

How to Deploy a FastAPI Application with Gunicorn and Nginx on Ubuntu 24.04

Learn how to deploy a FastAPI application using Gunicorn as the WSGI server and Nginx as a reverse proxy on Ubuntu 24.04 for improved performance and security.

Contents

01	Introduction	3
02	Prerequisites	3
03	Setup FastAPI Application	3
04	Deploy FastAPI using Gunicorn	5
05	Setup Nginx as Reverse Proxy	8
06	Secure Nginx with an SSL Certificate	11
07	Conclusion	12

Introduction

FastAPI is a modern Python web framework for building high-performance APIs and web applications. It supports asynchronous programming using `async` and `await`, enabling efficient handling of many simultaneous client connections. Built on the ASGI (Asynchronous Server Gateway Interface) standard, FastAPI is ideal for real-time applications and remains compatible with WSGI for traditional deployments. It also offers automatic, interactive API documentation through Swagger UI, making development and testing easier and faster.

This article explains how to deploy FastAPI applications using Gunicorn as the application server and Nginx as a reverse proxy on an Ubuntu 20.04 server.

Prerequisites

- Have access to an [Ubuntu 24.04 instance](#) as a non-root sudo user
- Set up a [domain A record pointing to the instance's IP address](#). For example, `fastapi.example.com`.

Setup FastAPI Application

In this section, you will create a FastAPI application inside a virtual environment. You will set up the project directory, install necessary dependencies, and test the application using the uvicorn ASGI server. Follow the steps below to setup the FastAPI application.

1. Navigate to your home directory.

```
CONSOLE
$ cd ~
```

2. Create a new directory for your FastAPI project.

```
CONSOLE
```

```
$ mkdir fastapi_demo
```

3. Navigate into the project directory.

```
CONSOLE
```

```
$ cd fastapi_demo
```

4. Update the APT package index.

```
CONSOLE
```

```
$ sudo apt update
```

5. Install the `python3-venv` package.

```
CONSOLE
```

```
$ sudo apt install -y python3-venv
```

6. Create a Python virtual environment named `env`.

```
CONSOLE
```

```
$ python3 -m venv env
```

7. Activate the virtual environment.

```
CONSOLE
```

```
$ source env/bin/activate
```

8. Install the `fastapi` package with all optional dependencies.

```
CONSOLE
```

```
$ pip install fastapi[all] wheel
```

9. Add the following FastAPI application code into the file.

PYTHON

```
from fastapi import FastAPI
app = FastAPI()

@app.get("/")
async def home():
    return {"message": "Hello World"}
```

Save and close the file by pressing Ctrl + X, then Enter.

10. Create a temporary web server using `uvicorn`.

CONSOLE

```
$ uvicorn app:app
```

11. Open a new terminal session and run the following command.

CONSOLE

```
$ curl http://localhost:8000
```

Your output should be similar to the one below:

```
{"message": "Hello World"}
```

Deploy FastAPI using Gunicorn

Gunicorn is a Python WSGI HTTP server for UNIX systems. It simplifies the management of FastAPI applications and supports ASGI via Uvicorn worker

classes. Follow the steps below to deploy your FastAPI application using Gunicorn.

1. Install the `gunicorn` Python package.

CONSOLE

```
$ pip install gunicorn
```

2. Start a temporary Gunicorn server with the Uvicorn worker.

CONSOLE

```
$ gunicorn app:app -k uvicorn.workers.UvicornWorker
```

3. Open a new session and test the **Gunicorn** deployment using `curl`.

CONSOLE

```
$ curl http://localhost:8000
```

Your output should be similar to the one below:

```
{"message": "Hello World"}
```

4. Create a Gunicorn configuration file named **gunicorn_conf.py**.

CONSOLE

```
$ nano gunicorn_conf.py
```

5. Add the following code into the file.

PYTHON

```
from multiprocessing import cpu_count

# Socket path
bind = 'unix:/home/linuxuser/fastapi_demo/gunicorn.sock'
```

```
# Worker options
workers = cpu_count() + 1
worker_class = 'uvicorn.workers.UvicornWorker'

# Logging options
loglevel = 'debug'
accesslog = '/home/linuxuser/fastapi_demo/access_log'
errorlog = '/home/linuxuser/fastapi_demo/error_log'
```

Save and close the file by pressing Ctrl + X, then Enter. Replace the paths of the variables if they are different in your case.

6. Create and open the systemd unit file for the FastAPI Gunicorn service.

CONSOLE

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

7. Add the following systemd service configuration to the file.

INI

```
[Unit]
Description=Gunicorn Daemon for FastAPI Demo Application
After=network.target

[Service]
User=linuxuser
Group=www-data
WorkingDirectory=/home/linuxuser/fastapi_demo
ExecStart=/home/linuxuser/fastapi_demo/venv/bin/gunicorn -c /
home/linuxuser/fastapi_demo/gunicorn_conf.py app:app

[Install]
WantedBy=multi-user.target
```

Save and close the file by pressing Ctrl + X, then Enter.

8. Reload the systemd daemon.

CONSOLE

```
$ sudo systemctl daemon-reload
```

9. Start and enable the FastAPI Gunicorn service.

CONSOLE

```
$ sudo systemctl enable --now fastapi_demo
```

Your output should be similar to the one below:

```
Created symlink /etc/systemd/system/multi-user.target.wants/fastapi_demo.service  
→ /etc/systemd/system/fastapi_demo.service.
```

10. View the status of newly created service.

CONSOLE

```
$ sudo systemctl status fastapi_demo
```

11. Test the response from the socket file using `curl`.

CONSOLE

```
$ curl --unix-socket /home/linuxuser/fastapi_demo/  
gunicorn.sock http://localhost
```

Your output should be similar to the one below:

```
{"message": "Hello World"}
```

Setup Nginx as Reverse Proxy

In this section, you will configure Nginx as a reverse proxy for your FastAPI application. Nginx is a lightweight, high-performance web server commonly used to forward client requests to application servers like Gunicorn. It allows you to serve your application on standard HTTP and HTTPS ports (80 and 443),

improves request handling and performance, and supports SSL termination for secure connections.

1. Install the Nginx Web Server.

CONSOLE

```
$ sudo apt install -y nginx
```

2. Create a virtual host configuration.

CONSOLE

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

3. Add the following configuration. Replace **fastapi.example.com** with your actual domain name.

INI

```
server {
    listen 80;
    server_name fastapi.example.com;

    location / {
        proxy_pass http://unix:/home/linuxuser/fastapi_demo/
gunicorn.sock;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For
$proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;

        # Optional: Handle WebSocket connections
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";

        # Timeout settings
        proxy_connect_timeout 60s;
        proxy_read_timeout 120s;
    }
}
```

Save and close the file with Ctrl + X, then Enter.

4. Create a symbolic link to the **sites-enabled/** directory to activate the vhost.

CONSOLE

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

5. Check the Nginx configuration syntax for any errors.

CONSOLE

```
$ sudo nginx -t
```

Your output should be similar to the one below:

```
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
```

6. Reload the Nginx service to apply the new configuration.

CONSOLE

```
$ sudo systemctl reload nginx
```

7. Allow HTTP and HTTPS traffic through the firewall.

CONSOLE

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

8. Verify the firewall status to confirm the rules are applied.

CONSOLE

```
$ sudo ufw status
```

Secure Nginx with an SSL Certificate

To encrypt traffic and protect user data, you can secure your FastAPI application with HTTPS using a free SSL certificate from Let's Encrypt. This section shows you how to configure SSL for your Nginx reverse proxy using Certbot.

1. Install Certbot and the Nginx plugin

CONSOLE

```
$ sudo apt install -y certbot python3-certbot-nginx
```

2. Request and install an SSL certificate.

CONSOLE

```
$ sudo certbot --nginx -d fastapi.example.com
```

Certbot will automatically update your Nginx configuration. When prompted, enter your email address and agree to the terms of service.

3. Visit the following link in a browser to confirm SSL is working.

```
https://fastapi.example.com
```

4. Test automatic certificate renewal.

CONSOLE

```
$ sudo certbot renew --dry-run
```

If no errors appear, your SSL certificate will renew automatically every 90 days.

Conclusion

You have deployed a FastAPI application with Gunicorn and Nginx on a production ready Ubuntu server. This setup uses a Unix socket for efficient communication between Gunicorn and Nginx, and leverages systemd to manage the application as a background service. By securing Nginx with a free SSL certificate from Let's Encrypt, you've ensured that all client traffic is encrypted over HTTPS. This configuration provides a robust, scalable, and secure environment for running FastAPI applications in production. For more information related to FastAPI, visit the official [FastAPI website](#).



VULTR

