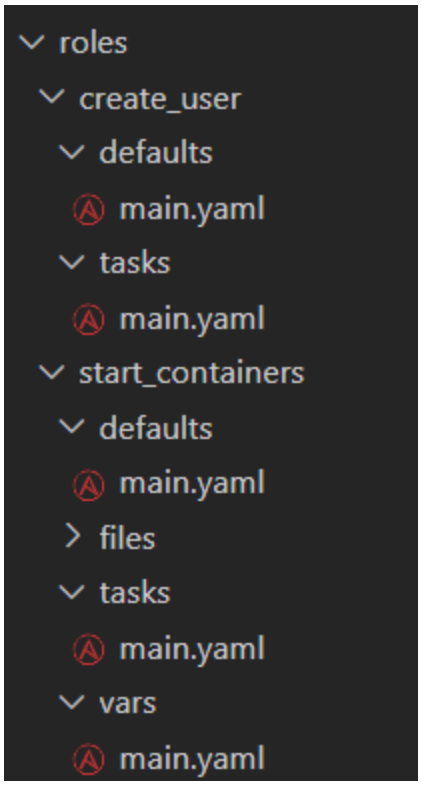
Demo Project: Structure Playbooks with Ansible Roles

Project Description

In this project, we will break up a large Ansible playbook into smaller, reusable **roles** to improve modularity and organization. Each role will handle a specific task, such as creating a user or managing Docker containers.

Step 1: Create Ansible Role Structure

1. **Create the roles directory** like this:



2. **Define the `create_user` role:**
   - **Defaults file (`roles/create_user/defaults/main.yaml`):**

     ```
     user_groups: admin,docker
     ```

   - **Tasks file (`roles/create_user/tasks/main.yaml`):**

     ```
     - name: Create new linux user
       user:
         name: eduardo
         groups: "{{ user_groups }}"
     ```

3. **Define the `start_containers` role:**
   - **Defaults file (`roles/start_containers/defaults/main.yaml`):**

     ```
     docker_registry: https://aws_account_id.dkr.ecr.region.amazonaws.com
     ```

```
docker_username: AWS
docker_password: xxxxx
```

- **Docker Compose file (`roles/start_containers/files/docker-compose.yaml`):**

```
version: '3'
services:
 java-app:
  image: eduardobautistamaciel/demo-app:java-maven-2.0
  environment:
   - DB_USER=user
   - DB_PWD=pass
   - DB_SERVER=mysql
   - DB_NAME=my-app-db
  ports:
  - 8080:8080
  container_name: my-java-app
 mysql:
  image: mysql
  ports:
   - 3306:3306
  environment:
   - MYSQL_ROOT_PASSWORD=my-secret-pw
   - MYSQL_DATABASE=my-app-db
   - MYSQL_USER=user
   - MYSQL_PASSWORD=pass
  volumes:
  - mysql-data:/var/lib/mysql
  container_name: mysql
 phpmyadmin:
  image: phpmyadmin
  environment:
   - PMA_HOST=mysql
  ports:
   - 8083:80
```

```
    container_name: myadmin
volumes:
  mysql-data:
    driver: local
```

- **Tasks file (`roles/start_containers/tasks/main.yaml`):**

```
- name: Copy docker compose
  copy:
    src: docker-compose.yaml
        dest: /home/eduardo/docker-compose.yaml

- name: Docker login
  docker_login:
    registry_url: "{{ docker_registry }}"
    username: "{{ docker_username }}"
    password: "{{ docker_password }}"
- name: Start containers from compose
  community.docker.docker_compose_v2:
    project_src: /home/eduardo
```

- **Variables file (`roles/start_containers/vars/main.yaml`):**

```
docker_registry: https://index.docker.io/v1/
docker_username: eduardobautistamaciel
```

Step 2: Define the Main Ansible Playbook

Create `deploy-docker-with-roles.yaml`:

```
---
- name: Install Docker
  hosts: all
  become: yes
```

```
  tasks:
  - name: Install Docker
    yum:
      name: docker
      update_cache: yes
      state: present
  - name: Start docker daemon
    systemd:
      name: docker
      state: started

- name: Create new linux user
  hosts: all
  become: yes
  vars:
    groups: adm,docker
  roles:
    - create_user

- name: Install Docker-compose
  hosts: all
  become: yes
  become_user: eduardo
  tasks:
  - name: Create docker-compose directory
    file:
      path: ~/.docker/cli-plugins
      state: directory
  - name: Get architecture of remote machine
    shell: uname -m
    register: remote_arch
  - name: Install docker-compose
    get_url:
      url: "https://github.com/docker/compose/releases/latest/download/docker
-compose-linux-{{lookup('pipe', 'uname -m')}}"
      dest: ~/.docker/cli-plugins/docker-compose
```

```
    mode: +x

- name: Start docker containers
  hosts: all
  become: yes
  become_user: eduardo
  vars_files:
  - project-vars
  roles:
    - start_containers
```

Step 3: Deploy the Infrastructure

1. **Deploy EC2 instances using Terraform from the terraform repo:**

   `terraform apply --auto-approve`

2. **Go back to the Ansible repo and run the playbook:**

   `ansible-playbook deploy-docker-with-roles.yaml -i inventory_aws_ec2.yaml -u ec2-user`

Step 4: Clean up

Destroy the infrastructure when no longer needed:

`terraform destroy --auto-approve`