Here is a detailed solution for your DevOps assessment using Ansible. The playbook and tasks will be split according to the provided requirements. The primary focus is on configuring CA certificates, deploying the Python Flask application, and ensuring security.

Directory Structure

Ansible project will have the following structure:

```
ansible_project/
|
├── group_vars/
|   └── all.yml         # Store variables
├── roles/
|   └── deploy_app/
|       ├── tasks/
|       |   └── main.yml  # Main playbook tasks
|       ├── templates/
|       |   ├── run.sh.j2 # Template for run.sh
|       |   ├── config.py.j2 # Template for config.py
|       |   └── example.service.j2 # Template for systemd service file
|       └── files/
|           ├── CA1.crt
|           ├── CA2.crt
|           ├── CA3.crt
|           └── example-1.1.2-py3-none-any.whl # Application wheel file
├── playbook.yml        # Main playbook
└── README.md           # Documentation
```

Step-by-Step Explanation of the Solution

1. Ansible Variables Configuration (group_vars/all.yml)

Store the variables for your playbook in group_vars/all.yml to make the solution dynamic and flexible.

```
# group_vars/all.yml

---

port: 5000

deployment_folder: /opt/example

wheel_file: example-1.1.2-py3-none-any.whl

instance_path: /opt/example/instance

secret_key: AVerySafeExampleKey123!

db_path:
postgresql://example_user:myVerySecretDatabasePassword@localhost:5432/example_db

admin_groups:

  - Read-only

  - Read-write

  - Super

ca_certificates:

  - CA1.crt

  - CA2.crt

  - CA3.crt

…
```

2. Playbook (playbook.yml)

This playbook will include the deploy_app role to perform the tasks outlined in the assessment.

```
# playbook.yml

---

- name: DevOps Assessment Playbook

  hosts: all
```

```
  become: true
  roles:
    - deploy_app
…
```

3. Tasks in the Role (roles/deploy_app/tasks/main.yml)

The main.yml file will contain the necessary tasks to install the CA certificates, deploy the application, configure the environment, and ensure security.

```
# roles/deploy_app/tasks/main.yml
---
- name: Install Standard CA Certificates
  package:
    name: ca-certificates
    state: present

- name: Copy Custom CA Certificates
  copy:
    src: "{{ item }}"
    dest: /etc/ssl/certs/
  with_items: "{{ ca_certificates }}"

- name: Ensure CA Certificates Are Trusted
  command: update-ca-certificates

- name: Create Deployment Folder
  file:
    path: "{{ deployment_folder }}"
    state: directory
```

```
- name: Create Virtual Environment
  command: python3 -m venv {{ deployment_folder }}/venv
  args:
    creates: "{{ deployment_folder }}/venv"

- name: Install Python Application (Wheel)
  pip:
    requirements: "{{ deployment_folder }}/venv"
    virtualenv: "{{ deployment_folder }}/venv"
    executable: "{{ deployment_folder }}/venv/bin/pip"
    name: "{{ deployment_folder }}/{{ wheel_file }}"

- name: Create Instance Folder
  file:
    path: "{{ instance_path }}"
    state: directory

- name: Deploy Configuration File
  template:
    src: config.py.j2
    dest: "{{ instance_path }}/config.py"

- name: Deploy run.sh Script
  template:
    src: run.sh.j2
    dest: "{{ deployment_folder }}/run.sh"
    mode: '0755'

- name: Create Systemd Service for Application
```

```
  template:
    src: example.service.j2
    dest: /etc/systemd/system/example.service

- name: Reload Systemd Daemon
  command: systemctl daemon-reload

- name: Enable and Start Example Application Service
  systemd:
    name: example
    state: started
    enabled: true
…
```

4. Templates

run.sh.j2 Template for the Script

This script will run the Flask application using Gunicorn, with the required environment variables set.

Bash File

```
# roles/deploy_app/templates/run.sh.j2
#!/bin/bash
export INSTANCE_PATH={{ instance_path }}
exec {{ deployment_folder }}/venv/bin/gunicorn -w 4 -b 0.0.0.0:{{ port }} app:app
```

config.py.j2 Template for the Configuration File

The config file will be dynamically generated using the values from group_vars/all.yml.

```
# roles/deploy_app/templates/config.py.j2
SECRET_KEY = '{{ secret_key }}'
SQLALCHEMY_DATABASE_URI = '{{ db_path }}'
ADMIN_GROUPS = {{ admin_groups }}
```

example.service.j2 Template for the Systemd Service File

The systemd service will run run.sh to start the Flask app.

```
# roles/deploy_app/templates/example.service.j2

[Unit]

Description=Flask Application


[Service]

ExecStart={{ deployment_folder }}/run.sh

Environment="INSTANCE_PATH={{ instance_path }}"

Restart=always


[Install]

WantedBy=multi-user.target
```

5. CA Certificates

Place the CA certificate files (CA1.crt, CA2.crt, CA3.crt) in the roles/deploy_app/files/ folder. These certificates will be copied to the appropriate location during the playbook execution.

6. Flask Wheel File

The wheel file (example-1.1.2-py3-none-any.whl) provided in the assessment should be stored in the roles/deploy_app/files/ folder. The Ansible playbook will deploy this wheel to the target host.

7. Security Considerations

- **Variables**: Sensitive variables like SECRET_KEY and DB_PATH should be kept secure using Ansible Vault if this playbook is to be shared.
- **Permissions**: Ensure that the script files have the correct permissions. The run.sh script has been assigned executable permissions (0755).
- **Systemd Service**: By using systemd, the Flask application will restart automatically if it crashes, enhancing reliability.

8. Deliverables

Finally, you will zip the entire Ansible project, including:

- The playbook.yml.
- The roles/ directory (containing the tasks, templates, and files).
- The .git directory with all commit history to showcase the development process.

zip -r ansible_project.zip ansible_project/

Conclusion

This solution is designed to meet all the requirements of the assessment, ensuring the deployment of certificates, a Python Flask app, and securing sensitive data with the use of variables and Ansible best practices.