THE UNIVERSITY OF SYDNEY

School of Computer Science
Dr. Ying Zhou

COMP5349: Cloud Computing Sem. 1/2025

Assignment 2: AWS Project

Individual Work: 20% **12.05.2025**

Tasks

In this assignment, you will deploy an enhanced image annotation application that integrates a serverless architecture for specific backend tasks. The application builds on the functionality explored in Assignment 1 and consists of two main components:

1. **Web Application Component:** A user-facing web application hosted on EC2 instance(s). It should provide a form to allow users to upload images and a page to display all previously uploaded images along with their captions/annotations. From an end user's perspective, the interface and behavior will be similar to Assignment 1.
2. **Serverless Component (AWS Lambda):** Two Lambda functions automatically triggered by specific events to process uploaded images using generative AI services and to generate thumbnails.

Detailed Requirements and Components

- **Web Application Component:** Deploy a standard web application on an EC2 instance that is part of an Auto Scaling Group (ASG). You must configure the ASG to scale out based on traffic, with a maximum capacity greater than 1 instance. The instances should be fronted by an Application Load Balancer (ALB) to distribute incoming traffic.

  This component must support the following:

  - Display an HTML form to upload images. On submission, the image should be stored in an AWS S3 bucket, and relevant metadata (such as filename and/or upload timestamp) saved to an RDS (MySQL) database.
  - Serve a page listing uploaded images and generated thumbnails, along with their captions/annotations retrieved from RDS and S3.

- **Serverless Component:** Implement two AWS Lambda functions that are automatically triggered when a new image is uploaded to the designated S3 bucket:
  - **Annotation Function:** Retrieves the image from S3, invokes the Gemini API to generate a description, and stores the results in the RDS database.
  - **Thumbnail Generator Function:** Retrieves the image from S3, generates a thumbnail, and stores it in a separate `thumbnails/` folder within the same S3 bucket.

Auto Scaling Test

To verify your Auto Scaling and Load Balancing setup, you must perform a load test by sending a large number of requests to the image listing page of your web application. You may use a tool such as ApacheBench or a Python-based concurrent request generator to simulate concurrent traffic.

You must provide evidence (e.g., screenshots from the AWS EC2 Console, CloudWatch metrics, and Load Balancer monitoring tools) demonstrating the following:

- **EC2 Instances Scaling Out:** Your Auto Scaling Group launches additional instances in response to increased CPU or memory utilization.
- **EC2 Instances Scaling In:** Instances are automatically terminated as load decreases.
- **Load Distribution:** Incoming requests are successfully distributed across multiple EC2 instances by the ALB.

Deployment Environment and Requirements

You are recommended to use AWS Learner's lab for this assignment. The Learner Lab provides a long-running environment with a $50.00 credit and access to all the AWS services covered in this unit. However, there are specific usage limits for each service. Please read the documentation carefully and ensure that you stay within these limits.

Exceeding service limits, even if you remain within your credit balance, can result in your account being automatically deactivated. One commonly exceeded limit is the concurrent number of AWS Lambda functions, which is capped at 10. This limit can be easily breached if your trigger configuration is incorrect.

If your account is deactivated, but still has credit remaining, it may be restored by AWS support, but the process could take several days. To avoid disruptions, plan carefully and monitor your usage.

In addition to the main services mentioned in the previous sections, you will likely need to use additional AWS services. You must justify your use of each service and the configuration choices you make. These decisions will be considered in the marking of your assignment.

Report Submission and Mark Distribution

Your submission must include a deployment report documenting the architecture with details and justifications. The report will be assessed based on the following criteria:

- **Introduction**
  A brief introduction that provides context for the assignment. Avoid repeating the assignment specification verbatim. Instead, describe the overall purpose and highlight key architectural features in your own words.

- **Architecture Diagram (5 points)**
  Provide two clearly labeled architecture diagrams:

  - **Web Application Architecture:** This diagram should show the key components involved in running the frontend application, including the EC2 Auto Scaling Group, Application Load Balancer (ALB), S3 for uploads, RDS for metadata storage, and any relevant networking/security layers (e.g., VPC, subnets, security groups, IAM roles, Bastion Host).
  - **Serverless Architecture:** This diagram should depict the AWS Lambda functions, their event source (e.g., S3), interactions with external APIs (e.g., Gemini), and downstream targets (e.g., RDS, S3 for thumbnails). Also include supporting services such as SNS, EvenBridge, VPC, Subnets or security groups whenever relevant.

  Integration Between Components: Clearly indicate the points of interaction between the two architectures, such as:

  - How the web application triggers Lambda functions (e.g., uploading to an S3 bucket monitored by S3 events)
  - How both components access shared resources (e.g., the same RDS database or S3 bucket)

  Each diagram should follow AWS architectural styles similar to those used in AWS Academy labs. They should be cover similar details as those used in the AWS Academy labs. The diagram should be prepared using a drawing software, such as draw.io. Hand-drawn diagrams will not be marked.

- **Web Application Deployment (4 points)**
  Provide a detailed description and justification of your web application deployment. This section should cover:
  - **Compute Environment:** Describe the use of EC2 within an Auto Scaling Group (ASG), and explain your configuration of:
    - Network settings (e.g., VPC, subnets)
    - Security configurations (e.g., security groups, IAM roles, Secrets Manager)
    - Administrative access (e.g., bastion host)
    - **Load Balancer setup:** Describe the use of an Application Load Balancer (ALB), including listener configuration, target groups, and health checks. Discuss how it ensures high availability and distributes traffic across EC2 instances.
  - **Database Environment:** Describe how the RDS (MySQL) database is provisioned, including its configuration, access policies, and integration with the application.
  - **Storage Environment:** Explain how the S3 bucket is configured for storing uploaded images, generated thumbnails and triggering the lambda functions.
- **Serverless Component Deployment (3 points)**
  Describe the implementation and deployment of your serverless functions. This section should address:
  - **Event-Driven Architecture:** How the Lambda functions are triggered (e.g., S3 ObjectCreated events, usage of EventBridge, SNS if relevant), and which resources they interact with.
  - **Annotation Function:** Describe how the Lambda function is packaged (e.g. as a simple script, a zip file or a container image) and deployed (e.g. through management console, cli or cloudformation) and the settings to allow it interact with S3, RDS and GeminiAPI.
  - **Thumbnail Generator:** Describe how the Lambda function is packaged (e.g. as a simple script, a zip file or a container image) and deployed (e.g. through management console, cli or cloudformation) and the settings to allow it interact with S3.
- **Auto Scaling Test Observation (1 point)**
  Include screenshots and a brief explanation showing evidence of auto scaling behavior, such as the number of EC2 instances increasing and decreasing, CPU/memory usage trends, and load distribution via the ALB. Be sure to explain how the test was performed (e.g., tool used and number of requests).

- **Summary and Lessons Learned** Conclude your report with a concise summary of your key findings. We encourage you to reflect on any challenges you faced during the assignment and explain how you addressed them.
- **Report Style and Professionalism (1 point)**
  Ensure your report is professionally written and well-organized. This includes using clear section headings, numbered figures and tables, and consistent formatting throughout. All AWS resources and components must be correctly named and referenced.

Demo Video Submission and Mark Distribution

In addition to the deployment report, you must submit a short demonstration video (strictly no longer than 10 minutes) that showcases the functionality and behavior of your application. The video should provide clear visual evidence of the system working as intended and highlight key architectural features in action.

The video must demonstrate the following:

- **Application Functionality (1 points):**
  - Uploading an image using the web interface
  - Viewing the list of uploaded images, thumbnails and their annotations
- **Lambda Function Execution (2 points):**
  - Triggering of the Lambda functions upon image upload
  - Evidence of successful execution: e.g., CloudWatch logs, S3 updates, or new entries in the RDS.
- **Infrastructure Configuration (3 points):**
  - **Application Load Balancer (ALB):** Show listener configuration, routing rules, target groups, and health check setup
  - **Auto Scaling Group (ASG):** Show scaling policies, launch configuration/template, instance limits, and attached target groups
  - **Lambda Functions:** Show each function's trigger configuration (e.g., S3 event), environment variables, VPC, subnet and relevent settings if the function is placed inside a VPC.
  - Include brief overviews of the RDS instance and EC2 instance together with their respective VPC/subnet, security groups, secrets managers if relevant.

Submission Format:

- The video must be submitted in MP4 format.
- Ensure that AWS Console screens are clearly visible when demonstrating the infrastructure. Your federated user information must be shown at some point during the video. Begin with a brief introduction displaying your student ID card, as done in Assignment 1.
- Trimming is allowed to keep the video within the time limit. However, stitching clips recorded at different times is generally discouraged.

Application Codebase

Since the functionality of this application is very similar to that in Assignment 1, we will not provide a new codebase. You may choose to reuse the Assignment 1 code or develop a new codebase from scratch. The only new required feature is thumbnail generation, which is relatively simple to implement, and you can find sample code online. There is also no restriction on programming language; however, please note that support will be limited if you choose a language other than Python.

You are not required to follow the same layout used in Assignment 1, you are free to design your own webpage with a different layout. Similarly, there is no strict requirement for the database schema; you may design your own tables to support the required functionality.

This assignment primarily assesses your ability to design and deploy a cloud-based application architecture. While code functionality is not the main focus, a certain level of coding is required to integrate services, decouple functionality into components, and make the system work end-to-end. Your solution should demonstrate good architectural decisions and proper use of AWS services.