Lab 1 Report – Amazon S3 Lifecycle Management

1. What worked as expected

When I created the S3 bucket and added the lifecycle policy, the rules behaved the way I expected. I was able to define transitions to cheaper storage classes and also set expiration for objects. The S3 console showed that the lifecycle rule was attached, and it reflected in the object properties, which confirmed the setup was correct.

2. What was challenging

One part I initially found confusing was understanding how lifecycle rules apply differently when using prefixes versus object tags. I also realized that lifecycle transitions don't happen immediately. It takes time for objects to move to Glacier or be deleted, which made testing a bit tricky. Another challenge was making sure I had the right permissions to create and manage lifecycle rules on the bucket.

3. Real-world application

This lab made me see how lifecycle policies can really save costs and enforce data management automatically. In a real project, I would apply this to logs, backups, or archival data that don't need to sit in S3 Standard. For example, logs could automatically move to Glacier after 30 days, while temporary files could be set to expire after a week. This would reduce manual cleanup and ensure compliance with retention policies.

4. Cost implications

From the lab, I observed that keeping everything in S3 Standard would cost more in the long run. Moving older or rarely accessed data to Glacier is cheaper, but I also noted that retrieval costs and delays are a trade-off. Another useful cost implication is that expiration rules help avoid paying for storage of unused objects. The lifecycle feature itself doesn't add extra charges, so the main costs depend on the storage classes chosen.