UNIVERSITAT ROVIRA i VIRGILI

AWS Academy

Distributed Systems

Introduction to AWS S3

- **What is AWS S3?**
  - Amazon Simple Storage Service (S3) is a highly scalable, durable, and secure object storage service.
  - It is used to store and retrieve any amount of data at any time.
- **Key Features:**
  - **Scalability:** Automatically scales storage capacity.
  - **Durability & Availability:** Designed for 99.999999999% (11 9's) durability.
  - **Security:** Supports encryption, access management, and compliance.
  - **Cost-Effective:** Pay only for the storage you use.
- **Use Cases:**
  - Data backup and archiving.
  - Hosting static websites.
  - Storing large amounts of unstructured data.

Introduction to AWS Lambda

- **What is AWS Lambda?**
  - AWS Lambda is a serverless compute service that allows you to run code without provisioning or managing servers.
  - It executes code in response to events, such as S3 uploads, API Gateway calls, or scheduled events.
- **Key Benefits:**
  - **No Server Management:** Automatically scales with demand.
  - **Event-Driven:** Code is triggered by specific events.
  - **Cost Efficiency:** You pay only for the compute time you consume.
  - **Flexible:** Supports multiple programming languages (e.g., Python, Node.js, Java).
- **Common Use Cases:**
  - Real-time file processing (e.g., processing CSV uploads in S3).
  - Backend for mobile or web applications.
  - Automation of operational tasks.

Introduction to AWS EC2

- **What is AWS EC2?**
  - Amazon Elastic Compute Cloud (EC2) provides scalable virtual servers in the cloud.
  - It gives you full control of your computing resources and allows you to run applications on a virtual machine.
- **Key Features:**
  - **Flexible Instance Types:** Choose from a variety of instance types optimized for different workloads.
  - **Scalability:** Scale up or down based on demand.
  - **Control:** Full access to the operating system, storage, and network.
  - **Integration:** Easily integrates with other AWS services like S3, RDS, and more.
- **Common Use Cases:**
  - Hosting web applications and microservices.
  - Running batch processing jobs.
  - Deploying legacy applications that require complete OS control.

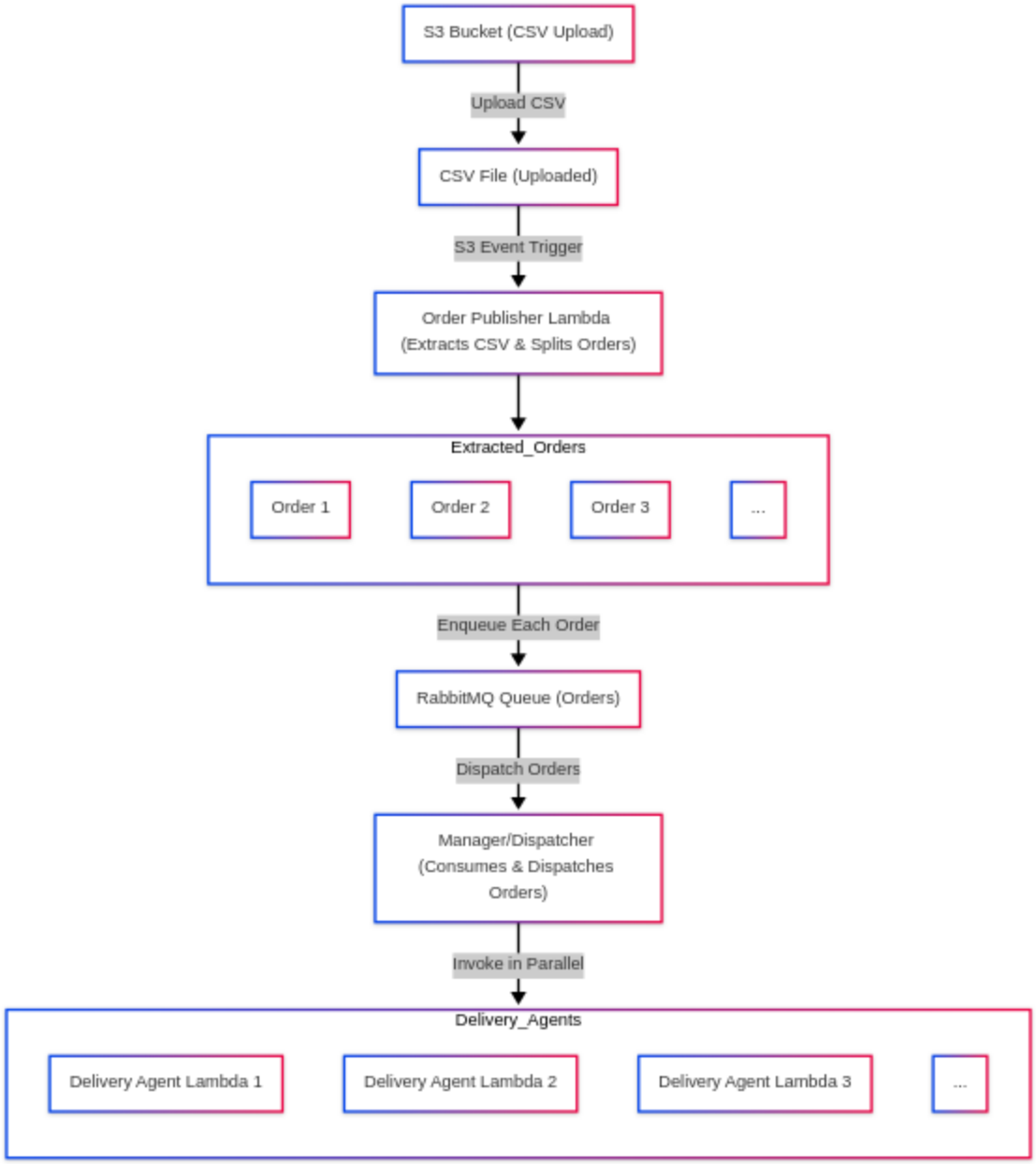
Conceptual Overview

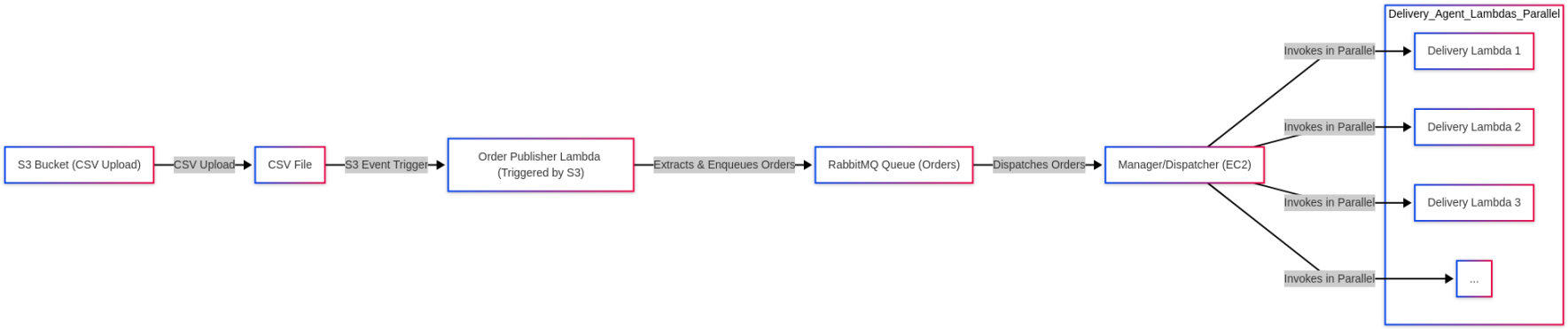
Distributed Order Processing System

- Objective
  - Simulate the operation of an order processing platform that receives customer orders, processes them, and assigns each order to a delivery agent. This system mimics real-world scenarios where orders must be handled concurrently by a fleet of couriers.

System Overview

- Build a distributed order processing system.
- The system starts by **receiving orders** via a CSV file uploaded to an **S3 bucket**.
- The uploaded CSV is processed by a Lambda function, which extracts each order and enqueues them in a **RabbitMQ queue**.
- A **Manager component** (or dispatcher) then takes the orders from the queue and **assigns each order to a different delivery agent**.
- Each delivery agent (simulated by a separate **Lambda function**) processes its order in parallel, mimicking concurrent delivery.


S3 Bucket (CSV Upload)
Upload CSV
CSV File (Uploaded)
S3 Event Trigger
Order Publisher Lambda
(Extracts CSV & Splits Orders)
Extracted_Orders
Order 1
Order 2
Order 3
...
Enqueue Each Order
RabbitMQ Queue (Orders)
Dispatch Orders
Manager/Dispatcher
(Consumes & Dispatches
Orders)
Invoke in Parallel
Delivery_Agents
Delivery Agent Lambda 1
Delivery Agent Lambda 2
Delivery Agent Lambda 3
...


S3 Bucket (CSV Upload)
CSV Upload
CSV File
S3 Event Trigger
Order Publisher Lambda
(Triggered by S3)
Extracts & Enqueues Orders
RabbitMQ Queue (Orders)
Dispatches Orders
Manager/Dispatcher (EC2)
Invokes in Parallel
Invokes in Parallel
Invokes in Parallel
Invokes in Parallel
Delivery_Agent_Lambdas_Parallel
Delivery Lambda 1
Delivery Lambda 2
Delivery Lambda 3
...

Key Steps

- **CSV Upload:** Orders are submitted in a CSV file.
- **Order Enqueueing:** A Lambda function reads the CSV, parses orders, and publishes them to a RabbitMQ queue.
- **Order Distribution:** A Manager component (running on an EC2 instance) subscribes to the queue and invokes a Delivery Simulator Lambda for each order.
- **Parallel Delivery:** Each invoked Lambda simulates order delivery concurrently.

Lab and Environment Setup

- Register on AWS Academy and log in to the LMS.
- Navigate to **Courses** → **All Courses** and open **AWS Academy Learner Lab**.
- In the module, click **Launch AWS Academy Learner Lab**.
- Start the lab with **Start Lab** and click the AWS icon once it turns green.

Creating an S3 Bucket

- **Task:**
  - Create a new S3 bucket in your AWS environment.
- **Purpose:**
  - This bucket will trigger the Order Publisher Lambda function when you upload the CSV file.
- **Resource:**
  - Creating an S3 Bucket Guide

Order Publisher Lambda Function

- **Objective:**
  - Write a Python Lambda function (named **lambda_order_publisher**) that triggers on S3 events.
  - This function reads the uploaded CSV, extracts orders, and publishes them to a RabbitMQ queue.
- **Key Code Components:**
  - **Extract Event Data:** Read the bucket name and CSV file key from the S3 event.
  - **Read and Parse CSV:** Use boto3 to get the file, then `csv.DictReader` to parse it.
  - **Publish Orders:** Use the pika library to connect to RabbitMQ and send each order as a JSON message.
- **Tip:**
  - First test the Lambda without RabbitMQ code—just log the orders. Then add the RabbitMQ connection.

Order Publisher Lambda Function

- Important – Add Required Python Packages Using a Lambda Layer:
  - Go to the **Lambda Dashboard** in AWS.
  - Create a **new Lambda Layer**.
  - **Upload the file:** `layer_contents.zip`
    (This ZIP file is already prepared and includes the necessary dependencies: `pika` and `boto3`, located inside a `python/` folder.)
  - After creating the layer, **copy its ARN**.
  - Go to your **`lambda_order_publisher`** function.
  - In the "Layers" section, click **“Add a layer”**.
  - Select **"Provide a layer version ARN"** and **paste the ARN** you copied.
    - **Note:** For detailed steps on how to create a Lambda layer, **refer to the resources on the next slide**.

Order Publisher Lambda Function

- **Important Notes:**
  - Make sure to configure your Lambda function's timeout to a sufficient value (for example, 30 seconds) in the AWS Lambda configuration. This ensures that the function has enough time to retrieve the file from S3, parse the CSV, establish a connection to RabbitMQ, and publish all the orders. A too-short timeout might cause your function to terminate before completing these tasks, leading to errors or incomplete processing.
  - Remember to redeploy your Lambda function after each modification.
- **Resources:**
  - Packaging your layer content
  - Boto3 S3 Documentation
  - Python CSV Module
  - Pika Documentation

Deploying RabbitMQ on EC2

- **Task:**
  - Launch a t2.micro EC2 instance to host RabbitMQ.
- **Key Steps:**
  - **Launch Instance:**
    - Choose Debian
    - Select t2.micro
    - Create a key pair, and assign an IAM instance profile (LabInstanceProfile).
  - **Configure Security Group:**
    - Allow traffic on ports 5672 (AMQP).
  - **Connect via SSH:**
    - Example command:
      - `ssh -i "your-key.pem" ec2-user@<EC2_PUBLIC_IP>`
      -

Deploying RabbitMQ on EC2

- **Install RabbitMQ and Dependencies**
  - Run the commands from the `ec2_steps.txt` file **on the EC2 instance you created**.
- **Resources:**
  - Get started with AWS EC2
  - RabbitMQ Installation Guide

Testing the Order Publisher Lambda

- **Action:**
  - With RabbitMQ running, update your Lambda to include the RabbitMQ connection code.
- **Verification:**
  - Re-upload the CSV file and check the RabbitMQ management console to see messages in the queue.
  -

Delivery Simulator Lambda Function

- **Objective:**
  - Create a Lambda function (named **lambda_delivery**) that simulates order delivery.
- **Key Code Elements:**
  - **Extract Order:** Retrieve order details from the event.
  - **Simulate Delay:** Use `random.randint(2, 10)` and `time.sleep(delay)` to simulate a delay.
  - **Return Success:** Log the delivery and return a confirmation message.
- **Important Note on Lambda Timeout:**
  - Make sure to configure your Lambda function's timeout to a sufficient value (for example, 15 seconds) in the AWS Lambda configuration.
- **Resources:**
  - Python time.sleep()
  - Random Module Documentation

Building the Subscriber (Manager) on EC2

- **Objective:**
  - Develop a Python client on an EC2 instance that subscribes to the RabbitMQ queue.
  - For each received order, the Manager invokes the Delivery Simulator Lambda asynchronously.
- **Key Code Components:**
  - **Connect to RabbitMQ:** Use pika to subscribe to the queue.
  - **Lambda Invocation:** Use boto3 to call **lambda_delivery** with `InvocationType='Event'` (specify region as us-east-1).
  - **Message Acknowledgement:** Ensure each message is acknowledged after processing.
- **Resources:**
  - Boto3 Lambda Client
  - Pika Consumption Example

Lab Summary and Final Verification

- **Process Recap:**
  - Upload CSV file with orders to S3.
  - Lambda (Order Publisher) processes the CSV and enqueues orders in RabbitMQ.
  - A Manager (subscriber) running on EC2 reads orders from RabbitMQ.
  - The Manager invokes the Delivery Simulator Lambda for each order.
  - Each Delivery Lambda simulates order delivery in parallel.
- **Verification:**
  - Monitor Lambda logs and RabbitMQ management console.
  - Confirm that each order is processed and delivered concurrently.

Explore AWS SQS as an Alternative

- **Why AWS SQS?**
  - Fully managed, serverless messaging service.
  - No need to manage EC2 instances or clusters.
  - Integrates seamlessly with AWS Lambda.
- **Advantages Over RabbitMQ:**
  - Simplified operations & lower maintenance.
  - Automatically scales with demand.
  - Cost-effective—pay per use.
- **Your Challenge:**
  - Research AWS SQS documentation.
  - Consider how you'd adapt your messaging code to use SQS instead of RabbitMQ.
  - Explore SQS to enhance your lab experience and reduce operational overhead!

Additional Resources and Useful Commands

- **AWS Documentation:**
  - AWS Lambda Developer Guide
  - AWS S3 User Guide
  - AWS EC2 User Guide

UNIVERSITAT
ROVIRA i VIRGILI