.service
```

```
[Unit]
```

```
Description=Gunicorn instance to serve myproject
```

```
After=network.target
```

```
[Service]
```

```
User=sammy
```

```
Group=www-data
```

```
WorkingDirectory=/home/sammy/myproject
```

```
Environment="PATH=/home/sammy/myproject/myprojectenv/bin"
```

```
ExecStart=/home/sammy/myproject/myprojectenv/bin/gunicorn --workers 3 --bind unix:myproject.sock
```

Finally, we'll add an [Install] section. This will tell systemd what to link this service to if we enable it to start at boot. We want this service to start when the regular multi-user system is up and running:

```
/etc/systemd/system/myproject.service
```

```
[Unit]
```

```
Description=Gunicorn instance to serve myproject
```

```
After=network.target
```

```
[Service]
```

```
User=sammy
```

```
Group=www-data
```

```
WorkingDirectory=/home/sammy/myproject
```

```
Environment="PATH=/home/sammy/myproject/myprojectenv/bin"
```

```
ExecStart=/home/sammy/myproject/myprojectenv/bin/gunicorn --workers 3 --bind unix:π
```

```
[Install]
```

```
WantedBy=multi-user.target
```



With that, our systemd service file is complete. Save and close it now.

We can now start the Gunicorn service we created and enable it so that it starts at boot:

```
$ sudo systemctl start myproject  
$ sudo systemctl enable myproject
```

## Configuring Nginx to Proxy Requests

Our Gunicorn application server should now be up and running, waiting for requests on the socket file in the project directory. We need to configure Nginx to pass web requests to that socket by making some small additions to its configuration file.

Begin by creating a new server block configuration file in Nginx's `sites-available` directory. We'll simply call this `myproject` to keep in line with the rest of the guide:

```
$ sudo nano /etc/nginx/sites-available/myproject
```



Open up a server block and tell Nginx to listen on the default port 80. We also need to tell it to use this block for requests for our server's domain name or IP address:

```
/etc/nginx/sites-available/myproject
```

```
server {  
    listen 80;  
    server_name server_domain_or_IP;  
}
```

The only other thing that we need to add is a location block that matches every request. Within this block, we'll include the `proxy_params` file that specifies some general proxying parameters that need to be set. We'll then pass the requests to the socket we defined using the `proxy_pass` directive:

```
/etc/nginx/sites-available/myproject
```

```
server {  
    listen 80;  
    server_name server_domain_or_IP;  
  
    location / {
```

```
location / {  
    include proxy_params;  
    proxy_pass http://unix:/home/sammy/myproject/myproject.sock;  
}  
}
```

That's actually all we need to serve our application. Save and close the file when you're finished.

To enable the Nginx server block configuration we've just created, link the file to the `sites-enabled` directory:

```
$ sudo ln -s /etc/nginx/sites-available/myproject /etc/nginx/sites-enabled
```

With the file in that directory, we can test for syntax errors by typing:

```
$ sudo nginx -t
```

If this returns without indicating any issues, we can restart the Nginx process to read the our new config:

```
$ sudo systemctl restart nginx
```

The last thing we need to do is adjust our firewall again. We no longer need access through port 5000, so we can remove that rule. We can then allow access to the Nginx server:

```
$ sudo ufw delete allow 5000  
$ sudo ufw allow 'Nginx Full'
```

You should now be able to go to your server's domain name or IP address in your web browser:

[http://server\\_domain\\_or\\_IP](http://server_domain_or_IP)

You should see your application's output:

**Hello There!**

## Note

After configuring Nginx, the next step should be securing traffic to the server using SSL/TLS. This is important because without it, all information, including passwords are sent over the network in plain text.

The easiest way get an SSL certificate to secure your traffic is using Let's Encrypt. Follow [this guide](#) to set up Let's Encrypt with Nginx on Ubuntu 16.04.

## Conclusion

In this guide, we've created a simple Flask application within a Python virtual environment. We create a WSGI entry point so that any WSGI-capable application server can interface with it, and then configured the Gunicorn app server to provide this function. Afterwards, we created a systemd unit file to automatically launch the application server on boot. We created an Nginx server block that passes web client traffic to the application server, relaying external requests.

Flask is a very simple, but extremely flexible framework meant to provide your applications with functionality without being too restrictive about structure and design. You can use the general stack described in this guide to serve the flask applications that you design.

By: Justin Ellingwood

♡ Upvote (39)

✚ Subscribe

---

---

# Introducing: DigitalOcean Marketplace

38 Pre-Built Open-Source Applications  
ready to deploy on DigitalOcean  
Droplets in less than 60 Seconds.  
Including LAMP, Docker, GitLab, Jenkins,  
Plesk, cPanel, WordPress, and many  
more.

[VIEW APPLICATIONS](#)

---

## Related Tutorials

How to Set Up a Scalable Django App with DigitalOcean Managed Databases and Spaces

Bias-Variance for Deep Reinforcement Learning: How To Build a Bot for Atari with OpenAI Gym

How To Set Up Jupyter Notebook with Python 3 on Ubuntu 18.04

How To Install and Configure pgAdmin 4 in Server Mode

How To Build a Neural Network to Recognize Handwritten Digits with TensorFlow

---

## 58 Comments

Leave a comment...

Log In to Comment

 [ixtora](#) May 26, 2016

0 Thanks, great tutorial!

Although I am having a problem and I hope you could help.  
.service

```
[Unit]
```

```
Description=Gunicorn instance to serve myproject
```

```
After=network.target
```

```
[Service]
```

```
User=deploy
```

```
Group=www-data
```

```
WorkingDirectory=/home/deploy/flasktest
```

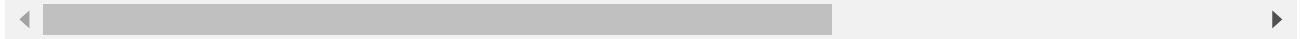
```
Environment="PATH=/home/deploy/.virtualenv/flasktest/bin"
```

```
ExecStart=/home/deploy/.virtualenv/flasktest/bin/gunicorn --workers 3 --bir
```



```
[Install]
```

```
WantedBy=multi-user.target
```

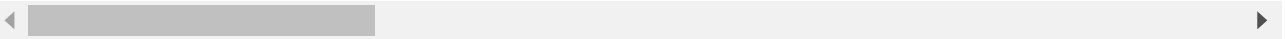


configuration file in Nginx's

```
server {  
    listen 80;  
    server_name 100.10.10.20;  
  
    location / {  
        include proxy_params;  
        proxy_pass http://unix:/home/deploy/flasktest/flasktest.sock;  
    }  
}
```

But if I connect to the address I get a "502 Bad Gateway" response...

```
2016/05/26 16:53:16 [error] 1052#0: *191 connect() failed (111: Connection
```



---

^ [jellingwood](#) May 31, 2016



0

[@ixtora](#): Hello. From a quick glance, it looks like the socket files in your two configuration files don't match. In the systemd file, you are telling Gunicorn to create a **myproject.sock** file. However, in your Nginx configuration, it looks like you are trying to connect to a **flasktest.sock** file. Try changing those to match and see if that works.

---

^ [ixtora](#) May 31, 2016



0

Hello. Thanks for the answer. This code I copied to the lesson, consequently made a mistake. In systemd everything correctly. Problem solved. I found the problem after viewing the log service.

---

^ [ozwiz](#) December 16, 2017



0

I have created the service and now it is running. Can I access my app at this point without proxying request through nginx ?

---

^ [m1368234107](#) June 24, 2016



1 Thanks, great tutorial!

I follow this and it worked, just 1 thing to note.

In my up to date ubuntu16.04(June 24) , In Configuring Nginx to Proxy Requests section, in /etc/nginx/sites-enabled folder , there is a "default" file. first time I enter

[http://server\\_domain\\_or\\_IP](http://server_domain_or_IP) in browser , get nginx welcome page. We should remove this file to get our 'hello there' page.

---

^ [SenderrSantos](#) June 28, 2016



0 Hi, My name is sender, and a trying to finish this instalation, but I dont know whats is the problem. 502, bad gateway. when simulation occurs, it appears the "hello there", but when I try to activate the second part appears this error, I am sending my settings.....

```
\etc\nginx\nginx.conf
```

```
server {  
    listen 80
```

```
listen 80;  
server_name 10.0.0.50;
```

```
    location / {  
        include uwsgi_params;  
        uwsgi_pass unix:/home/sender/meuprojeto2/meuprojeto2.sock;  
    }  
}
```

/etc/systemd/system/meuprojeto.service

[Unit]

Description=uwsgi instance to serve meuprojeto2

After=network.target

[Service]

User=sender

Group=nginx

WorkingDirectory=/home/sender/meuprojeto2

Environment="PATH=/home/sender/meuprojeto2/meuprojeto2env/bin"

ExecStart=/home/sender/meuprojeto2/meuprojeto2env/bin/uwsgi --ini /etc/nginx/conf.d/meuprojeto2.conf

```
ExecStart=/home/sender/meuprojeto2/meuprojeto2env/bin/uwsgi --ini meuprojeto2.ini
```

[Install]

```
WantedBy=multi-user.target
```

---

^ [rithythul](#) July 4, 2016

0 To clarify in case anyone else has this problem:

In the file: `/etc/systemd/system/myproject.service`

This line:

```
ExecStart=/home/sammy/myproject/myprojectenv/bin/gunicorn --workers 3 --bind  
unix:myproject.sock -m 007 wsgi:app
```

On our system it turns out gunicorn is not located here:

```
/home/sammy/myproject/myprojectenv/bin/gunicorn
```

rather it is located here:

```
/usr/local/bin/gunicorn
```

It took our apprentice several days to figure this one out.

✓ [rithythul](#) [comment](#) [comment](#) [comment](#) [comment](#) [comment](#)

You might want to clarify this alternative.

---

^ [jellingwood](#) July 5, 2016

1 [@rithythul](#): Hello! If your **gunicorn** process is located at `/usr/local/bin/gunicorn`, it might be an indication that you did not have your virtual environment enabled when you installed it. To install it to the virtual environment, as this guide demonstrates, make sure you activate your virtual environment first by typing:

```
$ source ~/myproject/myprojectenv/bin/activate
```

Afterwards, the Gunicorn installation command should install to the virtual environment:

```
$ pip install gunicorn flask
```

Hopefully that helps.

---

^  [msing2](#) July 4, 2016


0 I followed the steps exactly and in the end I got the "Hello There!" message to show up at my server's IP address. However, when I rebooted my server and ssh'ed in, I could not get it to work. After that I could no longer connect and get the "Hello There!" message to appear. I get the same message that the IP address refused to connect.

---

^  [oladapoadebowal](#) July 30, 2016

1 I followed the example but I get error 502. When I checked the error logs at /var/log/nginx/error.log. It says no directory or file /home/dapo/myproject/myproject.sock

---

^  [oladapoadebowal](#) July 30, 2016

1 I fix it, since the myproject.sock has to be created by gunicorn, all I did is to restart gunicorn like "sudo systemctl restart myproject" then I did "sudo systemctl restart nginx" and it works

---

^  [camleng](#) December 21, 2017

0 Thank you!!

---

 [ian796072667fc9](#) August 11, 2016

0 I am still getting the 502 Gateway error. I have checked the log of nginx and it says there is no "/home/root/myproject/myproject.sock file. Like the last the last person to post a question I have double checked my setup files. Here is the error.log file:

```
2016/08/11 12:10:41 [emerg] 13854#13854: listen() to 0.0.0.0:80, backlog 511 failed (98: Address already in use)
```

```
2016/08/11 12:10:41 [emerg] 13854#13854: listen() to [::]:80, backlog 511 failed (98: Address already in use)
```

```
2016/08/11 12:10:41 [emerg] 13854#13854: listen() to 0.0.0.0:80, backlog 511 failed (98: Address already in use)
```

```
2016/08/11 12:10:41 [emerg] 13854#13854: listen() to [::]:80, backlog 511 failed (98: Address already in use)
```

```
2016/08/11 12:10:41 [emerg] 13854#13854: listen() to 0.0.0.0:80, backlog 511 failed (98: Address already in use)
```

```
2016/08/11 12:10:41 [emerg] 13854#13854: listen() to [::]:80, backlog 511 failed (98: Address already in use)
```

```
2016/08/11 12:10:41 [emerg] 13854#13854: listen() to 0.0.0.0:80, backlog 511 failed (98: Address already in use)
```

```
2016/08/11 12:10:41 [emerg] 13854#13854: listen() to [::]:80, backlog 511 failed (98: Address already
```



in use)

2016/08/11 12:10:41 [emerg] 13854#13854: listen() to 0.0.0.0:80, backlog 511 failed (98: Address already in use)

2016/08/11 12:10:41 [emerg] 13854#13854: listen() to [::]:80, backlog 511 failed (98: Address already in use)

2016/08/11 12:10:41 [emerg] 13854#13854: still could not bind()

2016/08/11 12:29:50 [emerg] 14125#14125: listen() to 0.0.0.0:80, backlog 511 failed (98: Address already in use)

2016/08/11 12:29:50 [emerg] 14125#14125: listen() to [::]:80, backlog 511 failed (98: Address already in use)

2016/08/11 12:29:50 [emerg] 14125#14125: listen() to 0.0.0.0:80, backlog 511 failed (98: Address already in use)

2016/08/11 12:29:50 [emerg] 14125#14125: listen() to [::]:80, backlog 511 failed (98: Address already in use)

2016/08/11 12:29:50 [emerg] 14125#14125: listen() to 0.0.0.0:80, backlog 511 failed (98: Address already in use)

2016/08/11 12:29:50 [emerg] 14125#14125: listen() to [::]:80, backlog 511 failed (98: Address already in use)

2016/08/11 12:29:50 [emerg] 14125#14125: listen() to 0.0.0.0:80, backlog 511 failed (98: Address already in use)

2016/08/11 12:29:50 [emerg] 14125#14125: listen() to [::]:80, backlog 511 failed (98: Address already in use)

2016/08/11 12:29:50 [emerg] 14125#14125: listen() to 0.0.0.0:80, backlog 511 failed (98: Address

already in use)

2016/08/11 12:29:50 [emerg] 14125#14125: listen() to [::]:80, backlog 511 failed (98: Address already in use)

2016/08/11 12:29:50 [emerg] 14125#14125: still could not bind()

2016/08/11 13:43:46 [emerg] 14499#14499: listen() to 0.0.0.0:80, backlog 511 failed (98: Address already in use)

2016/08/11 13:43:46 [emerg] 14499#14499: listen() to 0.0.0.0:80, backlog 511 failed (98: Address already in use)

2016/08/11 13:43:46 [emerg] 14499#14499: listen() to 0.0.0.0:80, backlog 511 failed (98: Address already in use)

2016/08/11 13:43:46 [emerg] 14499#14499: listen() to 0.0.0.0:80, backlog 511 failed (98: Address already in use)

2016/08/11 13:43:46 [emerg] 14499#14499: listen() to 0.0.0.0:80, backlog 511 failed (98: Address already in use)

2016/08/11 13:43:46 [emerg] 14499#14499: still could not bind()

2016/08/11 13:45:21 [emerg] 14535#14535: listen() to 0.0.0.0:80, backlog 511 failed (98: Address already in use)

2016/08/11 13:45:21 [emerg] 14535#14535: listen() to 0.0.0.0:80, backlog 511 failed (98: Address already in use)

2016/08/11 13:45:21 [emerg] 14535#14535: listen() to 0.0.0.0:80, backlog 511 failed (98: Address already in use)

2016/08/11 13:45:21 [emerg] 14535#14535: listen() to 0.0.0.0:80, backlog 511 failed (98: Address already in use)

2016/08/11 13:45:21 [emerg] 14535#14535: still could not bind()

2016/08/11 13:45:21 [emerg] 14535#14535: listen() to 0.0.0.0:80, backlog 511 failed (98: Address already in use)

2016/08/11 13:45:21 [emerg] 14535#14535: still could not bind()

2016/08/11 13:48:55 [emerg] 14644#14644: listen() to 0.0.0.0:80, backlog 511 failed (98: Address already in use)

2016/08/11 13:48:55 [emerg] 14644#14644: listen() to 0.0.0.0:80, backlog 511 failed (98: Address already in use)

2016/08/11 13:48:55 [emerg] 14644#14644: listen() to 0.0.0.0:80, backlog 511 failed (98: Address already in use)

2016/08/11 13:48:55 [emerg] 14644#14644: listen() to 0.0.0.0:80, backlog 511 failed (98: Address already in use)

2016/08/11 13:48:55 [emerg] 14644#14644: still could not bind()

2016/08/11 13:51:55 [crit] 14722#14722: \*1 connect() to unix:/home/root/myproject/myproject.sock failed (2: No such file or directory) while connecting to upstream, client: 61.93.120.56, server: 188.166.226.199, request: "GET / HTTP/1.1", upstream: "http://unix:/home/root/myproject/myproject.sock:/", host: "188.166.226.199"

2016/08/11 13:51:55 [crit] 14722#14722: \*3 connect() to unix:/home/root/myproject/myproject.sock failed (2: No such file or directory) while connecting to upstream, client: 61.93.120.56, server: 188.166.226.199, request: "GET /favicon.ico HTTP/1.1", upstream: "http://unix:/home/root/myproject/myproject.sock:/favicon.ico", host: "188.166.226.199", referer: "http://188.166.226.199/"

2016/08/11 13:51:55 [crit] 14722#14722: \*4 connect() to unix:/home/root/myproject/myproject.sock failed (2: No such file or directory) while connecting to upstream, client: 61.93.120.56, server: 188.166.226.199, request: "GET / HTTP/1.1", upstream: "http://unix:/home/root/myproject/myproject.sock:/", host: "188.166.226.199"

```
2016/08/11 14:01:44 [crit] 148/0#148/0: *1 connect() to unix:/home/root/myproject/myproject.sock
failed (2: No such file or directory) while connecting to upstream, client: 61.93.120.56, server:
188.166.226.199, request: "GET / HTTP/1.1", upstream:
"http://unix:/home/root/myproject/myproject.sock:/", host: "188.166.226.199"
2016/08/11 14:09:54 [crit] 14870#14870: *3 connect() to unix:/home/root/myproject/myproject.sock
failed (2: No such file or directory) while connecting to upstream, client: 61.93.120.56, server:
188.166.226.199, request: "GET / HTTP/1.1", upstream:
"http://unix:/home/root/myproject/myproject.sock:/", host: "188.166.226.199"
2016/08/11 14:10:16 [crit] 14870#14870: *5 connect() to unix:/home/root/myproject/myproject.sock
failed (2: No such file or directory) while connecting to upstream, client: 61.93.120.56, server:
188.166.226.199, request: "GET / HTTP/1.1", upstream:
"http://unix:/home/root/myproject/myproject.sock:/", host: "188.166.226.199"
2016/08/11 14:12:34 [crit] 14955#14955: *1 connect() to unix:/home/root/myproject/myproject.sock
failed (2: No such file or directory) while connecting to upstream, client: 61.93.120.56, server:
188.166.226.199, request: "GET / HTTP/1.1", upstream:
"http://unix:/home/root/myproject/myproject.sock:/", host: "188.166.226.199"
2016/08/11 14:12:34 [crit] 14955#14955: *3 connect() to unix:/home/root/myproject/myproject.sock
failed (2: No such file or directory) while connecting to upstream, client: 61.93.120.56, server:
188.166.226.199, request: "GET /favicon.ico HTTP/1.1", upstream:
"http://unix:/home/root/myproject/myproject.sock/favicon.ico", host: "188.166.226.199", referrer:
"http://188.166.226.199/"
root@ianwj:~# sudo systemctl restart myproject
root@ianwj:~# sudo systemctl restart nginx
```

```
root@ianwj:~# cd myproject
root@ianwj:~/myproject# ls
myproject.py myproject.pyc myprojectenv wsgi.py wsgi.pyc
root@ianwj:~/myproject# cd etc/nginx/sites-available
-bash: cd: etc/nginx/sites-available: No such file or directory
```

Sorry for the longwinded post...

---

 [jellingwood](#) August 11, 2016

0 [@ian796072667fc9](#): Hello. There seem to be a few things going on here.

First, as it states in the prerequisites, you appear to be using the **root** user. This guide assumes that you are using a non-root user. Using root is not only insecure, it will cause problems when following this guide.

Second, it does not appear that you entered the project directory correctly in the Nginx configuration file. Since you are using the **root** user, the `/home/username/...` prefix does not fit anymore. The **root** user's home directory is `/root/...`, so you should use that in your **proxy\_pass** definition.

If your `.sock` file does not exist, that indicates that the the project's `systemd` step (the one calling `gunicorn` is having issues. You can check the logs produced when trying to start your project by typing:

```
$ sudo journalctl -u myproject
```

That should give you some indication as to what is going on with the project.

Hope that helps.

---

^ [ian796072667fc9](#) *August 11, 2016*



- 0 Thx for the information. I rebuilt the whole tutorial as "ian@ianwj" user. (Got out of "root") Had to do as the last person posted: "I fix it, since the myproject.sock has to be created by gunicorn, all I did is to restart gunicorn like "sudo systemctl restart myproject" then I did "sudo systemctl restart nginx" and it works" Now working.... Thx.
-

^  
♥ [ian796072667fc9](#) August 11, 2016

- 0 One last noob question: I changed the "myproject.py" to replace Hello There! with another message. Restarted myproject and nginx but still get Hello There! when browsing to my IP address.

---

^  
♥ [agquinn](#) September 17, 2016

- 1 "sudo systemctl restart myproject" -- that should do it.

---

^  
♥ [oladapoadebowal](#) October 22, 2016

- 1 I am very frustrated by the fact that at times, I get an error 502 bad gateway error or an error 500 internal server error. This will go away when I run the sudo systemctl restart myproject and after some days it comes back and this is really affecting my business.

When I check my error logs I often get this

```
2016/10/20 23:37:18 [crit] 12366#12366: *47260 connect() to
unix:/home/myusername/myproject/myproject.sock failed (2: No such file or directory) while
connecting to upstream, client: 78.206.179.140, server: 000.111.222.333, request: "GET /goods
HTTP/1.1" ...
```

HTTP/1.1", upstream: "http://unix:/home/myusername/myproject/myproject.sock:/goods", host: "000.111.222.333", referer: "http://000.111.222.333/goods"

Can it just work well once and for all or does it just kills it connections.

---

^ [louisb1979df452dc025c6cc83](#) July 26, 2018

- o same is happening to me right now. By looking at `sudo journalctl -u myproject` I did find some errors though... not sure if that's related

---

^ [tommyc38](#) October 26, 2016

- o I loved this tutorial. However, could you elaborate on what exactly is happening with the wsgi.py file and how my the app is speaking with gunicorn and nginx? Also, how can we clean up our working directory to store the wsgi.py and myproject.sock file to another folder so its out of the way? I would assume with git we would gitignore the sock file and wsgi.py file?

---

^ [dchilders21](#) October 28, 2016

- o Hey guys,



I'm pretty new to this but having some issues and I've done all that I think I could to troubleshoot.  
I'm not getting any major issues in the error.log for nginx or the unicorn log(journalctl).

I have this for my stats.service file

[Unit]

Description=Gunicorn instance to serve stats

After=network.target

[Service]

User=mainuser

Group=www-data

WorkingDirectory=/home/mainuser/stats

Environment="PATH=/home/mainuser/stats/stats/bin"

ExecStart=/home/mainuser/stats/stats/bin/gunicorn --workers 3 --bind unix:stats.sock -m 007  
wsgi:app

[Install]

WantedBy=multi-user.target

and this for my nginx file

```
server {  
listen 80;
```

```
server_name 192.241.204.247;  
allow 192.241.204.247;
```

```
    location / {  
        include proxy_params;  
        proxy_pass http://unix:/home/mainuser/stats/stats.sock;  
    }  
  
}
```

As I said before no errors show up in the logs but when I go to the IP (192.241.204.247) nothing shows, just the 'this site can't be reached - 192.241.204.247 refused to connect.' Thought it might be the firewall but I have that available to Nginx. When I did the test in the virtual env with port 5000 at the end that worked. But when I'm outside of the venv and trying to see something it's not working.

To Action From

OpenSSH ALLOW Anywhere

Apache Full ALLOW Anywhere

Nginx Full ALLOW Anywhere

5000 ALLOW Anywhere

OpenSSH (v6) ALLOW Anywhere (v6)

Apache Full (v6) ALLOW Anywhere (v6)

Nginx Full (v6) ALLOW Anywhere (v6)

5000 (v6) ALLOW Anywhere (v6)

---

 [elmq0022](#) December 21, 2016



0

So does anyone know the correct way to open a port in UFW so I can continue to use pip to install things? I tried:

```
sudo ufw allow 8123/tcp
```

But I didn't have any luck with that. Other thoughts?

---

^ [ArdaMavi](#) January 7, 2017



- 0 When i finish all settings, i show "Welcome to nginx!" page, not "Hello There" page. Please help ???

---

^ [jellingwood](#) January 9, 2017



- 0 [@ArdaMavi](#) Sorry to hear you had an issue. The part that should be preventing the default Nginx page from being served is setting the **server\_name** directive in the new file to the domain name or IP address used for the request.

By default, the **/etc/nginx/sites-enabled/default** file in the Nginx package in Ubuntu's repository will have the **server\_name** set to **\_**, which is just an invalid hostname. Since that server block is defined as the **default\_server** for port 80 (you can see that in the **listen** declaration), it means that this server block should only be used if no others match the request better:

`/etc/nginx/sites-available/default`

```

server {
    listen 80 default_server;
    listen [::]:80 default_server;

    . . .

    server_name _;

    . . .
}

```

I'm not sure why you're experiencing this issue, but I would check that the **server\_name** in **/etc/nginx/sites-enabled/default** is still set to **\_** and the **server\_name** in the new file is set to your server's IP address or domain name.

If you set the **server\_name** in the *new* file to the server's valid IP address or domain name, this should be a more exact match than the invalid **\_** match and should be selected to serve the request. The way you request your page has to be one of the things you add to the **server\_name** for it to match. You can find more about how Nginx chooses the server block to server requests [here](#).

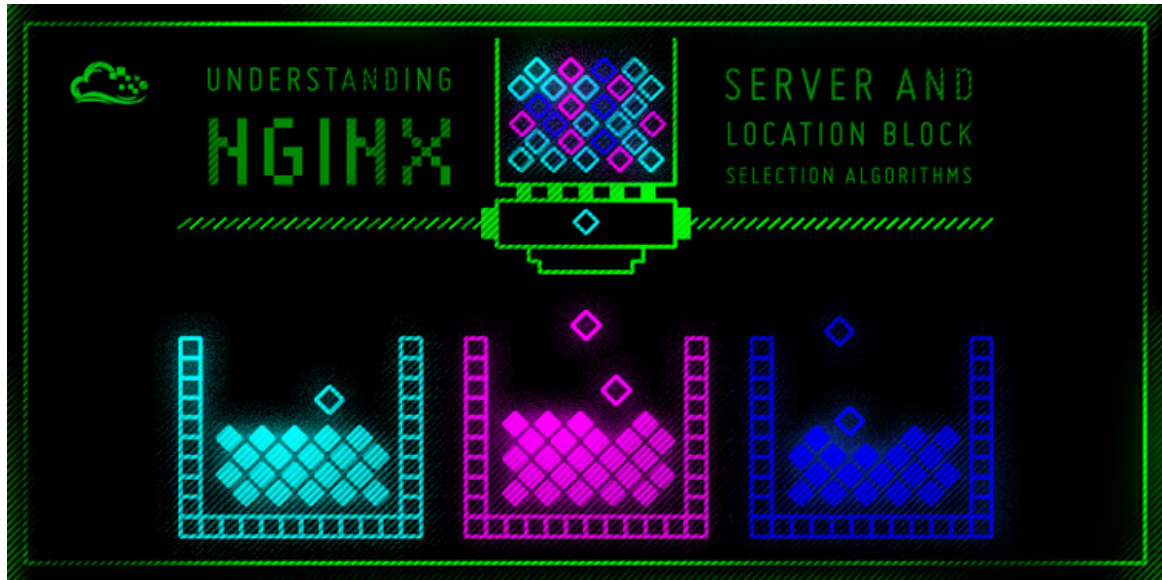
It is possible to provide more than one value, so I suggest adding any method you may be using to contact the server, like this:

```
                                /etc/nginx/sites-available/myproject

server {
    listen 80;
    server_name www.example.com example.com server_ip_address;
    . . .
}
```

Remember to restart Nginx after making changes.

Hopefully this helps a bit. Like I said, I'm unsure of why you're seeing this behavior, but this is my first best guess.



## Understanding Nginx Server and Location Block Selection Algorithms

by Justin Ellingwood

Nginx is one of the most popular web servers in the world. In this guide, we will discuss how Nginx selects the server and location block that will handle a given client's request. We will go over the algorithm in place, as well as the directives and options

---

^ [ArdaMavi](#) January 9, 2017



o I do all settings now and i get "bad gateway" ?

---

^ [ArdaMavi](#) January 11, 2017



o Ok!

Thank you i do.

I create user and try again all settings. Thats it.

But Why ? :D :D

---

^ [de](#) February 21, 2017



o So I got the tutorial site working, but now I would like to use my own flask app (which uses modules) and serve html,css and js files. I tried to create static and template folders to the myproject folder, but it doesn't work. I also tried to install all the modules my app requires. I just keep getting "502 bad gateway"

I already added this to some nginx file



```
location /static {
    root /home/user/myproject;
}

location ~ \.(jpg|jpeg|png|gif|css|js)$ {
    root /home/user/myproject/static;
}

location /templates {
    root /home/user/myproject;
}
```

I'm really bad with this kind of server setup things. I picked flask because it was easy to get working on localhost but now it seems very complicated. But yeah I would like to get my flask app working.

---

 [pipskweak](#) March 12, 2017

<sup>0</sup> I was having 502 bad gateway errors as well, I eventually fixed it by changing the

/etc/systemd/system/myproject.service file.

```
[Unit]
```

```
Description=Gunicorn instance to serve flmd
```

```
After=network.target
```

```
[Service]
```

```
User=root
```

```
Group=www-data
```

```
WorkingDirectory=/root/www/flmd
```

```
Environment="PATH=/root/www/flmd/flmd/bin"
```

```
ExecStart=/root/www/flmd/flmd/bin/gunicorn --workers 3 --bind unix:flmd.
```

```
[Install]
```

```
WantedBy=multi-user.target
```



it's related to the **myprojectenv** line, which should actually just be **myproject**

---

^ tkah March 1, 2017



1

I'm in the process of migrating a Flask application of mine to Digital Ocean and found this tutorial to be very helpful. However, though things worked perfectly fine when I tested the app in the earlier steps, once I made the systemd unit file and tried to test it I got bad gateway errors. I ran **sudo journalctl -u myproject** and saw that the errors were being thrown when my project was reading from a couple json/txt files I have in my project. The error being thrown was *UnicodeDecodeError: 'ascii' code cannot decode byte ....* After searching online a bit, I found that by adding **LC\_ALL="en\_US.UTF-8"** to `/etc/environment` I was able to access my project successfully.

Hope this helps someone else.

---

^ hanh April 5, 2017



1

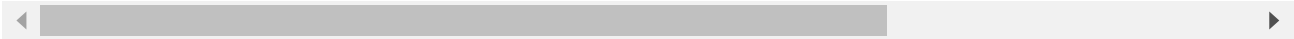
Thanks for this helpful tutorial.

I am trying a variation and struggling if you have any input.

I am using Anaconda environments, and so in my systemd unit file the [Service] part looks different.

With my conda env named 'flask', I tried:

```
[Service]
User=han
Group=www-data
WorkingDirectory=/home/han/myproject
Environment="PATH=/home/han/miniconda3/envs/flask/bin"
ExecStart=/home/han/miniconda3/envs/flask/bin/gunicorn --workers 3 --bind u
```



No error messages (or messages at all) when I run:

```
sudo systemctl start mysite
```

But when I run:

```
sudo systemctl enable mysite
```

I get:

`*Failed to execute operation: Invalid argument*`

I tried it out again using

```
ExecStart=/home/han/miniconda3/envs/flask/bin/gunicorn --workers 3 --bind 0
```



And saw the same error output from `sudo systemctl enable mysite` (but reassuringly could see the website start/stop with `systemctl start/stop`).

Also, running this works fine (from the `mysite` directory, with my *flask* conda env activated):

```
gunicorn --workers 3 --bind unix:mysite.sock -m 007 wsgi:app
```


Any ideas what's wrong?

Where-to-look-next guidance?

I am having a hard time finding anything about systemd error logs (if they exist) or how to troubleshoot systemd unit files.

Thanks

---

 [sahopkins93](#) May 9, 2017

0 I am logged in as root and have set permissions for root:www-data to rw for solytics\_api.sock.

I am seeing this error repeatedly however,

```
[crit] 3702#3702: *1 connect() to unix:/root/Solytics/SolyticsScript/SolyticsAPI/solytics_api.sock failed (13: Permission denied) while connecting to upstream,
```

under /etc/systemd/system/solytics\_api.service I have set

user: root

group: www-data

I have also run `chown -R root:www-data solytics_api.sock` from the working directory.

Any advice?

---

^ [peterschutt](#) June 23, 2017

0 This is a nice, thorough tutorial.

A couple of things that I found:

1. Item 31. I use a few special built python modules and have /path/to/my/packages added to my PYTHONPATH in ~/.bashrc. If I open a python interpreter I can import them no problems but the myproject.service fails when trying to import them. To get around this I added to the myproject.service file:

```
[Service]
```

```
Environment=PYTHONPATH=/path/to/my/packages
```

2. The tutorial that is linked in the note about setting up Lets Encrypt implies that there is not a fully automated option to set up Lets Encrypt with nginx. This is no longer the case, go here: <https://certbot.eff.org/> and then select nginx and operating system. Very easy.

---

[berryse13](#) June 30, 2017

^ I followed the instructions and got everything working on my IP address however when I visit  
o mydomain.com I get a "Welcome to nginx!"

Not a clue what is going on

---

^ [chris.h.devries](#) July 1, 2017

o I would guess it's that you have two configurations in `/etc/nginx/sites-enabled/`, including the one you want and one called `default`. I would remove the `default` one and change the server line of your nginx configuration file so it reads.

```
server_name server_domain_or_IP default_server;
```

---

^ [berryse13](#) July 1, 2017

o Thanks, I already solved it.

It was nothing to do with the code, I had a problem with FileZilla and it wasn't updating my



file edits.

Before I had:

```
server_name server_IP
```

The edit that work:

```
server_name server_IP server_domain www.server_domain
```

Load More Comments



This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.



Copyright © 2019 DigitalOcean™ Inc.

[Community](#) [Tutorials](#) [Questions](#) [Projects](#) [Tags](#) [Newsletter](#) [RSS](#) 

---

[Distros & One-Click Apps](#) [Terms, Privacy, & Copyright](#) [Security](#) [Report a Bug](#) [Write for DOnations](#) [Shop](#)