Demo Project: Automate displaying EKS cluster information

In this project, we will create a Python script that fetches and displays essential information about an AWS EKS cluster, including:

- EKS cluster status
- Kubernetes version
- Cluster endpoint

Step 1: Create EKS Cluster with Terraform

1. Navigate to the project folder from the provided link:

   https://gitlab.com/twn-devops-projects/automation-with-python/automation-projects/-/blob/main/eks-status-checks.py?ref_type=heads
2. Install the **boto3** library to interact with AWS APIs: `pip install boto3`
3. Confirm the installations by navigating to **External Libraries → Python 3.x → site-packages** in PyCharm.
4. Initialize Terraform: `terraform init`
5. Preview the planned changes: `terraform plan`
6. Apply the infrastructure configuration: `terraform apply -auto-approve`
7. Confirm in the AWS console that the EKS cluster and EC2 instances are created.

Step 2: Write the Python Script

Create the `eks-status-checks.py` file with the following content:

```
import boto3

client = boto3.client('eks', region_name="us-east-1")

clusters = client.list_clusters()['clusters']

for cluster in clusters:
    response = client.describe_cluster(
        name=cluster
    )
    cluster_info = response['cluster']
    cluster_status = cluster_info['status']
    cluster_endpoint = cluster_info['endpoint']
    cluster_version = cluster_info['version']

    print(f"Cluster {cluster} status is {cluster_status}")
    print(f"Cluster endpoint: {cluster_endpoint}")
    print(f"Cluster version: {cluster_version}")
```

Step 3: Run the Script

Execute the script directly from PyCharm or the terminal: `python3 eks-status-checks.py`

Example Output:

```
C:\Users\eduar\PycharmProjects\my-pythonProject\.venv\Scripts\python.exe C:\Users\eduar\PycharmProjects\my-pythonProject\eks-status-checks
Cluster myapp-eks-cluster status is ACTIVE
Cluster endpoint: https://74466CF42456512D0E3DF44A2D9D7EAA.gr7.us-east-1.eks.amazonaws.com
Cluster version: 1.27

Process finished with exit code 0
```

Step 4: Clean Up Resources

- Once you are finished, clean up the infrastructure to avoid additional costs: `terraform destroy --auto-approve`
- Verify that all resources are deleted by running: `terraform state list`

  If no resources are listed, the cleanup was successful.
