0x1A. Application server

DevOpsSysAdmin

- By: Sylvain Kalache, co-founder at Holberton School
- Weight: 1
- Project will start Nov 7, 2022 6:00 AM, must end by Nov 11, 2022 6:00 AM
- will be released at Nov 9, 2022 8:24 PM
- An auto review will be launched at the deadline

Concepts

For this project, we expect you to look at these concepts:

- Web Server
- Server
- Web stack debugging

Background Context

Your web infrastructure is already serving web pages via `Nginx` that you installed in your first web stack project. While a web server can also serve dynamic content, this task is usually given to an application server. In this project you will add this piece to your infrastructure, plug it to your `Nginx` and make is serve your Airbnb clone project.

Resources

Read or watch:

- Application server vs web server
- How to Serve a Flask Application with Gunicorn and Nginx on Ubuntu 16.04 (As mentioned in the video, do **not** install Gunicorn using `virtualenv`, just install everything globally)
- Running Gunicorn
- Be careful with the way Flask manages slash in route - `strict_slashes`
- Upstart documentation

Requirements

General

- A `README.md` file, at the root of the folder of the project, is mandatory
- Everything Python-related must be done using `python3`
- All config files must have comments

Bash Scripts

- All your files will be interpreted on Ubuntu 16.04 LTS
- All your files should end with a new line
- All your Bash script files must be executable
- Your Bash script must pass `Shellcheck` (version `0.3.7-5~ubuntu16.04.1` via `apt-get`) without any error
- The first line of all your Bash scripts should be exactly `#!/usr/bin/env bash`
- The second line of all your Bash scripts should be a comment explaining what is the script doing

Your servers

Name	Username	IP	State	
1609-web-01				Actions Toggle Dropdown
1609-web-02				Actions Toggle Dropdown
1609-lb-01				Actions Toggle Dropdown

Tasks

0. Set up development with Python

mandatory

Let's serve what you built for AirBnB clone v2 - Web framework on `web-01`. This task is an exercise in setting up your development environment, which is used for testing and debugging your code before deploying it to production.

Requirements:

- Make sure that task #3 of your SSH project is completed for `web-01`. The checker will connect to your servers.
- Git clone your `AirBnB_clone_v2` on your `web-01` server.
- Configure the file `web_flask/0-hello_route.py` to serve its content from the route `/airbnb-onepage/` on port `5000`.
- Your Flask application object must be named `app` (This will allow us to run and check your code).

Example:

Window 1:

```
ubuntu@229-web-01:~/AirBnB_clone_v2$ python3 -m web_flask.0-hello_route
 * Serving Flask app "0-hello_route" (lazy loading)
```

```
 * Environment: production
   WARNING: Do not use the development server in a production environment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
35.231.193.217 - - [02/May/2019 22:19:42] "GET /airbnb-onepage/ HTTP/1.1" 200 -
```

Window 2:

```
ubuntu@229-web-01:~/AirBnB_clone_v2$ curl 127.0.0.1:5000/airbnb-onepage/
Hello HBNB!ubuntu@229-web-01:~/AirBnB_clone_v2$
```

Repo:

- GitHub repository: `alx-system_engineering-devops`
- Directory: `0x1A-application_server`
- File: `README.md`

Done? Help Get a sandbox

1. Set up production with Gunicorn

mandatory

Now that you have your development environment set up, let's get your production application server set up with `Gunicorn` on `web-01`, port `5000`. You'll need to install `Gunicorn` and any libraries required by your application. Your `Flask` application object will serve as a WSGI entry point into your application. This will be your production environment. As you can see we want the production and development of your application to use the same port, so the conditions for serving your dynamic content are the same in both environments.

Requirements:

- Install `Gunicorn` and any other libraries required by your application.
- The Flask application object should be called `app`. (This will allow us to run and check your code)
- You will serve the same content from the same route as in the previous task. You can verify that it's working by binding a `Gunicorn` instance to localhost listening on port `5000` with your application object as the entry point.
- In order to check your code, the checker will bind a `Gunicorn` instance to port `6000`, so make sure nothing is listening on that port.

Example:

Terminal 1:

```
ubuntu@229-web-01:~/AirBnB_clone_v2$ gunicorn --bind 0.0.0.0:5000 web_flask.0-hello_r
oute:app
```

```
[2019-05-03 20:47:20 +0000] [3595] [INFO] Starting gunicorn 19.9.0
[2019-05-03 20:47:20 +0000] [3595] [INFO] Listening at: http://0.0.0.0:5000 (3595)
[2019-05-03 20:47:20 +0000] [3595] [INFO] Using worker: sync
[2019-05-03 20:47:20 +0000] [3598] [INFO] Booting worker with pid: 3598
```

Terminal 2:

```
ubuntu@229-web-01:~$ curl 127.0.0.1:5000/airbnb-onepage/
Hello HBNB!ubuntu@229-web-01:~$
```

Repo:

- GitHub repository: `alx-system_engineering-devops`
- Directory: `0x1A-application_server`

Done? Help Get a sandbox

2. Serve a page with Nginx

mandatory

Building on your work in the previous tasks, configure `Nginx` to serve your page from the route `/airbnb-onepage/`

Requirements:

- `Nginx` must serve this page both locally and on its public IP on port `80`.
- `Nginx` should proxy requests to the process listening on port `5000`.
- Include your `Nginx` config file as `2-app_server-nginx_config`.

Notes:

- In order to test this you'll have to spin up either your production or development application server (listening on port `5000`)
- In an actual production environment the application server will be configured to start upon startup in a system initialization script. This will be covered in the advanced tasks.
- You will probably need to `reboot` your server (by using the command `$ sudo reboot`) to have Nginx publicly accessible

Example:

On my server:

Window 1:

```
ubuntu@229-web-01:~/AirBnB_clone_v2$ gunicorn --bind 0.0.0.0:5000 web_flask.0-hello_route:app
[2019-05-06 20:43:57 +0000] [14026] [INFO] Starting gunicorn 19.9.0
```

```
[2019-05-06 20:43:57 +0000] [14026] [INFO] Listening at: http://0.0.0.0:5000 (14026)
[2019-05-06 20:43:57 +0000] [14026] [INFO] Using worker: sync
[2019-05-06 20:43:57 +0000] [14029] [INFO] Booting worker with pid: 14029
```

Window 2:

```
ubuntu@229-web-01:~/AirBnB_clone_v2$ curl 127.0.0.1/airbnb-onepage/
Hello HBNB!ubuntu@229-web-01:~/AirBnB_clone_v2$
```

On my local terminal:

```
vagrant@ubuntu-xenial:~$ curl -sI 35.231.193.217/airbnb-onepage/
HTTP/1.1 200 OK
Server: nginx/1.10.3 (Ubuntu)
Date: Mon, 06 May 2019 20:44:55 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 11
Connection: keep-alive
X-Served-By: 229-web-01

vagrant@ubuntu-xenial:~$ curl 35.231.193.217/airbnb-onepage/
Hello HBNB!vagrant@ubuntu-xenial:~$
```

Repo:

- GitHub repository: `alx-system_engineering-devops`
- Directory: `0x1A-application_server`
- File: `2-app_server-nginx_config`

Done? Help Get a sandbox

3. Add a route with query parameters

mandatory

Building on what you did in the previous tasks, let's expand our web application by adding another service for `Gunicorn` to handle. In `AirBnB_clone_v2/web_flask/6-number_odd_or_even`, the route `/number_odd_or_even/<int:n>` should already be defined to render a page telling you whether an integer is odd or even. You'll need to configure `Nginx` to proxy HTTP requests to the route `/airbnb-dynamic/number_odd_or_even/(any integer)` to a `Gunicorn` instance listening on port `5001`. The key to this exercise is getting `Nginx` configured to proxy requests to processes listening on two different ports. You are not expected to keep your application server processes

running. If you want to know how to run multiple instances of `Gunicorn` without having multiple terminals open, see tips below.

Requirements:

- `Nginx` must serve this page both locally and on its public IP on port `80`.
- `Nginx` should proxy requests to the route `/airbnb-dynamic/number_odd_or_even/(any integer)` the process listening on port `5001`.
- Include your `Nginx` config file as `3-app_server-nginx_config`.

Tips:

- Check out these articles/docs for clues on how to configure `Nginx`: Understanding Nginx Server and Location Block Selection Algorithms, Understanding Nginx Location Blocks Rewrite Rules, Nginx Reverse Proxy.
- In order to spin up a `Gunicorn` instance as a detached process you can use the terminal multiplexer utility `tmux`. Enter the command `tmux new-session -d 'gunicorn --bind 0.0.0.0:5001 web_flask.6-number_odd_or_even:app'` and if successful you should see no output to the screen. You can verify that the process has been created by running `pgrep gunicorn` to see its PID. Once you're ready to end the process you can either run `tmux a` to reattach to the processes, or you can run `kill <PID>` to terminate the background process by ID.

Example:

Terminal 1:

```
ubuntu@229-web-01:~/AirBnB_clone_v2$ tmux new-session -d 'gunicorn --bind 0.0.0.0:500
0 web_flask.0-hello_route:app'

ubuntu@229-web-01:~/AirBnB_clone_v2$ pgrep gunicorn

1661

1665

ubuntu@229-web-01:~/AirBnB_clone_v2$ tmux new-session -d 'gunicorn --bind 0.0.0.0:500
1 web_flask.6-number_odd_or_even:app'

ubuntu@229-web-01:~/AirBnB_clone_v2$ pgrep gunicorn

1661

1665

1684

1688


ubuntu@229-web-01:~/AirBnB_clone_v2$ curl 127.0.0.1:5000/airbnb-onepage/

Hello HBNB!ubuntu@229-web-01:~/AirBnB_clone_v2$
```

```
ubuntu@229-web-01:~/AirBnB_clone_v2$ curl 127.0.0.1:5001/number_odd_or_even/6
<!DOCTYPE html>
<HTML lang="en">
  <HEAD>
    <TITLE>HBNB</TITLE>
  </HEAD>
  <BODY><H1>Number: 6 is even</H1></BODY>
</HTML>ubuntu@229-web-01:~/AirBnB_clone_v2
ubuntu@229-web-01:~$
ubuntu@229-web-01:~/AirBnB_clone_v2$ curl 127.0.0.1/airbnb-dynamic/number_odd_or_even
/5
<!DOCTYPE html>
<HTML lang="en">
  <HEAD>
    <TITLE>HBNB</TITLE>
  </HEAD>
  <BODY><H1>Number: 5 is odd</H1></BODY>
</HTML>ubuntu@229-web-01:~/AirBnB_clone_v2$
```

Local machine:

```
vagrant@ubuntu-xenial:~$ curl 35.231.193.217/airbnb-dynamic/number_odd_or_even/6<!DOC
TYPE html>
<HTML lang="en">
  <HEAD>
    <TITLE>HBNB</TITLE>
  </HEAD>
  <BODY><H1>Number: 6 is even</H1></BODY>
</HTML>vagrant@ubuntu-xenial:~$
```

Repo:

- GitHub repository: `alx-system_engineering-devops`
- Directory: `0x1A-application_server`
- File: `3-app_server-nginx_config`

Done? Help Get a sandbox

4. Let's do this for your API

mandatory

Let’s serve what you built for AirBnB clone v3 - RESTful API on `web-01`.

Requirements:

- Git clone your `AirBnB_clone_v3`
- Setup `Nginx` so that the route `/api/` points to a `Gunicorn` instance listening on port `5002`
- `Nginx` must serve this page both locally and on its public IP on port `80`
- To test your setup you should bind `Gunicorn` to `api/v1/app.py`
- It may be helpful to import your data (and environment variables) from this project
- Upload your `Nginx` config file as `4-app_server-nginx_config`

Example:

Terminal 1:

```
ubuntu@229-web-01:~/AirBnB_clone_v3$ tmux new-session -d 'gunicorn --bind 0.0.0.0:500
2 api.v1.app:app'

ubuntu@229-web-01:~/AirBnB_clone_v3$ curl 127.0.0.1:5002/api/v1/states

[{"__class__":"State","created_at":"2019-05-10T00:39:27.032802","id":"7512f664-4951-4
231-8de9-b18d940cc912","name":"California","updated_at":"2019-05-10T00:39:27.032965"}
,{"__class__":"State","created_at":"2019-05-10T00:39:36.021219","id":"b25625c8-8a7a-4
c1f-8afc-257bf9f76bc8","name":"Arizona","updated_at":"2019-05-10T00:39:36.021281"}]

ubuntu@229-web-01:~/AirBnB_clone_v3$

ubuntu@229-web-01:~/AirBnB_clone_v3$ curl 127.0.0.1/api/v1/states

[{"__class__":"State","created_at":"2019-05-10T00:39:27.032802","id":"7512f664-4951-4
231-8de9-b18d940cc912","name":"California","updated_at":"2019-05-10T00:39:27.032965"}
,{"__class__":"State","created_at":"2019-05-10T00:39:36.021219","id":"b25625c8-8a7a-4
c1f-8afc-257bf9f76bc8","name":"Arizona","updated_at":"2019-05-10T00:39:36.021281"}]

ubuntu@229-web-01:~/AirBnB_clone_v3$
```

Local Terminal:

```
vagrant@ubuntu-xenial:~$ curl 35.231.193.217/api/v1/states

[{"__class__":"State","created_at":"2019-05-10T00:39:27.032802","id":"7512f664-4951-4
231-8de9-b18d940cc912","name":"California","updated_at":"2019-05-10T00:39:27.032965"}
,{"__class__":"State","created_at":"2019-05-10T00:39:36.021219","id":"b25625c8-8a7a-4
c1f-8afc-257bf9f76bc8","name":"Arizona","updated_at":"2019-05-10T00:39:36.021281"}]

vagrant@ubuntu-xenial:~$
```

Repo:

- GitHub repository: `alx-system_engineering-devops`
- Directory: `0x1A-application_server`
- File: `4-app_server-nginx_config`

Done? Help Get a sandbox

5. Serve your AirBnB clone

mandatory

Let’s serve what you built for AirBnB clone - Web dynamic on `web-01`.

Requirements:

- Git clone your `AirBnB_clone_v4`
- Your `Gunicorn` instance should serve content from `web_dynamic/2-hbnb.py` on port `5003`
- Setup `Nginx` so that the route `/` points to your `Gunicorn` instance
- Setup `Nginx` so that it properly serves the static assets found in `web_dynamic/static/` (this is essential for your page to render properly)
- For your website to be fully functional, you will need to reconfigure `web_dynamic/static/scripts/2-hbnb.js` to the correct IP
- `Nginx` must serve this page both locally and on its public IP and port `5003`
- Make sure to pull up your Developer Tools on your favorite browser to verify that you have no errors
- Upload your `Nginx` config as `5-app_server-nginx_config`

After loading, your website should look like this:

Repo:

- GitHub repository: `alx-system_engineering-devops`
- Directory: `0x1A-application_server`
- File: `5-app_server-nginx_config`