Project Report

Title: EC2 Auto-Start and Stop Schedule using IAM & Bash

1. Introduction

Amazon Web Services (AWS) provides on-demand cloud computing resources such as EC2 instances, which allow users to deploy and manage virtual servers. However, keeping EC2 instances running continuously can lead to unnecessary billing costs.
This project automates the process of starting and stopping EC2 instances at scheduled times using Bash scripts, IAM roles, and cron jobs.
By implementing this automation, users can ensure that their EC2 instances are active only during required hours, thereby optimizing costs and minimizing manual effort.

2. Necessity

In organizations or educational environments, EC2 instances are often used for development, testing, or training purposes during specific working hours.
If these instances remain running 24/7, they consume compute hours and generate significant unnecessary expenses.
Hence, an automation system that starts instances when required and stops them after hours is necessary to:

- Save cloud resource costs.
- Reduce manual intervention.
- Improve cloud resource management efficiency.

3. Motivation for the Project

The motivation for this project arises from the cost optimization challenges faced by small organizations, students, and developers using AWS.
Manually starting and stopping servers daily can be time-consuming and prone to human error.
Automation using AWS CLI and Linux scripting provides an elegant, low-cost, and reliable solution.

This project also demonstrates real-world DevOps automation skills, which are highly in demand in cloud-based careers.

4. Objectives

The main objectives of the project are:

1. To automate the start and stop operations of EC2 instances using Bash scripting.
2. To configure IAM roles and permissions for secure EC2 management.
3. To use cron scheduling for automatic execution at predefined times.
4. To verify and log automated actions for transparency and debugging.
5. To achieve cost optimization through controlled instance uptime.

5. Literature Survey

Author/Source	Concept/Technology	Key Takeaway
AWS Documentation (2024)	IAM Role & EC2 Control	Explains how IAM roles provide secure control over EC2 instances.
AWS CLI User Guide	AWS Command Line Interface	Provides commands like aws ec2 start-instances and stop-instances.
Linux Cron Manual	Task Scheduling	Enables automated periodic task execution using cron.
DevOps Automation Blogs	Cost Optimization in AWS	Demonstrate real-time use of scheduling to manage non-critical workloads.

The survey reveals that similar cost-saving automations are used widely in professional environments, confirming the relevance and practical value of this project.

6. Research Question

“How can we efficiently automate EC2 instance start and stop operations to reduce manual effort and cloud costs using only AWS-native tools (IAM, CLI, cron) without external software?”

7. System Structure

The system structure includes the following major components:

1. EC2 Instance: Virtual machine to be started or stopped.
2. IAM Role & Policy: Provides permission to control EC2.
3. Bash Scripts: Execute start and stop commands via AWS CLI.
4. Cron Job Scheduler: Automates the scripts at specific times.
5. AWS CLI: Interface used to communicate with AWS services.

8. System Design Framework (Flow Diagram)

Flow of Operation:

```
+-------------------+
|  User sets cron   |
|   schedule times   |
+---------+----------+
          |
          v
+-------------------+
| Cron triggers Bash|
|  script (start/stop)|
+---------+----------+
          |
          v
+-------------------+
| AWS CLI executes  |
|  EC2 command       |
| (start/stop)       |
+---------+----------+
          |
          v
+-------------------+
| EC2 Instance State|
| changes (Running/ |
| Stopped)           |
+---------+----------+
          |
          v
+-------------------+
| Log written to    |
|  ec2_log.txt file  |
+-------------------+
```

9. Methodology

The project follows these stages:

1. Setup Environment
   - Create EC2 instance and configure security group.
   - Install AWS CLI on the instance.
2. Create IAM Role
   - Define policy with permissions: `StartInstances`, `StopInstances`, `DescribeInstances`.
   - Attach the IAM role to the EC2 instance.
3. Write Bash Scripts
   - Two scripts created: `start_ec2.sh` and `stop_ec2.sh`.
   - Each script logs actions into `ec2_log.txt`.
4. Configure Cron Jobs
   - Schedule automatic execution at specific times using `crontab`.
5. Testing and Validation
   - Manual testing of scripts.
   - Time-based cron testing.
   - Verification of instance state and log file updates.

10. Proposed Learning Strategy

During the implementation, the learning approach included:

- Understanding AWS IAM security model.
- Hands-on use of AWS CLI commands.
- Learning Linux task automation via cron.
- Practicing cloud cost optimization techniques.
- Debugging permissions and verifying CLI-based instance control.

This strategy ensures deep practical understanding of both cloud automation and shell scripting concepts.

11. Requirement Specification

Hardware Requirements

- AWS EC2 instance (t2.micro or higher)
- Stable Internet connection
- Local system with SSH capability

Software Requirements

- AWS Account
- Amazon Linux 2 AMI
- AWS CLI installed
- IAM Role with EC2 permissions
- Cron (pre-installed in Linux)

12. Results and Discussion

After implementation:

- EC2 instance automatically started and stopped at defined times.
- Manual execution of scripts successfully controlled instance states.
- Cron logs and ec2_log.txt file confirmed automated runs.
- Testing proved the system's reliability and efficiency.

Observation:
Automation prevented the instance from staying "running" unnecessarily, saving hourly charges.
This validates the goal of cost-effective cloud resource management.

13. Advantages

1. Fully automated—no manual login required.
2. Reduces AWS billing significantly.
3. Secure—controlled via IAM roles.
4. Customizable timing using cron expressions.
5. Scalable—can manage multiple instances.

15. Applications

- Used by organizations to manage development and testing servers.
- Ideal for students and trainers using AWS labs for limited hours.
- Can be used in DevOps pipelines for cost optimization.
- Helpful in academic projects demonstrating cloud automation.

16. Conclusion

This project successfully demonstrates an automated system for managing EC2 instance uptime using IAM, AWS CLI, Bash scripting, and cron scheduling.
It simplifies cloud management, improves operational efficiency, and reduces unnecessary AWS costs.

Through this project, learners gain hands-on experience in:

- Cloud automation
- IAM role configuration
- Linux shell scripting
- Task scheduling and monitoring

Overall, the project serves as a simple yet powerful example of real-world DevOps automation in cloud computing.

Project Completed By:

Umesh Chimankar
Jay Soni
Sakshi Jain
Kaveri Kanawade
Vaishnavi .Kumbhar
Nikita Binnar