Objective Overview

This Proof of Concept (POC) is designed to demonstrate infrastructure automation and cloud-native resource management using AWS services. The deliverables include:

- A CloudFormation YAML template.
- A Launch Stack URL.
- A secure Python script (no hardcoded credentials).
- Documentation for implementation and execution.

📁 Required Deliverables

1. CloudFormation YAML file (`template.yaml`)
2. CloudFormation Stack Launch URL
3. Python script (using IAM role/profile)
4. This implementation documentation

🛠️ Implementation Steps

1. Design CloudFormation Template

Create a YAML file that describes the AWS resources to be deployed. For this task:

- Include an **S3 bucket** (with read-only access).
- Include an **EC2 instance** (e.g., Amazon Linux).
- Define outputs to easily reference the resource IDs (e.g., bucket name, instance ID).

> 📝 Use descriptive logical IDs, tags, and outputs to keep resources well-organized and queryable.

2. Host the Template Publicly

To generate a CloudFormation Launch URL, the template must be hosted at a public HTTPS-accessible URL.

Options include:

- **GitHub**: Upload to a public repository and use the **raw** file URL.
- **S3 bucket**: Upload with public-read permission and enable HTTPS access.

---

3. Generate the Launch Stack URL

Use the following format to generate a CloudFormation launch link:

```
https://console.aws.amazon.com/cloudformation/home?#/stacks/create/review?templateURL=<public-template-url>
```

Replace `<public-template-url>` with the actual URL of your hosted `template.yaml`.

This URL will allow anyone with AWS Console access to quickly launch the stack in their account.

---

4. Develop the Python Automation Script

Create a Python script that:

- Accepts the **CloudFormation Launch URL** as input.
- Extracts the `templateURL` parameter.
- Uses the AWS SDK (Boto3) to:
  - Create a new CloudFormation stack.
  - Wait for the stack to complete creation.
  - Retrieve inventory information like:
    - EC2 instance IDs
    - S3 bucket names
- Does **not** use any hardcoded keys or ARNs.

> 🛡️ The script should assume that it is running with proper IAM permissions (via EC2 instance profile or assumed role).

---

5. Run & Verify the Solution

To test your implementation:

1. **Ensure AWS CLI is configured** with assumed role or IAM profile (no manual credentials).
2. **Execute the script** in an environment with required IAM permissions.
3. **Verify the resources** have been created in the AWS Console.
4. **Check script output** to confirm inventory fetch is accurate.

✅ Best Practices Followed

- **Security**: No credentials hardcoded; uses environment-based IAM access.
- **Modular approach**: Clear logical separation of steps.
- **Clarity**: Well-commented and documented for review or reuse.
- **Reusability**: Stack and script can be reused with minor changes.
