Demo Project: Health Check: EC2 Status Checks

This project demonstrates how to fetch the statuses of EC2 instances using Python and AWS boto3, with continuous health checks at regular intervals.

Step 1: Create EC2 Instances with Terraform

1. Create the **main.tf** file using Terraform to provision EC2 instances.
    - Reference the Terraform configuration in this repo: https://gitlab.com/twn-devops-projects/automation-with-python/terraform
2. Create **terraform.tfvars** with your custom configurations (ensure it is excluded from version control using `.gitignore`).

    Example terraform.tfvars:

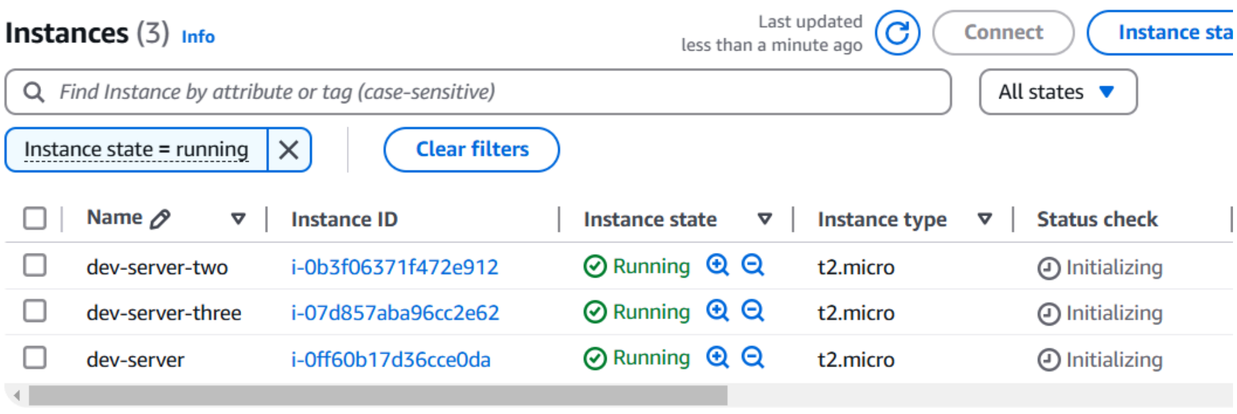
    ```
    vpc_cidr_block = "10.0.0.0/16"
    subnet_cidr_block = "10.0.10.0/24"
    avail_zone = "us-east-1a"
    env_prefix = "dev"
    my_ip = "167.57.134.210/32"
    instance_type = "t2.micro"
    public_key_location = "/home/eb/.ssh/id_rsa.pub"
    ```

3. Initialize and apply the Terraform configuration:

- `terraform init`
- `terraform plan`
- `terraform apply -auto-approve`

4. Confirm the instances are running in the AWS EC2 Console.

   Example output:



Instances (3) Info

Last updated less than a minute ago

Find Instance by attribute or tag (case-sensitive)

Instance state = running

Name	Instance ID	Instance state	Instance type	Status check
dev-server-two	i-0b3f06371f472e912	Running	t2.micro	Initializing
dev-server-three	i-07d857aba96cc2e62	Running	t2.micro	Initializing
dev-server	i-0ff60b17d36cce0da	Running	t2.micro	Initializing

Step 2: Install Required Libraries

1. Install the **boto3** library to interact with AWS APIs: `pip install boto3`
2. Install the **schedule** library to schedule periodic tasks: `pip install schedule`
3. Confirm the installations by navigating to **External Libraries → Python 3.x → site-packages** in PyCharm.

Step 3: Write the Python Script

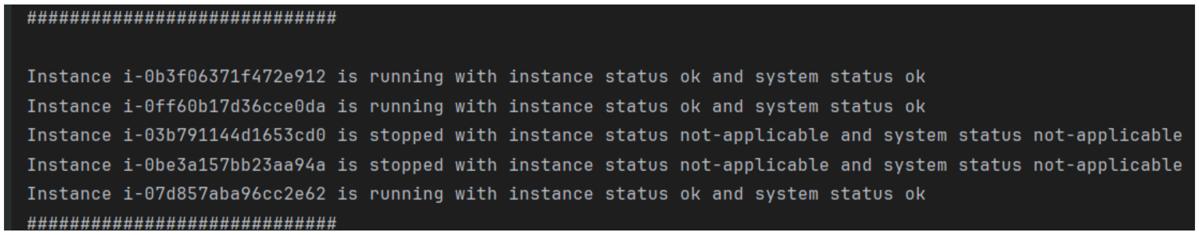
Create the **main.py** file.

Reference the file in this repo: https://gitlab.com/twn-devops-projects/automation-with-python/automation-projects

Step 4: Run the Script

- Execute the script via PyCharm or terminal: `python3 main.py`

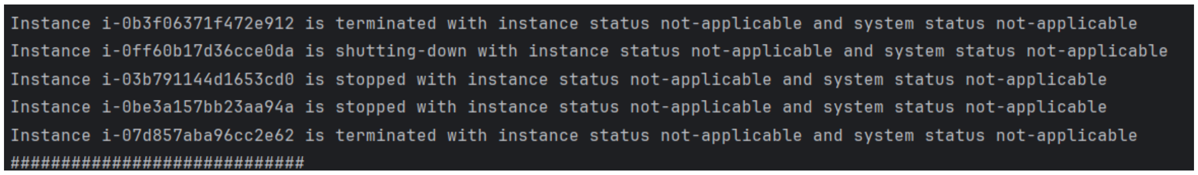
  Example Output:



```
#############################

Instance i-0b3f06371f472e912 is running with instance status ok and system status ok
Instance i-0ff60b17d36cce0da is running with instance status ok and system status ok
Instance i-03b791144d1653cd0 is stopped with instance status not-applicable and system status not-applicable
Instance i-0be3a157bb23aa94a is stopped with instance status not-applicable and system status not-applicable
Instance i-07d857aba96cc2e62 is running with instance status ok and system status ok
#############################
```

  Another example:



```
Instance i-0b3f06371f472e912 is terminated with instance status not-applicable and system status not-applicable
Instance i-0ff60b17d36cce0da is shutting-down with instance status not-applicable and system status not-applicable
Instance i-03b791144d1653cd0 is stopped with instance status not-applicable and system status not-applicable
Instance i-0be3a157bb23aa94a is stopped with instance status not-applicable and system status not-applicable
Instance i-07d857aba96cc2e62 is terminated with instance status not-applicable and system status not-applicable
#############################
```

Step 5: Clean Up Resources

1. Stop the Python script using **Ctrl + C**.
2. Destroy the Terraform infrastructure: `terraform destroy --auto-approve`
3. Verify that no EC2 instances are running in the AWS console.

Additional Notes

- Ensure your **AWS CLI** is configured with the correct credentials.
- Use **.gitignore** to exclude sensitive files like **terraform.tfvars** and AWS credentials.
- Modify the schedule interval according to your monitoring needs.
