# ICT & Infra S3 Automation & Orchestration, week 3

| | |
|---|---|
| Class: | *CB01* |
| Student number: | 4961854 |
| Student name: | *Heiko Morales* |

## *Introduction*

This week you will practice creating Ansible Playbook(s) to automate a complex process. Before executing the assignment, ensure that you have working Ansible control node. For this assignment, you must be familiar with preparing Apache server and hosting Flask application on it. To repeat the necessary steps, you can read this simple tutorial, which explains Apache installation steps; creation of simple Flask application; hosting a Flask app on Apache:
*https://www.codementor.io/@abhishake/minimal-apache-configuration-for-deploying-a-flask-app-ubuntu-18-04-phu50a7ft*

Similar (probably the same) steps you will need to automate in this assignment.

## *Assignment 1. Create a Playbook to automate hosting of a simple Flask application*

Difficulty: ★★★★☆.

You already know how to create a simple Ansible Playbook. Now it is time to create a Playbook that can prepare newly installed Linux machine to host a website. Additionally, you should be able to host Flask application automatically. For this assignment, you must execute the following steps:

1. Using Amazon EC2 service, manually create and configure (free tier) Ubuntu instance to host a website.
2. Test SSH connection to the Virtual Machine.
3. Read the steps (from codementor.io tutorial) of Apache configuration and simple Flask app hosting.
4. Create a simple Flask app (my_flask_app.py) on your local machine.
5. Create a Playbook to **automate**:
    a. apache2 service installation on the VM;
    b. mod_wsgi module installation;
    c. Flask installation;
    d. Generate and upload “my_flask_app.wsgi” file using Jinja2 templating. HOW TO: Template a file out to a remote server
    e. Generate and upload “ExampleFlask.conf” file using Jinja2 templating.
    f. Check if newly created/uploaded website is accessible. HOW TO: Interacts with webservices
6. Manually test if the website is working.

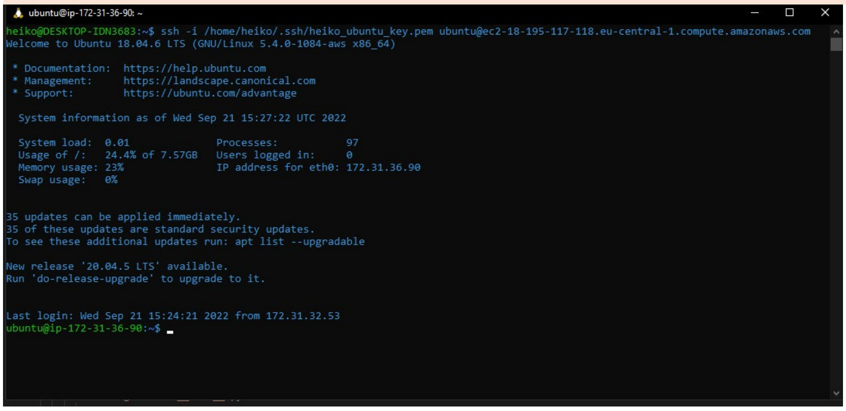
Provide screenshots (evidence) for your solution. Always explain your evidence! As a prof, we expect at least:

- A screenshot of running EC2 Ubuntu instance;
- A screenshot of successful SSH connection to the VM;
- Ansible Playbook file(s) that automates configuration of the VM.
- A prof that the website is working.

*Solution:* The first thing to do is to create the instance in aws. In our case we will use a linux OS, specifically ubuntu .

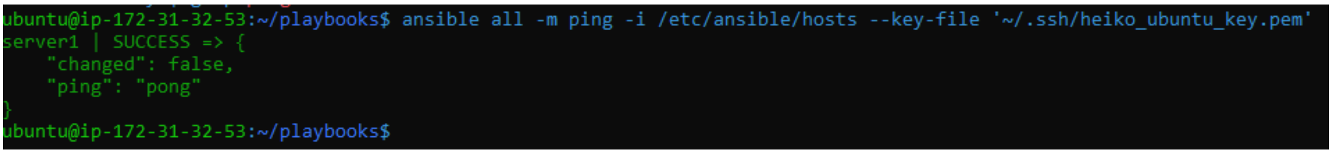
| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ☐ | heiko_ubuntu | i-007c6e38123138c46 | ✓ Running | ⊕⊖ | t… | ✓ 2/2 checks passed | + | eu-central-1b | e… | 18.195.117.118 | – | – | d… | l… | heiko_ubuntu… | 2022/09/21 |

When creating this service, a .pem key was created and thanks to it we are able to connect remotely to the aws instance.



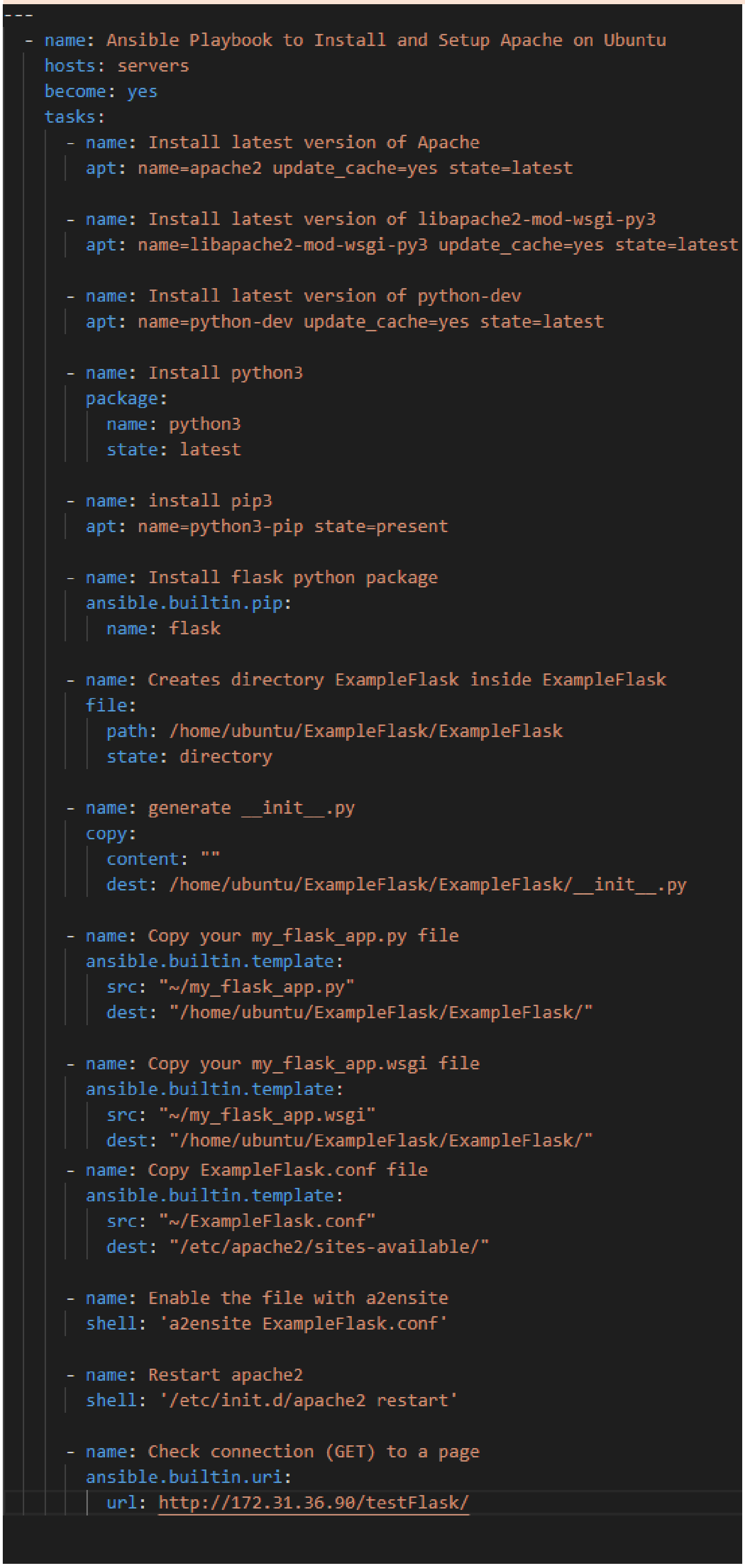
```
heiko@DESKTOP-IDN3683:~$ ssh -i /home/heiko/.ssh/heiko_ubuntu_key.pem ubuntu@ec2-18-195-117-118.eu-central-1.compute.amazonaws.com
Welcome to Ubuntu 18.04.6 LTS (GNU/Linux 5.4.0-1084-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Wed Sep 21 15:27:22 UTC 2022

  System load:  0.01               Processes:           97
  Usage of /:   24.4% of 7.57GB   Users logged in:     0
  Memory usage: 23%                IP address for eth0: 172.31.36.90
  Swap usage:   0%


35 updates can be applied immediately.
35 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

New release '20.04.5 LTS' available.
Run 'do-release-upgrade' to upgrade to it.


Last login: Wed Sep 21 15:24:21 2022 from 172.31.32.53
ubuntu@ip-172-31-36-90:~$ 
```

Later in another instance we installed ansible and configured all the files (mentioned in the practice "annotated-Infra-CB-S3-AO-W2-HW_v2") to the point of being able to execute commands or playbooks in the server that will host apache2.



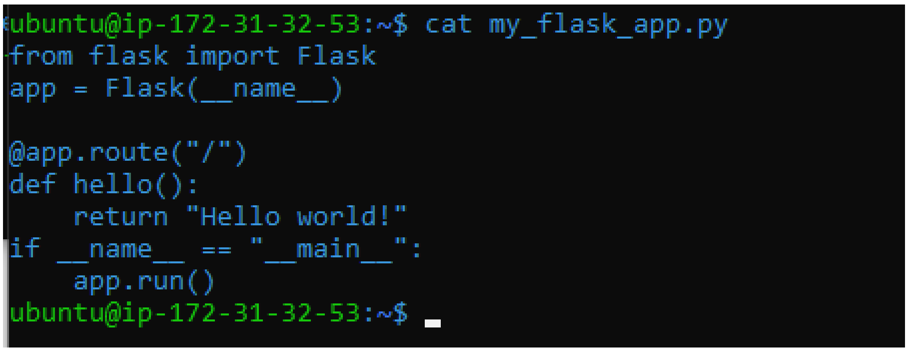
```
ubuntu@ip-172-31-32-53:~/playbooks$ ansible all -m ping -i /etc/ansible/hosts --key-file '~/.ssh/heiko_ubuntu_key.pem'
server1 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
ubuntu@ip-172-31-32-53:~/playbooks$
```

Once the installation is ready, we start to create the playbook in the following way:



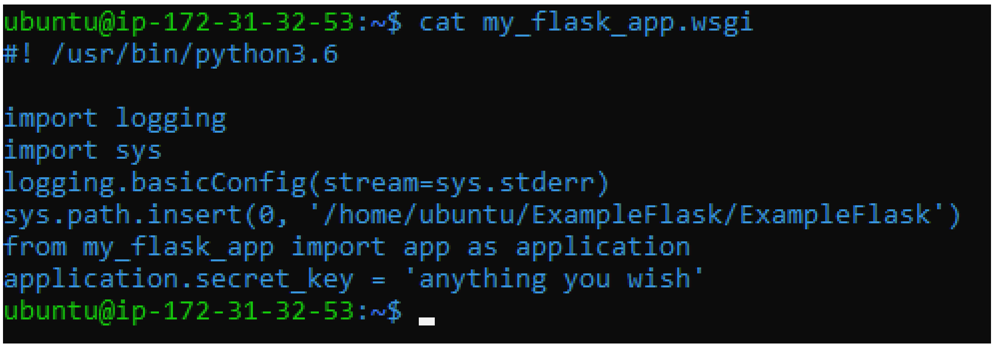
```yaml
---
  - name: Ansible Playbook to Install and Setup Apache on Ubuntu
    hosts: servers
    become: yes
    tasks:
      - name: Install latest version of Apache
        apt: name=apache2 update_cache=yes state=latest

      - name: Install latest version of libapache2-mod-wsgi-py3
        apt: name=libapache2-mod-wsgi-py3 update_cache=yes state=latest

      - name: Install latest version of python-dev
        apt: name=python-dev update_cache=yes state=latest

      - name: Install python3
        package:
          name: python3
          state: latest

      - name: install pip3
        apt: name=python3-pip state=present

      - name: Install flask python package
        ansible.builtin.pip:
          name: flask

      - name: Creates directory ExampleFlask inside ExampleFlask
        file:
          path: /home/ubuntu/ExampleFlask/ExampleFlask
          state: directory

      - name: generate __init__.py
        copy:
          content: ""
          dest: /home/ubuntu/ExampleFlask/ExampleFlask/__init__.py

      - name: Copy your my_flask_app.py file
        ansible.builtin.template:
          src: "~/my_flask_app.py"
          dest: "/home/ubuntu/ExampleFlask/ExampleFlask/"

      - name: Copy your my_flask_app.wsgi file
        ansible.builtin.template:
          src: "~/my_flask_app.wsgi"
          dest: "/home/ubuntu/ExampleFlask/ExampleFlask/"
      - name: Copy ExampleFlask.conf file
        ansible.builtin.template:
          src: "~/ExampleFlask.conf"
          dest: "/etc/apache2/sites-available/"

      - name: Enable the file with a2ensite
        shell: 'a2ensite ExampleFlask.conf'

      - name: Restart apache2
        shell: '/etc/init.d/apache2 restart'

      - name: Check connection (GET) to a page
        ansible.builtin.uri:
          url: http://172.31.36.90/testFlask/
```

Once the playbook has been created, we will configure the following files in the home/user directory my_flask_app.py



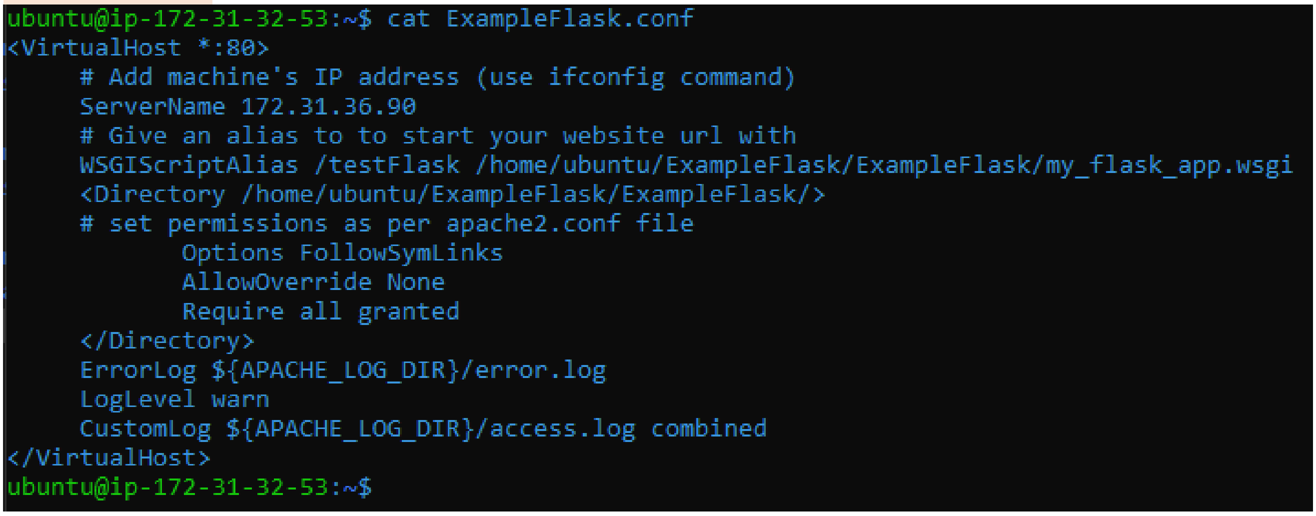
```
ubuntu@ip-172-31-32-53:~$ cat my_flask_app.py
from flask import Flask
app = Flask(__name__)

@app.route("/")
def hello():
    return "Hello world!"
if __name__ == "__main__":
    app.run()
ubuntu@ip-172-31-32-53:~$ 
```

my_flask_app.wsgi



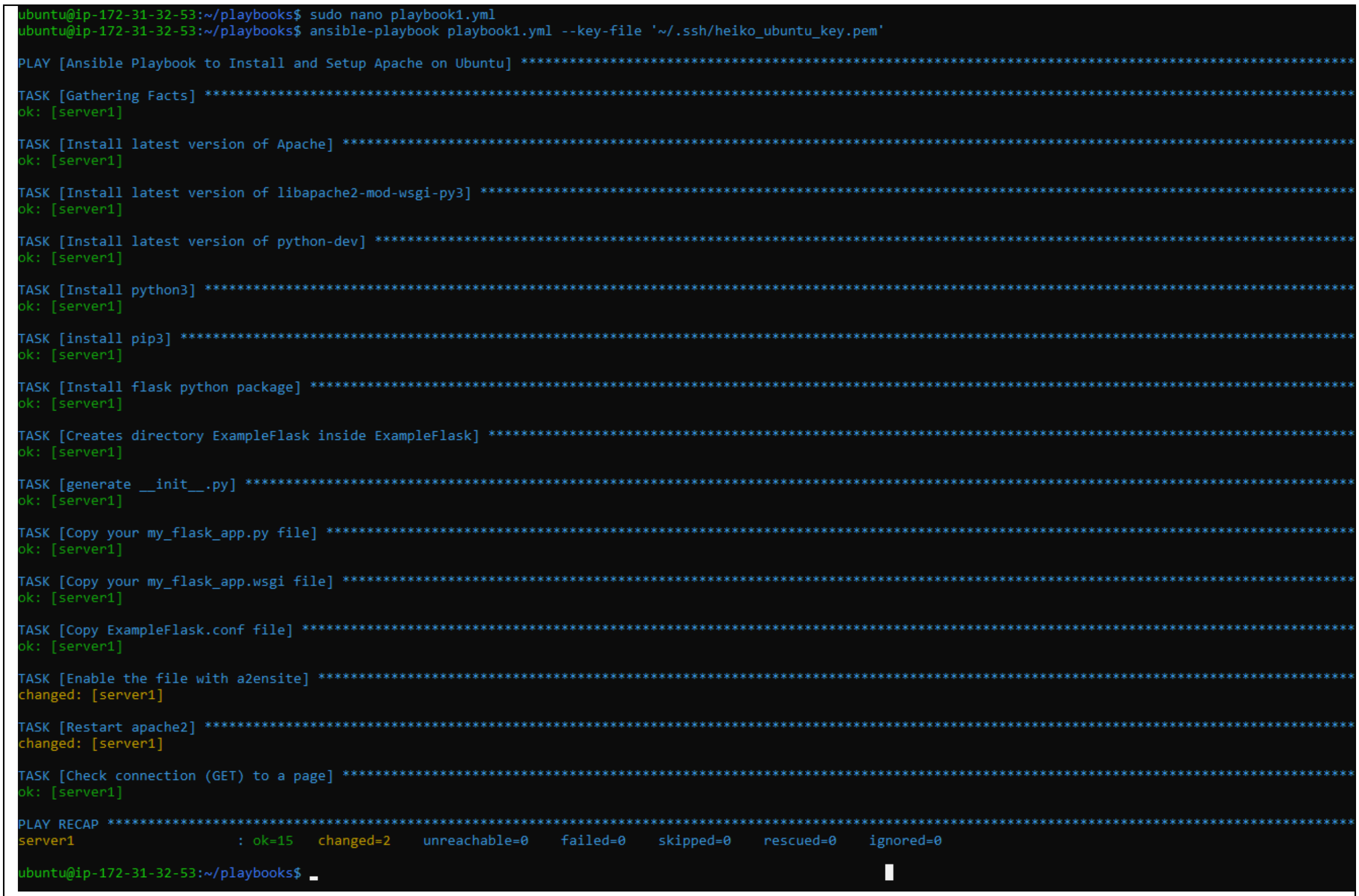
```
ubuntu@ip-172-31-32-53:~$ cat my_flask_app.wsgi
#! /usr/bin/python3.6

import logging
import sys
logging.basicConfig(stream=sys.stderr)
sys.path.insert(0, '/home/ubuntu/ExampleFlask/ExampleFlask')
from my_flask_app import app as application
application.secret_key = 'anything you wish'
ubuntu@ip-172-31-32-53:~$ 
```

ExampleFlask.conf



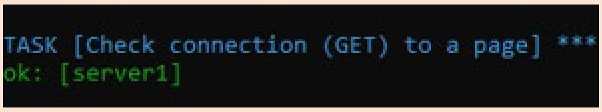
```
ubuntu@ip-172-31-32-53:~$ cat ExampleFlask.conf
<VirtualHost *:80>
     # Add machine's IP address (use ifconfig command)
     ServerName 172.31.36.90
     # Give an alias to to start your website url with
     WSGIScriptAlias /testFlask /home/ubuntu/ExampleFlask/ExampleFlask/my_flask_app.wsgi
     <Directory /home/ubuntu/ExampleFlask/ExampleFlask/>
     # set permissions as per apache2.conf file
            Options FollowSymLinks
            AllowOverride None
            Require all granted
     </Directory>
     ErrorLog ${APACHE_LOG_DIR}/error.log
     LogLevel warn
     CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
ubuntu@ip-172-31-32-53:~$
```

Once the configuration files have been created, run the playbook as follows:



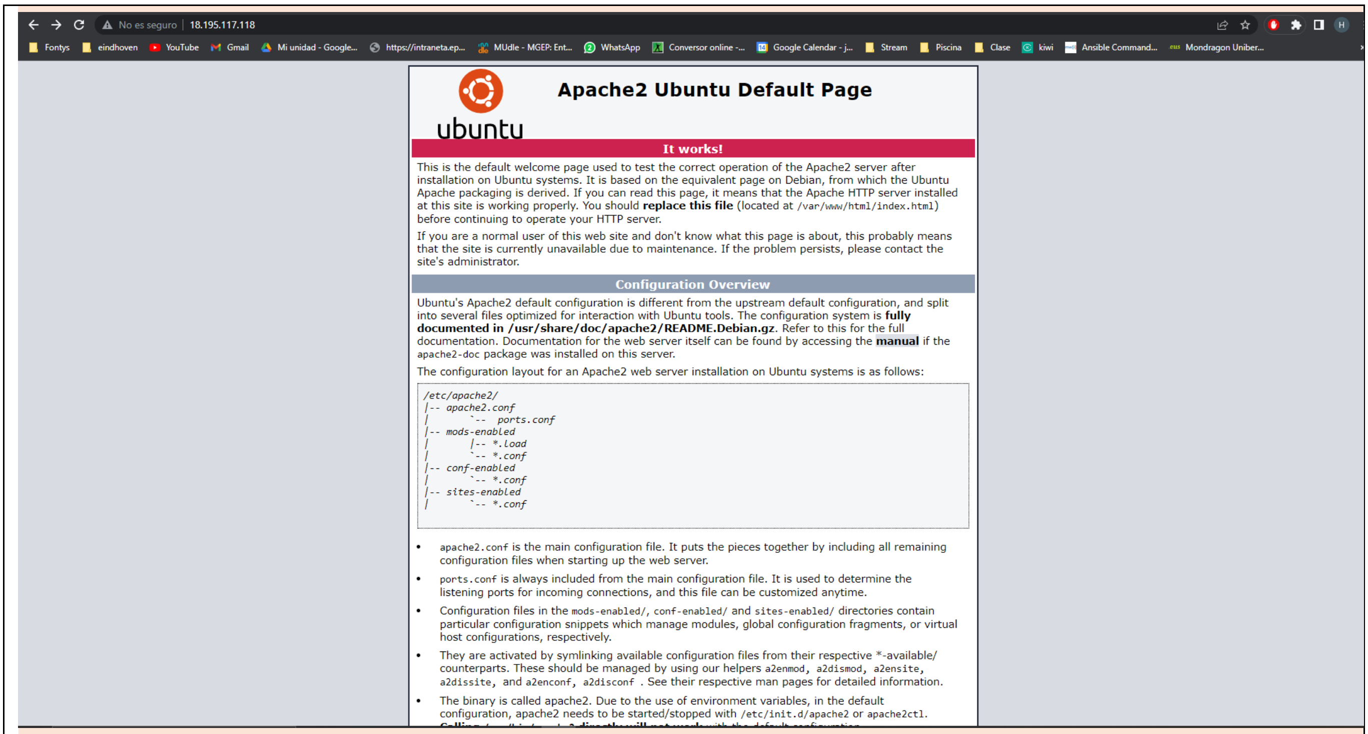
```
ubuntu@ip-172-31-32-53:~/playbooks$ sudo nano playbook1.yml
ubuntu@ip-172-31-32-53:~/playbooks$ ansible-playbook playbook1.yml --key-file '~/.ssh/heiko_ubuntu_key.pem'

PLAY [Ansible Playbook to Install and Setup Apache on Ubuntu] ******************************************************************************************************

TASK [Gathering Facts] *********************************************************************************************************************************************
ok: [server1]

TASK [Install latest version of Apache] ****************************************************************************************************************************
ok: [server1]

TASK [Install latest version of libapache2-mod-wsgi-py3] ***********************************************************************************************************
ok: [server1]

TASK [Install latest version of python-dev] ************************************************************************************************************************
ok: [server1]

TASK [Install python3] *********************************************************************************************************************************************
ok: [server1]

TASK [install pip3] ************************************************************************************************************************************************
ok: [server1]

TASK [Install flask python package] ********************************************************************************************************************************
ok: [server1]

TASK [Creates directory ExampleFlask inside ExampleFlask] **********************************************************************************************************
ok: [server1]

TASK [generate __init__.py] ****************************************************************************************************************************************
ok: [server1]

TASK [Copy your my_flask_app.py file] ******************************************************************************************************************************
ok: [server1]

TASK [Copy your my_flask_app.wsgi file] ****************************************************************************************************************************
ok: [server1]

TASK [Copy ExampleFlask.conf file] *********************************************************************************************************************************
ok: [server1]

TASK [Enable the file with a2ensite] *******************************************************************************************************************************
changed: [server1]

TASK [Restart apache2] *********************************************************************************************************************************************
changed: [server1]

TASK [Check connection (GET) to a page] ****************************************************************************************************************************
ok: [server1]

PLAY RECAP *********************************************************************************************************************************************************
server1                    : ok=15   changed=2    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

ubuntu@ip-172-31-32-53:~/playbooks$ _
```

As we can see in the image the playbook is able to install and configure everything needed to run apache correctly and securely, as you can see in the last image the page returns a status of 200 (ok).



```
TASK [Check connection (GET) to a page] ***
ok: [server1]
```

On the other hand, if we connect via the browser, we will see that the Apache is working correctly even if it does not have any web page.



**Apache2 Ubuntu Default Page**

**It works!**

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Ubuntu systems. It is based on the equivalent page on Debian, from which the Ubuntu Apache packaging is derived. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should **replace this file** (located at `/var/www/html/index.html`) before continuing to operate your HTTP server.

If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

**Configuration Overview**

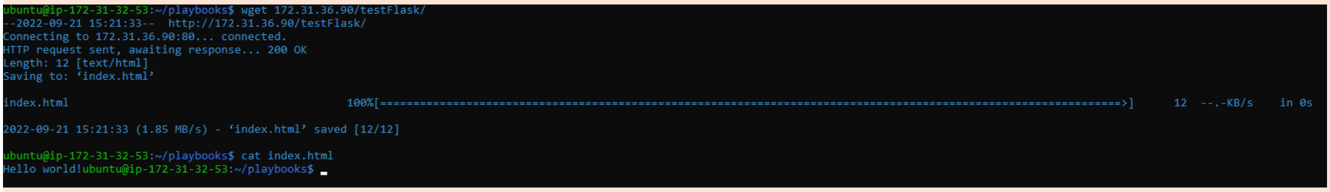
Ubuntu's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Ubuntu tools. The configuration system is **fully documented in /usr/share/doc/apache2/README.Debian.gz**. Refer to this for the full documentation. Documentation for the web server itself can be found by accessing the **manual** if the `apache2-doc` package was installed on this server.

The configuration layout for an Apache2 web server installation on Ubuntu systems is as follows:

```
/etc/apache2/
|-- apache2.conf
|       `--  ports.conf
|-- mods-enabled
|       |-- *.load
|       `-- *.conf
|-- conf-enabled
|       `-- *.conf
|-- sites-enabled
|       `-- *.conf
```

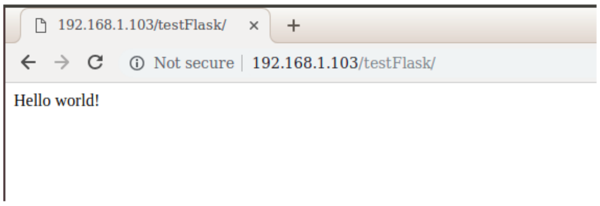
- `apache2.conf` is the main configuration file. It puts the pieces together by including all remaining configuration files when starting up the web server.
- `ports.conf` is always included from the main configuration file. It is used to determine the listening ports for incoming connections, and this file can be customized anytime.
- Configuration files in the `mods-enabled/`, `conf-enabled/` and `sites-enabled/` directories contain particular configuration snippets which manage modules, global configuration fragments, or virtual host configurations, respectively.
- They are activated by symlinking available configuration files from their respective *-available/ counterparts. These should be managed by using our helpers `a2enmod`, `a2dismod`, `a2ensite`, `a2dissite`, and `a2enconf`, `a2disconf` . See their respective man pages for detailed information.
- The binary is called apache2. Due to the use of environment variables, in the default configuration, apache2 needs to be started/stopped with `/etc/init.d/apache2` or `apache2ctl`.

Finally, in the case of using a wget from another instance to the machine, we can observe that it downloads the page correctly if we use the following URL:



```
ubuntu@ip-172-31-32-53:~/playbooks$ wget 172.31.36.90/testFlask/
--2022-09-21 15:21:33--  http://172.31.36.90/testFlask/
Connecting to 172.31.36.90:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 12 [text/html]
Saving to: ‘index.html’

index.html                           100%[==================================================================================================================>]      12  --.-KB/s    in 0s

2022-09-21 15:21:33 (1.85 MB/s) - ‘index.html’ saved [12/12]

ubuntu@ip-172-31-32-53:~/playbooks$ cat index.html
Hello world!ubuntu@ip-172-31-32-53:~/playbooks$ _
```

We see how the page returns a "hello world!" as in the example provided. ([https://www.codementor.io/@abhishake/minimal-apache-configuration-for-deploying-a-flask-app-ubuntu-18-04-phu50a7ft](https://www.codementor.io/@abhishake/minimal-apache-configuration-for-deploying-a-flask-app-ubuntu-18-04-phu50a7ft))



Hello world!