Amazon Web Services

AWS Cloud Foundations

Professional Training Program

AWS Training Agenda

Module 1: Cloud Basics & Identity

- What is Cloud Computing?
- Introduction to AWS
- Identity and Access Management (IAM)

Module 2: Core Compute Services

- Amazon EC2 (Elastic Compute Cloud)
- Amazon ECS (Elastic Container Service)
- AWS Lambda (Serverless Compute)

Module 3: Storage Solutions

- Amazon S3 (Simple Storage Service)
- Amazon EBS (Elastic Block Store)
- Amazon EFS (Elastic File System)

Module 4: Networking in the Cloud

- Amazon VPC (Virtual Private Cloud) & Subnets
- Security Groups, Gateways, and Endpoints

Module 5: Database Services

- Amazon RDS (Relational Database Service)
- Amazon DynamoDB (NoSQL Database)

Modules 6-7: Advanced Topics

- CloudWatch, CloudTrail, KMS & Secrets Manager
- API Gateway & Route 53
- Elastic Load Balancing
- Understanding Access from the FIL Network
- Conclusion & Key Takeaways

Module 1

Cloud Basics & Identity - What is Cloud Computing?

Cloud computing is the on-demand delivery of IT resources over the Internet with pay-as-you-go pricing.

On-demand

No upfront hardware investment.

Scalable

Easily scale resources up or down.

Global

Deploy applications in multiple regions around the world.

Secure

AWS provides a secure infrastructure.

What is Cloud Computing? ☁️

Core Definition: Cloud computing is the on-demand delivery of IT resources (like servers, databases, storage, and software) over the Internet with pay-as-you-go pricing.

💡 ***Think of it like this:*** *Instead of buying, owning, and maintaining your own physical servers, you can access these services from a cloud provider like Amazon Web Services (AWS), Google Cloud, or Microsoft Azure.*

Key Characteristics of the Cloud

⚡ On-demand

- Access resources instantly when you need them
- No upfront hardware investment

📈 Scalable

- Easily increase or decrease resources based on your needs
- Scale up for busy times, scale down for quiet times

🌍 Global

- Deploy your applications in data centres all around the world
- Serve your users faster & better

🔒 Secure

- Cloud providers offer highly secure infrastructure
- They handle physical security so you can focus on your application

Module 1

Introduction to AWS

Amazon Web Services (AWS) is the world's most comprehensive and broadly adopted cloud platform, offering over 200 fully featured services from data centers globally.

Global Infrastructure

Regions, Availability Zones (AZs), and Edge Locations.

Core Service Categories

Compute, Storage, Networking, Databases, etc.

Shared Responsibility Model

AWS manages the security **of** the cloud, while the customer is responsible for security **in** the cloud.

Introduction to AWS

The Big Picture

Amazon Web Services Overview for Beginners

What is AWS?

Amazon Web Services (AWS) is the world's leading cloud platform

→ Think of it like an electricity provider, but instead of power, it provides computing resources over the internet

→ Services include servers, storage, databases, and much more

→ Offers over 200+ services from data centers all over the globe

"Pay only for what you use - just like your electricity bill!"

🌎 Global Infrastructure

The Foundation of AWS

Regions

Geographic areas around the world (e.g., Mumbai, London, Ohio)

Availability Zones (AZs)

Multiple, isolated data centers within a single Region

→ *Ensures if one fails, others are still running*

Edge Locations

Smaller sites that deliver content to users with lower latency

→ *Content is cached closer to your users for faster access*

🧱 Core Service Categories

The Building Blocks

💻 Compute

The "brains" of the operation (e.g., EC2 for virtual servers)

💾 Storage

Where you keep your data (e.g., S3 for files and objects)

🔗 Networking

How your services connect to each other and the internet (e.g., VPC)

🗄️ Databases

Organized systems for storing and retrieving data (e.g., RDS)

🤝 Shared Responsibility Model

Who Secures What?

AWS's Responsibility

Security OF the Cloud

- ✓ Physical data centers
- ✓ Hardware infrastructure
- ✓ Global network security
- ✓ Virtualization layer

Your Responsibility

Security IN the Cloud

- ✓ Your data
- ✓ User access management
- ✓ Security configurations
- ✓ Application security

"Think of it like renting an apartment: The landlord secures the building, you secure your apartment!"

Key Takeaways

→ AWS is a cloud platform that provides computing resources on-demand

→ Global Infrastructure ensures reliability through Regions, AZs, and Edge Locations

→ Core services span Compute, Storage, Networking, and Databases

→ Security is a shared responsibility between AWS and customers

→ Pay only for what you use - scalable and cost-effective

Ready to start your AWS journey! 🚀

AWS

Questions and Answers

Check Your Understanding

Let's test your grasp of key AWS concepts through practical scenarios

Question 1

High Availability Architecture

Scenario

Imagine you are building a critical banking application and you want to make sure it stays online even if one data center completely fails. According to the AWS Global Infrastructure, where should you deploy your application?

Answer 1

Deploy Across Multiple Availability Zones

Solution

You should deploy it across **multiple Availability Zones (AZs)** within a single Region. This ensures that if one AZ goes down, the application can continue running in another isolated AZ.

Question 2

Shared Responsibility Model

Scenario

A developer in your team has stored sensitive customer data in an S3 bucket (which is an AWS storage service) but forgot to encrypt it and left it open to the public internet. According to the Shared Responsibility Model, who is responsible for this security mistake?

Answer 2

Customer Responsibility

Solution

The **customer (or your company)** is responsible. This falls under **"Security IN the cloud."** AWS provides the secure infrastructure (the S3 service), but the customer is responsible for configuring it correctly and securing the data they place within it.

Question 3

Core Service Categories

Task

Match the Core Service Category to its function. The categories are **Compute**, **Storage**, and **Database**.

A) A service for storing and retrieving user profile information in a structured table.

B) A service to run the code for your web application.

C) A service to store user-uploaded images and videos.

Answer 3

Service Category Matches

Solution

A) Database - Storing and retrieving structured user profile information

B) Compute - Running web application code

C) Storage - Storing user-uploaded images and videos

Module 1

Identity and Access Management (IAM)

IAM is the service you use to manage access to AWS services and resources securely.

Users

An entity that you create in AWS to represent the person or application that uses it to interact with AWS.

Groups

A collection of IAM users. You can use groups to specify permissions for multiple users at once.

Roles

An identity that you can assume to gain temporary access to permissions.

Policies

JSON documents that define permissions. They are attached to users, groups, or roles.

Pro-Tip:

Always follow the principle of least privilege. Grant only the permissions required to perform a task.

AWS IAM: The Security Guard of Your Cloud 💂

Identity and Access Management

A Comprehensive Guide for Freshers

What is IAM?

IAM stands for **Identity and Access Management**

It's the service that securely controls **who** (Identities) can do **what** (Access) in your AWS account.

> *Think of it as the bouncer at a club, checking IDs and deciding who gets in and where they can go.*

The Core Components of IAM

👤 Users

The 'Who'. An individual person or application that needs to interact with AWS.

Example: Developer 'Divya' or a deployment script

👥 Groups

A 'Team' or 'Department'. A collection of users for easy permission management.

Example: 'Developers' group or 'Testers' group

🎭 Roles

A 'Temporary Hat'. An identity with specific permissions that anyone can assume temporarily.

Example: 'Database-Admin-For-An-Hour' role

📜 Policies

The 'Rulebook'. A JSON document that explicitly lists the permissions.

Example: "Allow read access to specific S3 bucket"

The Golden Rule ⭐

Principle of Least Privilege

Always grant only the **minimum permissions necessary** to perform a task.

Don't give the keys to the whole building if someone just needs to enter one room!

Questions & Answers (1/2)

Q1: You have 10 new interns joining the data analysis team. What is the most efficient way to grant them access?

Ans - Create an IAM Group called 'Data-Analysts-Interns'. Create a Policy with read-only access. Attach the policy to the group. Create IAM Users for each intern and add them to the group.

Q2: What is the main difference between an IAM User and an IAM Role?

Ans - An IAM User is associated with a single identity and has permanent credentials. An IAM Role is not tied to a specific identity and is meant to be temporarily assumed by trusted entities.

Questions & Answers (2/2)

Q3: In what format are IAM Policies written?

Ans - IAM Policies are written in **JSON** (JavaScript Object Notation) format.

Q4: Can you explain the "Principle of Least Privilege" with an example?

Ans - It means giving only the required permissions. For example, if a web application only needs to read images from an S3 bucket, its IAM Role should only have s3:GetObject permission, not s3:DeleteObject or full s3:* access.

Hands-On Lab Overview

The Read-Only S3 User

Objective: Create a new IAM user who can only view S3 bucket contents but cannot create, modify, or delete anything.

What You Will Learn:

- How to create an IAM Group
- How to attach AWS Managed Policies
- How to create an IAM User and add them to a group
- How to verify permissions

Lab Part 1: Create the Group

Step 1-3: Navigate to IAM

Sign in to AWS Console → Search for "IAM" → Click on "User groups"

Step 4-5: Create Group

Click "Create group" → Enter name: **S3-ReadOnly-Group**

Step 6-8: Attach Policy

Search for "S3ReadOnly" → Select **AmazonS3ReadOnlyAccess** → Click "Create group"

Lab Part 2: Create the User

Step 1-3: Create User

Click "Users" → Click "Create user" → Enter name: **Test-ReadOnly-User**

Step 4-7: Set Console Access

Check "Provide user access to AWS Management Console" → Select "I want to create an IAM user" → Select "Autogenerated password" → Check "User must create a new password at next sign-in"

Step 8: Continue

Click "Next" to proceed to permissions

Lab Part 3: Add User to Group

Set Permissions Page

Select **"Add user to group"**

Select Group

Check the box next to **S3-ReadOnly-Group**

Review and Create

Click "Next" → Review details → Click "Create user"

Lab Part 4: Retrieve Credentials

Download Credentials

Click **"Download .csv file"** and save it securely

Sign In as New User

Open the CSV file → Copy the console sign-in link → Open in new browser/incognito window → Use username and password from CSV

Change Password

You'll be prompted to change password immediately → Enter old password → Create new password

Lab Part 5: Verify Permissions

Test Read Access (Should Work ✓)

Navigate to S3 service → You should see the list of S3 buckets → This confirms read-only access is working

Test Create Access (Should Fail ✗)

Click "Create bucket" → Try to create a bucket → You will receive **"Access Denied"** error

✓ Perfect! This proves the Principle of Least Privilege is working!

Lab Part 6: Clean Up

Step 1: Log Out

Log out of the Test-ReadOnly-User account

Step 2-3: Delete User

Log back in with admin account → Go to IAM → Users → Select Test-ReadOnly-User → Click "Delete"

Step 4: Delete Group

Go to User groups → Select S3-ReadOnly-Group → Click "Delete"

🎉 **Congratulations! You have successfully completed the lab!**

Summary

Today we learned about:

- IAM as the security guard of AWS
- Four core components: Users, Groups, Roles, and Policies
- The Principle of Least Privilege
- Hands-on experience with creating read-only access

> **Remember: Security is not a one-time setup, it's an ongoing practice!**

Module 2

Core Compute Services - Amazon EC2

EC2 provides scalable computing capacity in the AWS cloud. Think of it as a virtual server.

Amazon Machine Image (AMI)

A template that contains the software configuration (OS, application server, and applications).

Instance Types

Various combinations of CPU, memory, storage, and networking capacity.

Key Pairs

Used to connect to your instances securely.

Security Groups

A virtual firewall for your instances to control inbound and outbound traffic.

Introduction to Amazon EC2

Your Virtual Server in the Cloud

> **Key Idea:** EC2 lets you rent and manage virtual computers in the cloud without buying physical hardware.

What is EC2?

EC2 provides scalable computing capacity. Let's break that down:

Computing capacity means a computer's power—its processor, memory, and storage.

Scalable means you can easily increase or decrease this power on demand.

> The easiest way to think of an EC2 instance is as a **virtual server** or a computer that you rent in the AWS cloud. You don't have to buy any physical hardware; you just launch one with a few clicks.

Four Key Components

To launch a virtual computer, you need to define four key things:

1. Amazon Machine Image (AMI)

The blueprint for your server

2. Instance Type

The hardware specifications

3. Key Pairs

Secure keys to access the server

4. Security Groups

Firewall controlling traffic

AMI: The Blueprint for Your Server 

Think of it as a software template or an installation disc

It contains:

- **An Operating System:** Amazon Linux, Ubuntu, Windows Server
- **An Application Server:** Apache, Nginx
- **Applications:** Database, custom software suite
- You can choose from thousands of pre-configured AMIs or create your own

Instance Types: The Hardware 💻

Like choosing the specs for a new laptop (CPU, RAM, etc.)

It defines:

CPU
Number of virtual CPUs (vCPUs)

Memory
Amount of RAM (in GiB)

Storage
Type and speed of disk (e.g., SSD)

Networking Capacity
How fast it can talk to the internet

Example:

t2.micro is a popular choice for beginners as it's part of the AWS Free Tier

Key Pairs: Secure Keys 🔑

A digital key that is impossible to guess or forge

How it works:

- It uses **public-key cryptography**
- **Public Key:** Stored by AWS on your instance (the lock)
- **Private Key:** You download and keep it safe (the key)

> **Golden Rule:**
>
> If you lose your private key, you lose access to your server! Keep it safe.

Security Groups: The Bouncer

A virtual firewall controlling who can get in and who can get out

Key Features:

- **Inbound Rules:** Control incoming traffic to the instance (e.g., allow HTTP traffic from anywhere)
- **Outbound Rules:** Control outgoing traffic from the instance (e.g., allow the server to download software updates)
- **Stateful:** If you allow incoming traffic, the response is automatically allowed to go back out
- **Default Behavior:** By default, it denies all inbound traffic and allows all outbound traffic

Questions and Answers

In your own words, what is an EC2 instance?

Answer: It's a virtual server or computer that you can rent from AWS in the cloud.

If you want to launch a new EC2 instance with a pre-installed database, which of the four components would define that?

Answer: The AMI (Amazon Machine Image). You would choose an AMI that already has the database software included.

You have an application that needs a lot of memory. Which component would you focus on to meet this requirement?

Answer: The Instance Type. You would choose an instance type from a memory-optimised family.

What happens if you delete the .pem file you downloaded when creating your EC2 instance?

Answer: You will lose access to the instance via SSH. You cannot download the same private key again. (Advanced note: You would have to use more complex methods to try and regain access, but for a beginner, it's best to consider it lost).

Your web server is running on your EC2 instance, but no one can access it from the internet. What is the most likely component you need to check and fix?

Answer: The Security Group. You likely need to add an inbound rule to allow HTTP (port 80) and/or HTTPS (port 443) traffic from the internet (0.0.0.0/0).

Hands-On Exercise 2

Launch an EC2 Instance

🎯 Objective: Launch a basic web server on an EC2 instance.

Steps:

1. Navigate to the EC2 console.
2. Click "Launch instances".
3. Give your instance a name (e.g., My-Web-Server).
4. Choose an Amazon Machine Image (AMI), such as "Amazon Linux 2 AMI".
5. Choose an Instance Type, such as t2.micro (free tier eligible).
6. Create a new Key Pair and download it. Keep it safe!
7. In Network settings, create a security group that allows HTTP traffic from anywhere (Source: 0.0.0.0/0).
8. Expand "Advanced details" and in the "User data" field, paste the script below.
9. Click "Launch instance".

```
#!/bin/bash yum update -y yum install -y httpd systemctl start httpd systemctl enable httpd echo "<h1>Hello from your EC2 Instance!</h1>" > /var/www/html/index.html
```

✅ **Solution:** Once the instance is running, copy its Public IPv4 address and paste it into your web browser. You should see the "Hello" message.

Hands-on Lab

Objective: Launch a basic Amazon Linux web server, install a simple web page, and access it from your browser. This will use all four concepts we've discussed.

Prerequisites: An active AWS Account

Lab: Getting Started

Step 1: Navigate to the EC2 Dashboard

1. Log in to your AWS Management Console
2. In the search bar at the top, type "EC2" and select it
3. You will land on the EC2 Dashboard
4. Make sure you are in a region you want to work in (e.g., us-east-1 N. Virginia, ap-south-1 Mumbai)

Step 2: Launch a New EC2 Instance

On the dashboard, click the orange "Launch instance" button

Lab: Configure AMI & Instance Type

Step 3: Choose an AMI (The Blueprint)

1. You are now on the "Choose an Amazon Machine Image (AMI)" page
2. Select the **Amazon Linux 2 AMI** (free-tier eligible)
3. Click "Select"

Step 4: Choose an Instance Type (The Hardware)

1. Select the **t2.micro** type (Free tier eligible)
2. Click "Next: Configure Instance Details" (leave as default)
3. Click "Next: Add Storage" (leave as default)
4. Click "Next: Add Tags" (Optional: Add Tag, Key = Name, Value = My-First-Web-Server)
5. Click "Next: Configure Security Group"

Lab: Configure Security Group

Step 5: Configure the Security Group (Critical!)

1. Select "Create a new security group"
2. Security group name: **WebServer-SG**
3. Description: **Allows Web and SSH access**
4. You will see a default rule for SSH. Change "Source" from Anywhere to **My IP**
5. Click "Add Rule"
6. Change "Type" to **HTTP**
7. Port Range will automatically be set to **80**
8. Set "Source" to **Anywhere (0.0.0.0/0)**
9. Click "Review and Launch"

Lab: Connect & Install Web Server

Step 7: Connect to Your Instance

1. Go to EC2 Dashboard → Instances
2. Wait for "Status check" to show "2/2 checks passed"
3. Copy the **Public IPv4 address**
4. Open terminal and navigate to your .pem key folder

```
# Change permissions (Mac/Linux)
chmod 400 my-aws-key.pem

# Connect via SSH
ssh -i "my-aws-key.pem" ec2-user@YOUR_PUBLIC_IP
```

Lab: Install Apache & Test

Install Apache Web Server

```
sudo su
yum update -y
yum install -y httpd
systemctl start httpd
systemctl enable httpd
echo "<h1>Hello World from AWS EC2!</h1>" > /var/www/html/index.html
```

Step 8: View Your Website!

Open your browser and paste your **Public IPv4 address**. You should see "Hello World"! 🎉

Step 9: TERMINATE YOUR INSTANCE

Go to Instances → Select instance → Instance state → Terminate instance

EC2_inst_demo.txt

Module 2

ECS and Lambda

Amazon ECS (Elastic Container Service)

A highly scalable, high-performance container orchestration service that supports Docker containers.

AWS Lambda

A serverless compute service that lets you run code without provisioning or managing servers. You pay only for the compute time you consume.

Best Practice:

Use ECS for long-running containerized applications and Lambda for short-lived, event-driven functions.

AWS Compute Services

Choosing the Right Tool

Amazon ECS vs. AWS Lambda

📊 **Presenter:** Your Name

🗓️ **Date:** October 5, 2025

Meet the Services

Amazon ECS (Elastic Container Service)

- ✦ A highly scalable, high-performance service for running Docker containers.
- ✦ Think of it as a management service for your applications that are packaged in containers.
- ✦ You manage the underlying servers (EC2 instances) or use AWS Fargate for a serverless container experience.

AWS Lambda

- ✦ A serverless compute service that runs your code in response to events.
- ✦ You simply upload your code, and Lambda handles everything required to run and scale it.
- ✦ You don't think about servers at all!

An Analogy: Your Living Space 🏠

Amazon ECS is like Renting an Apartment

- You get a dedicated space that is always available.
- You can furnish it however you want and run multiple appliances (applications) 24/7.
- You are responsible for the space and pay a fixed rent (cost of running servers) whether you are home or not.

Best for: Long-running applications like web servers, APIs, or microservices that need to be always on.

AWS Lambda is like Using an On-Demand Service (e.g., a Taxi)

- You don't own the car; you just call for it when you need a ride (an event happens).
- It takes you from point A to point B (executes your function) and then it's gone.
- You only pay for the exact distance you traveled (the compute time you consume).

Best for: Short-lived, event-driven tasks like processing an image after it's uploaded, or reacting to a database change.

Key Differences at a Glance

Feature	Amazon ECS (with EC2)	AWS Lambda
Unit of Scale	Containers / Tasks	Events / Invocations
Management	You manage servers (patching, scaling)	No servers to manage (fully serverless)
Execution Time	Long-running (days, months)	Short-lived (max 15 minutes)
Billing Model	Pay for servers per hour/second	Pay per millisecond of execution time
Primary Use	Consistent, predictable workloads	Bursty, event-driven workloads

The Golden Rule (Best Practice)

⭐ Best Practice

- ✓ Use Amazon ECS for long-running, continuously operating applications (e.g., your main website backend).
- ✓ Use AWS Lambda for short-lived, event-driven functions (e.g., a quick task that runs when a user uploads a file).

Questions and Answers for Students

Let's Test Your Understanding

The following slides contain practical scenarios to help you understand when to use Amazon ECS vs. AWS Lambda.

Think about each question before moving to the answer!

Question 1

📷 **Scenario: A user uploads a photo to our app, and we need to resize it into a thumbnail immediately. Which service is a better fit for this task, ECS or Lambda, and why?**

Answer: AWS Lambda is the better fit. This is a classic event-driven, short-lived task. The function can be triggered by the file upload event, run for a few seconds to resize the image, and then shut down. You only pay for the exact time it takes to perform the resize operation.

Question 2

🛒 **Scenario: We are building the main API for our e-commerce website, which needs to be available 24/7 to handle customer traffic. Should we use ECS or Lambda?**

Answer: Amazon ECS is the more appropriate choice. An API for a website is a long-running application that needs to be constantly available to serve requests. ECS is designed for these types of consistent, always-on workloads.

Question 3

- **Concept: What does the term "serverless" mean in the context of AWS Lambda?**

Answer: "Serverless" doesn't mean there are no servers. It means that you, as the developer, do not have to provision, manage, or think about the underlying servers. AWS handles all the infrastructure management, including patching, scaling, and maintenance, allowing you to focus only on writing your code.

Hands-On Lab: Event-Driven Thumbnail Generation

🎯 Objective

To build a simple pipeline where uploading an image to an S3 bucket automatically triggers a Lambda function to create a thumbnail and save it in another S3 bucket. This demonstrates a real-world, event-driven use case for AWS Lambda.

📋 Prerequisites

- An AWS Account with administrative access
- Access to the AWS Management Console

Lab Step 1: Create Two S3 Buckets

📦 Create Source and Thumbnail Buckets

First, we need one bucket for source images and another for the generated thumbnails.

1. Navigate to the S3 service in the AWS Console.
2. Click **Create bucket**.
3. Enter a globally unique bucket name for your source images, for example, `yourname-source-images-123`.
4. Choose an AWS Region (e.g., us-east-1).
5. Keep all other default settings and click **Create bucket**.
6. Repeat the process to create a second bucket for thumbnails, for example, `yourname-thumbnails-123`.

Lab Step 2: Create an IAM Role for Lambda

🔐 Set Up Permissions

Our Lambda function needs permission to read from the source bucket and write to the thumbnails bucket.

1. Navigate to the **IAM** service.
2. Go to **Roles** in the left-hand menu and click **Create role**.
3. For "Trusted entity type," select **AWS service**.
4. For "Use case," choose **Lambda** and click **Next**.
5. In the "Add permissions" search bar, search for and select the policy named **AWSLambda_FullAccess** (for this lab; in production, you'd create a more restrictive policy). Click **Next**.
6. Give the role a name, like `LambdaS3ThumbnailRole`, and click **Create role**.

Lab Step 3: Create the Lambda Function

Build Your Function

Now we'll create the function that performs the image resizing.

1. Navigate to the **Lambda** service in the AWS Console.
2. Click **Create function**.
3. Select **Author from scratch**.
4. Enter a Function name, like `ThumbnailGenerator`.
5. For Runtime, select **Python 3.9** (or a newer Python version).
6. For Architecture, keep **x86_64**.
7. Under Permissions, expand "Change default execution role."
8. Select **Use an existing role** and choose the **LambdaS3ThumbnailRole** you created in the previous step.
9. Click **Create function**.

Lab Step 4: Add Code and Helper Layer (Part 1)

Add the Pillow Library Layer

The default Lambda runtime doesn't include image processing libraries. We'll use a public AWS Layer that contains the Pillow library.

1. In your function's configuration page, scroll down to the **Layers** section and click **Add a layer**.
2. Select **AWS Layers**.
3. From the dropdown, choose the layer named **AWSLambda-Python-Pillow** that matches your region and Python version. Select the latest version and click **Add**.

Lab Step 4: Add Code and Helper Layer (Part 2)

💻 Add the Python Code

Now, go back to the **Code source** section. Replace the existing code in **lambda_function.py** with the following:

```
import boto3 import os from urllib.parse import unquote_plus from PIL import Image import uuid s3_client = boto3.client('s3') def resize_image(image_path, resized_path): with Image.open(image_path) as image: image.thumbnail((128, 128)) image.save(resized_path) def lambda_handler(event, context): for record in event['Records']: source_bucket = record['s3']['bucket']['name'] key = unquote_plus(record['s3']['object']['key']) dest_bucket = os.environ['DEST_BUCKET'] download_path = f'/tmp/{uuid.uuid4()}{key}' upload_path = f'/tmp/resized-{key}' s3_client.download_file(source_bucket, key, download_path) resize_image(download_path, upload_path) s3_client.upload_file(upload_path, dest_bucket, f'resized-{key}') return {'statusCode': 200, 'body': 'Thumbnail generation successful!'}
```

Click the **Deploy** button to save your code.

lambda_example.txt

Lab Step 5: Configure Environment Variables

⚙️ Set Up Configuration

Environment Variables:

1. In your function's page, go to the **Configuration** tab and then **Environment variables**.
2. Click **Edit**, then **Add environment variable**.
3. For Key, enter `DEST_BUCKET`.
4. For Value, enter the name of your thumbnail bucket (e.g., yourname-thumbnails-123).
5. Click **Save**.

General Configuration:

1. Go to the **General configuration** tab, click **Edit**.
2. Set the Timeout to **30 seconds** to ensure the function has enough time to run. Click **Save**.

Lab Step 6: Add the S3 Trigger

🔗 Connect S3 to Lambda

Finally, let's connect the S3 bucket to the Lambda function.

1. In your function's page, go to the **Function overview** section at the top.
2. Click **+ Add trigger**.
3. In the "Select a source" dropdown, choose **S3**.
4. For Bucket, select your source bucket (`yourname-source-images-123`).
5. Ensure Event type is set to **All object create events**.
6. Acknowledge the recursive invocation warning by checking the box.
7. Click **Add**.

Lab Step 7: Test the Pipeline!

🎉 See Your Serverless Solution in Action

1. Navigate back to the **S3 service**.
2. Go into your source bucket (`yourname-source-images-123`).
3. Click **Upload** and add any **.jpg** or **.png** image from your computer.
4. Once the upload is complete, navigate to your thumbnails bucket (`yourname-thumbnails-123`).
5. After a few seconds, you should see a new file appear named **resized-your-image-name.jpg**.
6. Download and open it - it will be a 128x128 thumbnail of your original image!

🎊 Congratulations!

You have successfully built an event-driven serverless pipeline using AWS Lambda and S3.

Key Takeaways from Today's Session

🎯 Amazon ECS

Best for long-running, always-on applications like web servers, APIs, and microservices. You have more control but also more responsibility for infrastructure management.

⚡ AWS Lambda

Perfect for short-lived, event-driven tasks. Fully serverless - you pay only for what you use, and AWS handles all infrastructure management.

🤝 They Work Together

Modern applications often use both services together - ECS for core services and Lambda for event processing and automation.

Thank You! 🙏

Questions?

Keep Learning!

📚 AWS Documentation: docs.aws.amazon.com

💻 Practice: AWS Free Tier

🎓 Next Steps: Explore AWS SAM, Step Functions, and more!

Module 3
Storage Solutions

Amazon S3 (Simple Storage Service)

Object storage is built to store and retrieve any amount of data from anywhere.

- **Buckets:** Containers for objects (files). Bucket names must be globally unique.
- **Objects:** The fundamental entities stored in S3.
- **Storage Classes:** Different tiers for different use cases (e.g., S3 Standard, S3 Glacier).

Amazon EBS & EFS

Amazon EBS (Elastic Block Store): High-performance block storage volumes for use with Amazon EC2. Think of it as a virtual hard drive.

Amazon EFS (Elastic File System): A simple, scalable, elastic file system for Linux-based workloads for use with AWS Cloud services and on-premises resources.

AWS Storage Fundamentals

A Complete Guide for Freshers

Understanding S3, EBS, and EFS

AWS Storage Fundamentals

Where Does Your Data Live?

What is Cloud Storage?

Think of it as a massive, virtual warehouse for your digital stuff. Instead of saving files on your own computer's hard drive, you save them on the internet, accessible from anywhere.

Why AWS Storage?

- **Durability:** Keeps your data safe from being lost
- **Availability:** Ensures you can access your data whenever you need it
- **Scalability:** Store as little or as much as you want and only pay for what you use

The Three Musketeers of AWS Storage

Amazon S3 (Simple Storage Service)

The Infinite Locker

A storage service for objects (files, images, videos)

Amazon EBS (Elastic Block Store)

The Virtual Hard Drive

High-performance storage block that attaches to a single EC2 instance

Amazon EFS (Elastic File System)

The Shared Network Drive

Scalable, shared file storage system for multiple servers

Amazon S3

Simple Storage Service - The Infinite Locker

What it is

A storage service for objects (files, images, videos)

> **Analogy:** A limitless digital filing cabinet or a valet parking service for your data. You give your data a unique key (name), and S3 stores it for you. You can retrieve it anytime with that key.

Key Concepts

- **Buckets:** Like folders or containers where you store your objects. Bucket names must be globally unique!
- **Objects:** The actual files you store

Amazon S3

Use Cases

- Website hosting
- Data backups
- Application data storage
- Big data analytics

Amazon EBS

Elastic Block Store - The Virtual Hard Drive

What it is

A high-performance storage block that attaches to a single EC2 instance (a virtual server)

> **Analogy:** It's like plugging an external USB hard drive into your computer. It provides the C: drive or D: drive for your virtual server.

Key Features

- Tied to one EC2 instance in a specific Availability Zone
- Perfect for running operating systems, databases, and applications that need fast access to data

Amazon EFS

Elastic File System - The Shared Network Drive

What it is

A scalable, shared file storage system

> **Analogy:** It's like a shared network folder (like a Google Drive folder) that multiple computers can access and modify at the same time.

Key Features

- Can be accessed by multiple EC2 instances simultaneously
- Automatically grows and shrinks as you add/remove files

Amazon EFS

Use Cases

- Content management systems
- Shared code repositories
- Applications that need a common file system

Quick Recap

Choosing the Right Storage Service

S3: Valet Parking

For files/objects, accessible from anywhere

EBS: External Hard Drive

A dedicated hard drive for one server

EFS: Shared Network Drive

Questions & Answers

Test Your Understanding

Question 1

I have a web application that needs to be run on 10 different virtual servers (EC2 instances). All servers need to access and modify the same set of user-uploaded images. Which storage service should I use?

Answer: You should use Amazon EFS. EFS allows multiple EC2 instances to connect to and share the same file system simultaneously.

Question 2

You are trying to create an S3 bucket named "test". AWS gives you an error. What is the most likely reason?

Answer: S3 bucket names must be globally unique. Someone else in the world has already created a bucket named "test". You need to choose a more specific name.

Question 3

I need to install the Windows Operating System for my EC2 instance. Which AWS storage service would be the primary choice for the boot drive (C: drive)?

Answer: Amazon EBS. EBS volumes act as the boot volumes or virtual hard drives for EC2 instances.

Question 4

What is the main difference between "object storage" (S3) and "block storage" (EBS)?

Answer: Block storage (EBS) is like a raw hard drive that you format and manage; it's accessed by the operating system of a single server. Object storage (S3) stores data as self-contained objects (file + metadata) and is accessed via APIs over the internet, not as a local drive.

Question 5

Can a single EBS volume be attached to multiple EC2 instances at the same time?

Answer: No. A standard EBS volume can only be attached to one EC2 instance in the same Availability Zone.

(Note for advanced students: Multi-Attach EBS exists but has specific use cases and limitations)

Hands-On Lab

AWS Storage Exploration

Get practical experience with S3, EBS, and EFS

Hands-On Lab

Objective & Prerequisites

Objective

To get practical experience with S3, EBS, and EFS by creating and interacting with each service.

Prerequisites

An AWS Account with IAM user permissions to create:

- S3 buckets
- EC2 instances
- EBS volumes
- EFS file systems

Part 1: Amazon S3

Your First Bucket

1. Sign in to the AWS Console
2. Navigate to the S3 service
3. Click "Create bucket"
4. Bucket name: Enter a globally unique name
   (e.g., fil-fresher-lab-storage-yourname-randomnumber)
5. AWS Region: Choose a region near you (e.g., us-east-1)

Part 1: Amazon S3

Upload Your First Object

6. Leave all other settings as default and click "Create bucket"
7. Once created, click on your bucket name
8. Click "Upload", then "Add files". Choose any small text file or image from your computer
9. Click "Upload" at the bottom

✓ Success! You have just stored your first object in the cloud using S3.

Part 2: EC2 & EBS

Launching a Virtual Server with a Hard Drive

1. Navigate to the EC2 service
2. Click "Launch instances"
3. Name: Give it a name like Storage-Lab-Server
4. Application and OS Images (AMI): Select "Amazon Linux 2 AMI" (free tier eligible)
5. Instance type: Select t2.micro (free tier eligible)
6. Key pair (login): Proceed without a key pair for this lab
7. Network settings: Ensure a VPC and public subnet are selected. Make sure "Auto-assign public IP" is Enabled. Create a new security group named lab-sg and allow SSH traffic from "My IP"
8. Configure storage: This is where EBS comes in! You will see it's already configured with an 8 GiB gp2 EBS volume. This is the root volume (the C: drive). Leave this as is

Part 2: EC2 & EBS

Launch and Connect

9. Click "Launch instance"
10. Wait for the "Instance state" to change from Pending to Running
11. Select the instance, and click the "Connect" button at the top
12. Choose the "EC2 Instance Connect" tab and click "Connect". A new browser tab with a command line terminal will open

Part 3: Amazon EFS

The Shared Drive

1. In a new browser tab, navigate to the EFS service
2. Click "Create file system"
3. Name: Call it my-shared-drive
4. VPC: Select the same VPC that your EC2 instance is in (usually the only one, marked as 'default')
5. Click "Create"
6. Wait for the "Life cycle state" to become Available
7. Click on the file system's name to open its details page
8. Click the "Attach" button in the top right. You will see connection instructions
9. Copy the command under "using the EFS mount helper". It will look like:

```
sudo mount -t efs -o tls fs-xxxxxxxx:/ efs
```

Part 3: Amazon EFS

Mounting the Shared Drive

10. Go back to your EC2 Instance Connect terminal tab from Part 2
11. First, create a directory to mount the EFS drive to. Type:

```
sudo mkdir efs
```

12. Now, paste the command you copied from the EFS console and press Enter
13. If it gives a "command not found" error, install the EFS helper first:

```
sudo yum install -y amazon-efs-utils
```

Part 3: Amazon EFS

Verify and Test

14. You won't see any output if it's successful. To verify, create a file in the new shared drive:

```
sudo touch efs/hello_from_efs.txt
```

15. List the files to see if it was created:

```
ls efs/
```

16. You should see hello_from_efs.txt

✓ Success! You have attached a shared network drive to your EC2 instance.

Part 4: Cleanup

Very Important!

> ⚠️ To avoid any AWS charges, you must delete the resources you created

Follow the cleanup steps carefully to ensure all resources are properly deleted and you don't incur unexpected charges.

Part 4: Cleanup

Delete All Resources

1. Terminate EC2 Instance

Go to the EC2 Dashboard, select your instance, click "Instance state", and then "Terminate instance". Terminating the instance will also automatically delete the attached EBS root volume.

2. Delete EFS File System

Go to the EFS service, select your file system, click "Actions", and then "Delete".

3. Delete S3 Bucket

Go to the S3 service. First, you must empty the bucket. Select your bucket, select the file inside, click "Actions", then "Delete". After emptying it, go back to the bucket list, select the bucket, and click "Delete".

Thank You!

Questions?

You've learned the fundamentals of AWS Storage
S3 | EBS | EFS