# ****IST105 - Introduction to Programming****

# ****Midterm Exam:** Deploying a Mathematical Application with AWS Auto Scaling and GitHub**

**Course:** IST105 - Introduction to Programming  
**Instructor:** Washington Valencia  
**Version:** Feb 2025

## ****Important Notes:****

* Code with syntax or runtime errors will receive a grade of zero.
* The submission must include:
  + A **public URL** for the GitHub repository with all required branches and files.
  + A **screenshot** of the fully functional web application running on the AWS EC2 instance, showing the Load Balancer URL.
  + A **screenshot** of the AWS Auto Scaling Group showing the CPU consumption for the EC2 compute instance **before** stressing the system.
  + A **screenshot** of the AWS Auto Scaling Group showing the CPU consumption for the EC2 compute instance **after** stressing the system.
  + A **screenshot** of all EC2 compute instances **before** stressing the system.
  + A **screenshot** of all EC2 compute instances **after** stressing the system.
* All code must be fully functional and free of errors.
* The GitHub repository **must be public**; otherwise, the grade will be zero.
* All images must be **clear and legible**; otherwise, the grade will be zero.

## ****Prerequisites:****

1. **AWS Knowledge:**
   * Familiarity with EC2 instances, Auto Scaling Groups, Load Balancers, and EC2 Launch Templates.
2. **Git/GitHub Basics:**
   * Familiarity with creating repositories, branches, and managing code on GitHub.
3. **Software Installation:**
   * Ability to install and configure Python, PHP, and Apache on an EC2 instance.
4. **Programming Knowledge:**
   * Understanding of Python syntax, functions, conditionals, and mathematical operations.
   * Familiarity with creating forms in PHP and passing data to a Python script.

## ****Objective:****

You will create a web-based mathematical application using AWS EC2, Python, PHP, and Apache. The application will allow users to input values, perform a series of mathematical operations using Python, and display results through PHP.

This exam will assess your ability to:

* Work with AWS (EC2, Auto Scaling Groups, Load Balancers).
* Use GitHub for version control and deployment.
* Handle conditional logic, functions, and mathematical operations in Python.
* Deploy and scale a web application on AWS.

## ****Exam Tasks:****

### **1. Create the User Input Form (Mandatory):**

* Write a PHP script named **math\_form.php**to create a form that takes two numerical inputs from the user (**input1** and **input2**) and offers a dropdown menu to select a mathematical operation (**addition, subtraction, multiplication, or division**).
* The form must submit the data to a Python script (**math\_operations.py**) using the POST method.

**Form Fields:**

* Input for number 1 (numeric).
* Input for number 2 (numeric).
* Dropdown for operation selection (add, sub, mul, div).
* A submit button labeled "Calculate".

**Example:**  
Users can navigate to **http://<your-public-ip>/math\_form.php** to input values and submit the form.

### **2. Write a Python Script for Mathematical Operations:**

* Write a Python script named **math\_operations.py**that:
  1. Retrieves the user input for the two numbers and the selected operation.
  2. Implements the following operations based on the user's selection:
     + **Addition:** Add the two numbers.
     + **Subtraction:** Subtract the second number from the first.
     + **Multiplication:** Multiply the two numbers.
     + **Division:** If the second number is not zero, divide the first number by the second. If the second number is zero, return an error message indicating division by zero is not allowed.
  3. Additional conditions:
     + If the result of the operation is greater than 100, multiply the result by 2.
     + If the result is less than 0, add 50 to the result.
  4. Return the result formatted as HTML to be displayed on a webpage.

### **3. Create a PHP Script to Process Input:**

* Write a PHP script named **process\_math.php** that:
  + Receives input from the **math\_form.php** form.
  + Passes the inputs and selected operation to the Python script **math\_operations.py**.
  + Displays the results or any error messages returned by the Python script.

### **4. Set Up GitHub Repository:**

* Create a public GitHub repository named **IST105-Midterm**.
* 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 (**math\_form.php, math\_operations.py, process\_math.php**) to all three branches.

### **5. Deploy the Application on AWS with Auto Scaling:**

* Use an EC2 Launch Template to create an Auto Scaling Group with the following configuration:
  + Minimum: 1 EC2 instance.
  + Maximum: 7 EC2 instances.
* Configure the Target Tracking Policy for the Auto Scaling Group:
  + Policy type: Target tracking scaling.
  + Metric: Average CPU utilization at 35%.
  + 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.

### **6. Test the Application:**

* Open your browser and navigate to the Load Balancer's DNS name

(e.g., **http://<load-balancer-dns>/math\_form.php**) to test the input form.

* Submit values and see the result displayed on the results page.

### **7. Stress Test the System:**

* Connect to an EC2 instance in the Auto Scaling Group via SSH.
* Run the corresponding commands to generate high CPU usage:
* Observe the Auto Scaling Group creating additional EC2 instances to handle the load.

## ****Submission Requirements:****

### **GitHub Repository:**

* Provide the public URL for your GitHub repository below:  
  **GitHub Repository URL:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### **Screenshots:**

* Attach the following screenshots:
  1. **Fully Functional Web Application:**
     + Screenshots showing the web application running in different EC2 compute instances, showing the Load Balancer URL.  
       **Screenshot1 EC1:**

**Screenshot2 EC2:**

* 1. **AWS Auto Scaling Group - CPU Consumption (Before Stress):**
     + Screenshot showing the CPU consumption of the EC2 compute instance before stressing the system.  
       **Screenshot 3:**
  2. **AWS Auto Scaling Group - CPU Consumption (After Stress):**
     + Screenshot showing the CPU consumption of the EC2 compute instance after stressing the system.  
       **Screenshot 4:**
  3. **EC2 Compute Instances (Before Stress):**
     + Screenshot showing all EC2 compute instances before stressing the system.  
       **Screenshot 5:**
  4. **EC2 Compute Instances (After Stress):**
     + Screenshot showing all EC2 compute instances after stressing the system.  
       **Screenshot 6:**

1. **File Submission:**

* A document (e.g., **Midterm\_FirstNameLastName.doc**) containing:
  + - The GitHub repository URL.
    - All required screenshots.
* The doc file should be named Midterm\_FirstNameLastName.zip and uploaded to Canvas.

## ****Example Output:****

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

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

A computer screen shot of a computer program

AI-generated content may be incorrect.

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

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

A screen shot of a computer

AI-generated content may be incorrect.

## ****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.
* Use meaningful commit messages and branch names in your GitHub repository.