Web Application Deployment Using Ansible

Objective

This project demonstrates how to automate the deployment of a web application using Ansible on a remote Nginx server. The automation process includes installing Nginx, deploying the application, and configuring the server to host the application.

Problem Statement

Real-time scenario: You have joined XYZ Pvt. Ltd. as a DevOps engineer. The company provides a platform where individuals can create their profiles and start blogging on various topics. The application is ready to be hosted on a server. Your task is to implement an Ansible script to deploy this application on a remote Nginx server.

Industry Relevance

The tools and technologies used in this project are essential in the DevOps domain:

Ansible: Ansible is an open-source automation tool that simplifies IT tasks such as configuration management, application deployment, and orchestration. By using human-readable YAML files called playbooks, it enables easy and efficient management of complex IT environments.

Tools Used in the Project

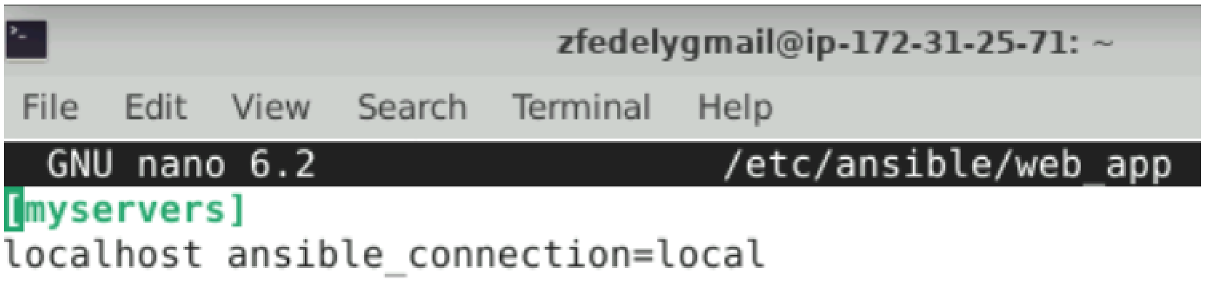
- **Ansible**: Automates configuration management and deployment tasks.
- **Nginx**: A high-performance web server used for hosting and serving the web application.

Steps to Deploy the Web Application Using Ansible

To deploy the web application using Ansible, we will follow the steps below:

1. **Create an Inventory File**

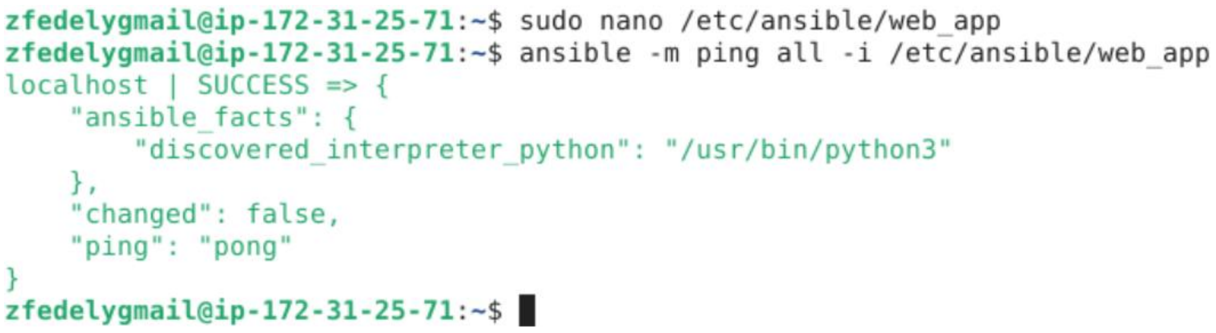
   We will create a simple Ansible inventory file named web_app to define our remote server. This process will allow us to manage and automate tasks on the server using Ansible.



zfedelygmail@ip-172-31-25-71: ~
File Edit View Search Terminal Help

```
  GNU nano 6.2                     /etc/ansible/web_app
[myservers]
localhost ansible_connection=local
```

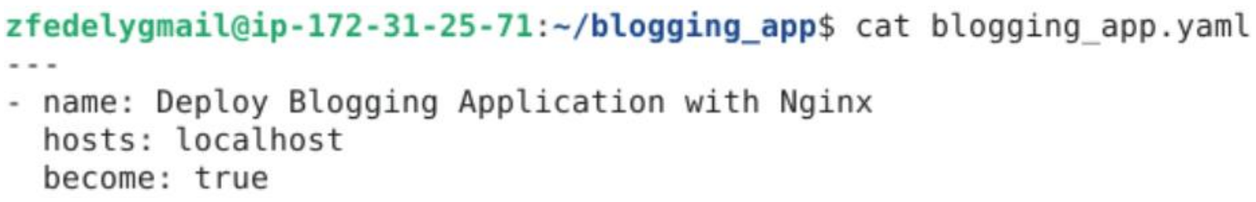
Run the Ansible ping command to test the connection



```
zfedelygmail@ip-172-31-25-71:~$ sudo nano /etc/ansible/web_app
zfedelygmail@ip-172-31-25-71:~$ ansible -m ping all -i /etc/ansible/web_app
localhost | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
zfedelygmail@ip-172-31-25-71:~$ 
```

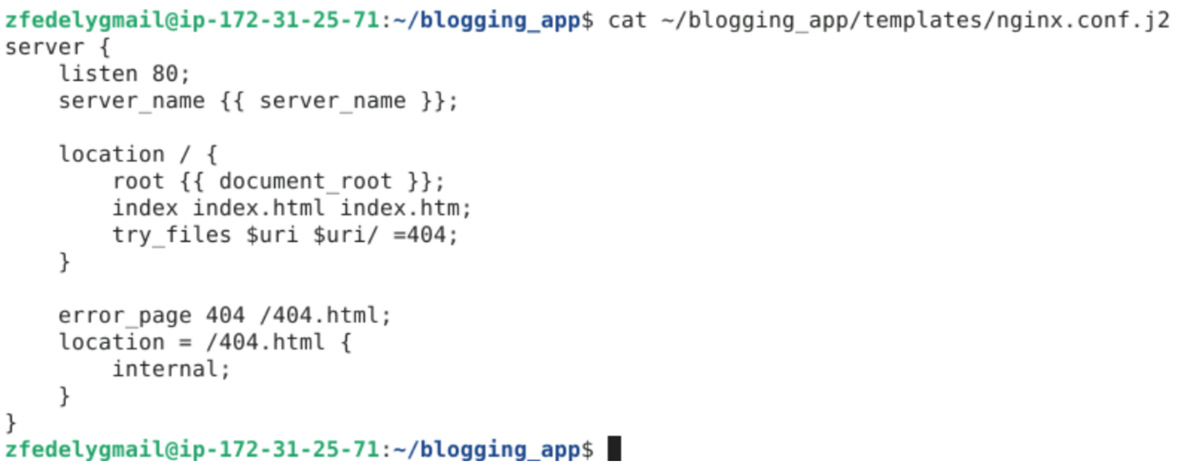
As you can see, we successfully created the Ansible inventory file “web_app” to define our server. The successful output of the ping command above confirms that Ansible can communicate with the server, indicating that the setup is correctly configured. This allows us to effectively manage and automate tasks on the defined server using Ansible.

2. **Write a YAML Playbook for Deployment**



```
zfedelygmail@ip-172-31-25-71:~/blogging_app$ cat blogging_app.yaml
---
- name: Deploy Blogging Application with Nginx
  hosts: localhost
  become: true
```

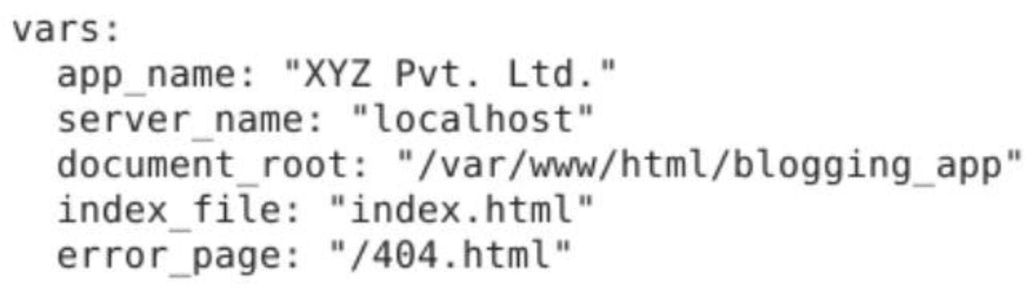
3. **Create a directory for templates and a Jinja2 template for the Nginx configuration**



```
zfedelygmail@ip-172-31-25-71:~/blogging_app$ cat ~/blogging_app/templates/nginx.conf.j2
server {
    listen 80;
    server_name {{ server_name }};

    location / {
        root {{ document_root }};
        index index.html index.htm;
        try_files $uri $uri/ =404;
    }

    error_page 404 /404.html;
    location = /404.html {
        internal;
    }
}
zfedelygmail@ip-172-31-25-71:~/blogging_app$
```

This Nginx configuration block sets up a server listening on port 80 for requests directed to a dynamic server name defined by the variable {{ server_name }}. It specifies the root directory for the web application as {{ document_root }}, and serves index.html or index.htm as the default page. If the requested resource is not found, it attempts to return a 404 error and directs users to a custom 404 error page located at /404.html, which can only be accessed internally. This configuration ensures users receive the correct pages and a friendly error message when resources are missing.

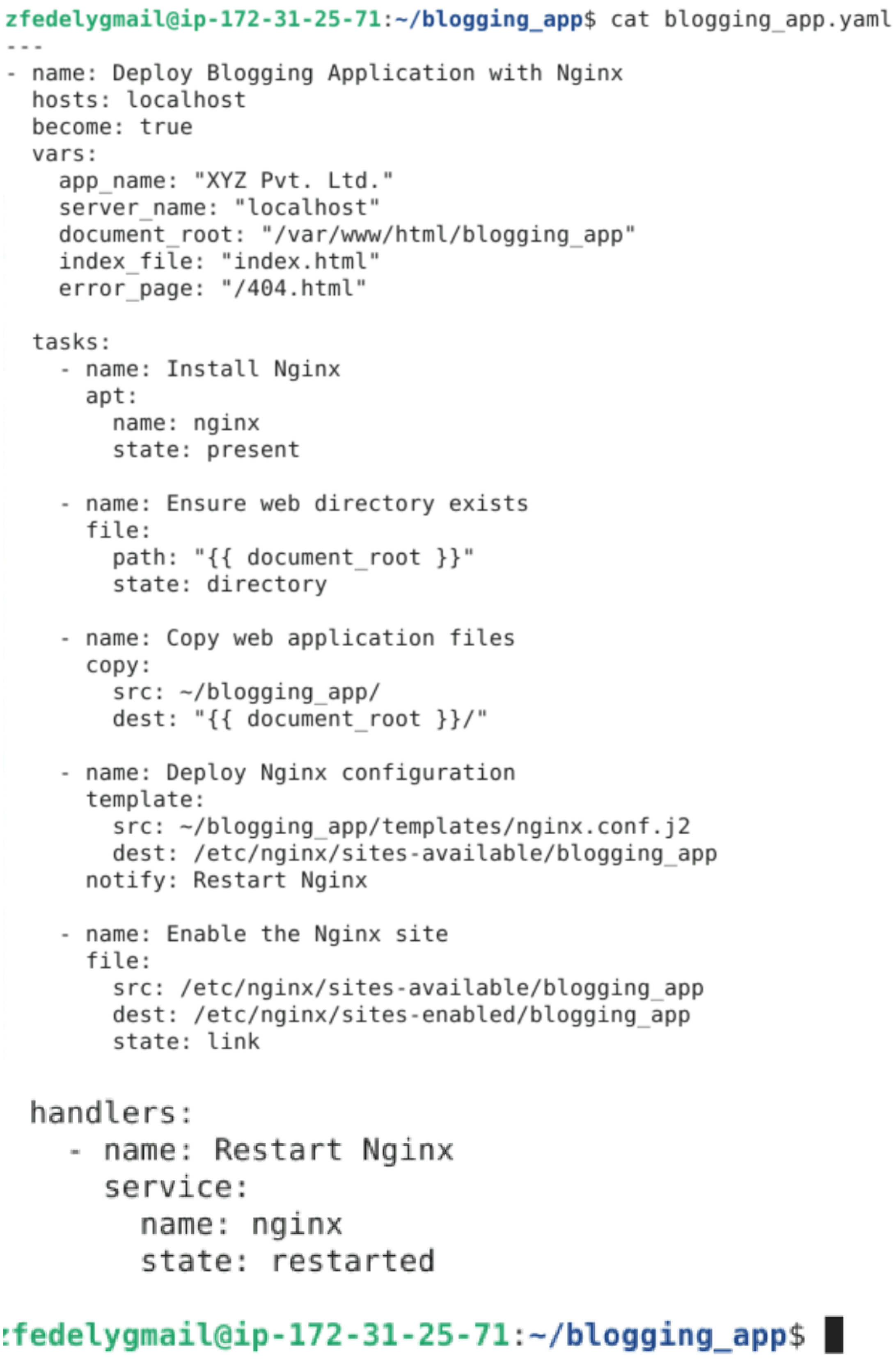
4. **Define variables in the playbook for application details and Nginx configuration**



```
vars:
  app_name: "XYZ Pvt. Ltd."
  server_name: "localhost"
  document_root: "/var/www/html/blogging_app"
  index_file: "index.html"
  error_page: "/404.html"
```

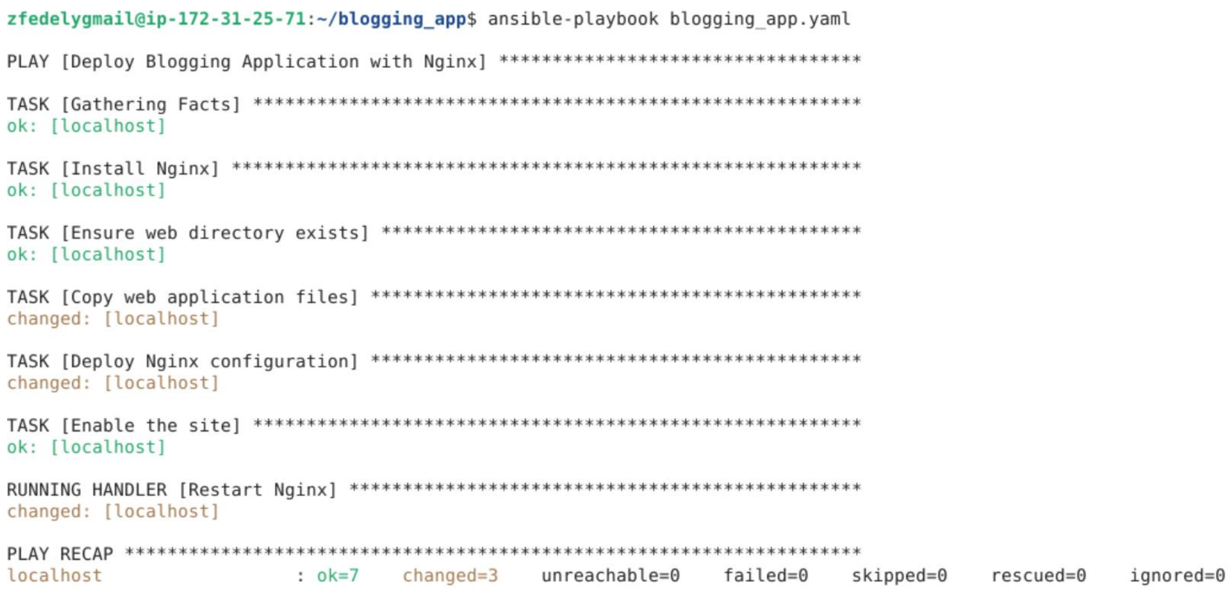
The **app_name** variable stores the application's name, "XYZ Pvt. Ltd.," which can be referenced throughout the playbook for consistency. The **server_name** variable specifies that the Nginx configuration will handle requests for "localhost." The **document_root** variable holds the path to the application's document root, allowing Nginx to locate the necessary files. The **index_file** variable defines the default file served by Nginx, guiding users to the application's entry point. Finally, the **error_page** variable specifies the path to a custom error page displayed when a resource cannot be found, enhancing user experience. These variables collectively simplify the Nginx configuration process for the application.

5. **Include tasks in the playbook for installing Nginx, copying application files, deploying Nginx configuration, and enabling the Nginx site**



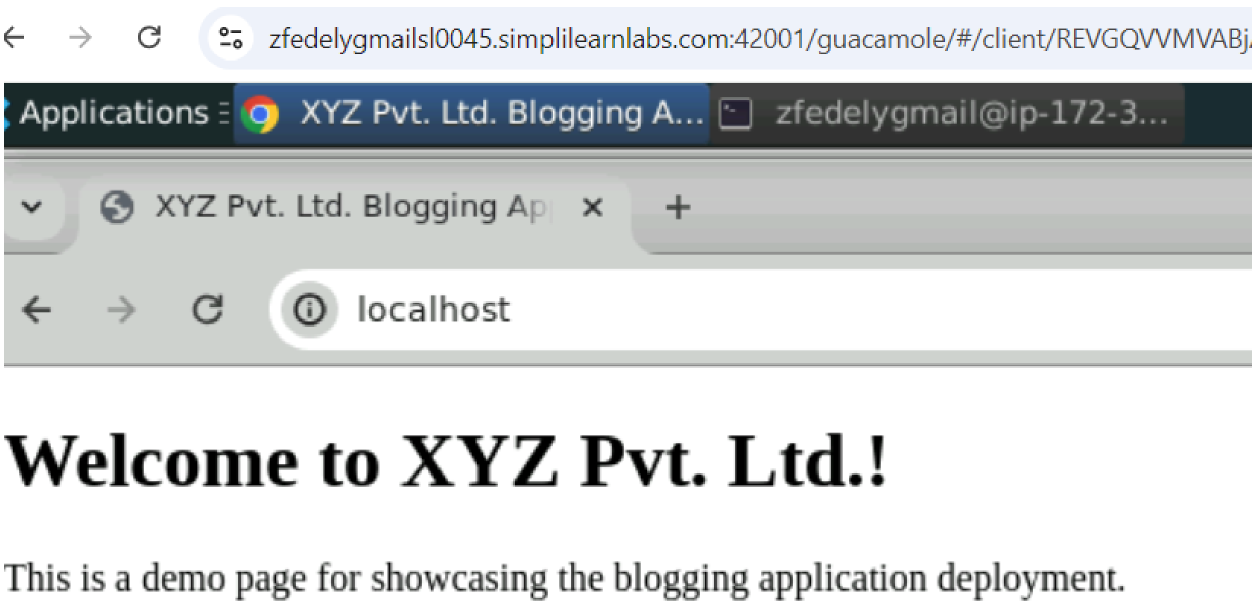
```
zfedelygmail@ip-172-31-25-71:~/blogging_app$ cat blogging_app.yaml
---
- name: Deploy Blogging Application with Nginx
  hosts: localhost
  become: true
  vars:
    app_name: "XYZ Pvt. Ltd."
    server_name: "localhost"
    document_root: "/var/www/html/blogging_app"
    index_file: "index.html"
    error_page: "/404.html"

  tasks:
    - name: Install Nginx
      apt:
        name: nginx
        state: present

    - name: Ensure web directory exists
      file:
        path: "{{ document_root }}"
        state: directory

    - name: Copy web application files
      copy:
        src: ~/blogging_app/
        dest: "{{ document_root }}/"

    - name: Deploy Nginx configuration
      template:
        src: ~/blogging_app/templates/nginx.conf.j2
        dest: /etc/nginx/sites-available/blogging_app
      notify: Restart Nginx

    - name: Enable the Nginx site
      file:
        src: /etc/nginx/sites-available/blogging_app
        dest: /etc/nginx/sites-enabled/blogging_app
        state: link

  handlers:
    - name: Restart Nginx
      service:
        name: nginx
        state: restarted

:fedelygmail@ip-172-31-25-71:~/blogging_app$ █
```

6. Execute the playbook to deploy the web application on the remote server



```
zfedelygmail@ip-172-31-25-71:~/blogging_app$ ansible-playbook blogging_app.yaml

PLAY [Deploy Blogging Application with Nginx] **********************************

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

TASK [Install Nginx] ***********************************************************
ok: [localhost]

TASK [Ensure web directory exists] *********************************************
ok: [localhost]

TASK [Copy web application files] **********************************************
changed: [localhost]

TASK [Deploy Nginx configuration] **********************************************
changed: [localhost]

TASK [Enable the site] *********************************************************
ok: [localhost]

RUNNING HANDLER [Restart Nginx] ************************************************
changed: [localhost]

PLAY RECAP *********************************************************************
localhost                  : ok=7    changed=3    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

The playbook was executed successfully, achieving all the intended tasks without any errors. Nginx was installed, the web directory was ensured to exist, the web application files were copied over, and the Nginx configuration was deployed and enabled. The "changed" status for copying the web files indicates that new content was successfully transferred to the server. With all tasks marked as "ok" and no failures, the web application is now properly deployed and configured on the Nginx server, making it ready to serve web requests.


zfedelygmailsl0045.simplilearnlabs.com:42001/guacamole/#/client/REVGQVVMVABj
Applications
XYZ Pvt. Ltd. Blogging A...
zfedelygmail@ip-172-3...
XYZ Pvt. Ltd. Blogging Ap
localhost

Welcome to XYZ Pvt. Ltd.!

This is a demo page for showcasing the blogging application deployment.

Conclusion

By following the osteps above, we successfully set up a complete deployment process for the blogging application using Ansible. We defined an inventory file for the remote server, created a YAML playbook for installing Nginx and configuring the application, developed a Jinja2 template for the Nginx configuration, and executed the playbook to deploy the application. environments.