# Triển khai lên server public

## Với django

### Tải project từ trên github về

- Như đã nói ở phần trên, project sẽ được lưu các phần tại github

- Sau đó ta sẽ “clone” các project này từ trên github về server và chạy project bình thường

### Cài đặt các requirements của venv

- Nếu bị lỗi không tải đưuọc psycopg2 thì trước hết phải tải hai cái sau

* Install 'libpq-dev' by running the command: sudo apt-get install libpq-dev.
* Install 'python-dev' by running the command: sudo apt-get install python-dev.

### Cài đặt postgresql trên linux

- sudo -u postgres psql; để vào command prompt của psql

- CREATE DATABASE smartfarm; để tạo database mới

- Tạo user mới

+ CREATE USER year3 WITH PASSWORD 'year3';

+ ALTER ROLE year3 SET client\_encoding TO 'utf8';

+ ALTER ROLE year3 SET default\_transaction\_isolation TO 'read committed';

+ ALTER ROLE year3 SET timezone TO 'UTC';

+ GRANT ALL PRIVILEGES ON DATABASE smartfarm TO year3;

NEUPDATE {

>> sudo -u postgres psql //login postgres as postgres

>> CREATE USER year3 WITH PASSWORD 'year3';

>> CREATE DATABASE smartfarm;

>> ALTER DATABASE smartfarm OWNER TO year3;

}

- Thay đổi setting của django để có thể kết nối được với postgresql do bây giờ ta đang dùng user year3 với database là smartfarm

DATABASES = {

'default': {

'ENGINE': 'django.db.backends.postgresql',

'NAME': 'smartfarm',

'USER': 'year3',

'PASSWORD': 'year3',

'HOST': 'localhost',

'PORT': '5432',

}

}

- Thay đổi các thông số của postgres trong các file dùng, như hàm insert\_to\_DB

def insert\_to\_DB(self, data,

\_\_database='smartfarm',

\_\_user='year3',

\_\_password='year3',

\_\_host='localhost',

\_\_port='5432') -> None:

- Sau đó chạy các lệnh “migrate” của django để tích hợp các thay đổi mới vào project và postgres

### Cài đặt django public

Source: <https://www.honeybadger.io/blog/deploy-django-ubuntu/>

\*CHÚ Ý: Từ bây giờ các câu lệnh ubuntu đều phải có từ khóa “sudo” đằng trước - Set Up a Firewall for Network Security

Setting up a firewall on your server provides protection for your server network and sets rules for incoming and outgoing requests to your server. We will use [Uncomplicated Firewall](https://help.ubuntu.com/community/UFW) (UFW) in this tutorial; it’s easy to set up, and you don't need to install anything new on your server.

UFW comes with Ubuntu by default, but its activity is disabled. You can enable UFW on your server by running the following command:

Note: Make sure that you have SSH into your remote server.

ufw enable

Allow all outgoing ports by running the command below. You can allow all outgoing requests because it doesn’t pose a threat from external sources.

ufw default allow outgoing

Run the command below to prevent access to your server from the outside world.

ufw default deny incoming

When you ran the command above, you also blocked SSH requests. To continue using SSH to access our server, you must enable SSH. To do this, run the following command:

ufw allow ssh

Allow *http/tcp* traffic on the firewall so that your browser can access your app.

sudo ufw allow http/tcp

## Với Frontend

Source: https://dev.to/xarala221/the-easiest-way-to-deploy-a-react-web-application-2l8a

Trước hết cũng như backend ta cần clone project Frontend của chúng ta từ github về

### Cài đặt các module cần thiết

- Cài môi trường cho reactjs

$ sudo apt-get install curl

$ curl -sL https://deb.nodesource.com/setup\_13.x | sudo -E bash –

(Sau đó gõ tiếp câu lệnh tiếp theo)

$ sudo apt-get install nodejs

$ sudo apt install npm

- Ta cũng cần cài nginx để làm một http server, đã được cài ở phần backend bên trên

- Tiếp đến đi vào thư mục react-admin và tải các denpencies cần thiết của project bằng lệnh “npm install”

- Chạy thử project bằng lệnh “npm run start”

+ nếu như bị lỗi  “ENOSPC: System limit for number of file watchers reached“ thì ta có thể sửa bằng hai câu lệnh sau

“echo fs.inotify.max\_user\_watches=524288 | sudo tee -a /etc/sysctl.conf” và tiếp theo là câu lệnh “sudo sysctl -p” rồi sau đó ta thử lại câu lệnh “npm start” và sẽ chạy thành công

=> Lúc này nếu react đã chạy thành công thì react sẽ chạy trong mạng local với cổng được kết nối đến là cổng 3000

- Chạy lệnh “npm run build” để có folder “build” của project react

+ Trước khi chạny lệnh build thì một số chú ý sau cần được làm rõ:

A screenshot of a computer program

Description automatically generated

1. Lệnh “npm run build” sẽ tạo ra một folder có cấu trúc y hệt như folder public khi chưa có lệnh và kèm theo đó là một folder static như là bản rút gọn của tất cả các code trong folder src

2. Trong bản folder public folder assets được đặt trong folder smartfarm nên trong code mỗi khi cần <img></img> một ảnh nào đó thì ta có thể lấy từ src “src={`../../smartfarm/assets/logo\_lab.png`}” và khi triển khai lên public thì các src này cũng sẽ được gọi bằng cách gắn link trên vào phía sau ipaddress của server “<http://27.71.227.1/smartfarm/>assets/image.png”, do server có ipaddress là [“27.71.227.1”](http://27.71.227.1/smartfarm/) nhưng ta sẽ dùng path “<http://27.71.227.1/smartfarm/>” để gọi chương trình react js nên khi chúng ta build xong ta cần phải đầu tiên đưa folder asset ra ngoài mức level của folder smartfarm và lúc này folder smartfarm không còn tác dụng nữa. Tiếp đến phần tiếp theo sẽ là các bước set up tiếp nginx để triển khai chương trình.

Lỗi

## Cài đặt nginx

### Set up nginx

- Set Up NGINX as a Reverse Proxy

In this section, you will need [NGINX](https://www.nginx.com/) for reverse proxy, which allows your server to listen for the application running and then redirect requests to your remote IP address.

To do this, install NGINX by running the following command:

apt-get install nginx

- Remove the default configuration file provided by the NGINX installation so that we can create our own.

rm /etc/nginx/sites-enabled/default

- Có thể chạy lệnh “sudo nginx -t” để kiểm tra xem đang có lỗi gì đang xảy ra với nginxv hay không

- Run *nano /etc/nginx/sites-enabled/project* to create and open a new NGINX configuration file,

+ This command opens the file "/etc/nginx/*sites-enabled/project*  " in the Nano text editor with superuser (sudo) privileges. You can use this command to modify the Nginx configuration for project

- and paste the NGINX script below; then

+ Tiếp đó sau khi đã paste đoạn code , ta ấn ‘ctrl-x’ tiếp đến ấn ‘y’ và cuối cùng là ấn enter

+ Correct script:

server {

listen 80;

server\_name 27.71.227.1;

location /smartfarm {

root /home/ubuntu/smartfarm/FrontEnd/Year3\_dev\_Frontend/Ver2\_use\_template/react-admin/build;

index index.html;

try\_files $uri $uri/ /index.html;

}

location /api {

proxy\_pass http://localhost:8000;

include /etc/nginx/proxy\_params;

proxy\_redirect off;

}

location / {

return 404;

}

}

>> Giải thích:

* **listen 80;**: This line specifies that Nginx should listen on port 80 for incoming HTTP requests.
* **server\_name 27.71.227.1;**: This line sets the server name or IP address for which this configuration block will be applied. In this case, it is set to the IP address "27.71.227.1".
* React App Configuration:
  + The **location /** block handles requests for your React app.
  + The **root** directive specifies the root directory where the static files for your React app are located.
  + The **index** directive specifies the default file to serve when a directory is requested.
  + The **try\_files** directive checks if the requested file exists. If it doesn't, it falls back to serving the **index.html** file. This is important for handling client-side routing in a React app.
* Django App Configuration:
  + The **location /api** block handles requests for your Django app's API.
  + The **proxy\_pass** directive specifies the URL where the requests should be forwarded. In this case, it's **http://localhost:8000**, which is the address of your Django app's server.
  + The **include /etc/nginx/proxy\_params** line includes additional proxy-related configurations.
  + The **proxy\_redirect off** directive disables the rewriting of HTTP redirects sent by the upstream server (your Django app).

By configuring Nginx in this way, any requests that start with **/api** will be forwarded to your Django app running on **localhost:8000**. This allows Nginx to act as a reverse proxy, seamlessly routing requests between the React app and the Django app.

Requests for the React app itself (e.g., **/**, **/about**, etc.) will be handled by Nginx directly. Nginx will serve the static files from the specified **root** directory, and if the requested file doesn't exist, it will serve the **index.html** file. This ensures that your React app's client-side routing works correctly.

In summary, this configuration enables you to serve both your React app and your Django app on the same server using Nginx. Nginx acts as a reverse proxy, forwarding API requests to your Django app and serving the static files of your React app.

### Với React

- Ta phải sửa lại cấu trúc folder build như sau:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Và bên trong file index.html của folder smartfarm phải sửa lại như sau:

Thông thường ở folder build bình thường hai link này sẽ không có /smartfarm đằng trước

<!doctype html>

<html lang="en">

<head>

<meta charset="utf-8"/>

<link rel="icon" href="/favicon.ico"/>

<meta name="viewport" content="width=device-width,initial-scale=1"/>

<meta name="theme-color" content="#000000"/>

<meta name="description" content="Web site created using create-react-app"/>

<link rel="apple-touch-icon" href="/logo192.png"/>

<link rel="manifest" href="/manifest.json"/>

<title>React App</title>

<script defer="defer" src="/smartfarm/static/js/main.ba43becb.js"></script>

<link href="/smartfarm/static/css/main.8af738c5.css" rel="stylesheet">

</head>

<body>

<noscript>You need to enable JavaScript to run this app.</noscript>

<div id="root"></div>

</body>

</html>

**- Giải thích**

location /smartfarm {

root /home/ubuntu/smartfarm/FrontEnd/Year3\_dev\_Frontend/Ver2\_use\_template/react-admin/build;

index index.html;

try\_files $uri $uri/ /index.html;

}

Khi ta đi đến url http:// 27.71.227.1/…, nếu đằng sau là smartfarm thì block location này sẽ được thông qua khi đó nginx sẽ đi đến đường root kia trong máy ubuntu nhưng kèm đằng sau là từ khóa “smartfarm” nghĩa là đi đến path “/home/ubuntu/smartfarm/FrontEnd/Year3\_dev\_Frontend/Ver2\_use\_template/react-admin/build/smartfarm/” để lấy file index.html ở đó và response lại cho client, đó là lý do mà nếu để là “location /” thì sẽ lấy file html ở “/home/ubuntu/smartfarm/FrontEnd/Year3\_dev\_Frontend/Ver2\_use\_template/react-admin/build” nên ta phải tạo thêm folder mới là “  
smartfarm”. Tiếp đó khi mà file html trong folder smartfarm đã được đưa về cho client thì hai dòng “<script defer="defer" src="/smartfarm/static/js/main.ba43becb.js"></script>

<link href="/smartfarm/static/css/main.8af738c5.css" rel="stylesheet">” sẽ tự động chạy bằng cách gắn hai src đó vào ipaddress 27.71.227.1 và do đó sẽ là <http://27.71.227.1/smartfarm/static/css/main.8af738c5.css> và sẽ đi vào “location /smartfarm” đi vào folder “/home/ubuntu/smartfarm/FrontEnd/Year3\_dev\_Frontend/Ver2\_use\_template/react-admin/build/smartfarm/” + “static/css/main.8af738c5.css” và mới load được. Nếu không sửa lại thì thành “<script defer="defer" src="/smartfarm/static/js/main.ba43becb.js"></script> ” thì sẽ đi vao`[http://27.71.227.1/ static/css/main.8af738c5.css](http://27.71.227.1/smartfarm/static/css/main.8af738c5.css) và sẽ bị 404 vì không tồn tại url đó.

**>>>>> Ngay từ lúc phát triển thì asset đã phải bị bọc trong một folder tên là smartfarm rồi thì khi build ra nó mới ở trong smartfarm đưuọc sửa lại**

**- Block location thứ 1**

location / {

root /home/ubuntu/smartfarm/FrontEnd/Year3\_dev\_Frontend/Ver2\_use\_template/react-admin/build;

index index.html;

try\_files $uri $uri/ /index.html;

}

-s

server {

server\_name 27.71.227.1 domain.com www.domain.com;

root /home/ubuntu/smartfarm/FrontEnd/Year3\_dev\_Frontend/Ver2\_use\_template/react-admin/build;

index index.html index.htm;

location / {

try\_files $uri /index.html =404;

}

}

+ server\_name put your IPaddress

+ root we use this to give the server the application located in the disk

+ index The main file

* **server**: Starts the configuration block for the server.
* **server\_name**: Specifies the server names or domain names that this configuration block will be applied to. In this case, it includes "27.71.227.1", "domain.com", and "[www.domain.com](http://www.domain.com/)".
* **root**: Sets the root directory where the static files of your React app are located. The specified path is "/home/ubuntu/smartfarm/FrontEnd/Year3\_dev\_Frontend/Ver2\_use\_template/react-admin/build".
* **index**: Defines the index files to look for when a directory is accessed. In this case, it specifies "index.html" and "index.htm".
* **location /**: Sets the configuration for the root location ("/") of the server.
* **try\_files $uri /index.html =404;**: Configures Nginx to first attempt to serve the requested file ($uri), and if it doesn't exist, fallback to serving the "index.html" file. If neither file is found, it will return a 404 error.

=> Ở đây không có line “listen 80” là do ở đây nginx sẽ mặc định nghe các request từ port 80 để serve ngược lại các context của react app làm response

Test script

server {

listen 80;

server\_name 27.71.227.1 domain.com www.domain.com;

location /smartfarm {

root /home/ubuntu/smartfarm/FrontEnd/Year3\_dev\_Frontend/Ver2\_use\_template/react-admin/build;

index index.html index.htm;

try\_files $uri /index.html =404;

}

}

suod ngin

server {

listen 80;

server\_name 27.71.227.1;

root /home/ubuntu/smartfarm/FrontEnd/Year3\_dev\_Frontend/Ver2\_use\_template/react-admin/build;

index index.html;

location / {

try\_files $uri $uri/ /index.html;

}

}

server {

listen 80;

server\_name 27.71.227.1;

location / {

root /home/ubuntu/smartfarm/FrontEnd/Year3\_dev\_Frontend/Ver2\_use\_template/react-admin/build;

index index.html;

try\_files $uri $uri/ /index.html;

}

location /API {

proxy\_pass http://localhost:8000;

}

}

- **Enable the file by linking to the sites-enabled dir**

sudo ln -s /etc/nginx/sites-available/react\_admin /etc/nginx/sites-enabled

+ This command creates a symbolic link (shortcut) between the Nginx configuration file "/etc/nginx/sites-available/react\_admin" and the "sites-enabled" directory. The "sites-enabled" directory is typically where active Nginx configurations are stored. By creating this symbolic link, you enable the configuration specified in "react\_admin" and make it active.

$ rm symlink\_name (to remove a symbolic link)

**Tổng quan**

Here's how the process works:

1. You have configured Nginx to listen on port 80 (default HTTP port) and specified the server block for the given server name(s) in your Nginx configuration.
2. When you access the URL "<http://27.71.227.1/>" from another laptop's browser (the last ‘/’ can go with anything other than “api”), a request is sent to the IP address "27.71.227.1" on port 80.
3. The network infrastructure, such as DNS resolution and routing, will direct the request to your Ubuntu server where Nginx is running.
4. Nginx, with the specified server block, receives the request and matches it with the configuration for the server name "27.71.227.1".
5. Nginx then looks for the requested resource within the specified root directory, which is set to "/home/ubuntu/smartfarm/FrontEnd/Year3\_dev\_Frontend/Ver2\_use\_template/react-admin/build".
6. If the requested resource is found (such as an HTML file), Nginx will serve it as the response.
7. If the requested resource is not found, Nginx will fall back to serving the "index.html" file due to the **try\_files $uri /index.html =404;** directive in the location block. This behavior is suitable for single-page applications (SPAs) where the client-side routing is handled by JavaScript.

Essentially, the above commands and script help you configure Nginx to serve your React app. The configuration block in the script specifies the server name, root directory, and location settings. By creating a symbolic link from the configuration file in "sites-available" to "sites-enabled," you enable the configuration and make it active in Nginx.

Make sure the root directory path in the configuration matches the location of your React app's build files. Additionally, ensure that you have built your React app and that the necessary files exist in the specified root directory.

After setting up Nginx and creating the symbolic link, you should be able to access your React app through the configured server names or domain names.

### Với Django

- Block thứ 2

location /api {

proxy\_pass http://localhost:8000;

include /etc/nginx/proxy\_params;

proxy\_redirect off;

}

}

The code will send requests to your IP address at [http://localhost:8000](http://localhost:8000/) because it is the host where your application will run. If you have a domain name, you can replace the IP address in the script.

- Now, restart NGINX to set the new configurations you just created.

sudo systemctl restart nginx

- Trường hợp gặp lỗi “Job for nginx.service failed because the control process exited with error code.” ta chạy các lệnh sau rồi chạy lại lệnh trên

$ sudo fuser -k 80/tcp

* 80/tcp: 7565 7579 7580 7582 7583 7586 8275 8277 8278 8279 8280

$ sudo fuser -k 443/tcp

- Tiếp đến, trong setting.py của django cần có host của server như sau:

ALLOWED\_HOSTS = ["27.71.227.1"]

* Cuối cùng là chạy lệnh “runserver” để chạy project django trên server và từ giờ chúng ta có thể truy cập đến project django thông qua url “<http://27.71.227.1>” từ bất cứ máy host nào trên internet thay vì chỉ truy cập được từ mạng local qua url “<http://localhost:8000>” như khi phát triển ứng dụng.

**- Cách nginx hoạt động**

Trước hết cần hiểu một cách cơ bản proxy là gì, proxy là một server hay là một phần mềm đứng giữa client với backend server để chịu trách nghiệm xử lý các yêu cầu request từ client đến backend server mà nó đứng giữa. Ở đây nginx cũng hoạt động như một proxy và chịu trách nghiệm xử lý các client gửi đến server ubuntu và chuyển hướng chúng đến django application.

Về cơ bản ta có thể hiểu đường đi của một request khi ta gọi đến máy chủ django như sau, đầu tiên khi ta truy cập url “<http://27.71.227.1/api/get/secondly_data/1>” thì mặc định do ta đang dùng protocal http, gateway hay model của mạng chúng ta dùng sẽ đưa request đó đến máy chủ có địa chỉ ip là “27.71.227.1” và đưa vào port 80 (là port mặc định để xử lý protocal http) nếu chúng ta đã cho phép server ubuntu mở port 80 trên nó (để xác nhận request sẽ mặc định được đưa vào port 80, chúng ta có thể trực tiếp khai báo port muốn truy cập vào url “<http://27.71.227.1:80/api/get/secondly_data/1>” và vẫn đạt được kết quả mong muốn) và lúc này nginx sẽ thực hiện công việc của nó.

Như chúng ta đã cài đặt một đoạn script server cho nginx như bên trên nginx sẽ liên tục nghe ở cổng port 80 xem có request gì không (tất nhiên chúng ta phải mở cổng 80 trên server ubuntu) và tiếp đó sẽ đưa request đó đến “<http://localhost:8000>” xử lý.

Một điều chú ý ở đây là server local của django vẫn chạy thông qua port 8000 (port này không được mở public trên serverr ubuntu mà chỉ có thể truy cập local), nginx ở đây đóng vai trò như một server đứng giữa client và django giúp các request public từ trên mạng internet có thể tiếp cận project django đang chạy local trên máy chủ ubuntu nhưng thông qua nginx sẽ được mã hóa, bảo mật, phân luồng, chia độ tắc nghẽn và tiếp cận django application đang chạy local để được xử lý. Django application sau khi sử lý xong request đó sẽ đưa response lại cho proxy đứng giữa là nginx, sau đó được mã hóa, bảo mật và được đưa về cho client. Do đó đằng sau django application vẫn đang chạy local như khi đang trong quá trình phát triển tuy nhiên nhờ có proxy nginx mà có thể được kết nối đến internet.

Test setup nginx

server {

listen 80;

server\_name 27.71.227.1;

location /smartfarm {

root /home/ubuntu/smartfarm/FrontEnd/Year3\_dev\_Frontend/Ver2\_use\_template/react-admin/build;

index index.html;

try\_files $uri $uri/ /index.html;

}

location /api {

proxy\_pass http://localhost:8000;

include /etc/nginx/proxy\_params;

proxy\_redirect off;

}

location / {

return 404;

}

}

### USING THIS ONE

NEUPDATE {

>> sudo -u postgres psql //login postgres as postgres

>> CREATE USER year3 WITH PASSWORD 'year3';

>> CREATE DATABASE smartfarm;

>> ALTER DATABASE smartfarm OWNER TO year3;

}

apt-get install nginx

- Remove the default configuration file provided by the NGINX installation so that we can create our own.

rm /etc/nginx/sites-enabled/default

- Có thể chạy lệnh “sudo nginx -t” để kiểm tra xem đang có lỗi gì đang xảy ra với nginxv hay không

- Run *nano /etc/nginx/sites-enabled/project* to create and open a new NGINX configuration file,

+ This command opens the file "/etc/nginx/*sites-enabled/project*  " in the Nano text editor with superuser (sudo) privileges. You can use this command to modify the Nginx configuration for project

* **FOR frontend**

**server {**

**listen 8001;**

**server\_name 27.71.227.1;**

**location / {**

**root /home/ubuntu/smartfarm/FrontEnd2/Year3\_dev\_Frontend/Ver2\_use\_template/react-admin/build;**

**index index.html;**

**try\_files $uri $uri/ /index.html;**

**}**

**}**

* **For backend**

server {

listen 8002; //cổng 8000 đã được dùng nên không được dùng

server\_name 27.71.227.1;

location / {

proxy\_pass http://localhost:8000;

include /etc/nginx/proxy\_params;

proxy\_redirect off;

}

}