NAS Alternative in AWS

1. Requirement Analysis:

The client requires a scalable, secure, and cost-effective NAS-like storage solution on AWS to support approximately 35 to 40 users, divided into two roles based on access privileges. Normal users should have limited access—specifically, the ability to upload files without the ability to view, read, or delete any data—while administrators must have full control over the system, including the ability to manage and retrieve all files. The solution must ensure strong security through proper identity and access management (IAM), enforcing the principle of least privilege and using OS-level or service-level access restrictions. It should also provide, in terms of scaling, consistent availability, and be capable of scaling seamlessly as data volume or user demand grows. Additionally, the solution must emphasize reliability, cost-efficiency, and maintainability to align with the organization's operational goals and compliance standards.

2. Purposed Solution:

a. AWS S3:

Background: Amazon Simple Storage Service (S3) is a highly durable, scalable, and secure object storage service offered by AWS. Designed to store and retrieve any amount of data from anywhere on the web, S3 provides 99.999999999% (11 nines) durability and offers flexible access control mechanisms, including IAM policies, bucket policies, and ACLs. It is ideal for use cases such as backup, archival, data lakes, and NAS-like storage solutions. With features like versioning, lifecycle policies, encryption, and native integration with AWS security and monitoring services, Amazon S3 enables organizations to build reliable, cost-effective, and compliant storage solutions without the need to manage traditional file servers or infrastructure. Plus for the storage requirements, S3 scales automatically with the objects in the bucket which greatly reduces operational overhead.

Flow of Operations:
Users authenticate via AWS Identity and Access Management (IAM) or an integrated identity provider to securely access a centralized Amazon S3 bucket. The bucket is logically partitioned into user-specific prefixes (sub-folders). Fine-grained access policies are enforced using IAM roles or bucket policies, ensuring that each user is restricted to uploading data only to their designated prefix. An administrator IAM role is granted full access to the entire bucket, with permissions to perform any S3 operation, including read, write, delete, and policy management.

As an alternative to a centralized bucket structure, a separate Amazon S3 bucket can be provisioned for each user to ensure strict isolation and simplified access control. Each user is granted permissions to access only their respective bucket using IAM policies or scoped roles. The administrator retains full control over all user buckets, enabling centralized governance and monitoring. This method enhances boundaries.

Purposed Architecture:

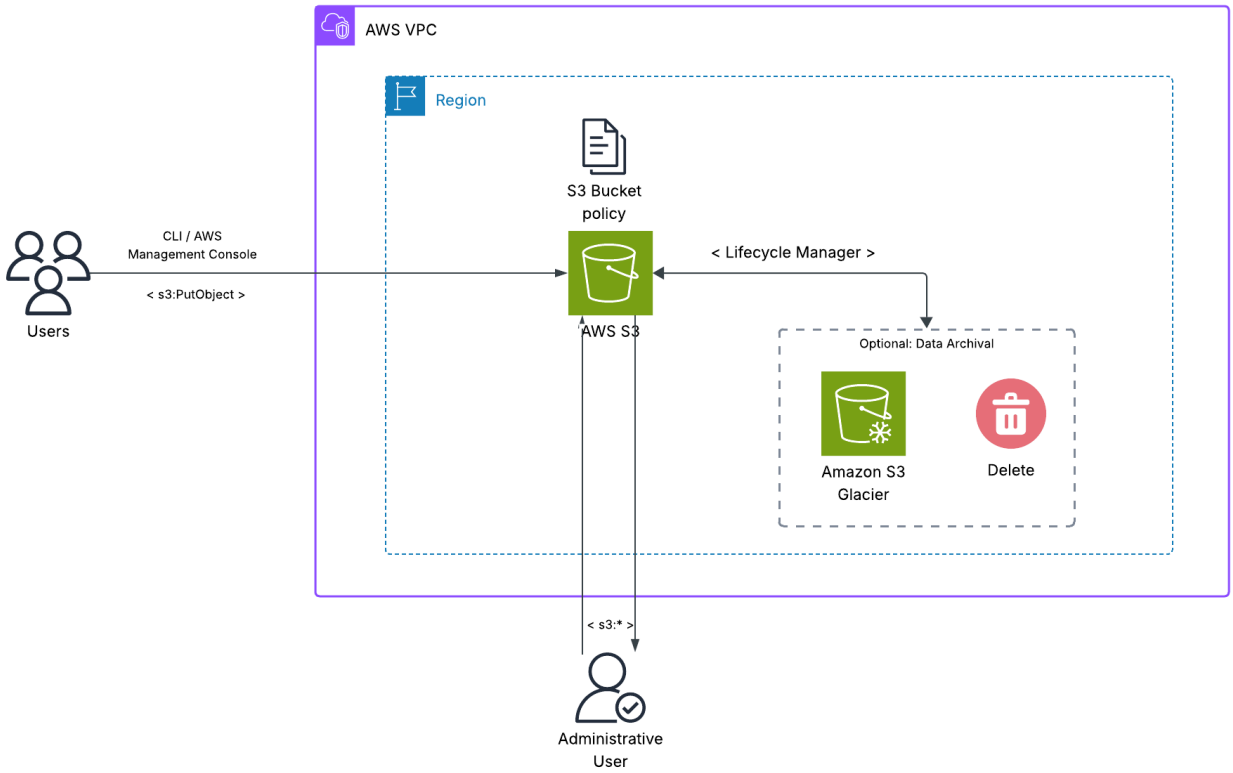

Figure: High Level Architecture for the S3 Storage Solution

Summary for the workings of the above architecture is below:

1. **User Access Strategy:**

   To securely and efficiently manage user interactions with the S3 bucket, we propose two options for providing access:

Option A: AWS Console Access (AWS Web UI)
In this approach, each user is provisioned with an individual IAM user account within a centralized AWS account. Access is granted through the AWS Management Console.

- Normal users will be assigned a policy allowing only the `s3:PutObject` action on the bucket (or on a user-specific prefix).
- Admin users will have a policy allowing `s3:*`, providing full access to the entire bucket.

This setup enables users to easily upload files using the AWS Console via a drag-and-drop interface.

Option B: AWS CLI Access
Alternatively, users can be provided with programmatic access via the AWS CLI. Each user will receive access keys and use the CLI to upload files securely. This method is ideal for users familiar with scripting, automation, or for batch uploads.

2. Storage Layout:

All uploaded data will be stored in a single S3 bucket. Optionally, documents from each user may be stored in separate folders such as `/user01` , `/user02` etc to isolate uploads and simplify access controls.

3. Lifecycle Management:

One of the best features about S3 is its ability to set Data Lifecycle Policy. We can automate data transitioning to different storage classes using **S3 Lifecycle Policies.** Once files are uploaded by users to the S3 bucket, they remain in the S3 Standard storage class for immediate availability and high performance. Over time, to optimize storage costs, objects that are not frequently accessed can be automatically transitioned to **Amazon S3 Glacier**, Deep Archive ;**a low-cost archival storage class** suitable for long-term retention. This transition process is seamless and policy-driven, requiring no manual intervention. Additionally, data can be configured to be automatically deleted after a specified retention period, supporting compliance requirements and minimizing unnecessary storage expenses. This tiered storage strategy ensures a balance between performance, availability, and cost-efficiency.

4. Estimated Cost for this Setup:

Considering cost and availability of our setup, the closest and cost effective region for us would be *ap-south-1 (Mumbai).* All the pricing calculations done below are based on the *Mumbai* region. Some quantities are estimated as we follow the pay as you go model.

Service	Quantity	Rate	Estimated Cost (per month)
S3 storage	3 TB per month	0.025 USD per GB per month	76.80 USD
PUT, COPY, POST, LIST requests to S3 Standard	~300,000 requests per month	0.000005 USD per request	1.50 USD
GET, SELECT, and all other requests from S3 Standard	~100,000 requests per month	0.0000004 USD per request	0.04 USD
Data transfer out cost	~100 GB	0.1093 USD per GB	10.93 USD
Total Cost			89.27 USD

Note: This cost does not include the archival solution, made optional in the architecture, refer to the official pricing estimate here.

b. Amazon EFS with SCP-Based Uploads:

Background: Amazon Elastic File System (EFS) is a fully managed, scalable, and cloud-native file storage service that provides shared access to data across multiple Amazon EC2 instances using the NFS (Network File System) protocol. It supports automatic scaling, high availability across multiple Availability Zones, and is ideal for applications that require a standard file system interface with POSIX compliance. EFS is designed for simplicity and durability, and offers lifecycle management to automatically move infrequently accessed files to a lower-cost storage class, making it suitable for both performance-sensitive and cost-sensitive workloads.

Purposed Architecture:

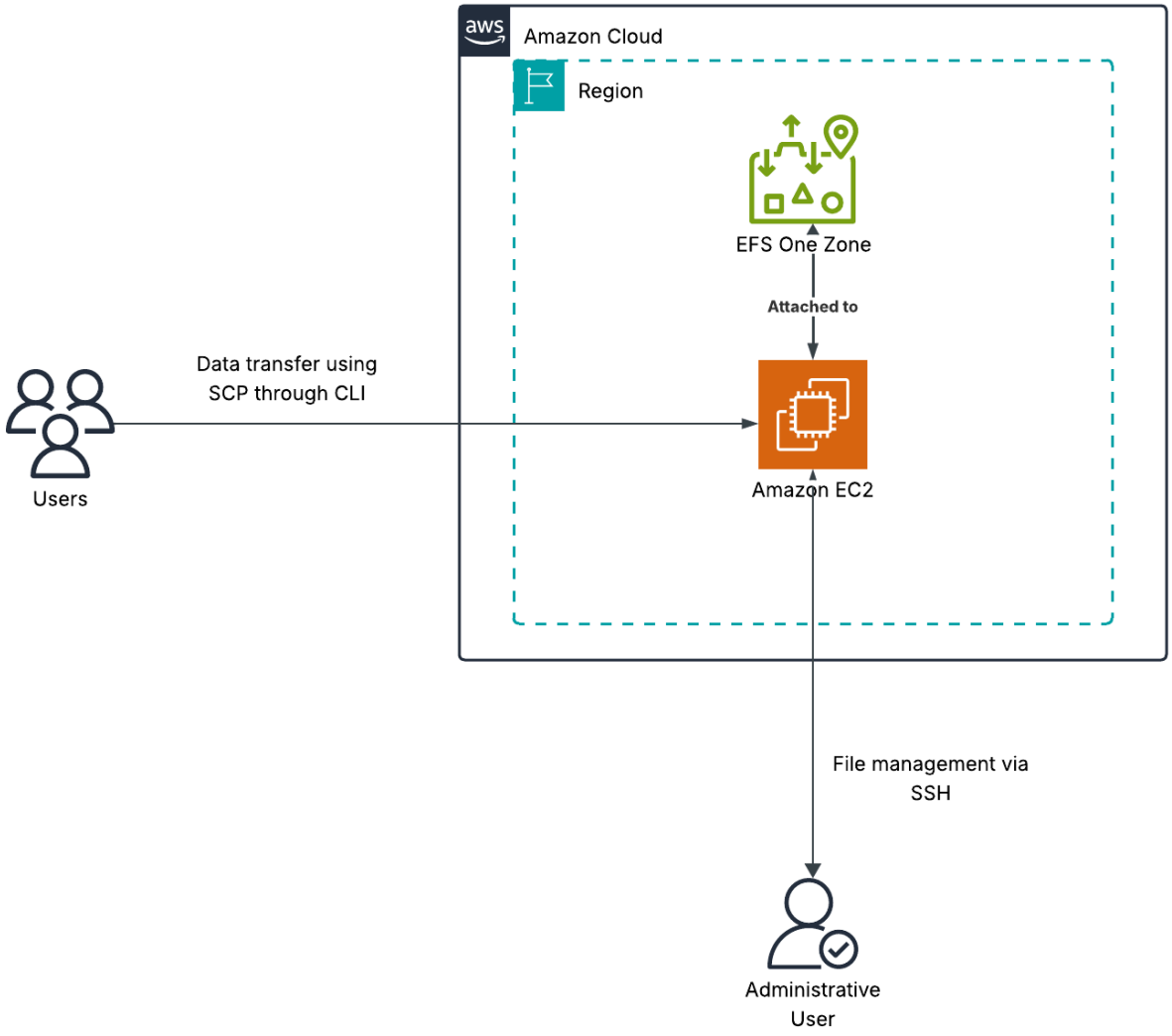

Figure: High Level Architecture for EFS with SCP-Based Uploads Solution

Flow of operations:

The AWS EFS is first mounted to the EC2 instance, preferably of linux based operating system. An administrative user connects to the EC2 instance via SSH for file management tasks, such as organizing directories, managing permissions, and performing maintenance. End users are granted restricted access via SCP (Secure Copy Protocol) over the command-line interface, enabling them to upload files exclusively to their designated directories within the EFS-mounted file system.

Summary for the workings of the proposed architecture is below:

1. **User Access Strategy:**
   - SCP-only access for users that need only uploads, users will have upload only access to their respective directories.
   - SSH based access for Administrative users which will have overall access to the system
2. **Storage layout:**

   In the Linux based Ec2 instance, we can have separate permissions for individual folders and corresponding folders visualized below:

```
/mnt/efs/users/
├── user01/
├── user02/
├── ...
└── user45/
```

3. **Advantages over S3 File System:**
   - POSIX-compliant, mountable file system
   - Secure upload via SCP with no user shell access
   - Centralized admin control
   - Scalable to petabytes

4. **Estimated Cost for this setup:**

   Considering cost and availability of our setup, the closest and cost effective region for us would be EFS one zone Storage configuration in *ap-south-1 (Mumbai).* All the pricing calculations done below are based on the *Mumbai* region.

Service	**Quantity**	**Rate**	**Estimated Cost (per month)**
EFS One Zone Storage	3 TB of storage, 20% of standard storage and 80% of Infrequent access	Standard (20%) = 614.4 GB × $0.176 = $108.13 Infrequent Access (80%) = 2,457.6 GB × $0.0145 = $35.64 Lifecycle Transition Cost (80 GB × $0.011) = $0.88	USD 144.65

EFS Provisioned Throughput Cost	50 MBPS per month	Baseline throughput from 614.4 GB: 22,425.6 MB/s-Hours Provisioned: 50 MB/s × 730 hrs = 36,500 MB/s-Hours Billable throughput = 36,500 − 22,425.6 = 14,074.4 MB/s-Hours Convert to MB/s-Month: 14,074.4 / 730 hours = 19.28 MB/s-Month 19.28 MB/s-Month x 6.60 USD = 127.25 USD (Provisioned Throughput monthly cost) Max (127.25 USD, 0 USD) = 127.25 USD	USD 127.25
EC2 On demand Instance - *t3.small*	1 Instance	1 instances x 0.0224 USD On Demand hourly cost x 730 hours in a month = 16.352000 USD	USD 16.352
Data Transfer	~300 GB	Internet: 300 GB x 0.1093 USD per GB = 32.79 USD	USD 32.79
Total			USD 321.042

Note: Official price estimate is linked here.