**[Que-48] - How would you deploy a Flask application to a production server using Gunicorn and Nginx**

Deploying a Flask application to a production server typically involves using Gunicorn as the WSGI server and Nginx as a reverse proxy server to handle client requests. Here’s a step-by-step guide on how to deploy a Flask application using Gunicorn and Nginx:

### **Step 1: Set Up Your Flask Application**

Make sure your Flask application is ready for deployment. Ensure it runs correctly on your local development environment.

### **Step 2: Install Required Packages**

SSH into your production server and install the necessary packages:

**Bash:**

sudo apt-get update  
sudo apt-get install nginx python3-pip python3-dev build-essential libssl-dev libffi-dev python3-setuptools  
sudo apt-get install python3-venv

### **Step 3: Set Up a Virtual Environment**

Create a virtual environment for your Flask application to isolate dependencies:

**Bash:**

python3 -m venv myprojectenv  
source myprojectenv/bin/activate

### **Step 4: Install Flask and Gunicorn**

Install Flask and Gunicorn within your virtual environment:

**Bash:**

pip install flask gunicorn

### **Step 5: Configure Gunicorn**

Create a Gunicorn configuration file (gunicorn\_config.py) in your project directory. This file should contain Gunicorn settings, such as the number of workers and the socket location:

**Python:**

# gunicorn\_config.py  
  
bind = 'unix:/tmp/myapp.sock'  
workers = 3

Adjust bind and workers according to your application's needs.

### **Step 6: Create a Systemd Service Unit for Gunicorn**

Create a systemd service unit file (myapp.service) to manage Gunicorn:

**Bash:**

sudo nano /etc/systemd/system/myapp.service

Add the following configuration (modify paths as necessary):

[Unit]  
Description=Gunicorn instance to serve myapp  
After=network.target  
  
[Service]  
User=your\_username  
Group=www-data  
WorkingDirectory=/path/to/your/app  
Environment="PATH=/path/to/your/app/myprojectenv/bin"  
ExecStart=/path/to/your/app/myprojectenv/bin/gunicorn --config gunicorn\_config.py wsgi:app  
  
[Install]  
WantedBy=multi-user.target

### **Step 7: Start and Enable the Gunicorn Service**

Start and enable the Gunicorn service:

**Bash:**

sudo systemctl start myapp  
sudo systemctl enable myapp

### **Step 8: Configure Nginx**

Create an Nginx server block configuration file (myapp) in /etc/nginx/sites-available/:

**Bash:**

sudo nano /etc/nginx/sites-available/myapp

Add the following configuration (modify paths and server\_name as necessary):

**Nginx:**

server {  
 listen 80;  
 server\_name your\_domain\_or\_server\_ip;  
  
 location / {  
 include proxy\_params;  
 proxy\_pass <http://unix:/tmp/myapp.sock>;  
 }  
}

### **Step 9: Enable the Nginx Server Block**

Enable the Nginx server block by creating a symbolic link to the sites-enabled directory:

**Bash:**

sudo ln -s /etc/nginx/sites-available/myapp /etc/nginx/sites-enabled

### **Step 10: Test Nginx Configuration and Restart Nginx**

Test the Nginx configuration for syntax errors:

**Bash:**

sudo nginx -t

If the test is successful, restart Nginx to apply the changes:

**Bash:**

sudo systemctl restart nginx

### **Step 11: Configure Firewall (if necessary)**

If your server uses a firewall, allow HTTP traffic (port 80):

**Bash:**

sudo ufw allow 'Nginx HTTP'

### **Step 12: Final Steps**

* Ensure your Flask application is configured for production (e.g., app.config['SECRET\_KEY'], DEBUG=False).
* Monitor logs (/var/log/nginx/error.log, /var/log/nginx/access.log, journalctl -u myapp) for troubleshooting.