*Great! Setting up Nginx as a reverse proxy for your Django application is a common approach for serving web applications. Here’s how you can do it step-by-step:*

*Reverse Proxy: It forwards client requests to another server and returns the response back to the client.*

Example of Reverse Proxy with Nginx:

Imagine you have a Django application running on a private server (VM3) with an IP like 10.0.3.4. To make this application accessible via the internet, you deploy Nginx on a public server (VM2). Nginx acts as the reverse proxy:

Client requests http://public-server-ip/.

Nginx forwards this request to the Django app running on the private server.

Django sends back the response to Nginx.

Nginx delivers the response to the client.

This way, the client interacts with Nginx (public-facing), but the actual application logic runs on the private server.

Visual Example of Reverse Proxy:

Client → Nginx (VM2) → Django Application (VM3)

Nginx (VM2) forwards the request to VM3.

VM3 processes the request and sends the response back via Nginx.

### Steps to Set Up Nginx as a Reverse Proxy

**1. \*\*Install Nginx\*\*:**

First, SSH into your public VM (VM2) and install Nginx:

```bash

sudo apt update

sudo apt install nginx

```

**2. \*\*Configure Nginx\*\*:**

You need to create a configuration file for your Django application. Open the default configuration file or create a new one for your app:

```bash

**sudo vim /etc/nginx/sites-available/my\_django\_app**

```

Here’s a basic configuration you can use:

```

server {

listen 80;

server\_name \_\_;

# Serve static files

location /{

alias /new\_chatapp/fundoo/static/;

}

location / {

proxy\_pass http://10.0.3.4:8000;

# Optional: Adjust proxy settings to improve performance

proxy\_read\_timeout 300;

proxy\_connect\_timeout 300;

    }

} ```

Make sure to replace `/path/to/your/django/project` with the actual path to your Django project's static files.

**3. \*\*Enable the Configuration\*\*:**

Create a symbolic link to enable your site configuration:

```bash

**sudo ln -s /etc/nginx/sites-available/my\_django\_app /etc/nginx/sites-enabled**

```

**4. \*\*Test the Nginx Configuration\*\*:**

Before restarting Nginx, test the configuration for any syntax errors:

```bash

sudo nginx -t

```

**5. \*\*Restart Nginx\*\*:**

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

```bash

sudo systemctl restart nginx

```

**6. \*\*Access Your Application\*\*:**

Now, you should be able to access your Django application through the public IP of VM2:

```

http://13.70.65.137/

```

### Additional Notes

- \*\*Static Files\*\*: Make sure to collect your static files by running:

```bash

python3 manage.py collectstatic

```

- \*\*Firewall Rules\*\*: Ensure that the firewall on VM2 allows traffic on port 80 (HTTP). You can check your firewall rules and add a rule if necessary:

```bash

sudo ufw allow 'Nginx Full'

```

Let me know if you need further assistance with any of these steps or have any questions!

This Nginx configuration file defines how Nginx will handle traffic on port `80` (HTTP) for a server identified by the IP address `16.170.155.60`. It also sets up a reverse proxy to forward requests to a backend service (e.g., a Django app running on Gunicorn) located at `10.0.2.88:8000`. Let's break it down step by step:

### 1. `listen 80;`

- \*\*What it does\*\*: This tells Nginx to listen for HTTP requests on port `80`.

- \*\*Why it's used\*\*: Port 80 is the default port for HTTP traffic, meaning when users access the server using `http://16.170.155.60/`, Nginx will listen for those requests.

### 2. `server\_name 16.170.155.60;`

- \*\*What it does\*\*: Specifies the domain name or IP address of the server.

- \*\*Why it's used\*\*: It tells Nginx to respond to requests that are sent to this specific IP address (`16.170.155.60` in this case). It can also be a domain name if you have one.

### 3. `location / { ... }`

- \*\*What it does\*\*: This block defines how Nginx serves static files (e.g., CSS, JS, images) when a request is made to a URL path starting with `/`.

```

location / {

alias /new\_chatapp/fundoo/static/;

}

```

- \*\*alias /new\_chatapp/fundoo/static/\*\*: This sets the directory where static files (like CSS, JavaScript, and images) are stored on the server. It serves files located in `/new\_chatapp/fundoo/static/` when someone requests `http://16.170.155.60/static/filename`.

- \*\*Why it's used\*\*: Static files (such as images, stylesheets, or JavaScript files) need to be served quickly. Nginx is excellent at serving static content efficiently without passing these requests to the application server (Django in this case).

### 4. `location / { ... }`

- \*\*What it does\*\*: This block handles all other requests (not for static files). It sets up a reverse proxy, which forwards incoming requests to the backend server (e.g., Django app running on Gunicorn at `10.0.2.88:8000`).

```nginx

location / {

proxy\_pass http://10.0.2.88:8000;

proxy\_read\_timeout 300;

proxy\_connect\_timeout 300;

}

```

#### Key Configuration Directives:

- \*\*`proxy\_pass http://10.0.2.88:8000;`\*\*

- \*\*What it does\*\*: Forwards the request to the Django application running on the private IP `10.0.2.88` on port `8000`.

- \*\*Why it's used\*\*: The Django app is running on a private machine (not directly accessible to the outside world). Nginx forwards traffic to the backend and returns the response to the client.

#### Optional: Proxy Timeout Settings

- \*\*`proxy\_read\_timeout 300;`\*\*

- \*\*What it does\*\*: Increases the amount of time Nginx will wait for a response from the backend server.

- \*\*Why it's used\*\*: Useful if the backend server takes time to respond, preventing Nginx from cutting off long-running processes (e.g., heavy database queries).

- \*\*`proxy\_connect\_timeout 300;`\*\*

- \*\*What it does\*\*: Sets how long Nginx should wait to establish a connection to the backend server.

- \*\*Why it's used\*\*: Ensures that the connection isn’t terminated prematurely when the backend is slow to respond.

---

### Summary:

- This Nginx configuration serves \*\*static files\*\* from `/new\_chatapp/fundoo/static/` and \*\*proxies all other requests\*\* to a backend Django server running on `10.0.2.88:8000`.

- Nginx handles the reverse proxy by forwarding requests to the backend server while passing necessary headers to preserve client information.

**Remove key:-**

ssh-keygen -f "/home/sysadmin/.ssh/known\_hosts" -R "10.0.2.6"

ssh sysadmin@10.0.2.6