**IST105 - Introduction to Programming**  
**Assignment #5**

### **Assignment #5 -**  The Interactive Treasure Hunt: Deploying with AWS Auto Scaling and GitHub

**Important Note:**

* Assignments with code that contains errors and does not run correctly will receive a grade of zero.
* Please ensure that your code is fully functional and free of syntax or runtime errors before submitting. It is recommended that you thoroughly test your code in Python IDLE and fix any issues that arise.
* Include comments where necessary to explain your logic and observations.

**Prerequisite:**

1. **Python Basics:**
   * Familiarity with Python syntax, including input/output, conditions, loops, and binary operators.
2. **PHP Basics:**
   * Familiarity with creating forms in PHP and passing data to a Python script.
3. **AWS Knowledge:**
   * Familiarity with EC2 instances, Auto Scaling Groups, Load Balancers, and EC2 Launch Templates.
4. **Git/GitHub Basics:**
   * Familiarity with creating repositories, branches, and managing code on GitHub.

**Objective:**

The goal of this assignment is to create an **interactive web application** where users input data through a PHP form, and Python processes the input to solve a fun and engaging puzzle. You will deploy the application on **AWS** using **Auto Scaling Groups**, a **Load Balancer**, and **GitHub** for version control. The application must scale automatically based on CPU utilization, and you will stress-test the system to demonstrate scaling.

### **The Problem: The Interactive Treasure Hunt**

You are building a treasure hunt game where users interact with the system **only once** through a PHP form. The form collects user input (numbers or text), and Python processes the input to solve the puzzle.

Here’s how it works:

1. **User Input Form (PHP):**
   * Create a PHP form (form.php) where users can input:
     + A **number** (e.g., their birth year).
     + A **text message** (e.g., their name or a secret word).
   * The form submits the data to a Python script (process.py) for processing.
2. **Python Processing:**
   * The Python script performs the following tasks:
     + **Task 1: Number Puzzle**
       - Check if the number is even or odd.
       - If the number is even, calculate its square root.
       - If the number is odd, calculate its cube.
     + **Task 2: Text Puzzle**
       - Convert the text message to binary.
       - Count the number of vowels in the text.
     + **Task 3: Treasure Hunt**
       - Generate a random number between 1 and 100.
       - Use a **while loop** to let the user guess the number (simulated in Python, not user input).
       - If the user guesses correctly within 5 attempts, they win the treasure!
3. **Display Results (PHP):**
   * The results of the Python processing are displayed on the web page, including:
     + The outcome of the number puzzle.
     + The binary-encoded text and vowel count.
     + Whether the user won the treasure hunt.

**Assignment Tasks:**

1. **Create the PHP Form (form.php):**
   * The form should have the following fields:
     + Input for a number (e.g., birth year).
     + Input for a text message (e.g., name or secret word).
     + A submit button labeled "Solve the Puzzle".
   * The form should submit the data to process.py using the POST method.
2. **Write the Python Script (process.py):**
   * The script should:
     + Retrieve the number and text message from the PHP form.
     + Perform the number puzzle (even/odd check and calculations).
     + Perform the text puzzle (binary conversion and vowel count).
     + Simulate the treasure hunt (random number guessing game).
     + Return the results to the PHP script for display.
3. **Set Up GitHub Repository:**
   * Create a public GitHub repository named IST105-Assignment5.
   * Create three branches:
     + main: Contains the final, working version of the code.
     + development: Used for testing and integrating features.
     + feature1: Used for developing the initial version of the application.
   * Commit and push all files (form.php, process.py) to all three branches.
4. **Deploy on AWS:**
   * Use an **EC2 Launch Template** to create an Auto Scaling Group with the following configuration:
     + Minimum: 1 EC2 instance.
     + Maximum: 5 EC2 instances.
   * Configure the **Target Tracking Policy** for the Auto Scaling Group:
     + Policy type: Target tracking scaling.
     + Metric: Average CPU utilization at 10%.
     + Instances need 30 seconds to warm up before including in the metric.
     + Scale-in: Enabled.
   * Set up a **Load Balancer** connected to the Auto Scaling Group.
   * Deploy the code from the main branch to the EC2 instances.

**8. Test the Setup:**

* Open your web browser and navigate to the load balancer's DNS name (e.g., http://<load-balancer-dns>/form.php) to see the user input form.
* Submit values to see the calculations and output displayed on the results page.

**9. Stress Test the System:**

* Connect to an EC2 instance in the Auto Scaling Group via SSH.
* Run the following command to generate high CPU usage:

sudo yum install -y stress

stress --cpu 6 --timeout 120

* Observe the Auto Scaling Group creating additional EC2 instances to handle the load.

### **Example Output:**

#### **1. User Interaction (PHP Form):**

The user visits the Load Balancer's public URL (e.g., http://<load-balancer-dns>/form.php) and sees the following form:

**Welcome to the Interactive Treasure Hunt!**

**Enter your details to solve the puzzle and find the treasure.**

Number (e.g., birth year): [25]

Text (e.g., name or secret word): [Treasure]

[Solve the Puzzle]

The user inputs:

* Number: 25
* Text: "Treasure"  
  And clicks **"Solve the Puzzle"**.

#### **2. Python Processing Results (Displayed on Web Page):**

After submitting the form, the user sees the following results:

**Number Puzzle:**

- The number 25 is odd. Its cube is 15625.

**Text Puzzle:**

- Binary: 01010100 01110010 01100101 01100001 01110011 01110101 01110010 01100101

- Vowel Count: 4

**Treasure Hunt:**

- The secret number is 42.

- Attempt 1: 50 (Too high!)

- Attempt 2: 30 (Too low!)

- Attempt 3: 42 (Correct!)

- You found the treasure in 3 attempts!

**Submission:**

1. **GitHub Repository:**
   * Provide the **public link** to your GitHub repository.

**[Add the link for GitHub here]**

* + Ensure the repository contains:
    - form.php
    - process.py
    - process.php
    - All files must be present in the **main**, **development**, and **feature1** branches.

1. **Screenshots:**
   * A screenshot of the Load Balancer showing the public URL and identifying which EC2 instance it is connected to.

**[Add the screenshot here]**

* A screenshot of the CPU Utilization graph for the Auto Scaling Group.

**[Add the screenshot here]**

* A screenshot of the Instance Management page showing:
  + The normal EC2 instance before stressing the CPU.

**[Add the screenshot here]**

* + All EC2 instances created automatically after stressing the CPU.

**[Add the screenshot here]**

1. **File:**
   * Upload this file named **Assignment5\_FirstNameLastName.doc** to Canvas containing:
     + The GitHub repository link.
     + All screenshots.

**Tips:**

* Ensure all paths and permissions are set correctly to avoid access issues.
* Test your Python and PHP scripts locally before deploying them on the EC2 instances.
* Double-check the public IP addresses and load balancer DNS name.
* Review the Apache and Python integration settings to ensure smooth operation.
* Use meaningful commit messages and branch names in your GitHub repository.
* Apply the knowledge and techniques from previous labs in your script. Your implementation should reflect the skills and concepts covered in our previous labs.