BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI

First Semester 2025-26

SSWT ZG628T DISSERTATION

Dissertation Outline

BITS ID No.: 2021wa15838 **Name of Student:** Ajad Ramjit Chauhan

E-mail ID of the student: 2021wa15838@wilp.bits-pilani.ac.in

Name of Supervisor: Shafique Khan
Designation of Supervisor: Technical Lead
Qualification and Experience: B.E. 11+ years
E- mail ID of Supervisor: shafique.khan1@wipro.com

Title of Dissertation: Intelligent AWS EC2 Cost Optimization and Stale Resource Management.

Name of First Examiner: Rahul Gaikwad
Designation of First Examiner: Associate project manager
Qualification and Experience: B.E. 4+ years
E- mail ID of First Examiner: rahul.gaikwad@wipro.com

Name of Second Examiner: GANESH PAWADE
Designation of Second Examiner: Lead Engineer
Qualification and Experience: B.E 8+ years
E- mail ID of Second Examiner: ganesh.pawade@wipro.com

Supervisor's rating of the Technical Quality of this Dissertation Outline

EXCELLENT / GOOD / FAIR/ POOR (Please specify): GOOD

Supervisor's suggestions and remarks about the outline:
Your project outline reflects a thoughtful and well-structured approach to tackling one of the most critical challenges in cloud infrastructure - cost optimization and stale resource management. By leveraging AWS-native services such as Lambda, CloudWatch, Cost explorer, and S3 or DynamoDB reflect intelligent resource monitoring and automated decision-making operational efficiency.

[illegible]

AJAD RAMJIT CHAUHAN

(Signature of Student)

Date: 2nd Aug 2025

S.Ahmed/.

Shafique Khan

(Signature of Supervisor)

Date: 2nd Aug 2025

Intelligent AWS EC2 Cost Optimization and Stale Resource Management

Discussion on the Chosen Topic

1. The purpose of the work and expected outcome of the work

The purpose of this dissertation is to design and implement a cloud-native system that monitors EC2 performance, analyzes AWS service costs, and identifies stale resources such as unused volumes, snapshots, and idle instances. The system will automate cleanup actions using AWS Lambda and notify users of cost-impacting resources. The expected outcome is a scalable, cost-efficient solution that reduces cloud expenditure and improves operational governance.

2. Literature Review Done in Connection with the Work

- **AWS Well-Architected Framework (2023)**: Emphasizes cost optimization and operational excellence.
- **Serverless Architectures on AWS** by Peter Sbarski: Highlights automation and event-driven design.
- **Armbrust et al. (2010)**: Discusses reliability and cost challenges in cloud computing.
- **Li et al. (2020)**: Explores serverless resilience using Lambda and event-driven recovery.
- **Gartner Reports (2022)**: Show that 70% of cloud outages stem from human error, reinforcing the need for automation.

3. Brief Discussion on the Existing Process and Its Limitations

Existing Process

- EC2 instances are manually started and often left running even during idle periods.
- Snapshots and EBS volumes are created for backup or migration but are rarely deleted.
- Elastic IPs are allocated and remain unused, incurring charges.
- Cost tracking is done via the AWS Billing Console, without service-level granularity or automation.
- CloudWatch is used for basic monitoring, but without actionable automation.

Limitations

- Idle Resource Costs: EC2 instances and other resources continue to incur charges even when not in use.
- Stale Resources: Unattached volumes, old snapshots, and unused IPs accumulate over time, increasing storage and network costs.
- Lack of Automation: No built-in mechanism to detect and clean up stale resources or stop idle instances.
- Manual Oversight: Users must manually check usage and billing, which is error-prone and inefficient.
- No Real-Time Alerts: Users are not notified when resources become stale or exceed cost thresholds.

Public

4. Justification for Selecting a Particular Methodology for Completing the Tasks

Serverless Architecture Justification

Traditional server-based approaches (e.g., EC2-hosted applications) require continuous resource allocation, manual scaling, and higher operational overhead. In contrast, serverless services like **AWS Lambda** and **EventBridge** offer:

- **Automatic scaling** based on demand
- **Pay-per-use pricing**, reducing idle-time costs
- **Fault-tolerant design**, improving reliability
- **Minimal maintenance**, aligning with cost optimization goals

AWS Service Selection

- **CloudWatch**: For real-time monitoring of EC2 performance metrics.
- **Cost Explorer CLI**: For service-level cost analysis and trend tracking.
- **Lambda**: For automation of resource cleanup and EC2 control.
- **SNS**: For sending notifications to users about stale resources.
- **Streamlit (Python)**: For building a custom dashboard to visualize metrics and cost data.

5. Brief Discussion on the Dissertation Methodology

Scope of Work

The project involves developing a cloud-native, serverless system that includes:

- **Backend automation** using AWS Lambda for resource cleanup and EC2 control.
- **Monitoring infrastructure** using CloudWatch Agent and Dashboards.
- **Cost analysis tools** using AWS CLI and Cost Explorer API.
- **Visualization layer** using a Python-based dashboard (Streamlit).
- **Infrastructure as Code (IaC)** using CloudFormation for reproducibility.

Background of Previous Work

The methodology builds upon established serverless patterns and AWS best practices documented in literature. It extends these with custom logic for detecting stale resources and automating cost-saving actions, which are not commonly addressed in existing implementations.

Public

Ways and Means to Achieve Objectives

- **Design Phase**:
  - Define architecture and data flow.
  - Identify key AWS services and metrics.
  - Create architecture diagrams and dashboard layout.
- **Implementation Phase**:
  - Configure CloudWatch Agent and dashboards.
  - Simulate EC2 load using stress-ng.
  - Develop Lambda functions for automation.
  - Build Streamlit dashboard for cost and resource visualization.
  - Integrate AWS CLI for cost data collection.
- **Evaluation Phase**:
  - Collect cost and usage data before and after optimization.
  - Compare trends using visual dashboards.
  - Validate automation effectiveness through functional testing.

Benefits Expected

- **Enhanced reliability** through automated monitoring and cleanup.
- **Reduced downtime and manual effort** via Lambda-based automation.
- **Scalable management** of EC2 and related resources.
- **Quantifiable cost savings** demonstrated through real-time dashboards and historical comparisons.

6. Benefits Derivable from Work

- **Cost Savings**: Reduction in EC2 and storage costs through automated cleanup.
- **Operational Efficiency**: Real-time monitoring and automated actions reduce manual effort.
- **Scalability**: Works across multiple users and services without additional infrastructure.
- **Educational Value**: Demonstrates real-world AWS cost optimization and DevOps practices.
- **Security and Governance**: Enforces tagging and usage policies to prevent shadow IT.

Public

7. Any Other Details in Support of the Work

- **Innovation**: Combines real-time monitoring, cost analysis, and stale resource automation.
- **Tools Used**: AWS CLI, Lambda, CloudWatch, Python, Streamlit.
- **Ethical Considerations**: Ensures secure access and tagging for accountability.
- **Risk Mitigation**: Includes rollback scripts and alerting mechanisms.
- **Broader Impact**: Can be extended to other AWS services and multi-cloud environments.

Detailed Plan of Work (for 16 weeks)

Sr. No.	Task	Planned Duration (Weeks)	Specific Deliverable in Terms of the Project
1	Project Research, Setup & Requirement Analysis	2	- Literature review document - AWS Free Tier account setup - Functional requirements document
2	EC2 Instance Launch & Monitoring Setup	2	- EC2 instance (t2.micro) launched - CloudWatch Agent installed and configured - Metrics collection setup (CPU, memory, disk, network)
3	Load Simulation & Metrics Collection	2	- stress-ng installed and configured - Load tests executed - Usage pattern data collected
4	CloudWatch Dashboard Creation	1	- CloudWatch dashboards created - Widgets for CPU, memory, disk, and network metrics
5	Cost Analysis Setup & Streamlit Dashboard	2	- Cost Explorer enabled - AWS CLI scripts for cost data retrieval - Cost data stored in S3/local - Streamlit dashboard for cost visualization
6	Stale Resource Detection	2	- Boto3 scripts for detecting unused resources - Dashboard displaying stale resources with metadata - Action buttons for cleanup
7	Lambda Automation Development	2	- Lambda functions for cleanup and EC2 management - EventBridge rules for scheduling - SNS notifications setup
8	Testing & System Integration	2	- Unit testing of Lambda and Boto3 scripts - Integration of all modules - End-to-end workflow validation - Final system test report

Public

9	Cost Comparison & Optimization Demonstration	1	- Cost data before and after optimization collected - Streamlit dashboard showing savings - Graphs and percentage savings visualized
10	Documentation & Final Report	1	- Technical documentation - User guide - Final dissertation report - Architecture diagram - Dashboard screenshots - Cost analysis summary

8. Learning Outcomes

1. **Cloud Infrastructure Management**
    - Gained hands-on experience in launching and configuring AWS EC2 instances within the Free Tier.
    - Learned to assign IAM roles and set up billing alerts for cost control.
2. **Monitoring and Metrics Collection**
    - Developed skills in configuring CloudWatch Agent to collect and visualize system metrics such as CPU, memory, disk, and network usage.
    - Understood how to simulate load using stress-ng to generate meaningful performance data.
3. **Cost Analysis and Visualization**
    - Learned to enable and use AWS Cost Explorer and AWS CLI for daily cost tracking.
    - Built interactive dashboards using Streamlit to visualize cost trends and identify high-cost services.
4. **Resource Optimization Techniques**
    - Implemented logic to detect stale resources (e.g., unused volumes, snapshots, Elastic IPs) using Boto3.
    - Designed dashboards to display resource usage and provide cleanup actions.
5. **Automation with AWS Lambda**
    - Created Lambda functions to automate cost-saving actions such as stopping idle EC2 instances and deleting unused resources.

Public

- Integrated EventBridge and SNS for scheduled checks and notifications.

6. **System Integration and Testing**
    - Successfully integrated multiple AWS services into a cohesive cost optimization system.
    - Conducted unit and system-level testing to ensure reliability and performance.
7. **Cost Optimization Impact Analysis**
    - Compared pre- and post-optimization cost data to quantify savings.
    - Visualized the impact of automation and cleanup strategies on overall AWS billing.
8. **Technical Documentation and Presentation Skills**
    - Produced comprehensive documentation including user guides, deployment steps, and architecture diagrams.
    - Delivered a final presentation showcasing system design, dashboards, and cost analysis results,

Public