Detailed Step-by-Step Guide for Implementing the Cloud-Based Retail Inventory Management System on AWS Console

This guide provides a detailed step-by-step walkthrough for implementing the **Cloud-Based Retail Inventory Management System** on the AWS Console. The steps are organized to align with the project objectives and scope, ensuring a structured and efficient implementation process.

Step 1: Set Up AWS Infrastructure

1.1 Create an AWS Account

- If you don't already have an AWS account, sign up at AWS.
- Ensure you have the necessary permissions to create and manage resources.

1.2 Set Up IAM Roles and Policies

- Go to the **IAM (Identity and Access Management)** console.
- Create a new IAM role for EC2 instances with permissions for S3, RDS, CloudWatch, and Lambda.
- Attach the following managed policies:
  - **AmazonS3FullAccess**
  - **AmazonRDSFullAccess**
  - **AmazonCloudWatchFullAccess**
  - **AWSLambda_FullAccess**
-
- Create an IAM user for administrative access and generate access keys for programmatic access.

1.3 Create a Virtual Private Cloud (VPC)

- Go to the **VPC Dashboard**.
- Click **Create VPC** and configure:
  - **VPC Name**: Inventory-Management-VPC
  - **IPv4 CIDR Block**: 10.0.0.0/16
-
- Create **Subnets**:
  - Two public subnets (e.g., 10.0.1.0/24 and 10.0.2.0/24) for web servers.
  - Two private subnets (e.g., 10.0.3.0/24 and 10.0.4.0/24) for RDS and Lambda functions.
-
- Create an **Internet Gateway** and attach it to the VPC.

- Update **Route Tables** to allow internet access for public subnets.

Step 2: Set Up EC2 Instances

2.1 Launch EC2 Instances

- Go to the **EC2 Dashboard**.
- Click **Launch Instance** and select an Amazon Linux 2 AMI.
- Choose an instance type (e.g., t2.micro for testing).
- Configure instance details:
  - Select the VPC and public subnets created earlier.
  - Assign the IAM role created in Step 1.2.
-
- Add storage (default 8GB is sufficient for testing).
- Configure security groups:
  - Allow SSH (port 22) from your IP.
  - Allow HTTP (port 80) for web access.
-
- Launch the instance and download the key pair for SSH access.

2.2 Install Required Software

- SSH into the EC2 instance using the key pair.

Update the system:
```
sudo yum update -y
```

-
  content_copy download
  Use code with caution.Bash

Install Python and required libraries:
```
sudo yum install python3 -y
pip3 install boto3 pandas flask
```

-
  content_copy download
  Use code with caution.Bash
- Install a web server (e.g., Nginx or Apache) if needed for hosting dashboards.

Step 3: Set Up S3 for Data Storage

3.1 Create S3 Buckets

- Go to the **S3 Dashboard**.
- Click **Create Bucket**:
    - **Bucket Name**: inventory-management-data
    - **Region**: Select your preferred region.
    - Enable **Versioning** and **Server-Side Encryption** (SSE-S3).
-
- Create another bucket for backups: inventory-management-backups.

3.2 Upload Sample Data

- Upload sample inventory and sales data (CSV files) to the inventory-management-data bucket.
- Use the AWS CLI or S3 Console to upload files.

Step 4: Set Up RDS for Database

4.1 Create an RDS Instance

- Go to the **RDS Dashboard**.
- Click **Create Database**:
    - Choose **MySQL** or **PostgreSQL** as the engine.
    - Select the **Free Tier** template for testing.
    - Configure database settings:
        - **DB Instance Identifier**: inventory-db
        - **Master Username**: admin
        - **Master Password**: Set a strong password.
    -
    - Configure network settings:
        - Select the VPC and private subnets created earlier.
        - Ensure the database is not publicly accessible.
    -
    - Enable **Automated Backups** and **Encryption**.
-
- Click **Create Database**.

4.2 Connect to the Database

- Once the database is available, note the **Endpoint** and **Port**.
- Use a MySQL client (e.g., MySQL Workbench) or Python script to connect to the database.
- Create tables for **Products**, **Inventory**, **Sales**, and **Users**.

Step 5: Set Up Lambda Functions

5.1 Create Lambda Functions

- Go to the **Lambda Dashboard**.
- Click **Create Function**:
  - **Function Name**: inventory-update
  - **Runtime**: Python 3.9
  - **Execution Role**: Use the IAM role created earlier.
-
- Write a Python script to update inventory levels based on sales data.
- Repeat the process to create functions for:
  - sales-reporting: Generate sales reports.
  - low-stock-alerts: Send alerts for low stock levels.
-

5.2 Trigger Lambda Functions

- Set up triggers for Lambda functions:
  - Use **S3 Event Notifications** to trigger inventory-update when new sales data is uploaded.
  - Use **CloudWatch Events** to schedule sales-reporting and low-stock-alerts functions.
-

Step 6: Set Up CloudWatch for Monitoring

6.1 Create CloudWatch Alarms

- Go to the **CloudWatch Dashboard**.
- Create alarms for:
  - EC2 CPU utilization.
  - RDS storage and CPU usage.
  - Lambda function errors.
-
- Set up notifications to send alerts via **SNS (Simple Notification Service)**.

6.2 Set Up CloudWatch Logs

- Enable logging for EC2 instances, RDS, and Lambda functions.
- Use CloudWatch Logs Insights to analyze logs and troubleshoot issues.

Step 7: Set Up Dashboards and Reporting

7.1 Create Dashboards with QuickSight

- Go to the **QuickSight Dashboard**.
- Connect QuickSight to your RDS database and S3 buckets.
- Create dashboards for:
    - Real-time inventory tracking.
    - Sales trends and analytics.
-

7.2 Host Dashboards on a Web Interface

- Use **Flask** or **Django** to create a simple web application.
- Host the application on the EC2 instance.
- Use **Nginx** or **Apache** to serve the web application.

Step 8: Test and Optimize

8.1 Perform Unit and Integration Testing

- Test individual components (e.g., Lambda functions, database queries).
- Perform integration testing to ensure seamless data flow between components.

8.2 Optimize Costs

- Use **AWS Cost Explorer** to monitor costs.
- Right-size EC2 instances and RDS configurations.
- Use **Reserved Instances** or **Savings Plans** for long-term cost savings.

Step 9: Documentation and Delivery

9.1 Document the Architecture

- Create a detailed architecture diagram using **AWS Architecture Icons**.
- Document the setup process, including IAM roles, security groups, and network configurations.

9.2 Prepare User Manual

- Write a user manual for retail staff to use the inventory management system.
- Include instructions for accessing dashboards, uploading data, and generating reports.

Conclusion

By following this step-by-step guide, you will successfully implement a **Cloud-Based Retail Inventory Management System** on AWS. This system will provide real-time inventory tracking, automated stock replenishment, and detailed sales analytics, leveraging Yanga Mgudwa's AWS cloud skills, Python scripting, and Linux knowledge. The system is designed to be scalable, secure, and cost-effective, making it a valuable tool for retail businesses.
write a duplicate