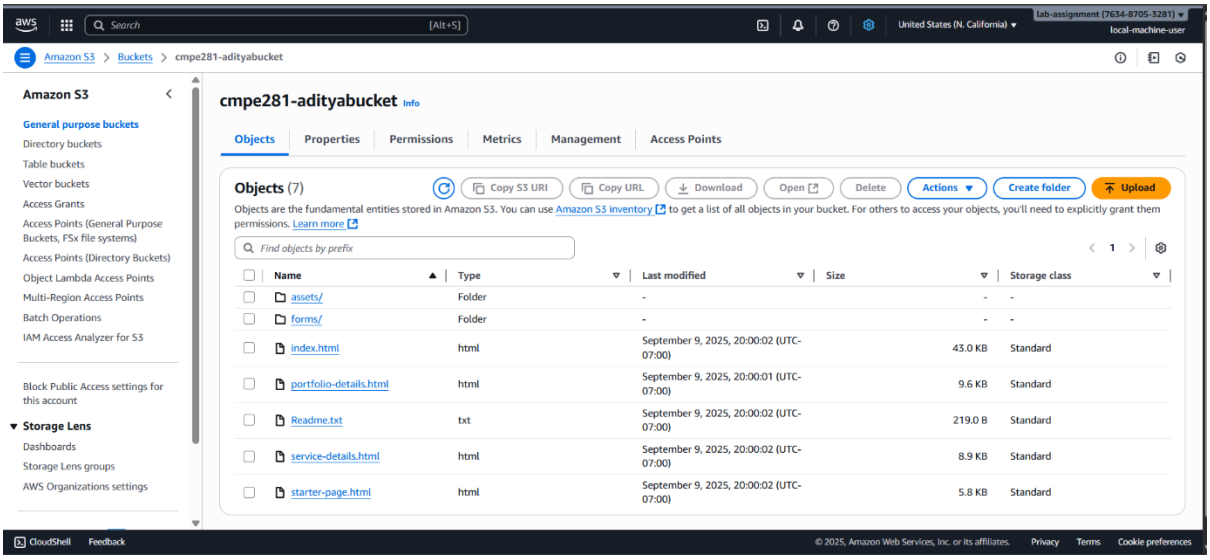
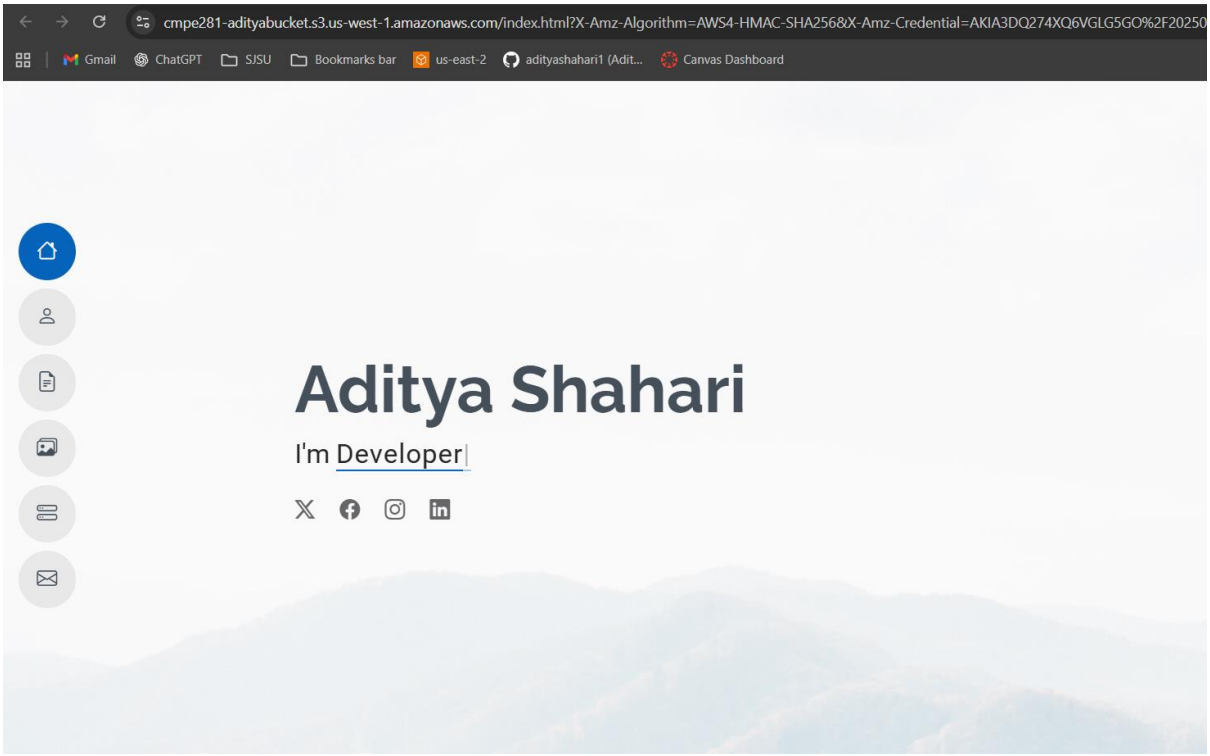
1. Screenshot of S3 bucket and files in AWS Console



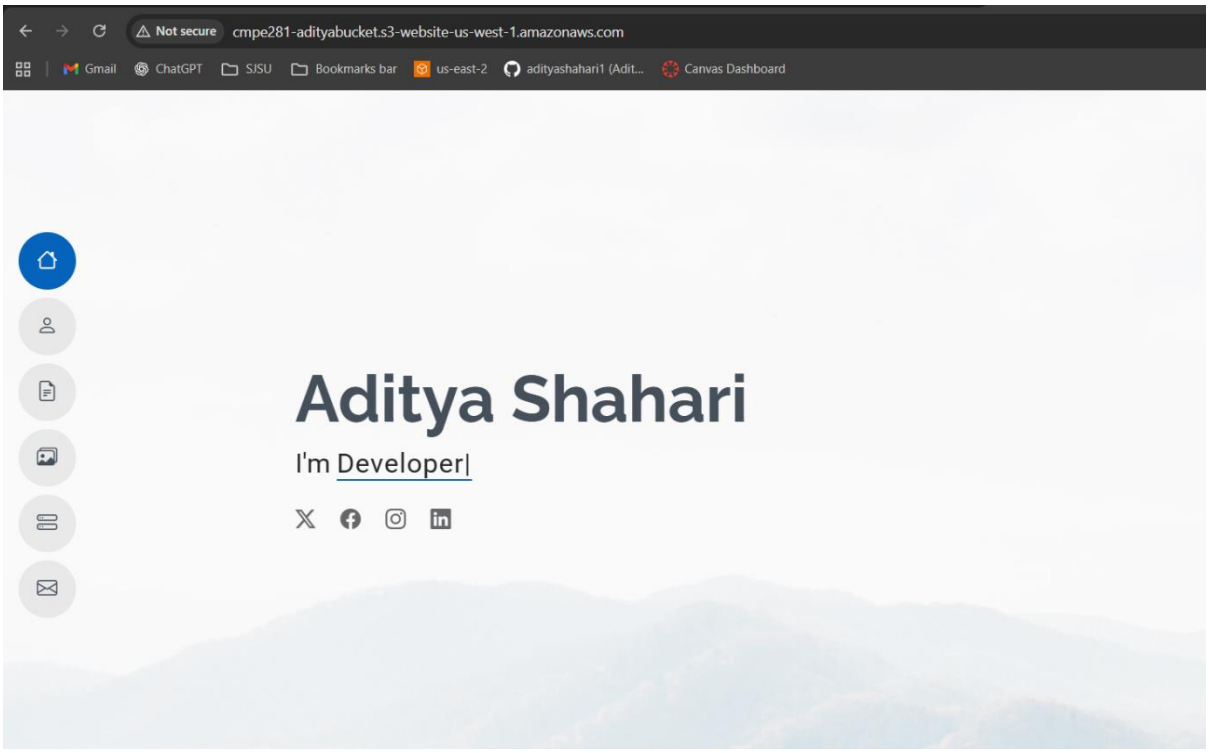
2. Screenshot of presigned URL and displayable page

```
aditya@Aditya:~/IntelligentCloud/lab-storage-adityashahari15 aws s3 presign s3://cmpe281-adityabucket/index.html --expires-in 600
https://cmpe281-adityabucket.s3.us-west-2.amazonaws.com/index.html?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIA3DQ274XQ6VGLG5GO%2F20250916%2Fus-west-2%2Fs3%2Faws4_request&X-Amz-Date=20250916T185859Z&X-Amz-Expires=600&X-Amz-SignedHeaders=host&X-Amz-Signature=84d1ecf7c3b280100da570835a8abd3cc14d6e2ba0f304666b1212e92d178b77
```

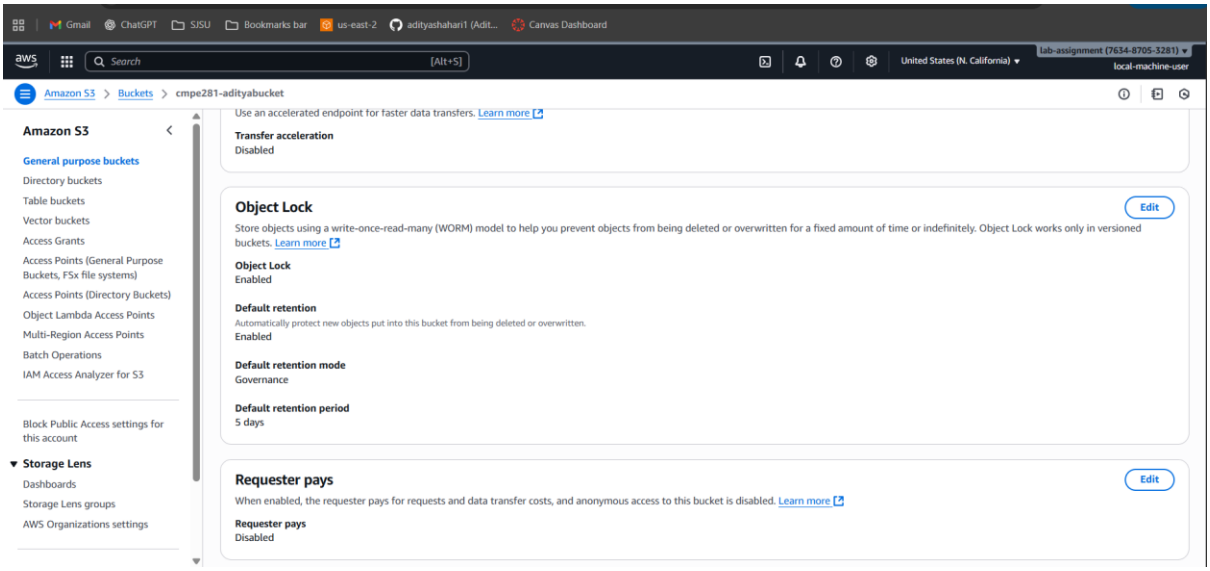


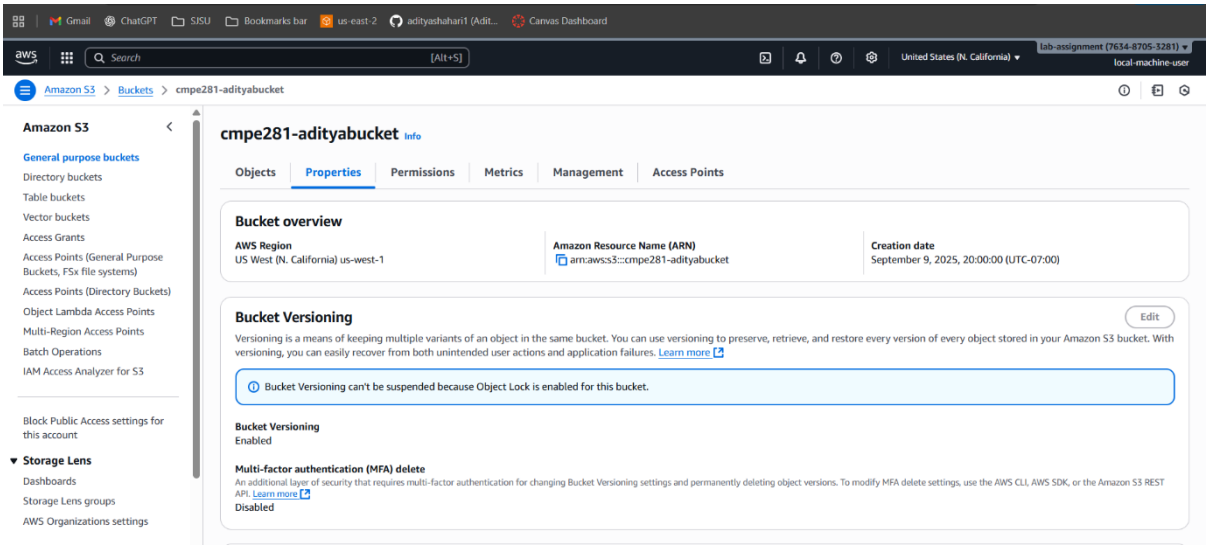
3. Link to Static URL and displayable page

```
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

aws_instance.instance1: Creating...
aws_instance.instance2: Creating...
aws_instance.instance2: Still creating... [00m10s elapsed]
aws_instance.instance1: Still creating... [00m10s elapsed]
aws_instance.instance1: Creation complete after 12s [id=i-02d9cb16cfd234000]
aws_instance.instance2: Creation complete after 13s [id=i-0eec306642456edf1]

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.

Outputs:

endpoint = "cmpe281-adityabucket.s3-website-us-west-1.amazonaws.com"
```

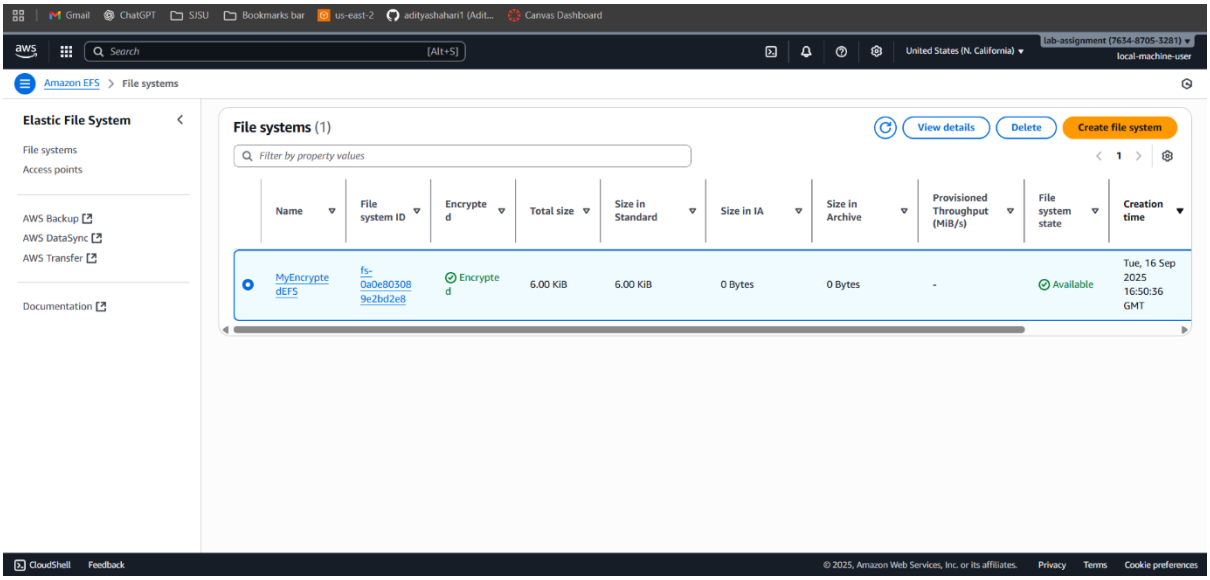


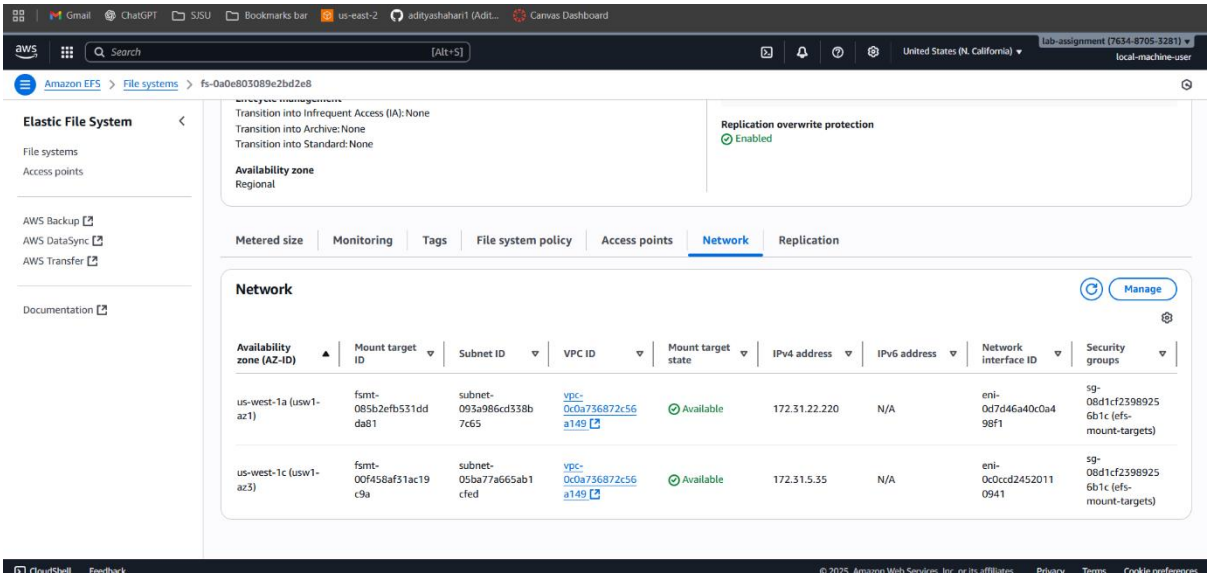
4. Screenshots of lifecycle management rules, object lock and versioning enabled





5. Screenshots of EFS console, Terminals from Instance 1 and Instance 2 –





Instance 1 -

```
ubuntu@ip-172-31-20-117:~$ df -h | grep efs
fs-0a0e803089e2bd2e8.efs.us-west-1.amazonaws.com:/  8.0E     0  8.0E   0% /mnt/efs
ubuntu@ip-172-31-20-117:~$ ls -l /mnt/efs
total 4
-rw-r--r-- 1 root root 10 Sep 16 18:45 hello.txt
ubuntu@ip-172-31-20-117:~$
```

Instance 2 -

```
ubuntu@ip-172-31-13-95:~$ ls -l /mnt/efs
total 4
-rw-r--r-- 1 root root 10 Sep 16 18:45 hello.txt
ubuntu@ip-172-31-13-95:~$ cat /mnt/efs/hello.txt
hello-efs
ubuntu@ip-172-31-13-95:~$
```

1. Explain how we use a single resource definition in task 1 to upload multiple files.

   Ans -

   In Task 1, we used a single aws_s3_object with for_each = fileset("${path.module}/s3-template", "**/*"), which turns one resource block into many objects at apply time. each.value becomes the S3 key and the local source path, etag = filemd5(...) ensures only changed files are reuploaded, and the content_type lookup assigns the right MIME type so the browser knows how to render each file.

2. Explain why the bucket policy is required in task 2 to display the web-page properly.

   Ans -

   The presigned URL authorizes only the one object you signed, but HTML page pulls other assets like CSS, JS, and images as separate requests. Without a bucket policy allowing public s3:GetObject on bucket/*, those assets stay private and the page cannot render correctly.

3. What are the pros and cons of the resources you used in task 4?

   Ans -

   Versioning protects against accidental deletes or overwrites and is useful for rollbacks. Object Lock in governance mode gives tamper protection and helps with compliance or ransomware scenarios, it prevents deletes or changes until retention ends and must be planned at bucket creation. Keeping the bucket ACL private with ownership controls is simple and safer, though cross-account or legacy ACL workflows can be harder without additional configuration.

4. Imagine you are managing an S3 bucket for a company that stores log files generated by their web applications. These log files are frequently accessed during the first 30 days after creation for analysis but are rarely accessed after that period. After 1 year, the logs don't need to be accessible immediately, but should be held for 5 years. Describe how you could use Amazon S3 Lifecycle Management to optimize storage costs for these log files. Include details on the lifecycle rules you would configure and how they would help reduce expenses.

   Ans -

   For the logs, keep new objects in S3 Standard for the first 30 days, transition them at day 30 to Standard-IA to cut costs, then at day 365 move them to Glacier Deep Archive since immediate access is not needed, and finally expire them at 5 years. If versioning is on, add matching noncurrent version transitions and expiration. This setup keeps frequently accessed logs fast and cheap enough early, then shifts them to much lower cost storage as they age, and removes them when retention is over.