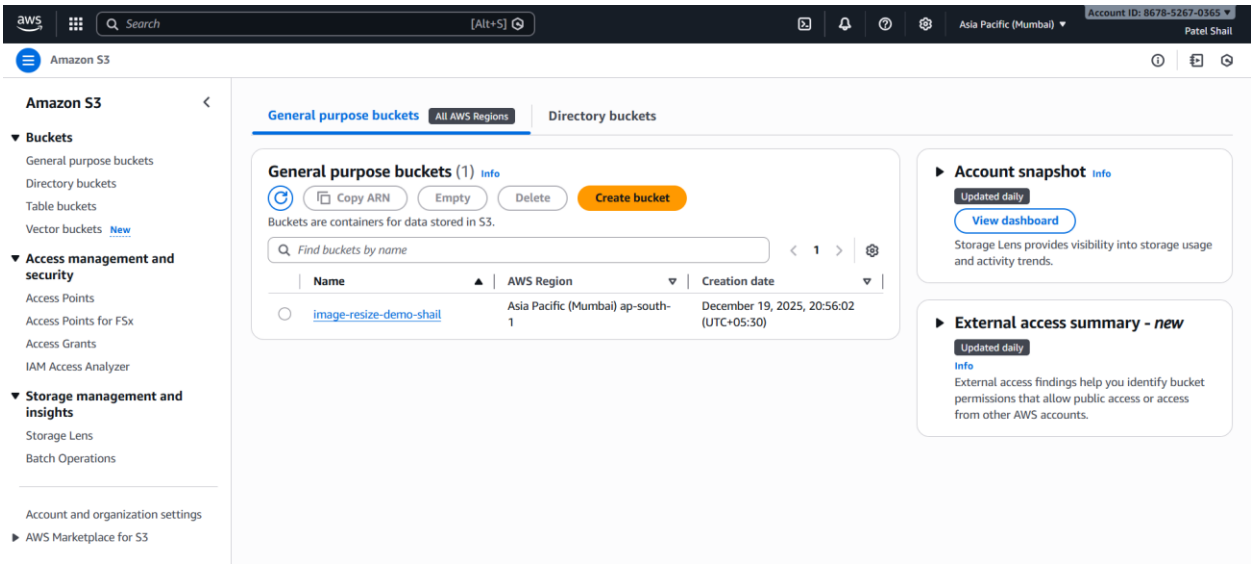
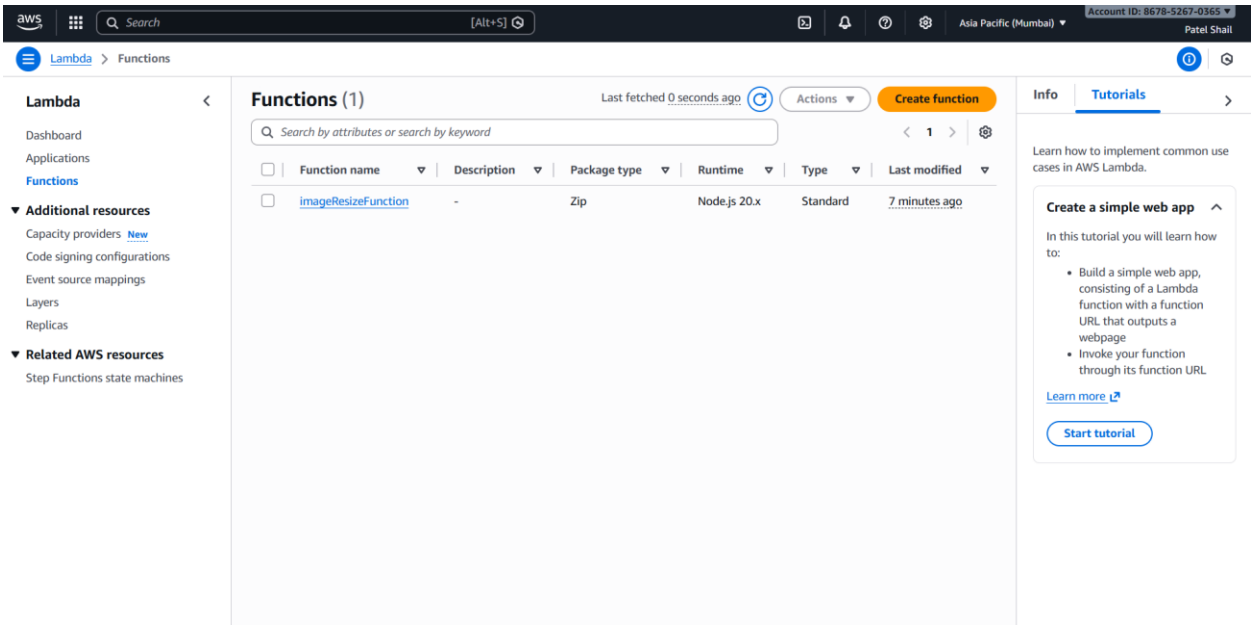
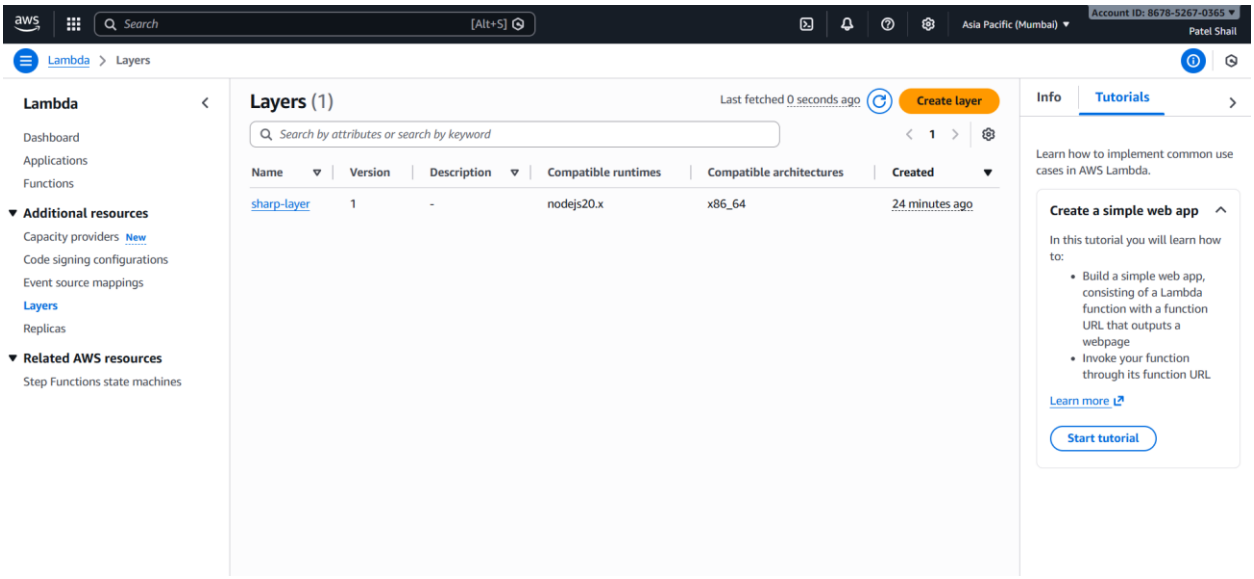
Create a S3 Bucket:

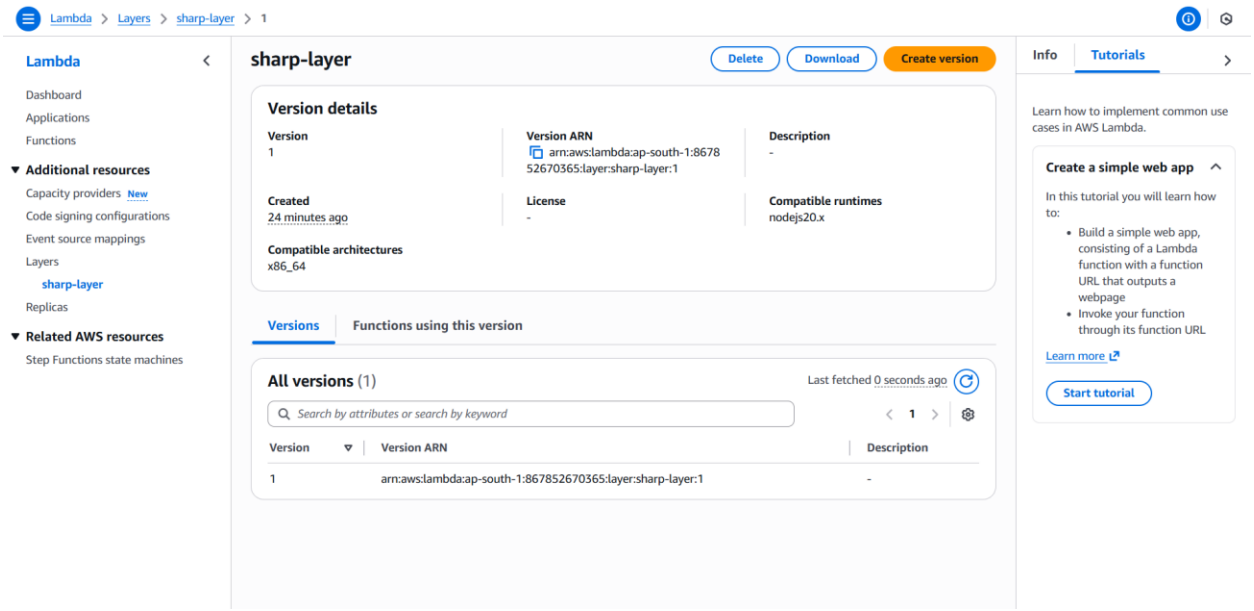


Create a Lambda Function :

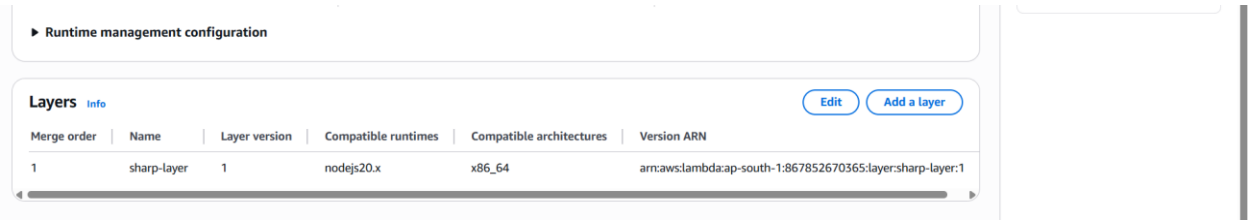


First Make a Layer using the CloudShell:

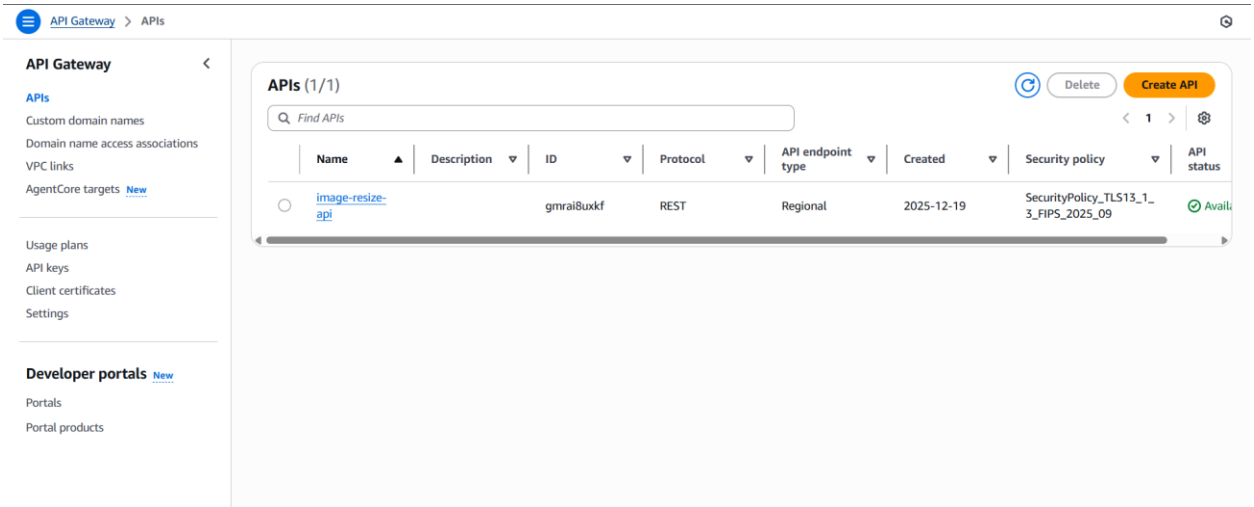


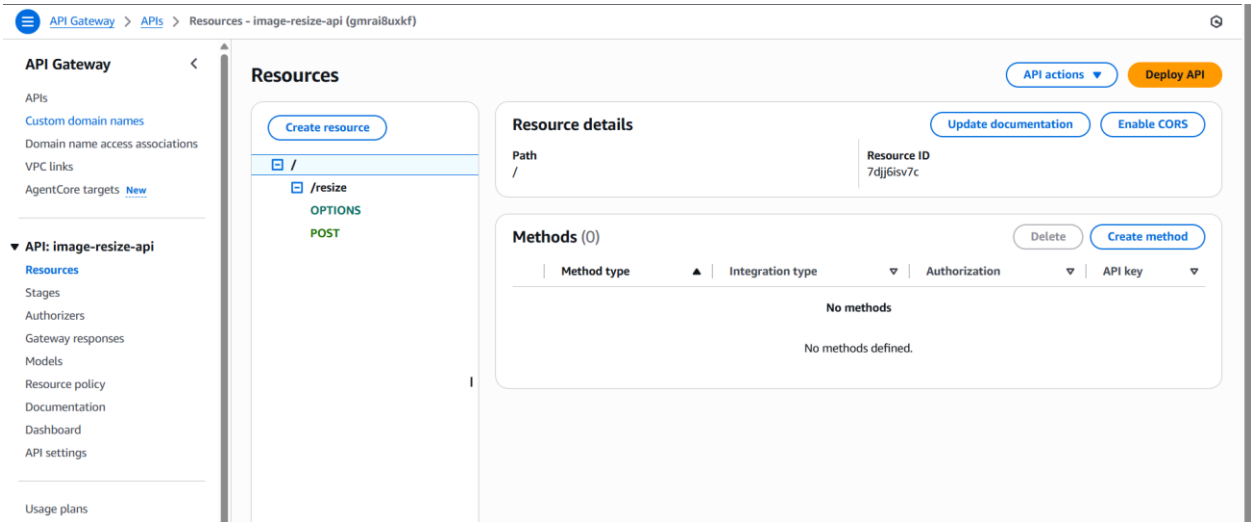


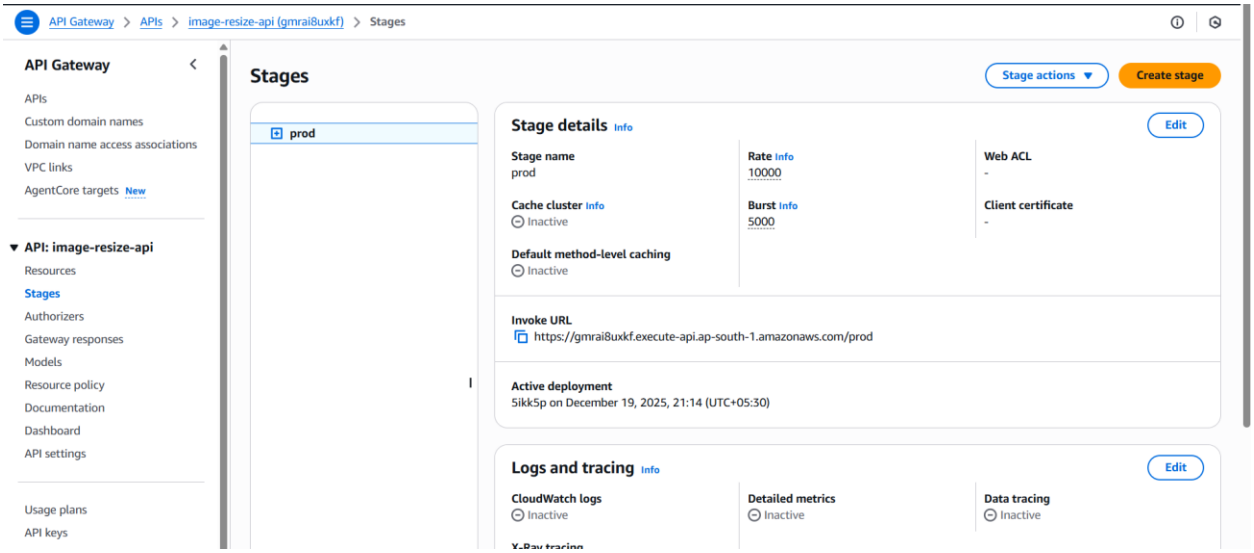
Add a Layer:



Make a API Gateway :







Index.html :

```
<!DOCTYPE html>

<html>

<head>

  <title>AWS Image Resizer</title>

</head>

<body>

  <h2>Upload Image</h2>

  <input type="file" id="file" />

  <button onclick="upload()">Upload & Resize</button>


  <p id="status"></p>

  <a id="download" target="_blank"></a>
```

```
  <script>
    async function upload() {
      const file = document.getElementById("file").files[0];
      const reader = new FileReader();

      reader.onloadend = async () => {
        const base64 = reader.result.split(",")[1];

        const res = await fetch("https://gmrai8uxkf.execute-api.ap-south-
1.amazonaws.com/prod/resize", {
          method: "POST",
          headers: { "Content-Type": "application/json" },
          body: JSON.stringify({ image: base64 }),
        });

        const data = await res.json();
        document.getElementById("status").innerText = data.message;
        document.getElementById("download").href = data.downloadUrl;
        document.getElementById("download").innerText = "Download Resized Image";
      };

      reader.readAsDataURL(file);
    }
  </script>
</body>
</html>
```

Lamda Function :

```
import {
  S3Client,
  PutObjectCommand,
  GetObjectCommand
} from "@aws-sdk/client-s3";
import { getSignedUrl } from "@aws-sdk/s3-request-presigner";
import sharp from "sharp";
import { randomUUID } from "crypto";

const s3 = new S3Client({ region: process.env.AWS_REGION });
const BUCKET_NAME = "image-resize-demo-shail";

export const handler = async (event) => {
  try {
    const body = JSON.parse(event.body);
    const imageBuffer = Buffer.from(body.image, "base64");

    const originalFileName = `original-${randomUUID()}.jpg`;
    const resizedFileName = `resized-${randomUUID()}.jpg`;
```

```
// 1️⃣ Upload ORIGINAL image to uploads/
await s3.send(
  new PutObjectCommand({
    Bucket: BUCKET_NAME,
    Key: `uploads/${originalFileName}`,
    Body: imageBuffer,
    ContentType: "image/jpeg",
  })
);

// 2️⃣ Resize image
const resizedImage = await sharp(imageBuffer)
  .resize(300, 300)
  .jpeg()
  .toBuffer();

// 3️⃣ Upload RESIZED image to resized/
await s3.send(
  new PutObjectCommand({
    Bucket: BUCKET_NAME,
```

```
    Key: `resized/${resizedFileName}`,
    Body: resizedImage,
    ContentType: "image/jpeg",
  })
);

// 4️⃣ Generate presigned URL for resized image
const getCommand = new GetObjectCommand({
  Bucket: BUCKET_NAME,
  Key: `resized/${resizedFileName}`,
});

const signedUrl = await getSignedUrl(s3, getCommand, {
  expiresIn: 3600, // 1 hour
});

return {
  statusCode: 200,
  headers: {
    "Access-Control-Allow-Origin": "*",
  },
```

```
      body: JSON.stringify({
        message: "Image uploaded and resized successfully",
        downloadUrl: signedUrl,
      }),
    };
  } catch (error) {
    console.error("ERROR:", error);
    return {
      statusCode: 500,
      headers: {
        "Access-Control-Allow-Origin": "*",
      },
      body: JSON.stringify({ error: error.message }),
    };
  }
};
```