Cloud Cost Optimization

Strategies to Reduce Cloud Spending

Why Cost Optimization Matters

- AWS offers scalability but costs can rise quickly.
- Unused resources = wasted money.
- Goal: Maximize value and minimize waste.

Key Principles of AWS Cost Optimization

Right Sizing – Match instance types to actual needs.

Elasticity – Use Auto Scaling and on-demand resources.

Purchase Options – Leverage Savings Plans, Reserved Instances.

Monitor & Analyze – Use AWS tools to track and optimize spend.

Avoid Waste – Remove unused resources and storage.

AWS Tools for Cost Management

AWS Cost Explorer – Visualize cost and usage patterns.

AWS Budgets – Set spending limits and alerts.

AWS Trusted Advisor – Recommends cost-saving actions.

AWS Compute Optimizer – Suggests instance right-sizing.

Right-Sizing Resources

- Analyze CPU, memory, network usage.
- Downsize underutilized EC2, RDS, and Lambda functions.
- Example: t3.large → t3.medium saves ~20–30%.

Use Spot & Reserved Instances

- **Spot Instances:** Up to 90% cheaper, best for flexible workloads.
- **Reserved Instances:** 1–3 year commitment, big savings.
- **Savings Plans:** More flexible than RIs.

Optimize Storage

- **S3**: Use lifecycle policies (Standard → Infrequent Access → Glacier).
- **EBS**: Delete unattached volumes, snapshot cleanup.
- **RDS**: Use storage auto-scaling and monitor IOPS usage.

Auto Scaling & Scheduling

- **Auto Scaling Groups (ASGs)** scale based on demand.
- **Instance Scheduler**: Automatically stop dev/test environments during off-hours.
- Use Lambda + EventBridge for automation.

Monitoring and Alerts

- Enable **CloudWatch** metrics and alarms.
- Use **AWS Budgets** for threshold alerts.
- Visualize costs with **Cost Explorer Dashboards**.

Cost Optimization Use Case (Example)

Company X reduced AWS bill by 40%:

- Right Sized EC2 instances
- Moved infrequent S3 data to Glacier
- Used Spot instances for batch jobs
- Scheduled dev environments to shut down at night

Repo

https://github.com/ankit20000/EC2_Controller.git

Task

1-Check different way to save the cloud cost

2- Explore all Service available to save the cloud cost without impacting the application performance

3- Write the lambda function to stop/start the ec2_instance

4- Create the CI/CD pipeline for developer friendly so that they can stop/start the instances in non-business hours

5- Change pipeline to accept multiple IP