# **5-Hour Hands-On Test: AWS Automation with Terraform, Jinja2, and Boto3**

## **Objective**

You are required to build a **Python-based AWS Infrastructure as Code (IaC) tool** that:

1. **Generates a Terraform configuration dynamically** using **Jinja2**.
2. **Executes Terraform using a Python wrapper** (python-terraform library) to deploy an EC2 instance attached to an Application Load Balancer (ALB).
3. **Verifies deployment using boto3**, ensuring that the EC2 instance and ALB exist and are correctly configured.
4. **Implements clean code practices** using functions and classes with **error handling** (try-except).

## **Test Breakdown & Time Allocation (5 Hours)**

| **Task** | **Description** | **Estimated Time** |
| --- | --- | --- |
| **1. Jinja2 Terraform Templating** | Complete the given Jinja2 template to create a Terraform .tf file dynamically with user inputs. | **1.5 hours** |
| **2. Terraform Execution via Python** | Use python-terraform to execute Terraform commands based on user selections. | **1 hour** |
| **3. AWS Validation with Boto3** | Use boto3 to fetch AWS data and validate that the deployment was successful. | **1 hour** |
| **4. Error Handling & Modularization** | Ensure structured code using classes, functions, and try-except blocks. | **1 hour** |
| **5. Code Documentation & Submission** | Ensure clean documentation and provide a JSON file containing AWS validation data. | **30 min** |

## **Project Requirements**

### **1. User Input for Cloud Deployment**

Your script should:

1. Prompt the user to select:  
   * **AMI**: Choose between Ubuntu (ami-???) or Amazon Linux (ami-???).
   * **Instance Type**: Choose between t3.small and t3.medium.
   * **Availability Zone & Region**:
     + If the user selects any region **other than us-east-1**, **reject** the input and default to us-east-1.
   * **Load Balancer Name**: Ask for a custom ALB name.
2. Store these selections as **variables** to be passed into the Jinja2 template.

### **2. Terraform Configuration Using Jinja2**

You are provided with a 90% **complete Jinja2 template** for Terraform. Complete / fix the missing part (you may use chatgpt here)

#### **Given Terraform Jinja2 Template (You Need to Complete the Missing Sections)**

| # **Map user inputs** to jinja2 template variables ami = ami\_options.get(ami\_choice, "ami-?????") instance\_type = instance\_types.get(instance\_type\_choice, "t3.small")  terraform\_template = """ provider "aws" {  region = "{{ region }}" }  resource "aws\_instance" "web\_server" {  ami = "{{ ami }}"  instance\_type = "{{ instance\_type }}"  availability\_zone = "{{ availability\_zone }}"   tags = {  Name = "WebServer"  } }  resource "aws\_lb" "application\_lb" {  name = "{{ load\_balancer\_name }}"  internal = false  load\_balancer\_type = "application"  security\_groups = [aws\_security\_group.lb\_sg.id]  subnets = aws\_subnet.public[\*].id }  resource "aws\_security\_group" "lb\_sg" {  name = "lb\_security\_group"  description = "Allow HTTP inbound traffic"   ingress {  from\_port = 80  to\_port = 80  protocol = "tcp"  cidr\_blocks = ["0.0.0.0/0"]  } }  resource "aws\_lb\_listener" "http\_listener" {  load\_balancer\_arn = aws\_lb.application\_lb.arn  port = 80  protocol = "HTTP"   default\_action {  type = "forward"  target\_group\_arn = aws\_lb\_target\_group.web\_target\_group.arn  } }  resource "aws\_lb\_target\_group" "web\_target\_group" {  name = "web-target-group"  port = 80  protocol = "HTTP"  vpc\_id = aws\_vpc.main.id }  resource "aws\_lb\_target\_group\_attachment" "web\_instance\_attachment" {  target\_group\_arn = aws\_lb\_target\_group.web\_target\_group.arn  target\_id = aws\_instance.web\_server.id }  resource "aws\_subnet" "public" {  count = 2  vpc\_id = aws\_vpc.main.id  cidr\_block = "10.0.${count.index}.0/24"  availability\_zone = element(["us-east-1a", "us-east-1b"], count.index) }  resource "aws\_vpc" "main" {  cidr\_block = "10.0.0.0/16" } """ |
| --- |

#### 

### **3. Execute Terraform Using Python**

Use **python-terraform** to apply the generated Terraform script.

#### **Required Steps:**

1. Run terraform init, terraform plan, and terraform apply and print the output to screen so you can debug issues
2. Capture the Terraform output (instance ID, LB DNS name).
3. If Terraform fails, display the error message and terminate.

### **4. AWS Validation with boto3**

Once Terraform deployment is complete, use **boto3** to verify the resources.

#### **Required Validations:**

1. **Fetch EC2 instance details** using the machine ID got from terraform or using a filter
   * Ensure the instance exists and is running.
   * Retrieve its **Public IP**.
2. **Fetch ALB details**:
   * Ensure the ALB exists.
   * Retrieve its **DNS name**.
3. **Store the verification data as a JSON file (aws\_validation.json)**:

{

"instance\_id": "i-0123456789abcdef0",

"instance\_state": "running",

"public\_ip": "3.92.102.45",

"load\_balancer\_dns": "my-alb-123456.elb.amazonaws.com"

}

You may use chatGPT to get help

### **5. Error Handling & Code Structure**

**Points will be awarded for:**

* **Using functions & classes** to structure the code.
* **Error handling with try-except** for Terraform execution and AWS API calls.
* **Validating user input** (e.g., rejecting invalid regions).

## **Submission Requirements**

**Students must submit:**

1. **Python Script**: The complete deployment script.
2. **Generated Terraform File**: The .tf file created from the Jinja2 template.
3. **AWS Validation JSON (aws\_validation.json)**: Proof that the deployment succeeded.
4. **README**: Explain how the script works and how to run it.

## **Evaluation Criteria**

| **Criteria** | **Points** |
| --- | --- |
| **Correctly Generates Terraform File Using Jinja2** | 20 |
| **Executes Terraform from Python & Captures Outputs** | 20 |
| **Uses boto3 to Validate EC2 & ALB** | 20 |
| **Proper Code Structure (Functions & Classes)** | 15 |
| **Error Handling (try-except for Terraform & AWS API)** | 15 |
| **Code Readability & Documentation** | 10 |
| **Bonus (Code working end2end)** | **10** |
| **Total** | **110** |

### **Final Notes**

* You may only use chatgpt for help BUT you should not copy past code that chatgpt provided to you.
* You will be reviewed for the code quality and not necessary if you were able to complete the functionality
* Don’t forget to have termination capability in your code

Good luck