



Academic Year	Module	Task
2025	5CS022/HJ1: Distributed and Cloud Systems Programming	2

Name: Swoyam Pokharel

ID: 2431342

Group: 26

Tutor: Ms. Jenny Rajak

Table Of Contents:

Question:	3
Solution:	4
1. Understanding the requirements	4
2. Preparation & Initial Setup:	4
Images:	4
Project Structure:	5
Screenshots showing the project structure	5
Screenshot showing tailwind compiling & bun server running on :3000	6
3. Finalized Code:	6
Index.html:	6
Screenshot of the headers of <code>index.html</code>	6
Screenshot showing the navigation of <code>index.html</code>	7
Screenshot Of The Main Page Content Of <code>Index.html</code>	7
Screenshots Showing gsap animations	8
Screenshot of <code>index.html</code> on my local machine	9
Screenshot of <code>index.html</code> with the dropdown expanded showing the 3 images I've used	9
Page1.html:	10
Screenshot showing differences between <code>page1.html</code> and <code>index.html</code> using diff piped to bat for clarity.	10
Screenshot showing animation in <code>page1.html</code> , exactly the same as <code>index.html</code>	11
Page2.html:	12
Screenshot showing differences between <code>page2.html</code> and <code>index.html</code> using diff piped to bat for clarity.	12
Screenshot showing animation in <code>page2.html</code> , exactly the same as <code>index.html</code>	13
4. Deployment:	14
S3 Bucket & Static Hosting:	14
I: Navigating To AWS:	14
Screenshot showing my AWS landing page	14
II: Navigating To S3:	15
Screenshot showing my buckets	15
III: Creating a new bucket	16
Screenshot of the bucket creation page with filled inputs	17
Screenshot Showing Successful Bucket Creation	17
IV: Uploading Relevant Files	18
Screenshot showing files for upload	18
Screenshot showing successful upload	19
V: Editing The Bucket Policy:	19
Screenshots showing editing bucket policy.	19
Screenshot showing confirmation	20
VI: Checking If Our Files Are Accessible Via Object URL:	21
Screenshot showing <code>index.html</code> accessible via S3 Object URL	21
Screenshot showing <code>page1.html</code> accessible via S3 Object URL	22
Screenshot showing images accessible via S3 Object URL	22
VII: Enabling Static Hosting:	23
Screenshot Of Static Hosting Page	23
Screenshot showing successful save	24
Screenshots showing 403 forbidden	24
Screenshot showing http link to statically hosted S3 bucket	25
Screenshot of bucket policy that was causing the problem	26
Successful static website hosting	27
AWS Amplify	28
I: Navigating To AWS Amplify	28
Screenshot of Amplify Landing Page	28
II: Deploying Amplify App	29

Screenshots of the <code>create new app</code> page	29
Screenshot showing deployment in progress	30
Screenshot showing successful deployment	30
Screenshot showing hosted website in amplify	31
III: Updating Code & Changing Navigation	31
Screenshot Of Updated Navigation In Index.html	32
Screenshot Of Updated Navigation In page1.html	32
Screenshot Of Updated Navigation In page2.html	33
IV: Uploading Changes	34
Screenshot Of Uploading Changes	34
Screenshot Confirming Upload	34
V: Screenshots	35
Screenshot Of Static Page Hosted On S3	35
Screenshot Of Amplify Hosted Page 1	35
Screenshot Of Amplify Hosted Page 2	36
Screenshot Of Redirection Back to Index Hosted On S3	36
Example Screenshot of the image when accessed via index.html (loaded from the S3 domain).	38
Example Screenshot of the image when accessed via page1.html (loaded from the Amplify domain).	39

Question:

In a similar approach to Workshop on AWS, create a static website on AWS S3 Bucket, consisting of 3 HTML web pages, and one image. The 3 web pages should be named “index.html”, “page1.html”, and “page2.html”. The image file should be named “image1.png” and should contain either a picture of a kitten or flowers. The “index.html” page should display the image in a tag, similar to the workshop exercise. In addition, it should also contain a link to “page1.html” and a link to “page2.html”. The two files “page1.html” and “page2.html” should both contain a link back to “index.html”, and to each other. The image file “image1.png”, as well as “page1.html” and “page2.html”, should then be deployed to the AWS Amplify service, and all the links updated. The only non-Amplify link should be only to “index.html”. Your submission should be a zip of the 3 HTML files plus the image and a well documented MS Word file containing your name, student ID and explanation and screenshots of the 3 HTML pages, as well as the image on AWS Amplify, in your web browser showing the link URLs clearly.

This task will contribute 20% of the marks to the Portfolio.

Solution:

1. Understanding the requirements

So, the question wants us to create 3 html pages, explicitly named:

`index.html`
`page1.html`
`page2.html`

In the `index.html` page, the question wants us to display images which can either be of kittens, or flowers.

We then want to create a static website on AWS S3 Bucket hosting those 3 html pages and the images.

Then, we need to deploy `page1.html`, `page2.html` along with the images on AWS Amplify service, and we need to update the links so that the following navigation path is achieved:

`index.html` (hosted on S3) → Links To `page1.html` and `page2.html` (both hosted on amplify)
`page1.html` (hosted on Amplify) → Links To `page2.html` (hosted on amplify) and `index.html` (hosted on S3)
`page2.html` (hosted on Amplify) → Links to `page1.html` (hosted on amplify) and `index.html` (hosted on s3)

2. Preparation & Initial Setup:

Images:

Since we require images, of either kittens or flowers, I've decided to go with the following 3 images:



Project Structure:

```
[wizard@archlinux Swoyam_Pokharel_2431342]$ ls
[wizard@archlinux Swoyam_Pokharel_2431342]$ touch {index,page1,page2}.html
[wizard@archlinux Swoyam_Pokharel_2431342]$ tree

.
├── index.html
├── page1.html
└── page2.html

1 directory, 3 files
[wizard@archlinux Swoyam_Pokharel_2431342]$ 
```

```
[wizard@archlinux Swoyam_Pokharel_2431342]$ tree
.
├── index.html
├── page1.html
├── page2.html
└── static
    ├── images
    │   ├── image1.jpg
    │   ├── image2.jpg
    │   └── image3.jpg
    ├── input.css
    └── output.css

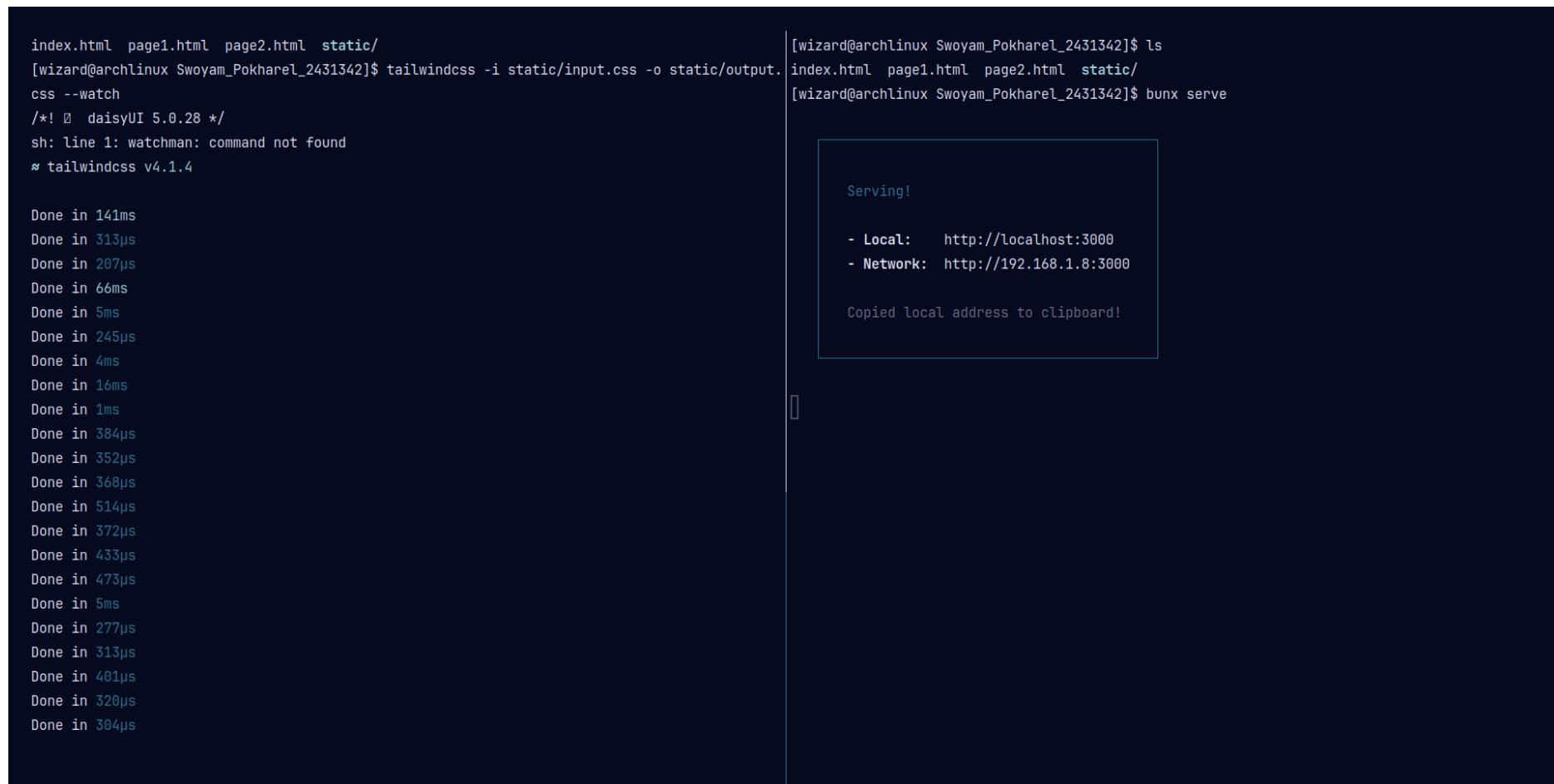
3 directories, 8 files
[wizard@archlinux Swoyam_Pokharel_2431342]$ 
```

```
1
2 @import url('https://fonts.googleapis.com/css2?family=Geist&display=swap');
3 @import "tailwindcss";
4 @plugin "@tailwindcss/typography";
5 @plugin "daisyui";
6
7 *{
8     font-family:"Geist",sans-serif;
9 }
10
11 @plugin "daisyui/theme" {
12     name: "black";
13     default: true;
14     prefersdark: true;
15     color-scheme: "dark";
16     --color-base-100: white;
17     --color-base-200: white;
18     --color-base-300: oklch(14% 0.004 49.25);
19     --color-base-content: oklch(12% 0.042 264.695);
20     --color-primary: oklch(13% 0.028 261.692);
21     --color-primary-content: white;
22     --color-secondary: white;
23     --color-secondary-content: oklch(12% 0.042 264.695);
24     --color-accent: oklch(84% 0.238 128.85);
25     --color-accent-content: oklch(12% 0.042 264.695);
26     --color-neutral: oklch(20% 0 0);
27     --color-neutral-content: oklch(100% 0 0);
28     --color-info: oklch(60% 0.25 292.717);
29     --color-info-content: oklch(12% 0.042 264.695);
30     --color-success: oklch(50% 0.118 165.612);
31     --color-success-content: oklch(90.395% 0.035 142.495);
32     --color-warning: oklch(96.798% 0.211 109.769);
33     --color-warning-content: oklch(19.359% 0.042 109.769);
34     --color-error: oklch(62.795% 0.257 29.233);
35 }

: 
```

Screenshots showing the project structure

I've placed the images in the `static/images/` directory. For styling, I'm using `Tailwind CSS` along with `DaisyUI` for convenience. The `input.css` file includes base styles and a custom theme generated using the [DaisyUI-Theme-Generator](#). `Tailwind` compiles this into `output.css`, which is the stylesheet used in the HTML pages. This is all that we will be doing for the initial setup. For the live development server I'm using `bun`.



```

index.html page1.html page2.html static/
[wizard@archlinux Swoyam_Pokharel_2431342]$ tailwindcss -i static/input.css -o static/output.css --watch
/*! / daisyUI 5.0.28 */
sh: line 1: watchman: command not found
≈ tailwindcss v4.1.4

Done in 141ms
Done in 313µs
Done in 207µs
Done in 66ms
Done in 5ms
Done in 245µs
Done in 4ms
Done in 16ms
Done in 1ms
Done in 384µs
Done in 352µs
Done in 368µs
Done in 514µs
Done in 372µs
Done in 433µs
Done in 473µs
Done in 5ms
Done in 277µs
Done in 313µs
Done in 401µs
Done in 320µs
Done in 304µs

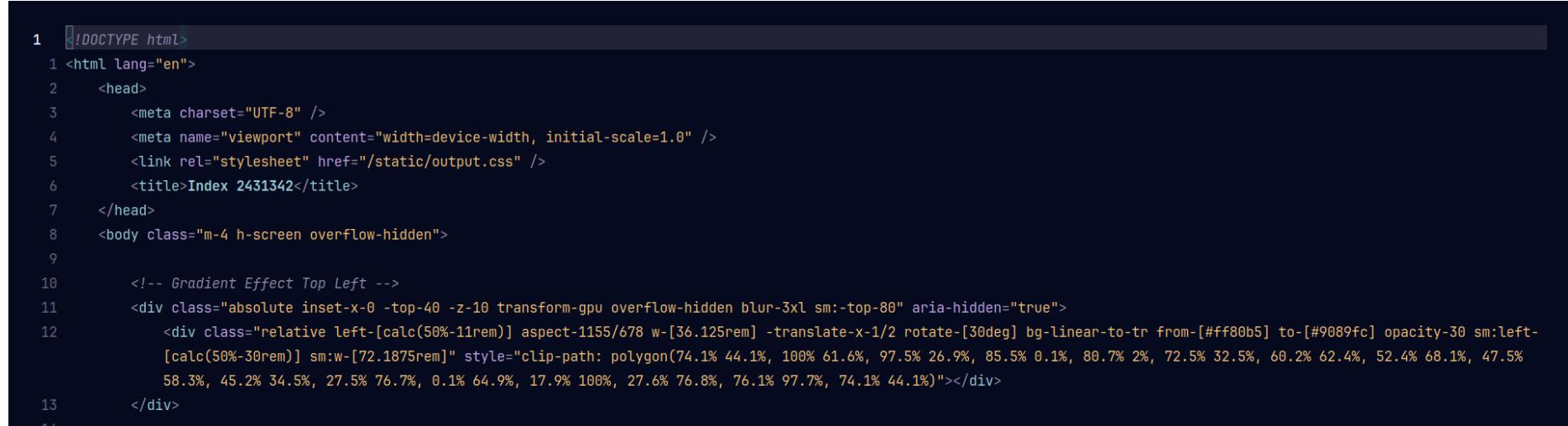
[wizard@archlinux Swoyam_Pokharel_2431342]$ ls
index.html page1.html page2.html static/
[wizard@archlinux Swoyam_Pokharel_2431342]$ bunx serve
Serving!
- Local: http://localhost:3000
- Network: http://192.168.1.8:3000
Copied local address to clipboard!

```

Screenshot showing tailwind compiling & bun server running on :3000

3. Finalized Code:

Index.html:



```

1  !DOCTYPE html>
2  <html lang="en">
3    <head>
4      <meta charset="UTF-8" />
5      <meta name="viewport" content="width=device-width, initial-scale=1.0" />
6      <link rel="stylesheet" href="/static/output.css" />
7      <title>Index 2431342</title>
8    </head>
9    <body class="m-4 h-screen overflow-hidden">
10
11      <!-- Gradient Effect Top Left -->
12      <div class="absolute inset-x-0 -top-40 -z-10 transform-gpu overflow-hidden blur-3xl sm:-top-80" aria-hidden="true">
13        <div class="relative left-[calc(50%-11rem)] aspect-1155/678 w-[36.125rem] -translate-x-1/2 rotate-[30deg] bg-linear-to-tr from-[#ff80b5] to-[#9089fc] opacity-30 sm:left-[calc(50%-30rem)] sm:w-[72.1875rem]" style="clip-path: polygon(74.1% 44.1%, 100% 61.6%, 97.5% 26.9%, 85.5% 0.1%, 80.7% 2%, 72.5% 32.5%, 60.2% 62.4%, 52.4% 68.1%, 47.5% 58.3%, 45.2% 34.5%, 27.5% 76.7%, 0.1% 64.9%, 17.9% 100%, 27.6% 76.8%, 76.1% 97.7%, 74.1% 44.1%)></div>
14    </div>

```

Screenshot of the headers of `index.html`

This is mostly standard boilerplate HTML. The only notable addition is a gradient blob positioned at the top left. It's just a polygon placed behind a blurred background, creating the illusion of a gradient flowing from that corner.

```

20      <!-- Navigation Bar -->
19      <div class="px-30 navbar sticky top-0 z-50 bg-transparent backdrop-blur-md">
18          |   <div class="flex justify-between items-center w-full">
17              |       <!-- Left -->
16          |           <div class="flex-1">
15              |               <a class="btn btn-ghost text-xl nav-link tooltip tooltip-bottom" data-tip="Navigate Back!" onclick="window.history.back()">&larr;</a>
14          |           </div>
13
12          |       <!-- Right -->
11          |           <div class="flex-1 justify-end flex">
10              |               <ul class="menu menu-horizontal px-1 flex gap-3">
9                  |                   <li>
8                      |                       <a class="text-base font-medium uppercase nav-link tooltip tooltip-bottom" data-tip="Meow Rotate 45" href="/page1.html">Page 1</a>
7
6                  |                   <li>
5                      |                       <a class="text-base font-medium uppercase nav-link tooltip tooltip-bottom" data-tip="Meow Rotate 135" href="/page2.html">Page 2</a>
4
3                  |                   <li>
2
1                  |                       <details>
1                      |                           <summary class="text-base font-medium uppercase nav-link tooltip tooltip-left" data-tip="Click For Links To Images on S3 Bucket">Images</summary>
36
1                         |                               <ul class="bg-base-100 rounded-t-none p-2 uppercase font-bold">
1                             |                                   <li><a class="text-md" href="/static/images/image1.jpg">Image-1</a></li>
2                             |                                   <li><a class="text-md" href="/static/images/image2.jpg">Image-2</a></li>
3                             |                                   <li><a class="text-md" href="/static/images/image3.jpg">Image-3</a></li>
4
5                             |                               </ul>
6                         |                       </details>
7
8                     |                 </li>
9                 </ul>
10            </div>
11
12
index.html
[0] 0:nvim* 1:bunx-

```

Screenshot showing the navigation of index.html

The above code is a navigation bar for [index.html](#). The navigation bar includes a back button on the left, that utilizes the browser's native history to go back, and links to [page1.html](#) and [page2.html](#) on the right. I've also included a drop down menu for the images I've used.

```

5      <!-- Page Content -->
4      <div class="h-2/3 w-screen flex-col flex justify-center gap-20 items-center">
3          |   <h1 class="text-8xl font-bold uppercase tracking-wide hero-title"> Index Page </h1>
2          |   <div class="flex gap-4">
1              |       
54
1                         |                         
1                         |                         
2
3             </div>
4
5             <!-- Gradient Effect Bottom Right -->
6             <div class="absolute inset-x-0 top-[calc(100%-13rem)] -z-10 transform-gpu overflow-hidden blur-3xl sm:top-[calc(100%-30rem)]" aria-hidden="true">
7                 <div class="relative left-[calc(50%+3rem)] aspect-1155/678 w-[36.125rem] -translate-x-1/2 bg-linear-to-tr from-[#ff80b5] to-[#9089fc] opacity-30 sm:left-[calc(50%+
36rem)] sm:w-[72.1875rem]" style="clip-path: polygon(74.1% 44.1%, 100% 61.6%, 97.5% 26.9%, 85.5% 0.1%, 80.7% 2%, 72.5% 32.5%, 60.2% 62.4%, 52.4% 68.1%, 47.5% 58.3%, 45.
2% 34.5%, 27.5% 76.7%, 0.1% 64.9%, 17.9% 100%, 27.6% 76.8%, 76.1% 97.7%, 74.1% 44.1%)"></div>
8
9
10    </body>

```

Screenshot Of The Main Page Content Of index.html

The above code shows the main content of the [index.html](#) page, it simply is a flex container with the 3 images of kittens that we've selected above. Furthermore, just like in the header, we have a similar gradient positioned at the bottom right of the page.

```

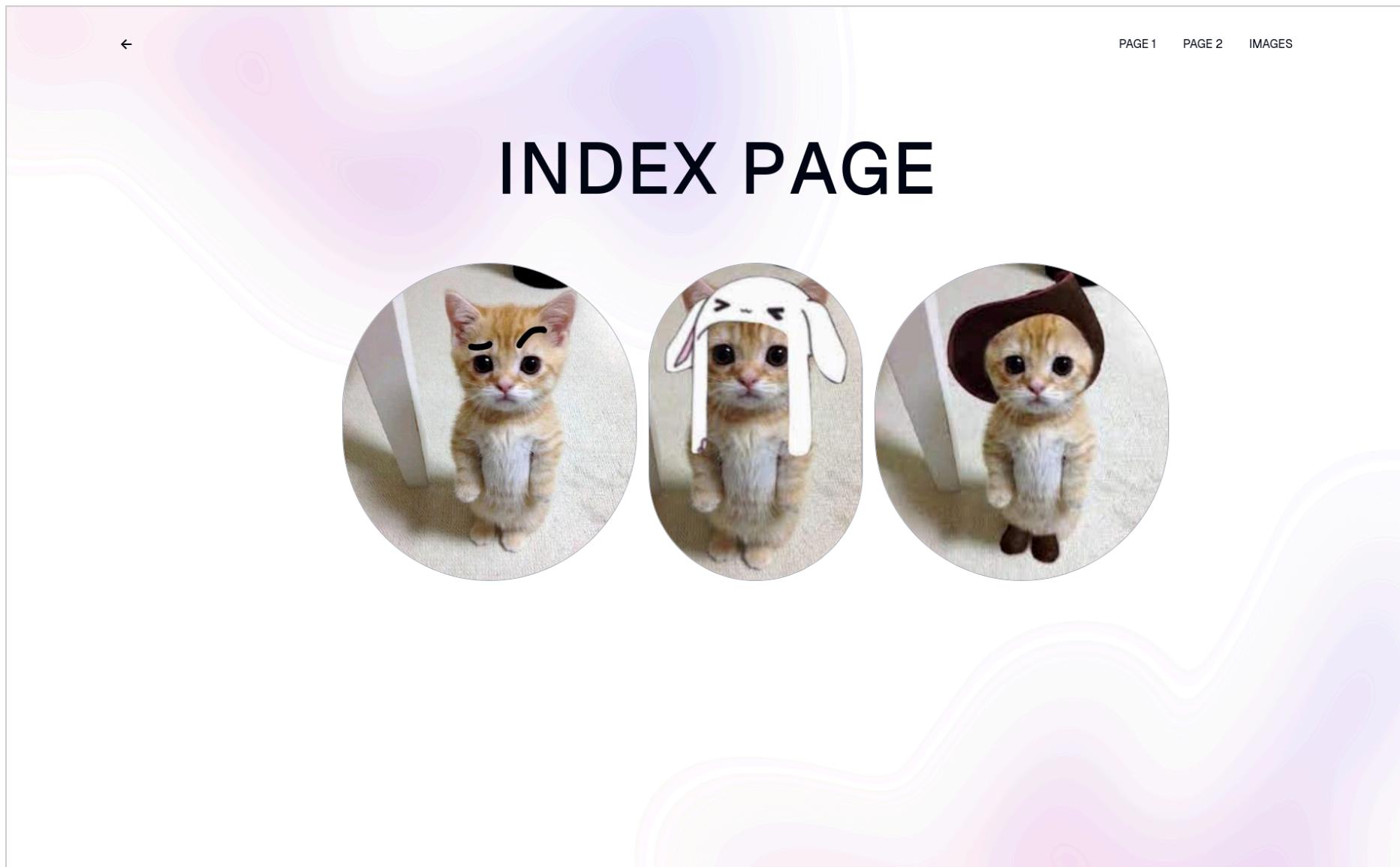
22 <script src="https://cdn.jsdelivr.net/npm/gsap@3.13.0/dist/gsap.min.js"></script>
21 <script src="https://cdn.jsdelivr.net/npm/gsap@3.13.0/dist/SplitText.min.js"></script>
20
19 <script>
18 |   document.addEventListener("DOMContentLoaded", () => {
17 |     const title = document.querySelector(".hero-title");
16 |     const split = new SplitText(title, { type: "chars" });
15 |
14 |     // hero title split text animation
13 |     gsap.from(split.chars, {
12 |       opacity: 0,
11 |       y: 50,
10 |       rotateX: 90,
9 |       stagger: 0.05,
8 |       ease: "back.out(1.7)",
7 |       duration: 1,
6 |     });
5 |
4 |     // navigation links fly in
3 |     gsap.from(gsap.utils.toArray(".nav-link"), {
2 |       y: -20,
1 |       opacity: 0,
88 ||       stagger: 0.1,
1 |       delay: 0.3,
2 |       duration: 0.6,
3 |       ease: "power2.out"
4 |     });
5 |
6 |     // scale up them kittens
7 |     gsap.from(gsap.utils.toArray("img"), {
8 |       opacity: 0,
9 |       stagger: 0.2,
10 |       delay: 0.3,
11 |       scale: 0.2,
12 |       duration: 0.6,
13 |     });
14 |   });
15 </script>
16 </html>
index.html" 104L, 5351B
[0] 0:nvim* 1:bunx-

```

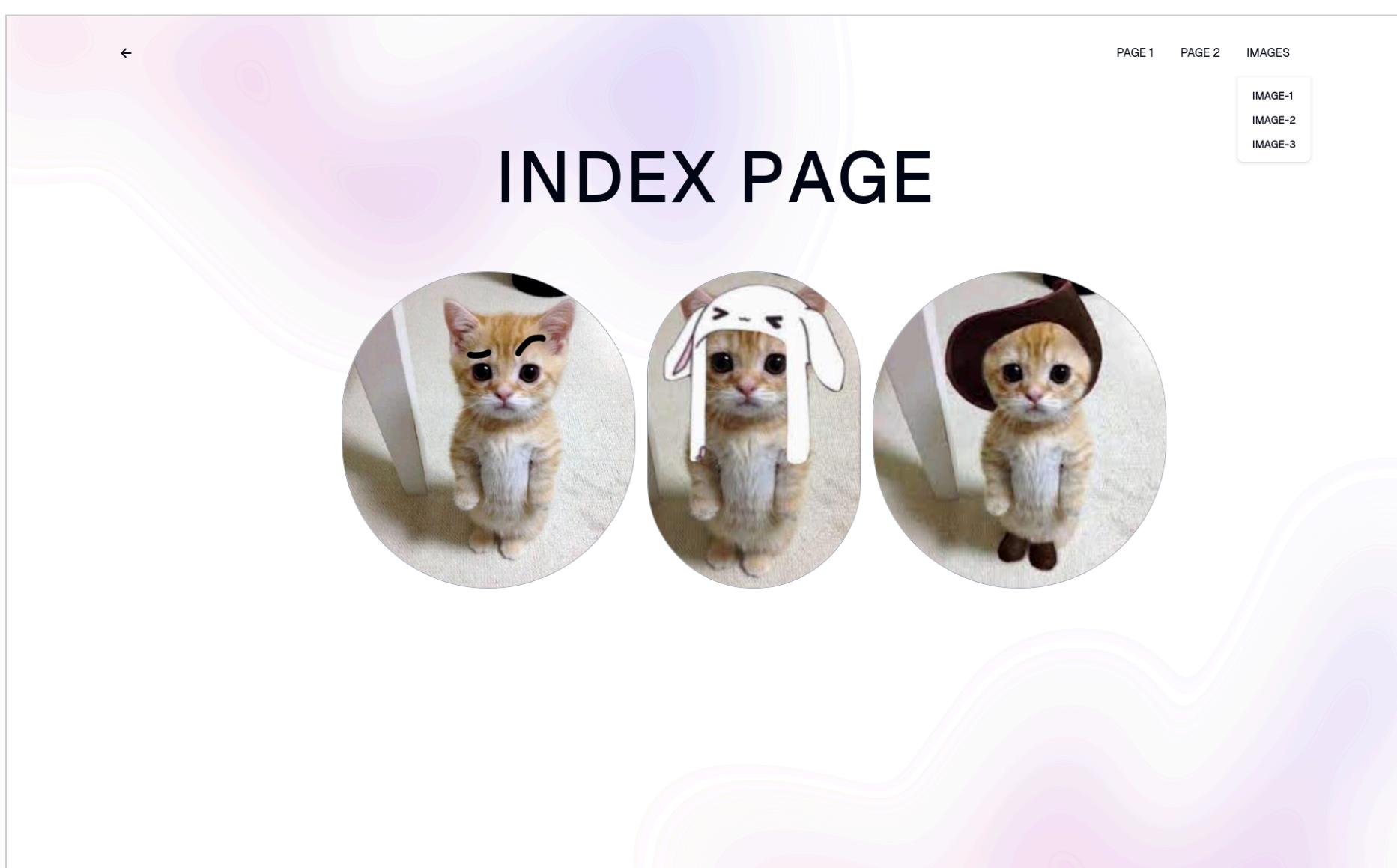
Screenshots Showing gsap animations

Finally, at the bottom of the page, I've added some basic **GSAP** animations to bring a bit of life to the page and keep it from feeling too static.

That concludes the content for **index.html** we will be following a similar pattern in the rest of the pages. This is how index.html looks:



Screenshot of index.html on my local machine



Screenshot of index.html with the dropdown expanded showing the 3 images I've used

Page1.html:

Since the question doesn't specify any particular content for this page, I've kept the layout almost identical to `index.html`. The main differences are the page title, updated navigation (which now links to `page2.html` and `index.html`), and the images—which I rotated 45 degrees to give the page a different visual feel compared to the index. The following is the only difference, removed lines are marked (-) and added lines are marked (+)

```

STDIN
1 --- index.html 2025-05-12 18:17:00.936390751 +0545
2 +++ page1.html 2025-05-12 18:10:56.561958119 +0545
3 @@ -4,7 +4,7 @@
4     <meta charset="UTF-8" />
5     <meta name="viewport" content="width=device-width, initial-scale=1.0" />
6     <link rel="stylesheet" href="/static/output.css" />
7 -     <title>Index 2431342</title>
8 +     <title>Page1 2431342</title>
9     </head>
10    <body class="m-4 h-screen overflow-hidden">
11
12 @@ -25,7 +25,7 @@
13     <div class="flex-1 justify-end flex">
14         <ul class="menu menu-horizontal px-1 flex gap-3">
15             <li>
16 -                 <a class="text-base font-medium uppercase nav-link tooltip tooltip-bottom" data-tip="Meow Rotate 45" href="/page1.html">Page 1</a>
17 +                 <a class="text-base font-medium uppercase nav-link tooltip tooltip-bottom" data-tip="Cat No Rotate" href="/index.html">Index</a>
18             </li>
19             <li>
20                 <a class="text-base font-medium uppercase nav-link tooltip tooltip-bottom" data-tip="Meow Rotate 135" href="/page2.html">Page 2</a>
21 @@ -46,13 +46,14 @@
22         </div>
23
24
25 +
26     <!-- Page Content -->
27     <div class="h-2/3 w-screen flex-col flex justify-center gap-20 items-center">
28 -         <h1 class="text-8xl font-bold uppercase tracking-wide hero-title"> Index Page </h1>
29 +         <h1 class="text-8xl font-bold uppercase tracking-wide hero-title"> Page 1</h1>
30         <div class="flex gap-4">
31 -             
32 -             
33 -             
34 +             
35 +             
36 +             
37         </div>
38     </div>
39
[0] 0:nvim 1:bunx- 2: bash*

```

Screenshot showing differences between `page1.html` and `index.html` using `diff` piped to `bat` for clarity.

The animations stays the same as `index.html`:

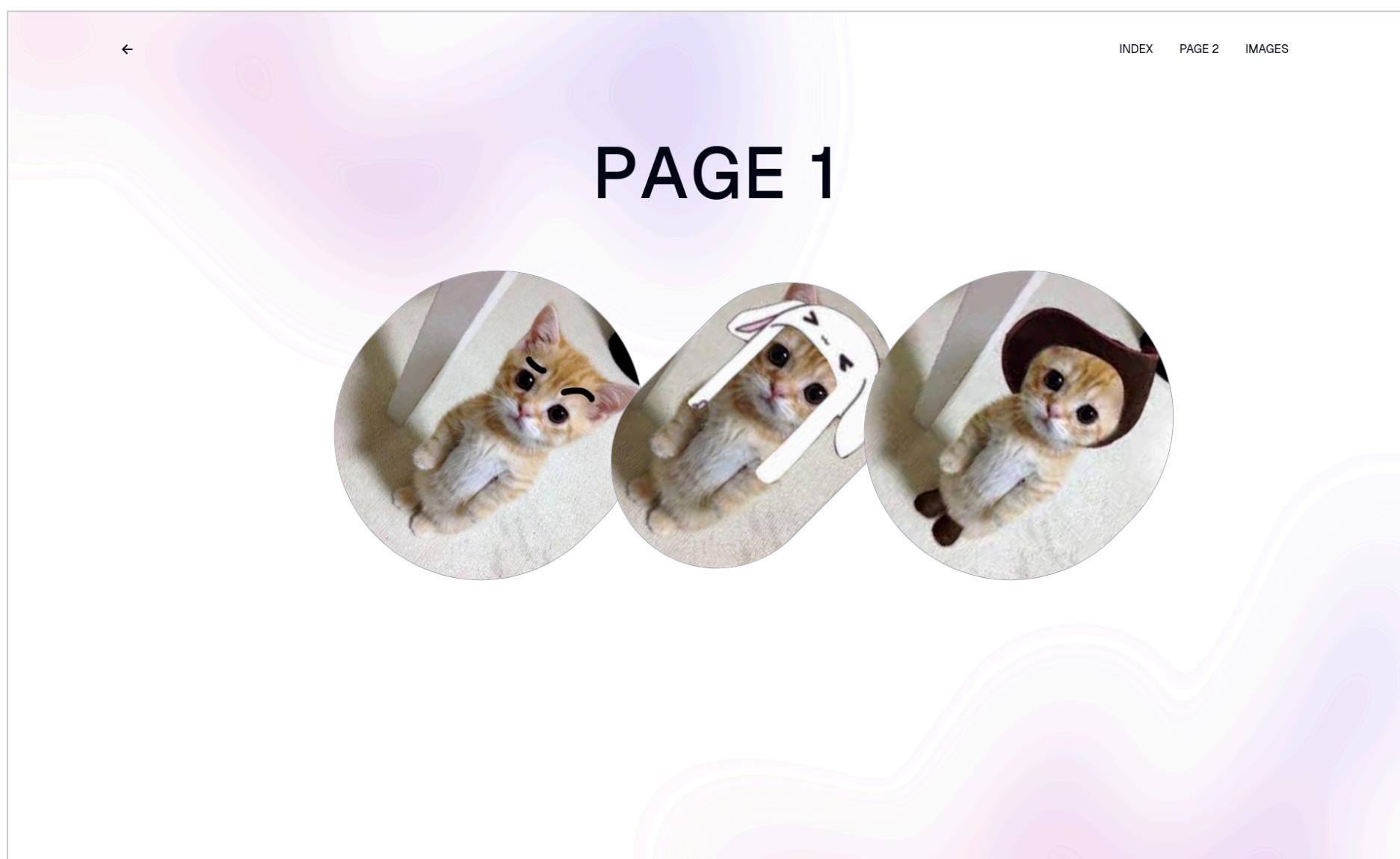
```

22   <script src="https://cdn.jsdelivr.net/npm/gsap@3.13.0/dist/gsap.min.js"></script>
21   <script src="https://cdn.jsdelivr.net/npm/gsap@3.13.0/dist/SplitText.min.js"></script>
20
19   <script>
18   |   document.addEventListener("DOMContentLoaded", () => {
17   |       const title = document.querySelector(".hero-title");
16   |       const split = new SplitText(title, { type: "chars" });
15   |
14   |       // hero title split text animation
13   |       gsap.from(split.chars, {
12   |           opacity: 0,
11   |           y: 50,
10   |           rotateX: 90,
9   |           stagger: 0.05,
8   |           ease: "back.out(1.7)",
7   |           duration: 1,
6   |       });
5   |
4   |       // navigation links fly in
3   |       gsap.from(gsap.utils.toArray(".nav-link"), {
2   |           y: -20,
1   |           opacity: 0,
88   |           stagger: 0.1,
1   |           delay: 0.3,
2   |           duration: 0.6,
3   |           ease: "power2.out"
4   |       });
5   |
6   |       // scale up them kittens
7   |       gsap.from(gsap.utils.toArray("img"), {
8   |           opacity: 0,
9   |           stagger: 0.2,
10  |           delay: 0.3,
11  |           scale: 0.2,
12  |           duration: 0.6,
13  |       });
14  |   });
15   </script>
16 </html>
index.html
"index.html" 104L, 5351B
[0] 0:nvim* 1:bunx-

```

Screenshot showing animation in `page1.html`, exactly the same as `index.html`

That's pretty much it for `page1.html`, as we've only changed the links and rotated the image. Here's how it looks now running locally on my machine:



Page2.html:

This page also shares a similar story to [page1.html](#). Again, the only differences are the page title, updated navigation (which now links to [page1.html](#) and [index.html](#)), and the images – which I rotated 135 degrees to give the page a different visual feel compared to both the [index.html](#) and [page1.html](#). The following is the only difference, removed lines are marked (-) and added lines are marked (+)

```
2     +++ page2.html 2025-05-12 18:10:01.269679160 +0545
3 @@ -4,7 +4,7 @@
4         <meta charset="UTF-8" />
5         <meta name="viewport" content="width=device-width, initial-scale=1.0" />
6         <link rel="stylesheet" href="/static/output.css" />
7 -         <title>Index 2431342</title>
8 +         <title>Page2 2431342</title>
9     </head>
10    <body class="m-4 h-screen overflow-hidden">
11
12 @@ -25,10 +25,10 @@
13         <div class="flex-1 justify-end flex">
14             <ul class="menu menu-horizontal px-1 flex gap-3">
15                 <li>
16 -                     <a class="text-base font-medium uppercase nav-link tooltip tooltip-bottom" data-tip="Meow Rotate 45" href="/page1.html">Page 1</a>
17 +                     <a class="text-base font-medium uppercase nav-link tooltip tooltip-bottom" data-tip="Cat No Rotate" href="/index.html">Index</a>
18                 </li>
19                 <li>
20 -                     <a class="text-base font-medium uppercase nav-link tooltip tooltip-bottom" data-tip="Meow Rotate 135" href="/page2.html">Page 2</a>
21 +                     <a class="text-base font-medium uppercase nav-link tooltip tooltip-bottom" data-tip="Meow Rotate 45" href="/page1.html">Page 1</a>
22                 </li>
23                 <li>
24                     <details>
25 @@ -46,13 +46,14 @@
26             </div>
27
28 +
29 +         <!-- Page Content -->
30         <div class="h-2/3 w-screen flex-col flex justify-center gap-20 items-center">
31 -             <h1 class="text-8xl font-bold uppercase tracking-wide hero-title"> Index Page </h1>
32 +             <h1 class="text-8xl font-bold uppercase tracking-wide hero-title"> Page 2</h1>
33         <div class="flex gap-4">
34 -             
35 -             
36 -             
37 +             
38 +             
39 +             
40         </div>
41     </div>
42 </div>
:
```

Screenshot showing differences between page2.html and index.html using diff piped to bat for clarity.

The animations again are unchanged:

```

22  <script src="https://cdn.jsdelivr.net/npm/gsap@3.13.0/dist/gsap.min.js"></script>
21  <script src="https://cdn.jsdelivr.net/npm/gsap@3.13.0/dist/SplitText.min.js"></script>
20
19  <script>
18  |   document.addEventListener("DOMContentLoaded", () => {
17  |     const title = document.querySelector(".hero-title");
16  |     const split = new SplitText(title, { type: "chars" });
15  |
14  |     // hero title split text animation
13  |     gsap.from(split.chars, {
12  |       opacity: 0,
11  |       y: 50,
10  |       rotateX: 90,
9  |       stagger: 0.05,
8  |       ease: "back.out(1.7)",
7  |       duration: 1,
6  |     });
5  |
4  |     // navigation links fly in
3  |     gsap.from(gsap.utils.toArray(".nav-link"), {
2  |       y: -20,
1  |       opacity: 0,
88  |       stagger: 0.1,
1  |       delay: 0.3,
2  |       duration: 0.6,
3  |       ease: "power2.out"
4  |     });
5  |
6  |     // scale up them kittens
7  |     gsap.from(gsap.utils.toArray("img"), {
8  |       opacity: 0,
9  |       stagger: 0.2,
10 |       delay: 0.3,
11 |       scale: 0.2,
12 |       duration: 0.6,
13 |     });
14  |   });
15  </script>
16 </html>
index.html
"index.html" 104L, 5351B
[0] 0:nvim* 1:bunx-

```

Screenshot showing animation in page2.html, exactly the same as index.html

That's also it for [page2.html](#). Here is how the final version looks locally running on my machine:



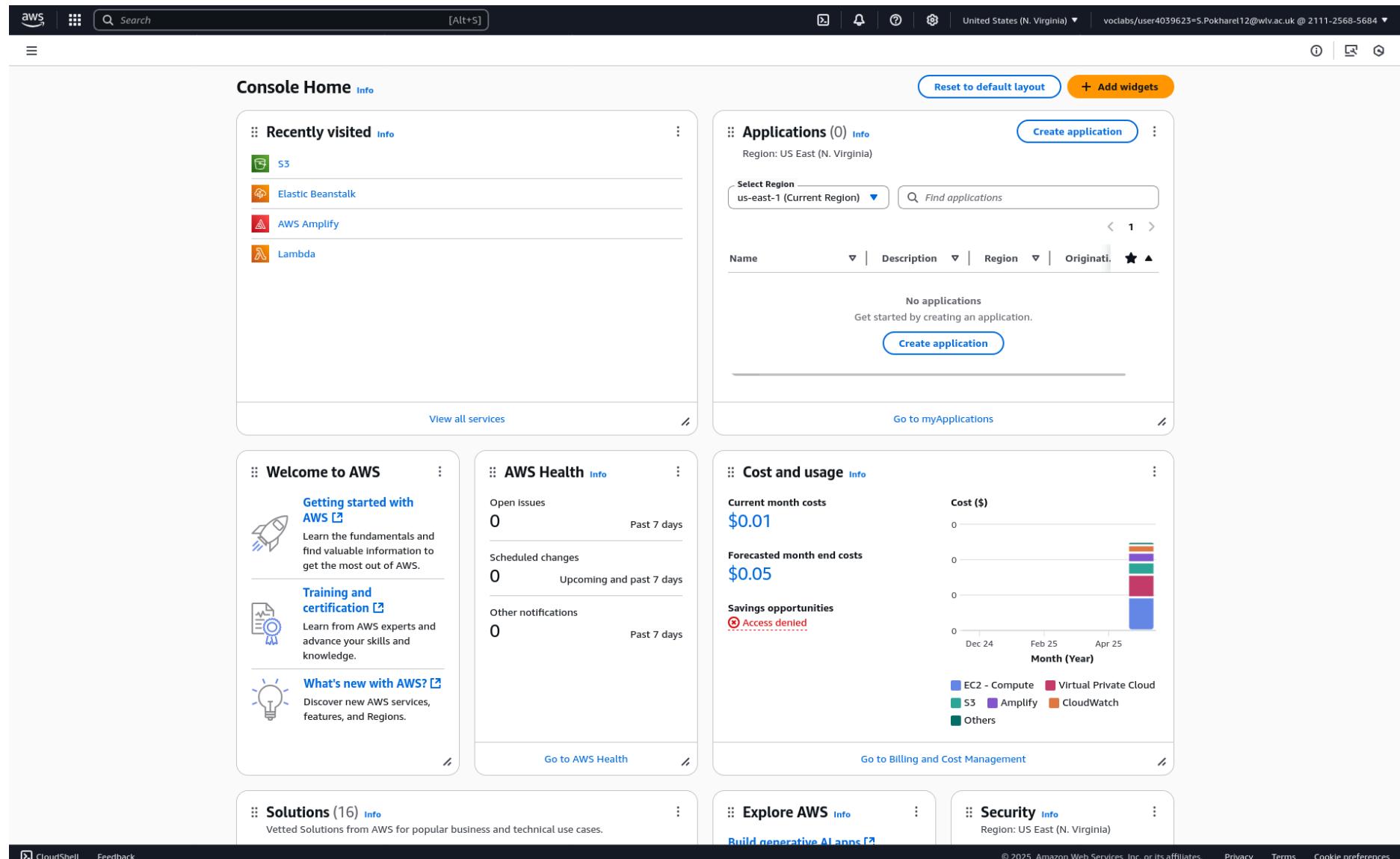
That will be all for the coding side of things, now moving on to hosting.

4. Deployment:

S3 Bucket & Static Hosting:

We will firstly be uploading the files and creating a static website on S3.

I: Navigating To AWS:

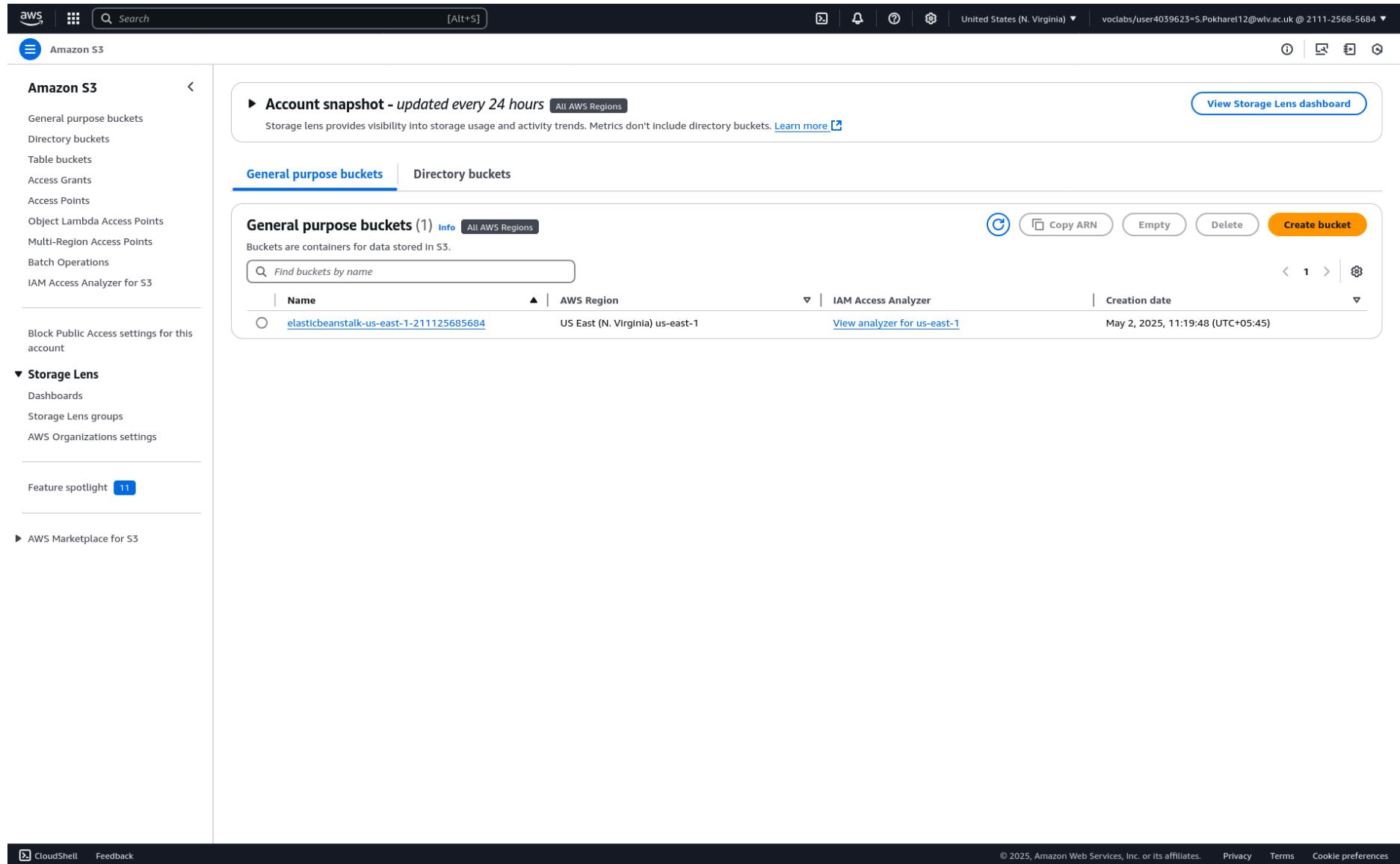


The screenshot shows the AWS Console Home page. At the top, there's a search bar and navigation links for S3, Elastic Beanstalk, AWS Amplify, and Lambda under 'Recently visited'. Below this, there are sections for 'Applications' (0), 'Welcome to AWS' (with links to Getting started with AWS, Training and certification, and What's new with AWS?), 'AWS Health' (with Open Issues, Scheduled changes, and Other notifications), 'Cost and usage' (showing current month costs of \$0.01 and a forecasted end-of-month cost of \$0.05, with a bar chart for EC2, S3, Amplify, CloudWatch, and Others), 'Solutions' (16), 'Explore AWS' (with a link to Build generative AI apps), and 'Security' (Region: US East (N. Virginia)). The bottom of the page includes links for CloudShell, Feedback, and various legal and preference links.

Screenshot showing my AWS landing page

Conveniently enough, there is a link to S3 in my recently visited tab. We now navigate to Amazon S3.

II: Navigating To S3:



The screenshot shows the AWS S3 console with the 'General purpose buckets' tab selected. A single bucket, 'elasticbeanstalk-us-east-1-211125685684', is listed. The bucket details are as follows:

Name	AWS Region	IAM Access Analyzer	Creation date
elasticbeanstalk-us-east-1-211125685684	US East (N. Virginia) us-east-1	View analyzer for us-east-1	May 2, 2025, 11:19:48 (UTC+05:45)

Screenshot showing my buckets

Here, we can see that we have an instance of Elastic Beanstalk running, which we won't be needing. We will be creating a new bucket for this specific task.

III: Creating a new bucket

Create bucket Info

Buckets are containers for data stored in S3.

General configuration

AWS Region
US East (N. Virginia) us-east-1

Bucket type Info

- General purpose
Recommended for most use cases and access patterns. General purpose buckets are the original S3 bucket type. They allow a mix of storage classes that redundantly store objects across multiple Availability Zones.
- Directory
Recommended for low-latency use cases. These buckets use only the S3 Express One Zone storage class, which provides faster processing of data within a single Availability Zone.

Bucket name Info
swoyam-pokharel-2431342

Bucket names must be 3 to 63 characters and unique within the global namespace. Bucket names must also begin and end with a letter or number. Valid characters are a-z, 0-9, periods (.), and hyphens (-). [Learn More](#)

Copy settings from existing bucket - optional
Only the bucket settings in the following configuration are copied.

[Choose bucket](#)

Format: s3://bucket/prefix

Object Ownership

Info

Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can specify access to objects.

ACLs disabled (recommended)
All objects in this bucket are owned by this account. Access to this bucket and its objects is specified using only policies.

ACLs enabled
Objects in this bucket can be owned by other AWS accounts. Access to this bucket and its objects can be specified using ACLs.

⚠ We recommend disabling ACLs, unless you need to control access for each object individually or to have the object writer own the data they upload. Using a bucket policy instead of ACLs to share data with users outside of your account simplifies permissions management and auditing.

Object Ownership

Bucket owner preferred
If new objects written to this bucket specify the bucket-owner-full-control canned ACL, they are owned by the bucket owner. Otherwise, they are owned by the object writer.

Object writer
The object writer remains the object owner.

ⓘ If you want to enforce object ownership for new objects only, your bucket policy must specify that the bucket-owner-full-control canned ACL is required for object uploads. [Learn more](#)

Block Public Access settings for this bucket

[CloudShell](#) [Feedback](#)

© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

[Search](#) [Alt+S]

[United States \(N. Virginia\)](#) voclabs/user4039623=5.Pokharel12@wlv.ac.uk @ 2111-2568-5684

[Amazon S3](#) > [Buckets](#) > Create bucket

Block Public Access settings for this bucket

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to this bucket and its objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to this bucket or objects within, you can customize the individual settings below to suit your specific storage use cases. [Learn more](#)

Block all public access
Turning this setting on is the same as turning on all four settings below. Each of the following settings are independent of one another.

- Block public access to buckets and objects granted through new access control lists (ACLs)**
S3 will block public access permissions applied to newly added buckets or objects, and prevent the creation of new public access ACLs for existing buckets and objects. This setting doesn't change any existing permissions that allow public access to S3 resources using ACLs.
- Block public access to buckets and objects granted through any access control lists (ACLs)**
S3 will ignore all ACLs that grant public access to buckets and objects.
- Block public access to buckets and objects granted through new public bucket or access point policies**
S3 will block new bucket and access point policies that grant public access to buckets and objects. This setting doesn't change any existing policies that allow public access to S3 resources.
- Block public and cross-account access to buckets and objects through any public bucket or access point policies**
S3 will ignore public and cross-account access for buckets or access points with policies that grant public access to buckets and objects.

⚠ Turning off block all public access might result in this bucket and the objects within becoming public
AWS recommends that you turn on block all public access, unless public access is required for specific and verified use cases such as static website hosting.

I acknowledge that the current settings might result in this bucket and the objects within becoming public.

Bucket Versioning

Versioning is a means of keeping multiple variants of an object in the same bucket. You can use versioning to preserve, retrieve, and restore every version of every object stored in your Amazon S3 bucket. With versioning, you can easily recover from both unintended user actions and application failures. [Learn more](#)

Bucket Versioning

Disable

Enable

Tags - optional (0)

You can use bucket tags to track storage costs and organize buckets. [Learn more](#)

No tags associated with this bucket.

[Add tag](#)

Default encryption

Info

Server-side encryption is automatically applied to new objects stored in this bucket.

Bucket Versioning

Versioning is a means of keeping multiple variants of an object in the same bucket. You can use versioning to preserve, retrieve, and restore every version of every object stored in your Amazon S3 bucket. With versioning, you can easily recover from both unintended user actions and application failures. [Learn more](#)

Bucket Versioning

Disable
 Enable

Tags - optional (0)

You can use bucket tags to track storage costs and organize buckets. [Learn more](#)

No tags associated with this bucket.

[Add tag](#)

Default encryption [Info](#)

Server-side encryption is automatically applied to new objects stored in this bucket.

Encryption type [Info](#)

Server-side encryption with Amazon S3 managed keys (SSE-S3)
 Server-side encryption with AWS Key Management Service keys (SSE-KMS)
 Dual-layer server-side encryption with AWS Key Management Service keys (DSSE-KMS)
 Secure your objects with two separate layers of encryption. For details on pricing, see DSSE-KMS pricing on the Storage tab of the [Amazon S3 pricing page](#).

Bucket Key

Using an S3 Bucket Key for SSE-KMS reduces encryption costs by lowering calls to AWS KMS. S3 Bucket Keys aren't supported for DSSE-KMS. [Learn more](#)

Disable
 Enable

Advanced settings

[After creating the bucket, you can upload files and folders to the bucket, and configure additional bucket settings.](#)

[Cancel](#) [Create bucket](#)

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Screenshot of the bucket creation page with filled inputs

Here, we specify the name of our bucket, in my case `swoyam-pokharel-2431342`; we enable ACLs and uncheck **Block Public Access**, because we want our statically hosted website to be accessible to anyone. We then acknowledge that change and leave the rest of options as is.

Success Successfully created bucket "swoyam-pokharel-2431342"
 To upload files and folders, or to configure additional bucket settings, choose [View details](#).

Account snapshot - updated every 24 hours [All AWS Regions](#)

Storage lens provides visibility into storage usage and activity trends. Metrics don't include directory buckets. [Learn more](#)

[View Storage Lens dashboard](#)

[General purpose buckets](#) [Directory buckets](#)

General purpose buckets (2) [Info](#) [All AWS Regions](#)

Buckets are containers for data stored in S3.

Name	AWS Region	IAM Access Analyzer	Creation date
elasticbeanstalk-us-east-1-211125685684	US East (N. Virginia) us-east-1	View analyzer for us-east-1	May 2, 2025, 11:19:48 (UTC+05:45)
swoyam-pokharel-2431342	US East (N. Virginia) us-east-1	View analyzer for us-east-1	May 12, 2025, 18:48:50 (UTC+05:45)

[Copy ARN](#) [Empty](#) [Delete](#) [Create bucket](#)

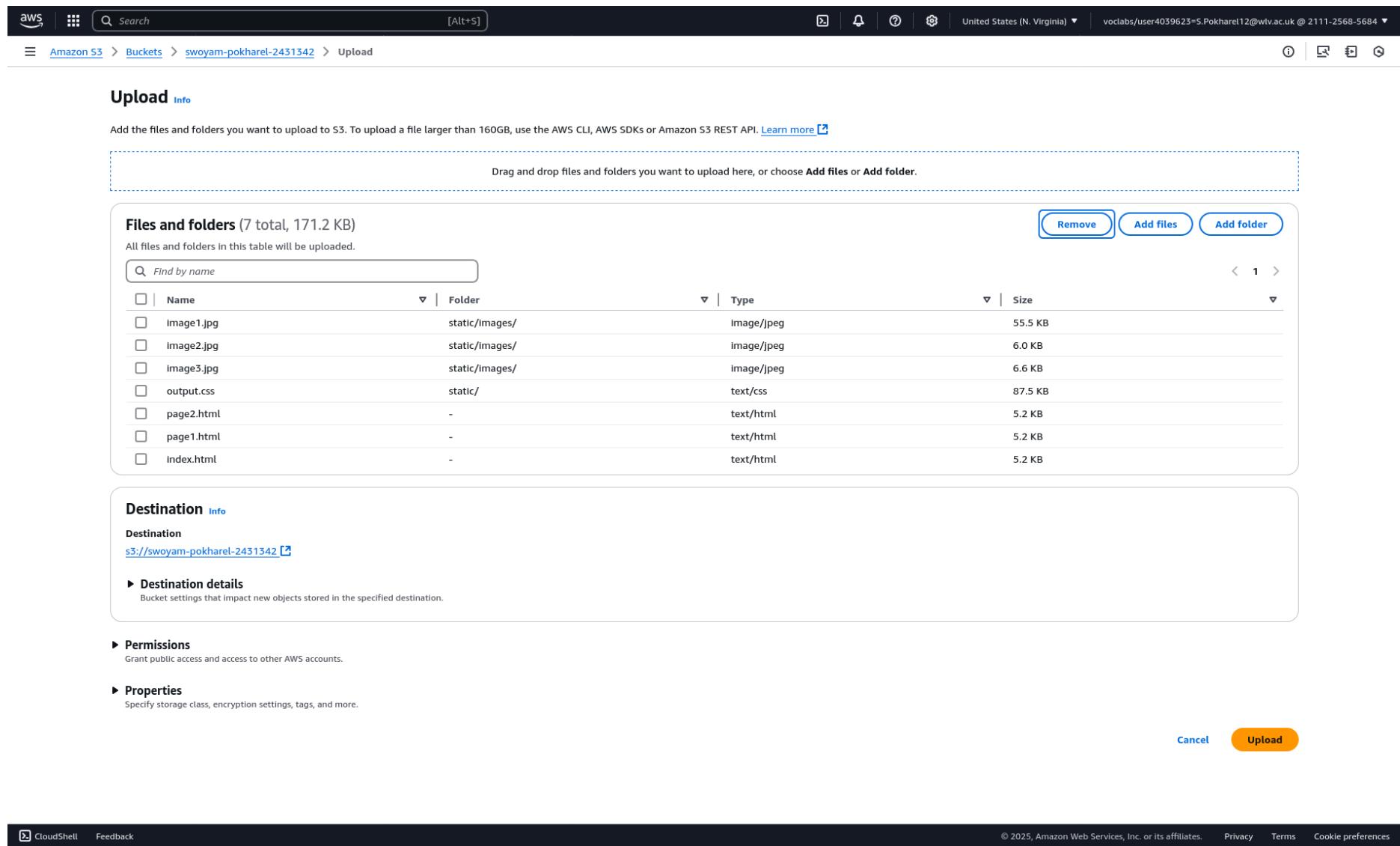
CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Screenshot Showing Successful Bucket Creation

Here, we can see that our bucket has been successfully created and now we may proceed to upload relevant files

IV: Uploading Relevant Files

In our case, we would like to upload all 3 of our html pages, all of the images and our **output.css** file. We do exactly that:



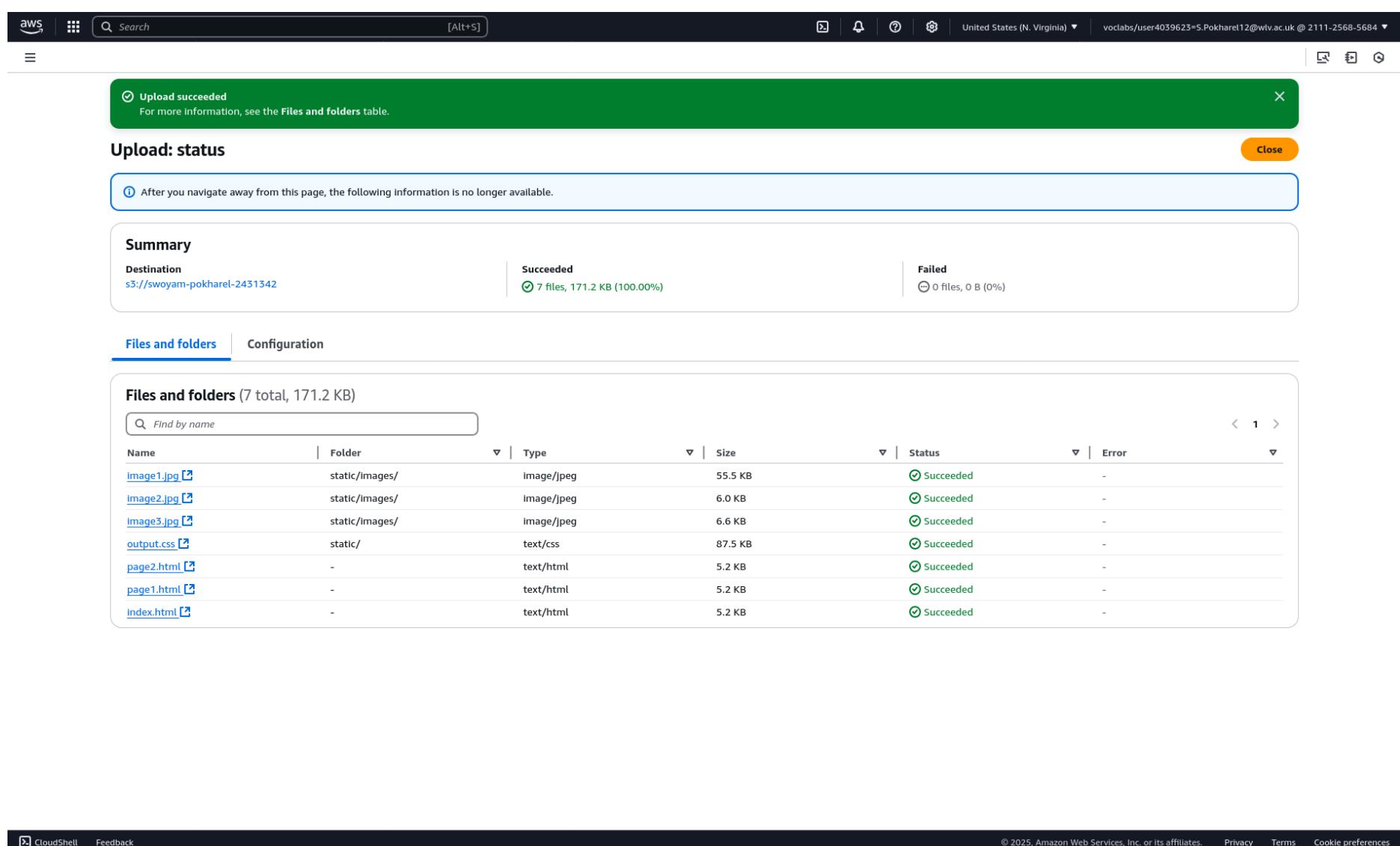
The screenshot shows the AWS S3 'Upload' interface. At the top, there's a search bar and navigation links for 'Amazon S3 > Buckets > swoyam-pokharel-2431342 > Upload'. The main area is titled 'Upload Info' with a sub-section 'Files and folders (7 total, 171.2 KB)'. It lists the following files:

Name	Folder	Type	Size
Image1.jpg	static/Images/	Image/jpeg	55.5 KB
Image2.jpg	static/Images/	Image/jpeg	6.0 KB
Image3.jpg	static/Images/	Image/jpeg	6.6 KB
output.css	static/	text/css	87.5 KB
page2.html	-	text/html	5.2 KB
page1.html	-	text/html	5.2 KB
index.html	-	text/html	5.2 KB

Below this is a 'Destination' section with a dropdown set to 's3://swoyam-pokharel-2431342'. There are sections for 'Destination details', 'Permissions', and 'Properties'. At the bottom right are 'Cancel' and 'Upload' buttons.

Screenshot showing files for upload

I dragged and dropped all the relevant files, the images and **output.css** lie inside **static/images/** and **static/** accordingly, while **{index,page1,page2}.html** remain in the project's root directory.



The screenshot shows the AWS S3 'Upload: status' page. A green banner at the top says 'Upload succeeded' with the note 'For more information, see the Files and folders table.' Below this is a summary table:

Destination	Succeeded	Failed
s3://swoyam-pokharel-2431342	7 files, 171.2 KB (100.00%)	0 files, 0 B (0%)

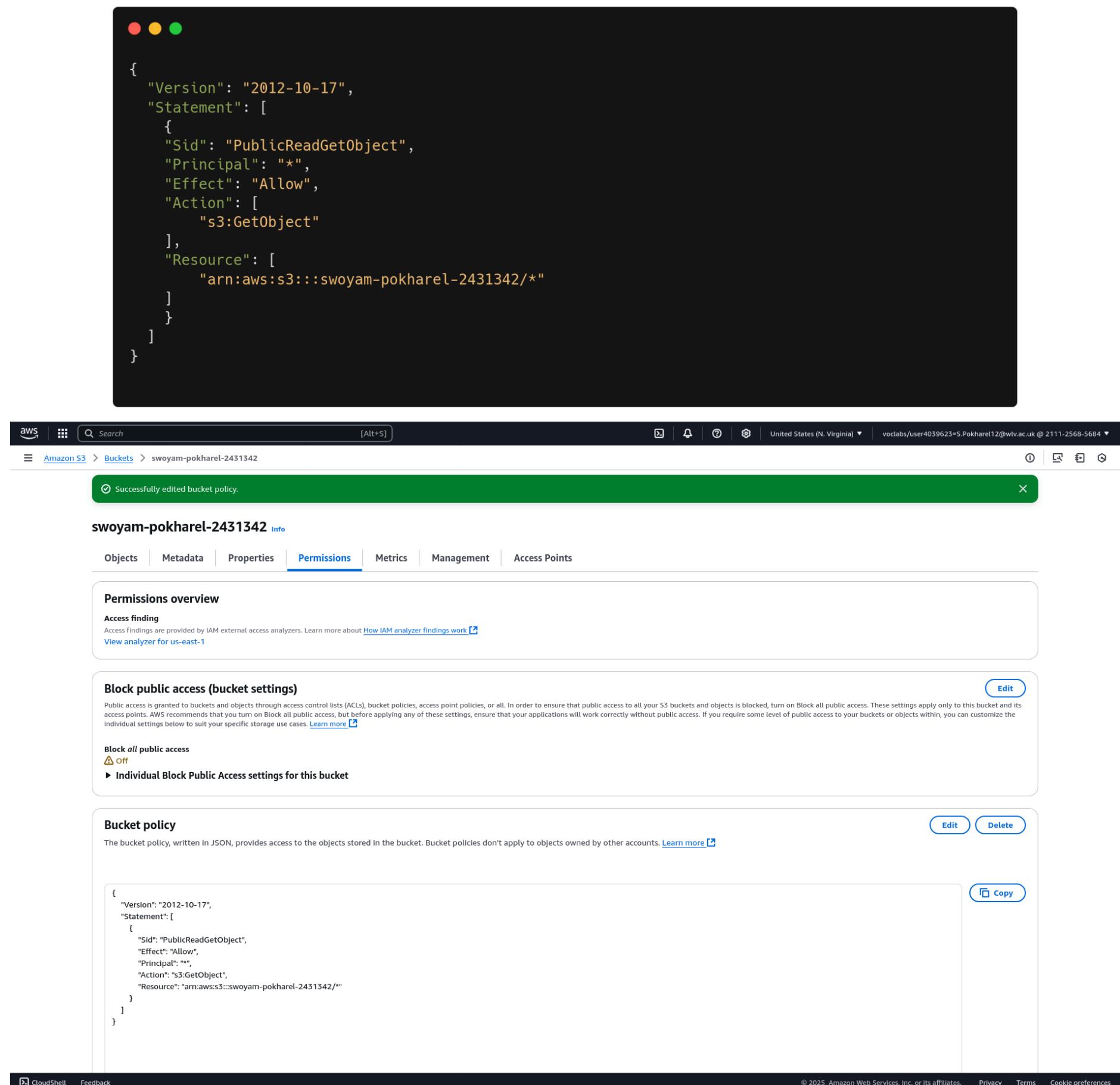
At the bottom, there are tabs for 'Files and folders' (selected) and 'Configuration'. The 'Files and folders' table shows the same 7 files as the upload screen, each with a 'Status' column showing 'Succeeded' with a green checkmark icon.

Screenshot showing successful upload

Here, we simply get confirmation that our files have been uploaded and now we may move on to editing the bucket policy.

V: Editing The Bucket Policy:

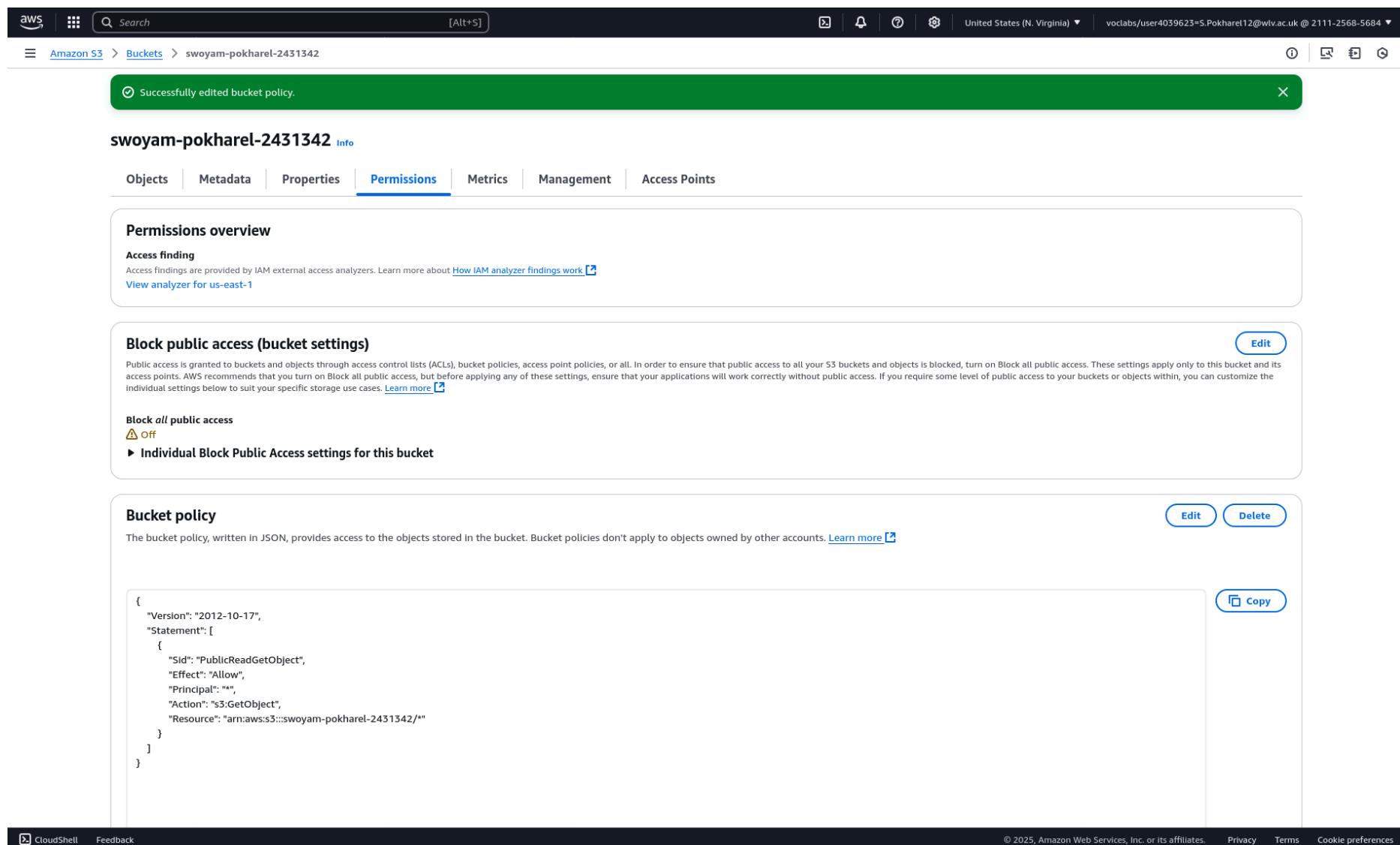
We now proceed to edit the bucket policy to:



The screenshot shows the AWS S3 console interface. At the top, there's a dark window containing a JSON policy document. Below it, the AWS navigation bar includes 'aws' and 'Search' fields, and a breadcrumb trail 'Amazon S3 > Buckets > swoyam-pokharel-2431342'. The main content area is titled 'swoyam-pokharel-2431342' with an 'Info' link. A green success message banner says 'Successfully edited bucket policy.' Below this, the 'Permissions' tab is selected, showing the 'Permissions overview' section with an 'Access finding' summary. The 'Block public access (bucket settings)' section has its status set to 'Off'. In the 'Bucket policy' section, the JSON policy code from the first screenshot is pasted into the editor, and a 'Copy' button is visible to its right. The bottom of the screen shows standard AWS footer links like CloudShell, Feedback, Privacy, Terms, and Cookie preferences.

Screenshots showing editing bucket policy.

We edit the bucket policy to allow public read access to the objects in the S3 bucket. In this case, the policy grants the `s3:GetObject` permission to everyone by using the wildcard `Principal: "*"`, meaning that everyone can access the files within the specified bucket. This is typically done when hosting a publicly accessible website like ours with static assets like images, CSS and HTML files in an S3 bucket. Without this policy, the default permissions would block public access to those files.



Successfully edited bucket policy.

swoyam-pokharel-2431342 [Info](#)

Objects | Metadata | Properties | **Permissions** | Metrics | Management | Access Points

Permissions overview

Access finding
Access findings are provided by IAM external access analyzers. Learn more about [How IAM analyzer findings work](#).

[View analyzer for us-east-1](#)

Block public access (bucket settings)

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to all your S3 buckets and objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to your buckets or objects within, you can customize the individual settings below to suit your specific storage use cases. [Learn more](#)

Block all public access
⚠ Off

► Individual Block Public Access settings for this bucket

Bucket policy

The bucket policy, written in JSON, provides access to the objects stored in the bucket. Bucket policies don't apply to objects owned by other accounts. [Learn more](#)

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PublicReadGetObject",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::swoyam-pokharel-2431342/*"
    }
  ]
}
```

[Edit](#) [Delete](#) [Copy](#)

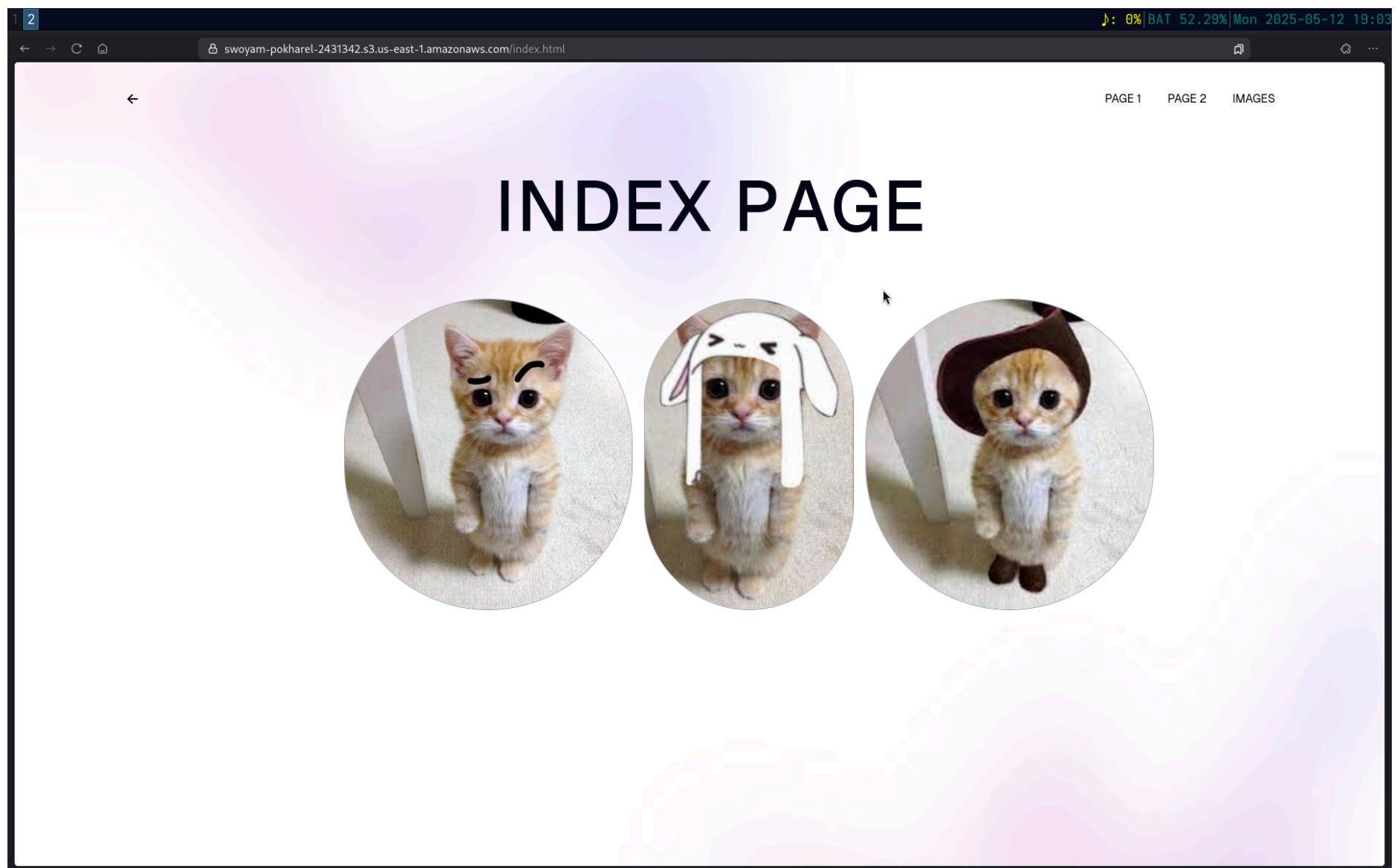
CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Screenshot showing confirmation

Here, we just get confirmation that our bucket policy has changed, which implies that now we can access the files inside this bucket via their object url. Let's see if that's the case in the next step.

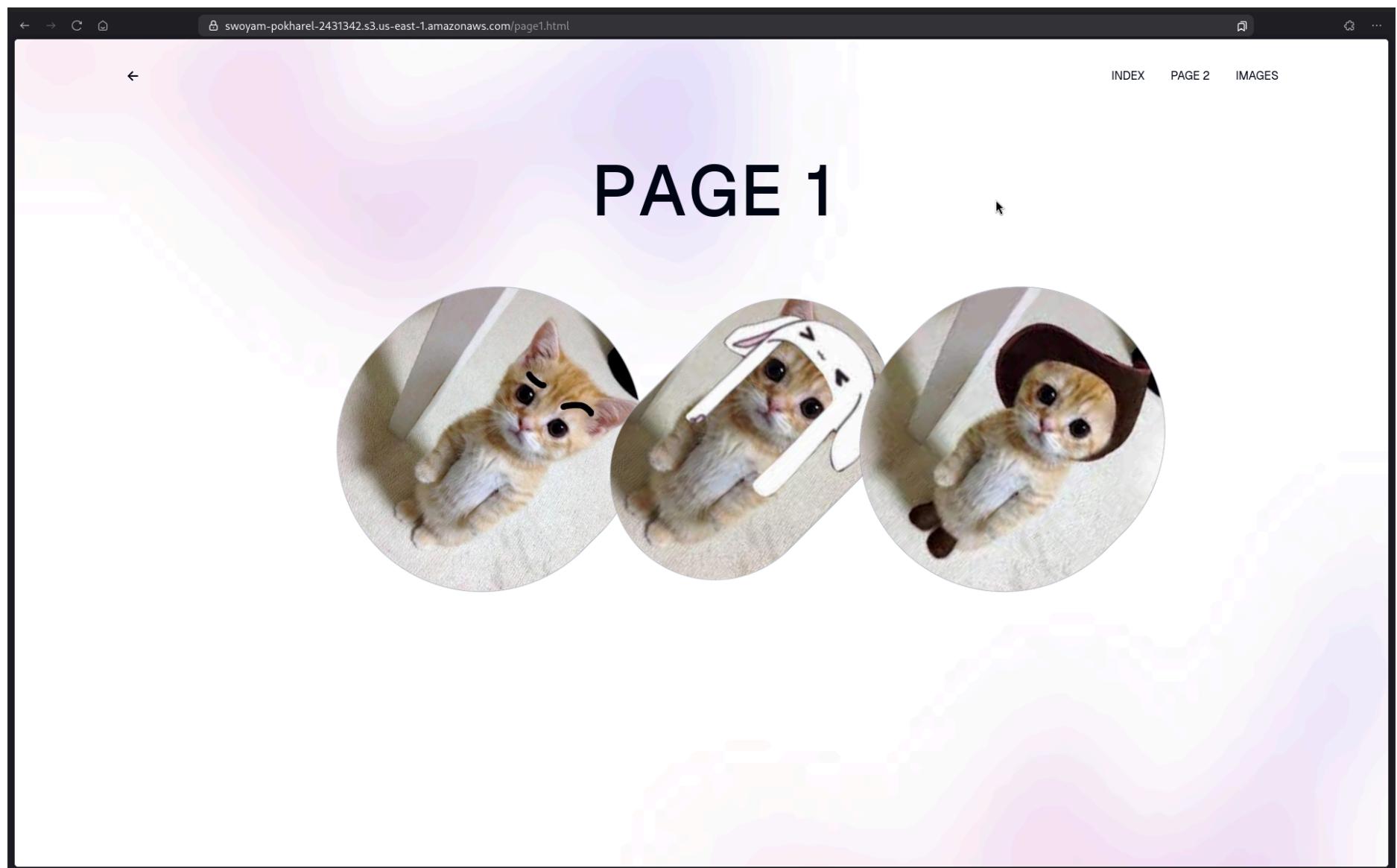
VI: Checking If Our Files Are Accessible Via Object URL:

So navigating to the [link](#) S3 gave us we can see the following:

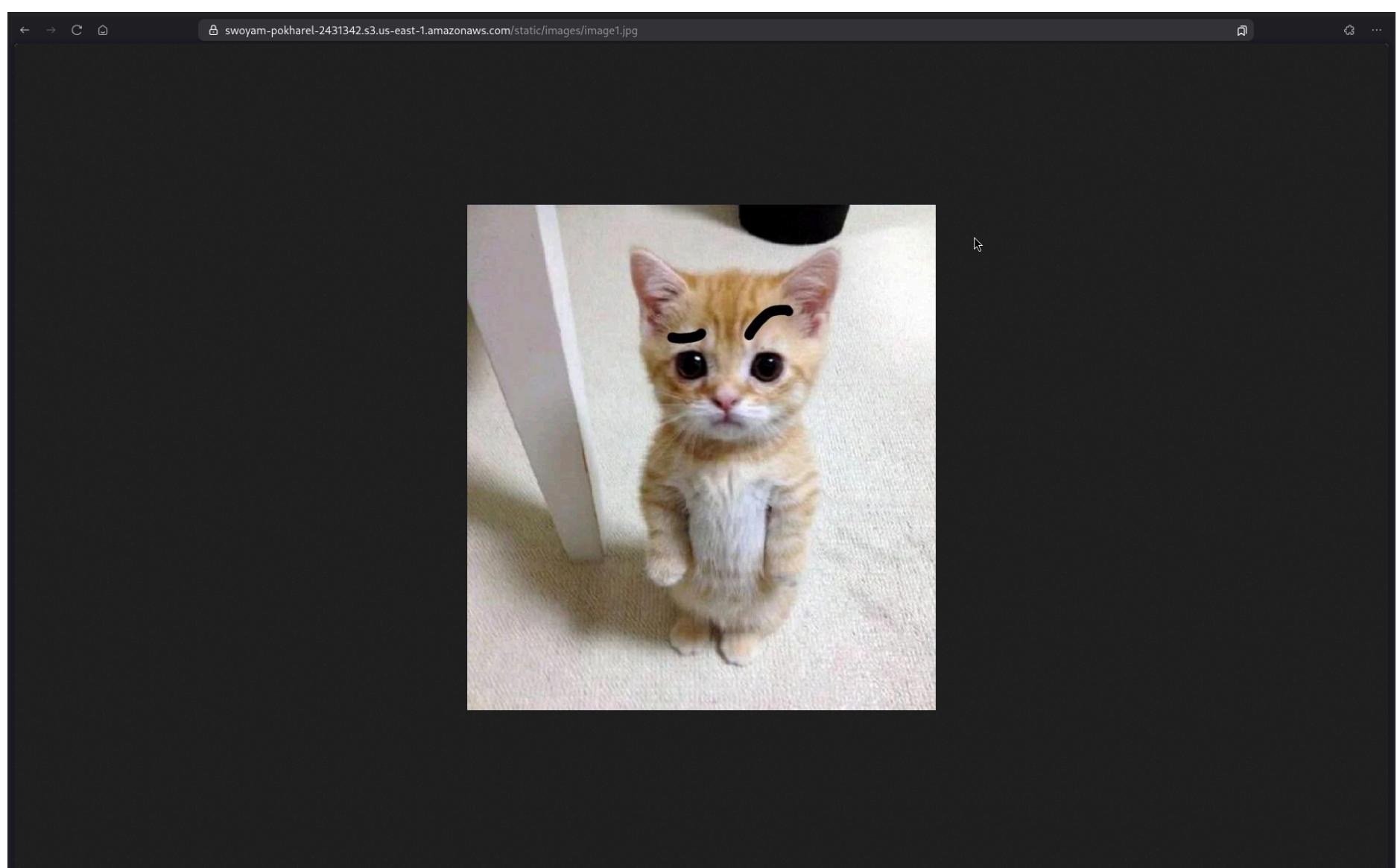


Screenshot showing index.html accessible via S3 Object URL

So, our bucket policy worked and now we can access [index.html](#) via the object url. Similarly the rest of the pages and the images are also accessible:



Screenshot showing page1.html accessible via S3 Object URL

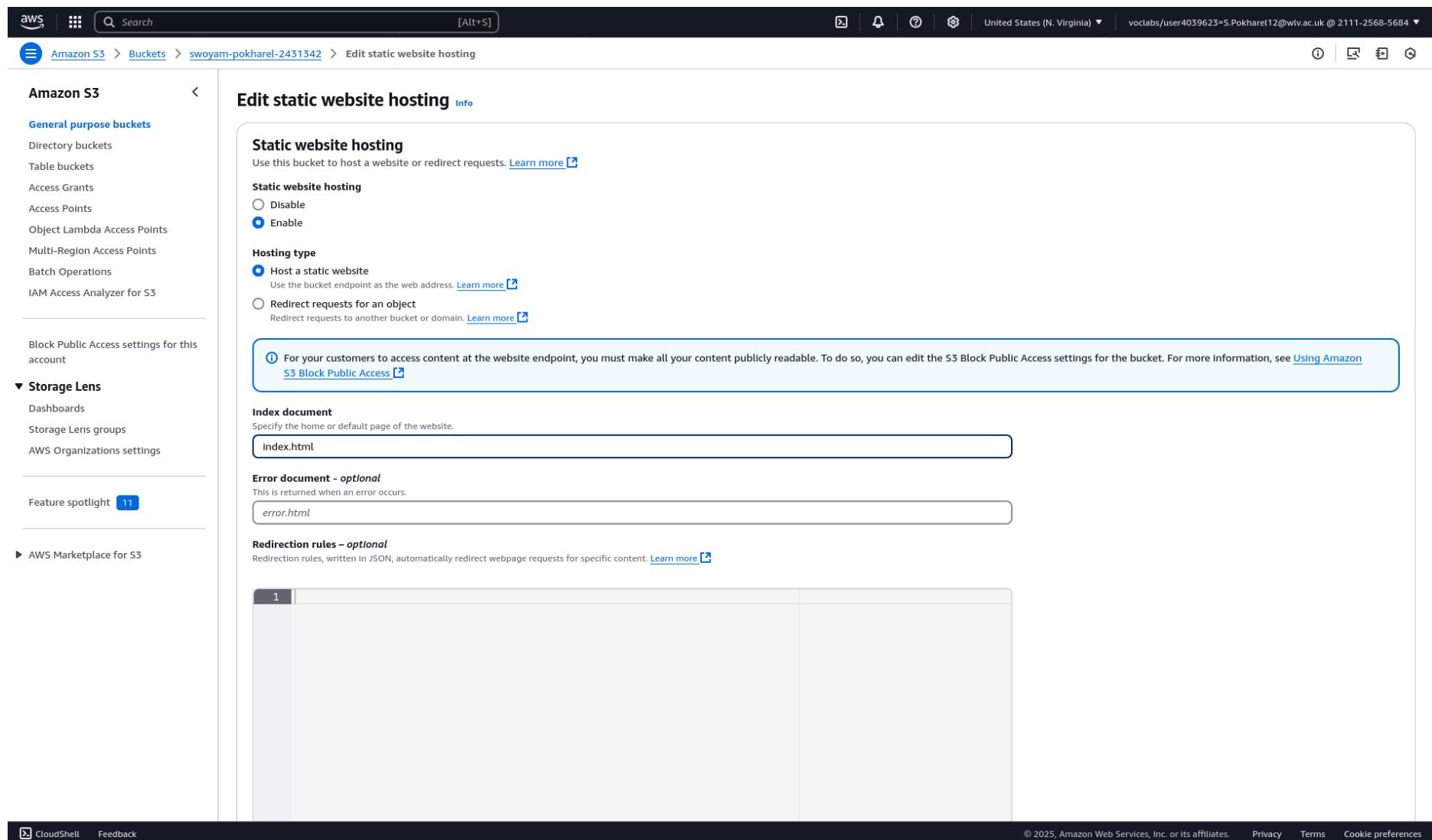


Screenshot showing images accessible via S3 Object URL

So now that it's evident that the bucket policy worked and we can access the objects in the S3 bucket, we proceed to enable static hosting in this bucket.

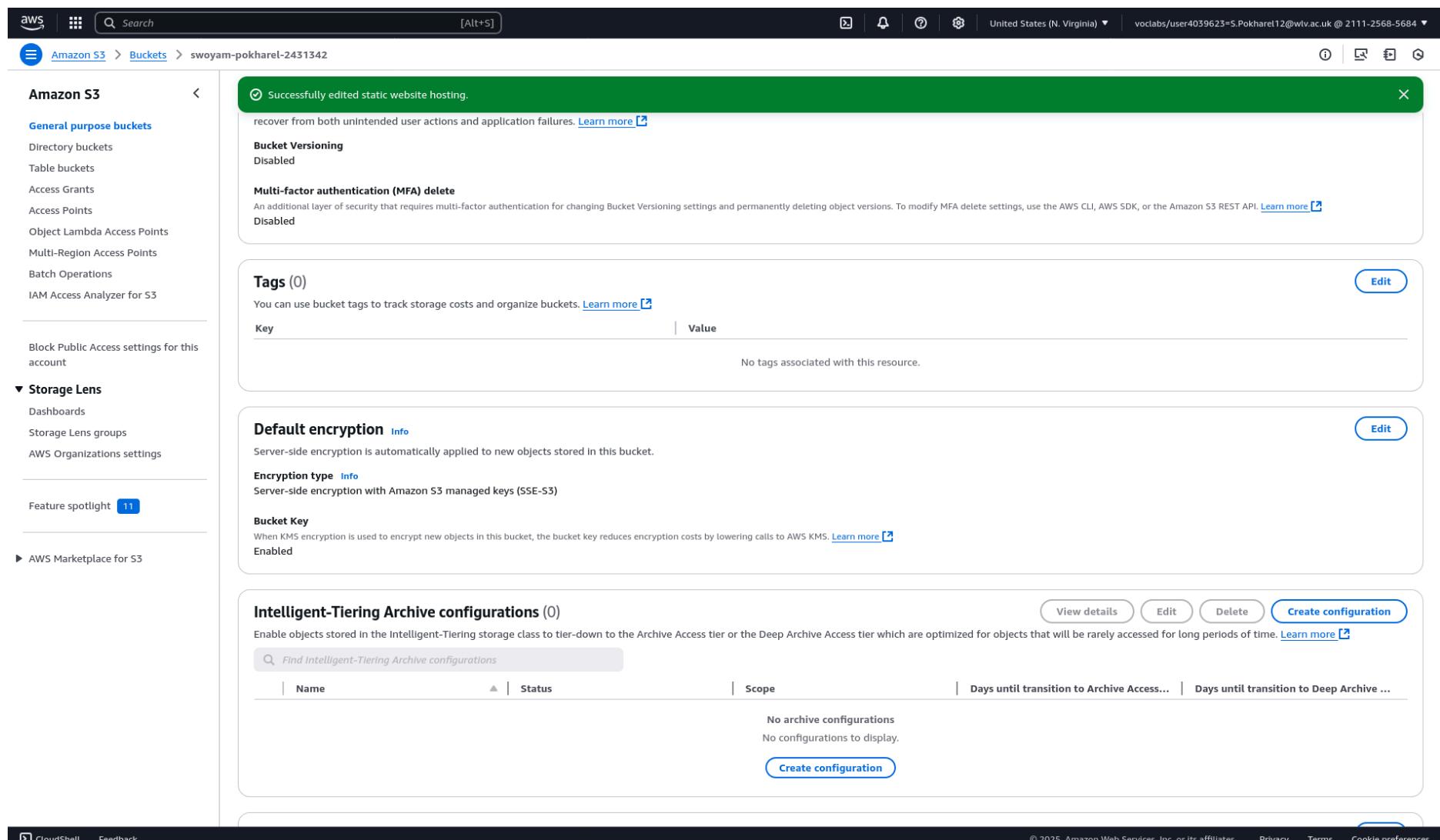
VII: Enabling Static Hosting:

Enabling static hosting is as simple as navigating all the way down in the properties tab and clicking enable static hosting. This brings us to the following page:



Screenshot Of Static Hosting Page

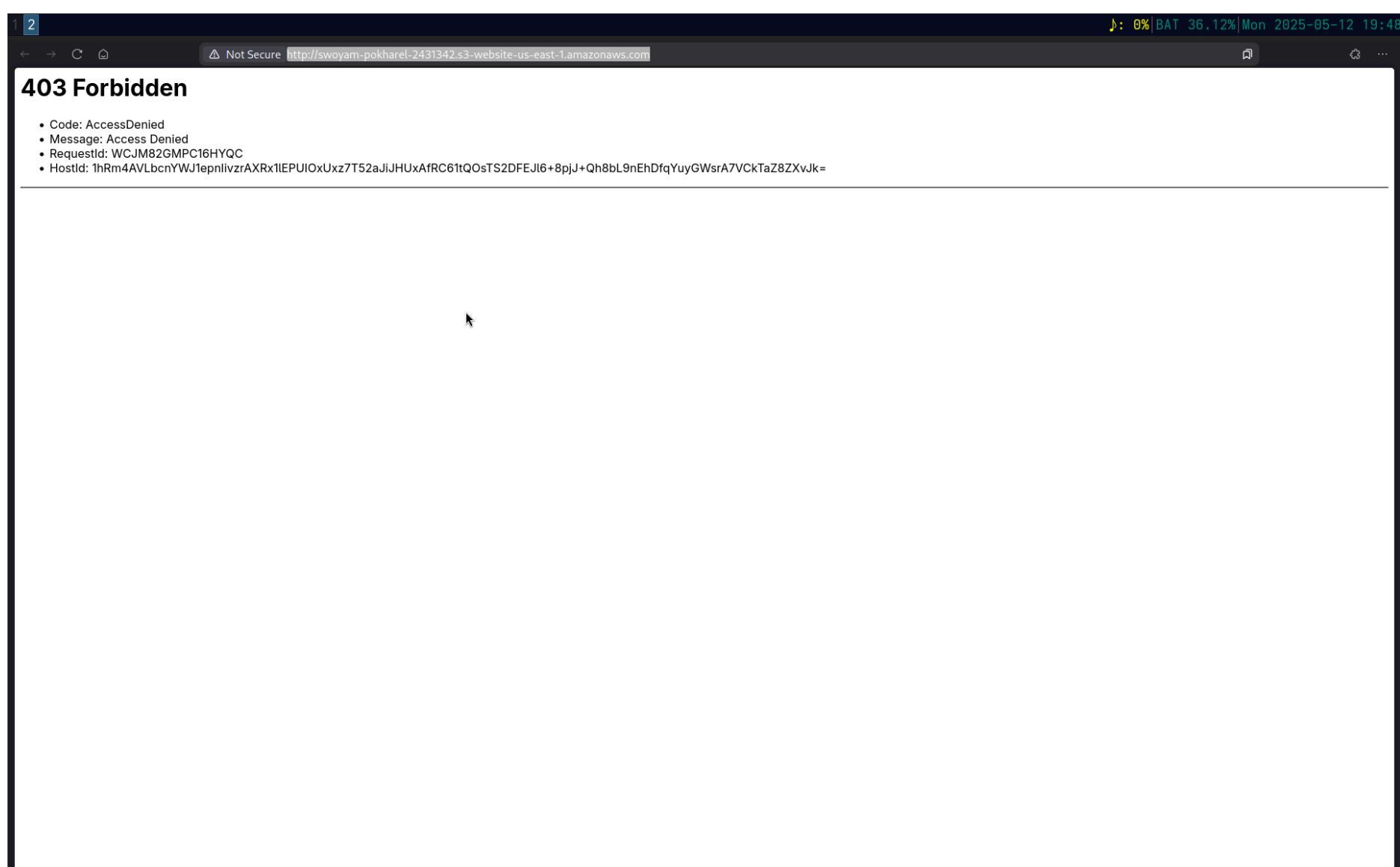
Here, we enable static website hosting and set `index.html` as the index document. This tells S3 which file to serve by default when someone accesses the bucket's static website URL. We also have the option to specify the error page, which we leave empty in this case. Now we simply save the changes:



The screenshot shows the AWS S3 Bucket configuration page for the bucket 'swoyam-pokharel-2431342'. A green success message at the top states 'Successfully edited static website hosting.' Below this, there are sections for 'Bucket Versioning' (Disabled), 'Multi-factor authentication (MFA) delete' (Disabled), and 'Tags (0)' (No tags associated with this resource). Under 'Default encryption' (Info), it says 'Server-side encryption is automatically applied to new objects stored in this bucket.' and 'Encryption type' (Info) shows 'Server-side encryption with Amazon S3 managed keys (SSE-S3)'. In the 'Bucket Key' section, it says 'When KMS encryption is used to encrypt new objects in this bucket, the bucket key reduces encryption costs by lowering calls to AWS KMS.' and 'Enabled'. The 'Intelligent-Tiering Archive configurations (0)' section has a 'Create configuration' button. The bottom of the page includes standard AWS navigation links like CloudShell, Feedback, and a footer with copyright information.

Screenshot showing successful save

Now that that's done, we should be able to navigate to the [provided url](#), but in doing so, we get hit with a

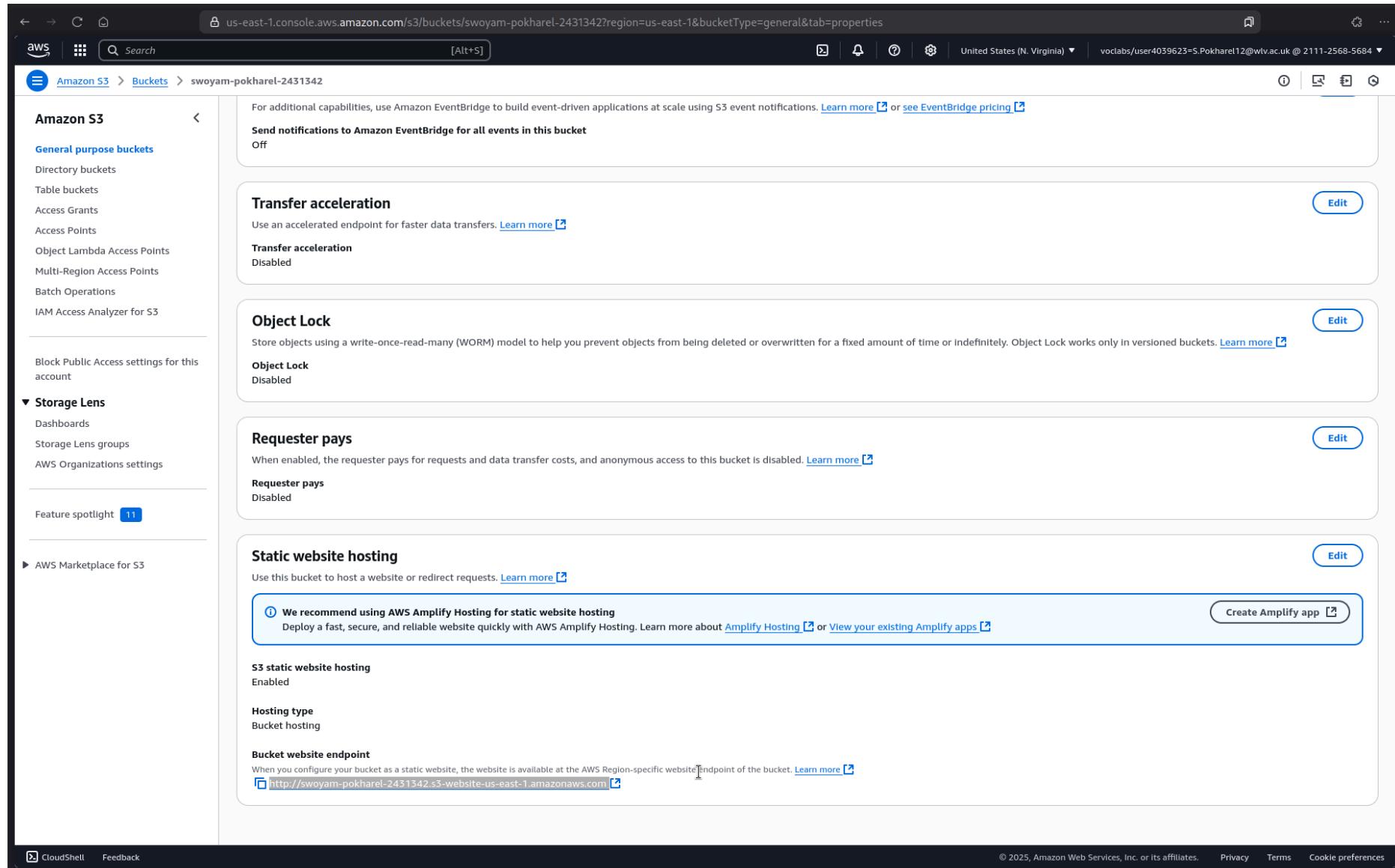


The screenshot shows a browser window displaying a 403 Forbidden error page. The URL in the address bar is 'Not Secure http://swoyam-pokharel-2431342.s3-website-us-east-1.amazonaws.com'. The error message '403 Forbidden' is prominently displayed. Below it, a list of details includes:

- Code: AccessDenied
- Message: Access Denied
- RequestID: WCJM82GMPG16HYQC
- HostId: 1hRm4AVLbcnYWJ1epnlivzrAXRx1lEPUIOxUxz7T52aJiJHUXAfRC61tQOsTS2DFEJ6+8pjJ+Qh8bL9nEhDfqYuyGwsrA7VCKTaZ8ZXvJk=

Screenshots showing 403 forbidden

For some reason, we get hit with a **403 Unauthorized**, Why is that? Well I did a little digging, and if we look closely to the highlighted link,

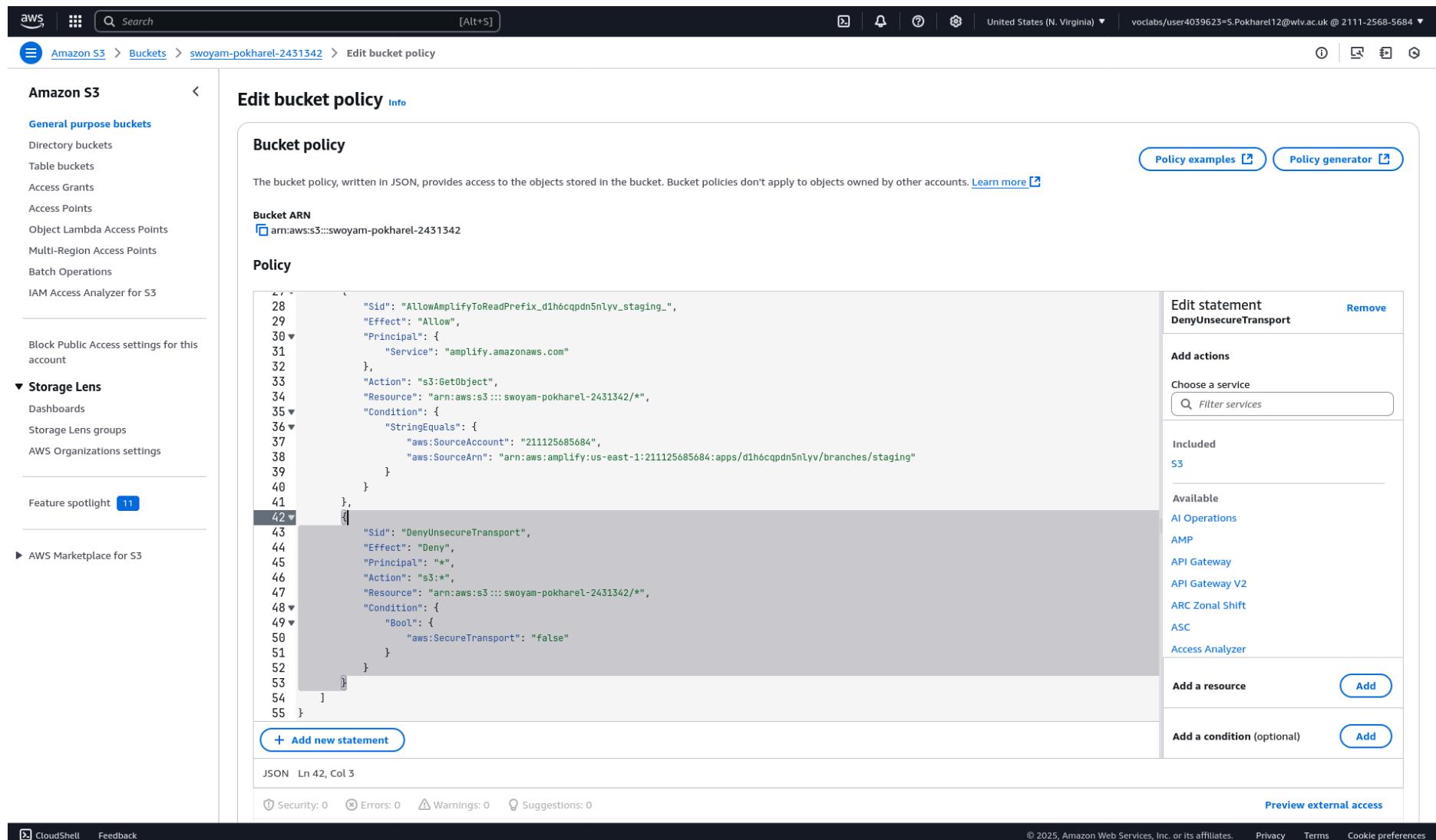


The screenshot shows the AWS S3 console interface. On the left, there's a sidebar with options like 'Amazon S3', 'General purpose buckets', 'Storage Lens', and 'AWS Marketplace for S3'. The main content area is titled 'swoyam-pokharel-2431342'. It contains several sections: 'Send notifications to Amazon EventBridge for all events in this bucket' (disabled), 'Transfer acceleration' (disabled), 'Object Lock' (disabled), 'Requester pays' (disabled), and 'Static website hosting'. The 'Static website hosting' section is expanded, showing a note about using AWS Amplify Hosting, the status 'Enabled', 'Hosting type' as 'Bucket hosting', and the 'Bucket website endpoint' as 'http://swoyam-pokharel-2431342.s3-website-us-east-1.amazonaws.com'.

Screenshot showing http link to statically hosted S3 bucket

We can see that the S3 statically hosted link is at:

<http://swoyam-pokharel-2431342.s3-website-us-east-1.amazonaws.com> which is **http**. The **http** in itself isn't the problem. The problem is that when S3 later populated the Bucket Policy, it added in:



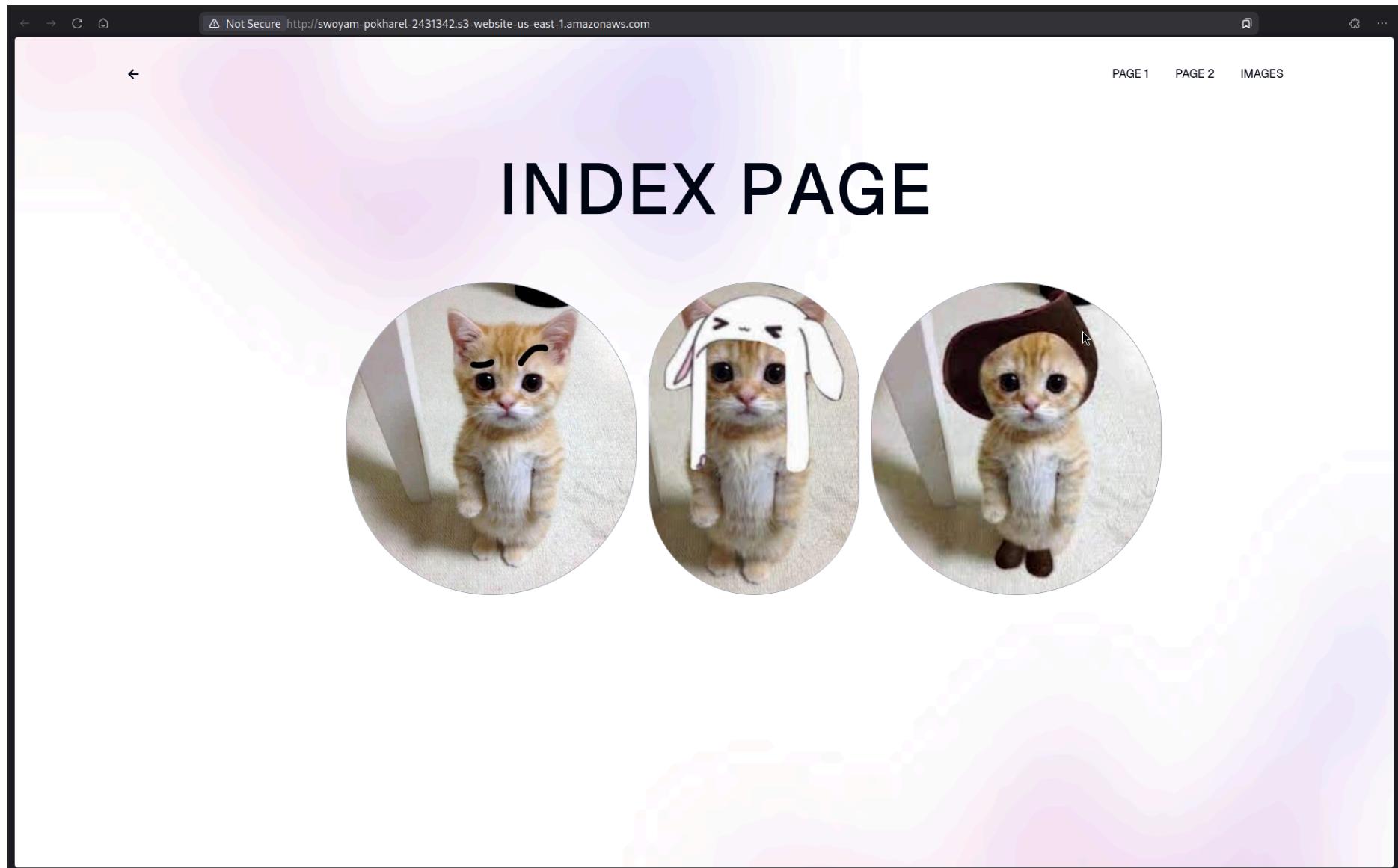
The screenshot shows the 'Edit bucket policy' page in the AWS S3 console. The left sidebar includes sections for General purpose buckets, Storage Lens, and AWS Marketplace for S3. The main area displays a JSON-based bucket policy. A specific line of code, line 42, is highlighted with a gray box. This line contains a 'DenyUnsecureTransport' statement. The right side of the screen shows a sidebar with service categories like IAM Operations, API Gateway, and Access Analyzer, along with buttons for adding resources or conditions.

```

    "Sid": "DenyUnsecureTransport",
    "Effect": "Deny",
    "Principal": "*",
    "Action": "s3:GetObject",
    "Resource": "arn:aws:s3:::swoyam-pokharel-2431342/*",
    "Condition": {
        "Bool": {
            "aws:SecureTransport": "false"
        }
    }
}
    
```

Screenshot of bucket policy that was causing the problem

If we look closely, it requires `aws::SecureTransport`, or in simpler words **https**. So the bucket policy was wanting **https** but the link to the static site itself was in **http**, that's our problem right there. Thankfully, it's an easy fix, simply removing that section from the json did the trick. Now, navigating to the [url](#) where our S3 bucket is statically hosted, we get:



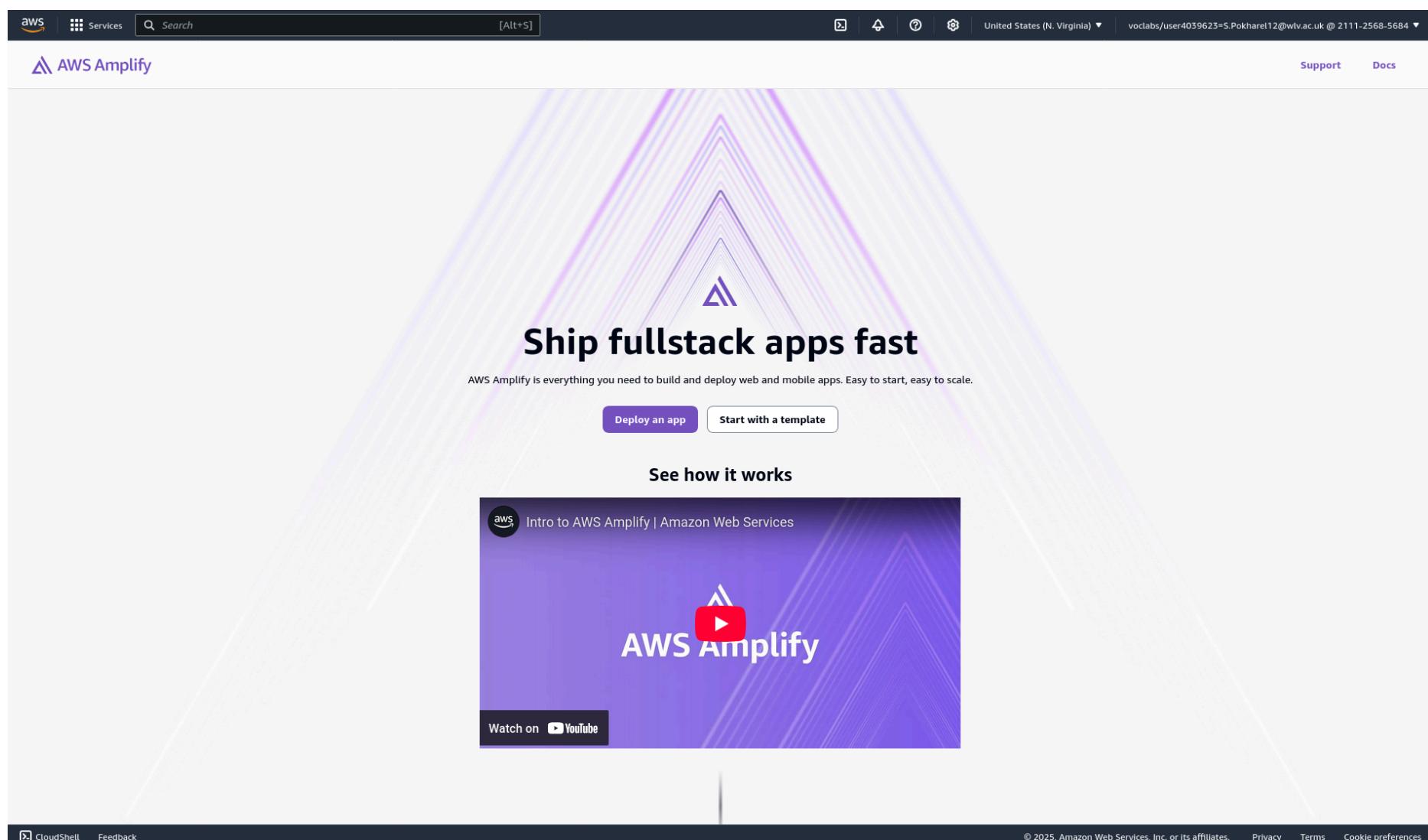
Successful static website hosting

Now that we have successfully hosted the static site, time to move on to amplify hosting.

AWS Amplify

Hosting on amplify is pretty straightforward, firstly we:

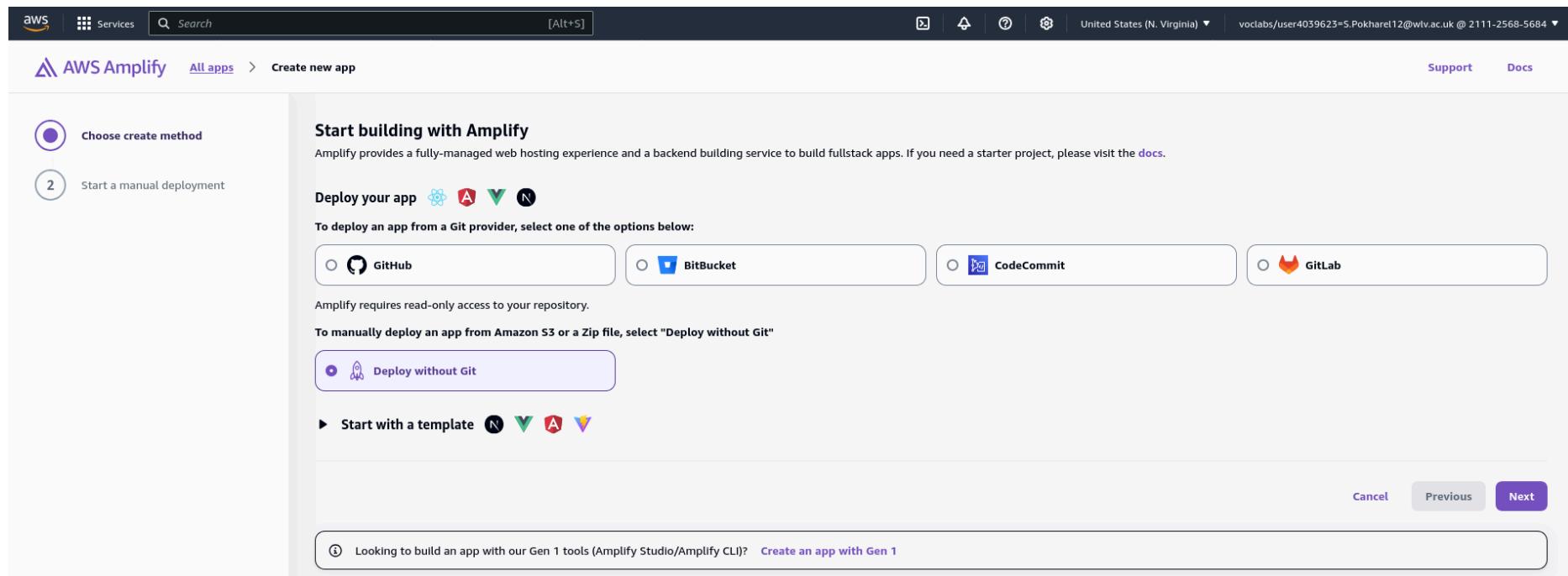
I: Navigating To AWS Amplify



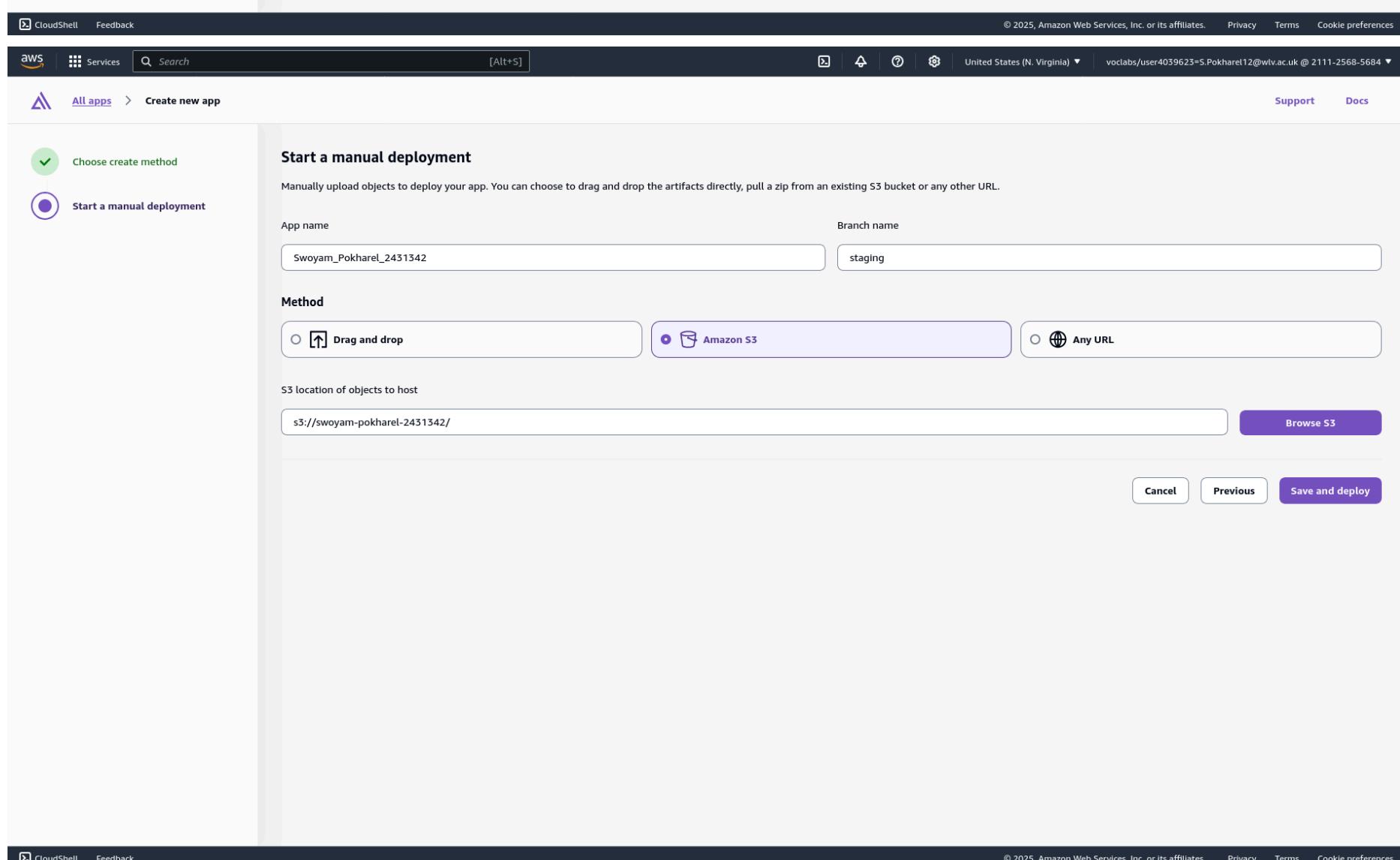
Screenshot of Amplify Landing Page

We navigate to the landing page of aws amplify and click on deploy an app

II: Deploying Amplify App



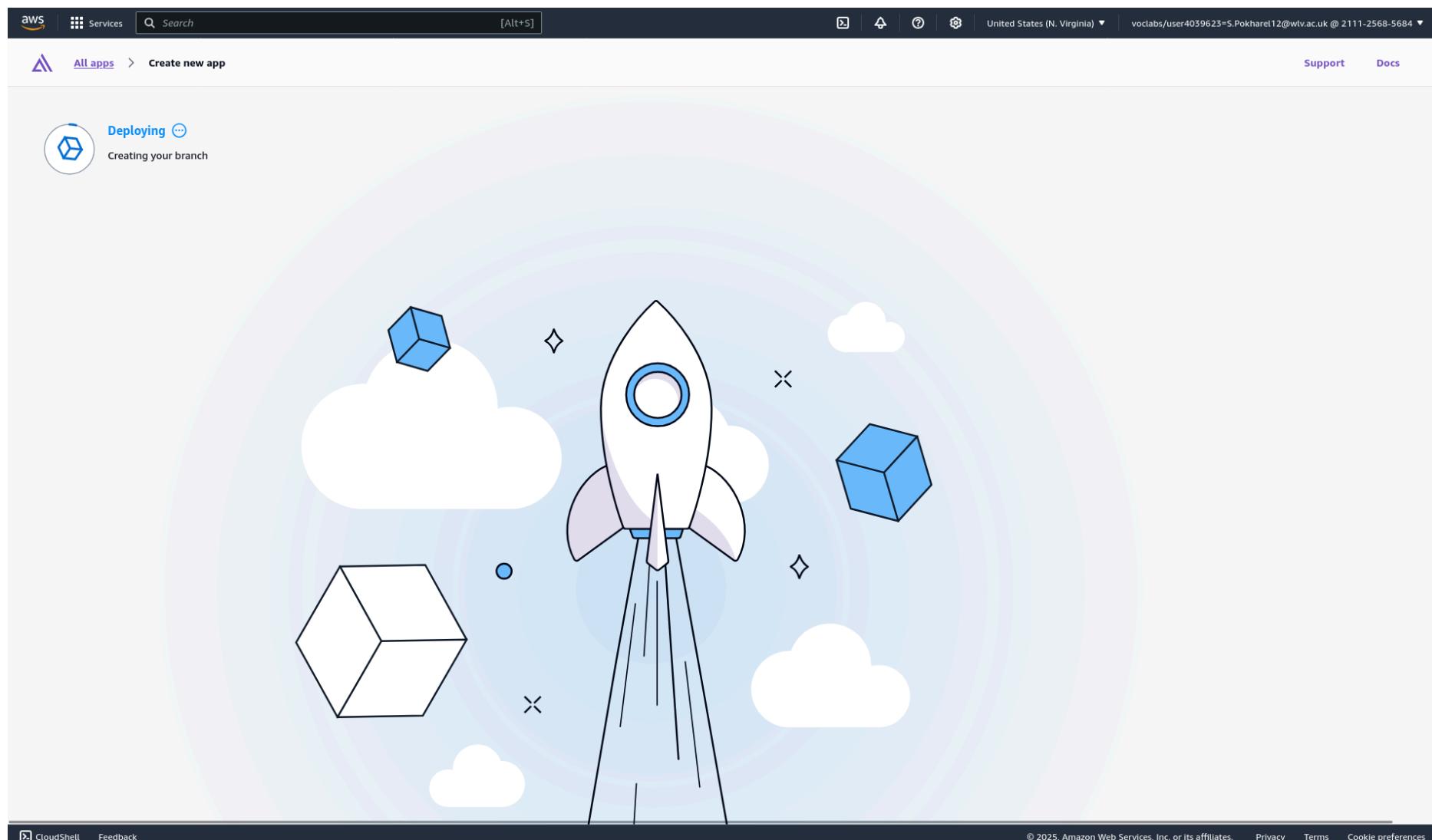
The screenshot shows the 'Create new app' wizard. Step 1: 'Choose create method' is selected. Step 2: 'Start a manual deployment' is selected. The main content area is titled 'Start building with Amplify' and includes a note about full-stack app support. It lists deployment methods: GitHub, BitBucket, CodeCommit, and GitLab. Below this, a purple box highlights 'Deploy without Git'. A note says 'To manually deploy an app from Amazon S3 or a Zip file, select "Deploy without Git"'. A link to 'Create an app with Gen 1' is shown at the bottom.



The screenshot shows the 'Create new app' wizard. Step 1: 'Choose create method' is selected. Step 2: 'Start a manual deployment' is selected. The main content area is titled 'Start a manual deployment' and includes a note about manually uploading objects. It asks for 'App name' (Swoyam_Pokharel_2431342) and 'Branch name' (staging). Under 'Method', 'Amazon S3' is selected. The 'S3 location of objects to host' field contains 's3://swoyam-pokharel-2431342/'. Buttons for 'Cancel', 'Previous', and 'Save and deploy' are visible at the bottom.

Screenshots of the [create new app](#) page

Here, we simply choose to deploy without git and select Amazon S3 as our method to upload the files, and as simple as that, we are now deploying on aws amplify



Screenshot showing deployment in progress

Swoyam_Pokharel_2431342: Overview

Swoyam_Pokharel_2431342

App ID: d1h6cqpdn5nlyv

Get to production

- 1 Add a custom domain
- 2 Enable firewall protections
- 3 Connect new branches

0 of 3 steps complete

Branches 1

staging Deployed

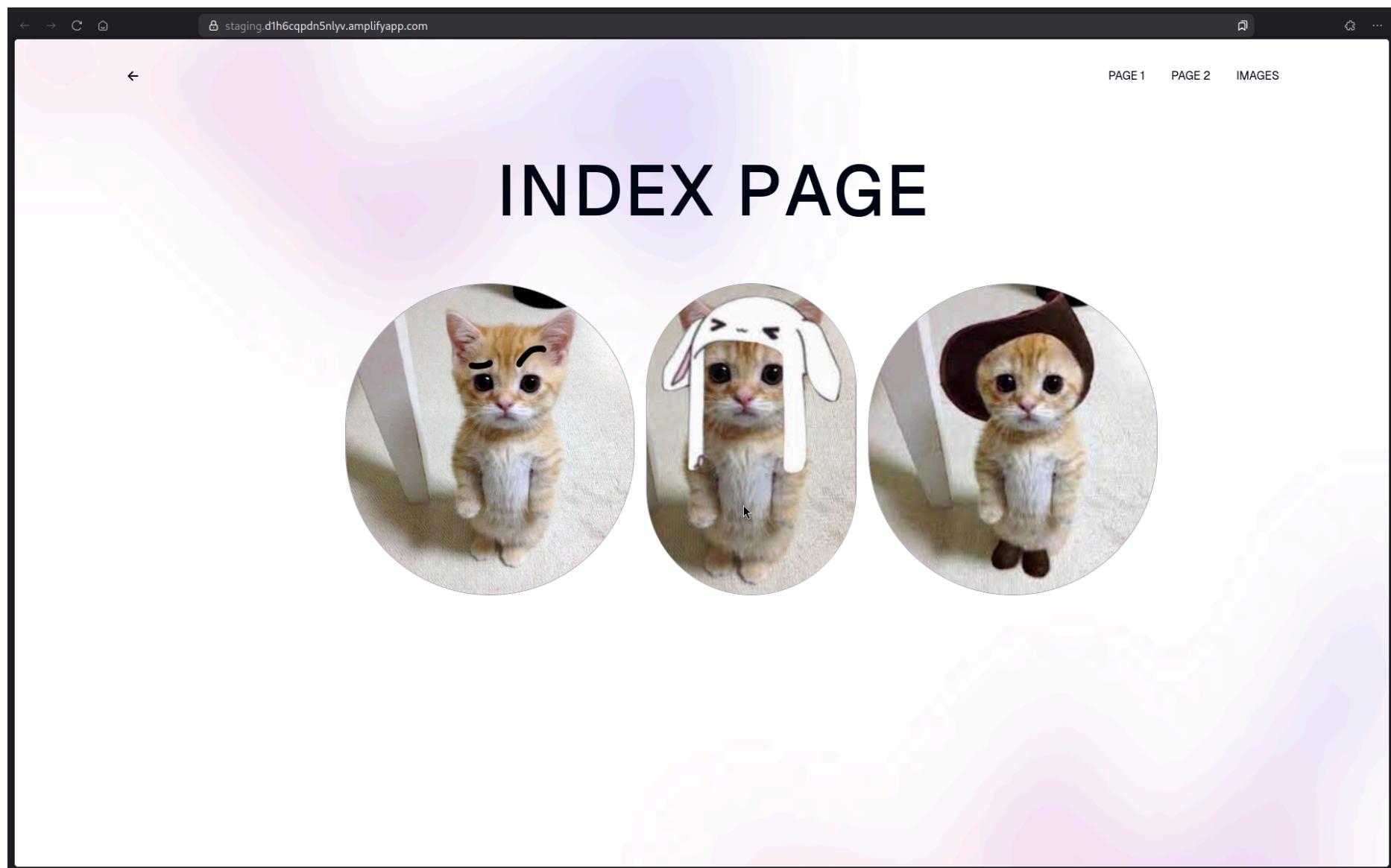
Domain <https://staging.d1h6cqpdn5nlyv.amplifyapp.com>

Last deployment 0 minutes ago

Deploy updates ★ Production branch

Screenshot showing successful deployment

And we are done, we have successfully deployed our application. We can navigate to the [provided amplify link](#) to view our webpage.



Screenshot showing hosted website in amplify

All of the pages work but the only problem we have now is the navigation takes us to page1.html page2.html and the images hosted in amplify.

III: Updating Code & Changing Navigation

The question specifies that `index.html` should be the only non-Amplify link. However, since we're using Amazon S3 as the source for our Amplify deployment, there's no way to exclude specific files from being served via the Amplify domain. Amplify simply maps a domain to the entire contents of the S3 bucket. It doesn't duplicate files; it just exposes them all under its own domain.

To meet the requirement while working within these constraints, I've structured our links like this:

- `index.html` is loaded using the S3 static hosting URL (the only non-Amplify link, as required).
- Inside `index.html`, links to `page1.html` and `page2.html` use the Amplify domain.
- `page1.html` links back to `index.html` using the S3 link, and to `page2.html` using a relative link `/page2.html` so it stays within the Amplify domain.
- `page2.html` also links back to `index.html` via the S3 link, and to `page1.html` using a relative link.
- Images are referenced using relative paths `/static/images/...` so that:
 - If you're on `index.html` (S3 domain), images load from S3.
 - If you're on `page1.html` or `page2.html` (Amplify domain), images load through Amplify.

This setup ensures that:

- Only `index.html` is accessed through a non-Amplify link.
- Page navigation works across both domains.
- Images load correctly respective to the domain you're on.

So, now we can act accordingly:

```
<div class="px-30 navbar sticky top-0 z-50 bg-transparent backdrop-blur-md">
  <div class="flex justify-between items-center w-full">
    <!-- Left -->
    <div class="flex-1">
      <a class="btn btn-ghost text-xl nav-link tooltip tooltip-bottom" data-tip="Navigate Back!" onclick="window.history.back()">&lt;&gt;</a>
    </div>

    <!-- Right -->
    <div class="flex-1 justify-end flex">
      <ul class="menu menu-horizontal px-1 flex gap-3">
        <li>
          |   <a class="text-base font-medium uppercase nav-link tooltip tooltip-bottom" data-tip="Meow Rotate 45" href="https://staging.d1h6cqpdn5nlyv.amplifyapp.com/page1.html">Page 1</a>
        </li>
        <li>
          |   <a class="text-base font-medium uppercase nav-link tooltip tooltip-bottom" data-tip="Meow Rotate 135" href="https://staging.d1h6cqpdn5nlyv.amplifyapp.com/page2.html">Page 2</a>
        </li>
        <li>
          <details>
            <summary class="text-base font-medium uppercase nav-link tooltip tooltip-left" data-tip="Click For Links To Images on S3 Bucket">Images</summary>
            <ul class="bg-base-100 rounded-t-none p-2 uppercase font-bold">
              <li><a class="text-md" href="/static/images/image1.jpg">Image-1</a></li>
              <li><a class="text-md" href="/static/images/image2.jpg">Image-2</a></li>
              <li><a class="text-md" href="/static/images/image3.jpg">Image-3</a></li>
            </ul>
          </details>
        </li>
      </ul>
    </div>
  </div>
</div>
```

Screenshot Of Updated Navigation In Index.html

Here we change the navigation in the `index.html` to point to the amplify links for `page1.html` and `page2.html`

```
<!-- NAVIGATION bar -->
<div class="px-30 navbar sticky top-0 z-50 bg-transparent backdrop-blur-md">
  <div class="flex justify-between items-center w-full">
    <!-- Left -->
    <div class="flex-1">
      <a class="btn btn-ghost text-xl nav-link tooltip tooltip-bottom" data-tip="Navigate Back!" onclick="window.history.back()">&lt;&gt;</a>
    </div>

    <!-- Right -->
    <div class="flex-1 justify-end flex">
      <ul class="menu menu-horizontal px-1 flex gap-3">
        <li>
          |   <a class="text-base font-medium uppercase nav-link tooltip tooltip-bottom" data-tip="Cat No Rotate" href="http://swoyam-pokharel-2431342.s3-website-us-east-1.amazonaws.com/">Index</a>
        </li>
        <li>
          |   <a class="text-base font-medium uppercase nav-link tooltip tooltip-bottom" data-tip="Meow Rotate 135" href="/page2.html">Page 2</a>
        </li>
        <li>
          <details>
            <summary class="text-base font-medium uppercase nav-link tooltip tooltip-left" data-tip="Click For Links To Images on S3 Bucket">Images</summary>
            <ul class="bg-base-100 rounded-t-none p-2 uppercase font-bold">
              <li><a class="text-md" href="/static/images/image1.jpg">Image-1</a></li>
              <li><a class="text-md" href="/static/images/image2.jpg">Image-2</a></li>
              <li><a class="text-md" href="/static/images/image3.jpg">Image-3</a></li>
            </ul>
          </details>
        </li>
      </ul>
    </div>
  </div>
</div>
```

Screenshot Of Updated Navigation In page1.html

Here we change the navigation in the `page1.html` to point to the S3 link for `index.html` and amplify link for `page2.html`

```

<!-- Navigation Bar -->
<div class="px-30 navbar sticky top-0 z-50 bg-transparent backdrop-blur-md">
  <div class="flex justify-between items-center w-full">
    <!-- Left -->
    <div class="flex-1">
      <a class="btn btn-ghost text-xl nav-link tooltip tooltip-bottom" data-tip="Navigate Back!" onclick="window.history.back()">&larr;</a>
    </div>

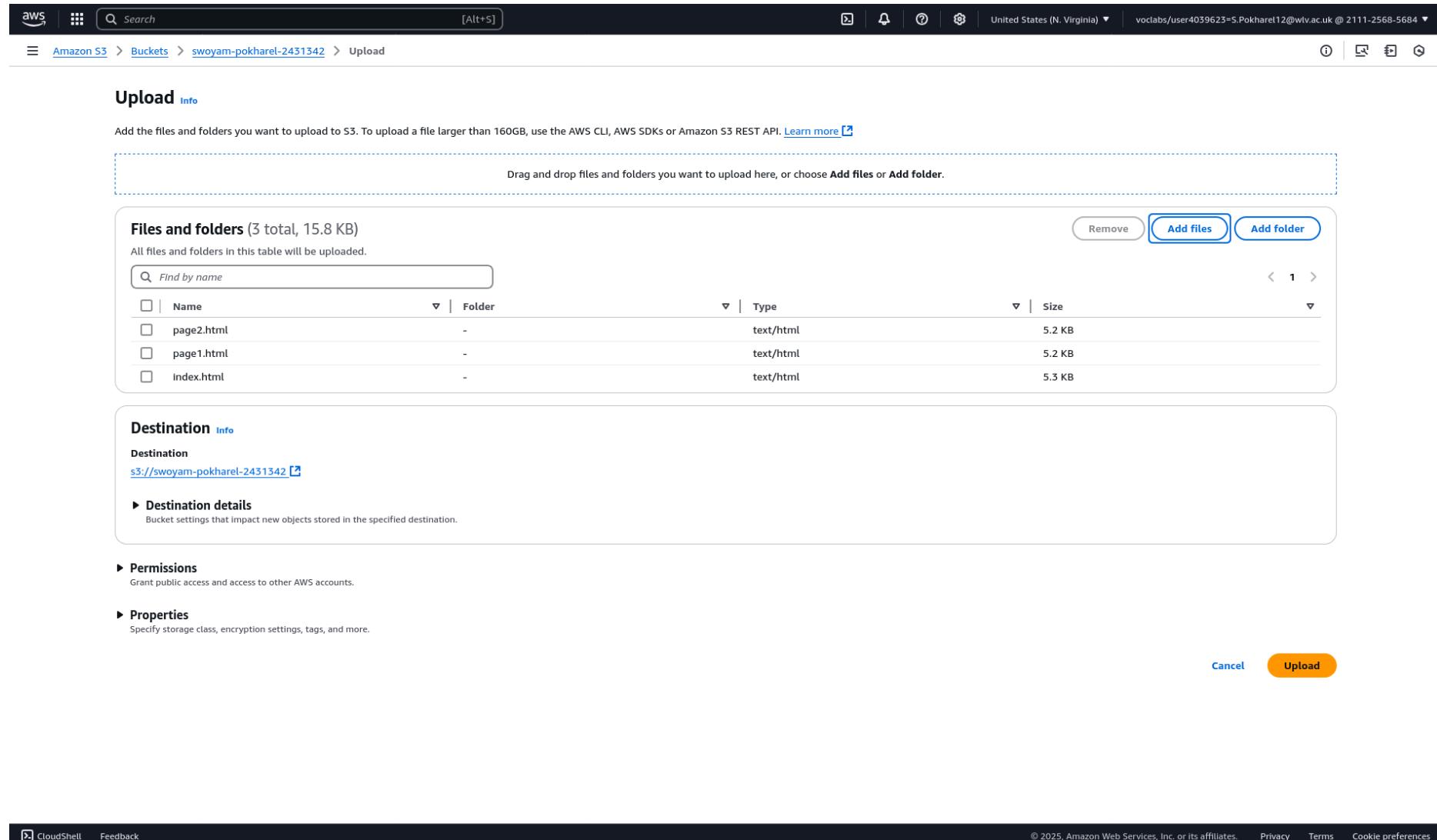
    <!-- Right -->
    <div class="flex-1 justify-end flex">
      <ul class="menu menu-horizontal px-1 flex gap-3">
        <li>
          |   <a class="text-base font-medium uppercase nav-link tooltip tooltip-bottom" data-tip="Cat No Rotate" href="http://swoyam-pokharel-2431342.s3-website-us-
          |   east-1.amazonaws.com/">Index</a>
        </li>
        <li>
          <a class="text-base font-medium uppercase nav-link tooltip tooltip-bottom" data-tip="Meow Rotate 135" href="/page2.html">Page 2</a>
        </li>
        <li>
          <details>
            <summary class="text-base font-medium uppercase nav-link tooltip tooltip-left" data-tip="Click For Links To Images on S3 Bucket">Images</summary>
            <ul class="bg-base-100 rounded-t-none p-2 uppercase font-bold">
              <li><a class="text-md" href="/static/images/image1.jpg">Image-1</a></li>
              <li><a class="text-md" href="/static/images/image2.jpg">Image-2</a></li>
              <li><a class="text-md" href="/static/images/image3.jpg">Image-3</a></li>
            </ul>
          </details>
        </li>
      </ul>
    </div>
  </div>
</div>

```

Screenshot Of Updated Navigation In page2.html

Here we change the navigation in the `page2.html` to point to the S3 link for `index.html` and amplify link for `page1.html`. Now, we upload these changes that they take effect.

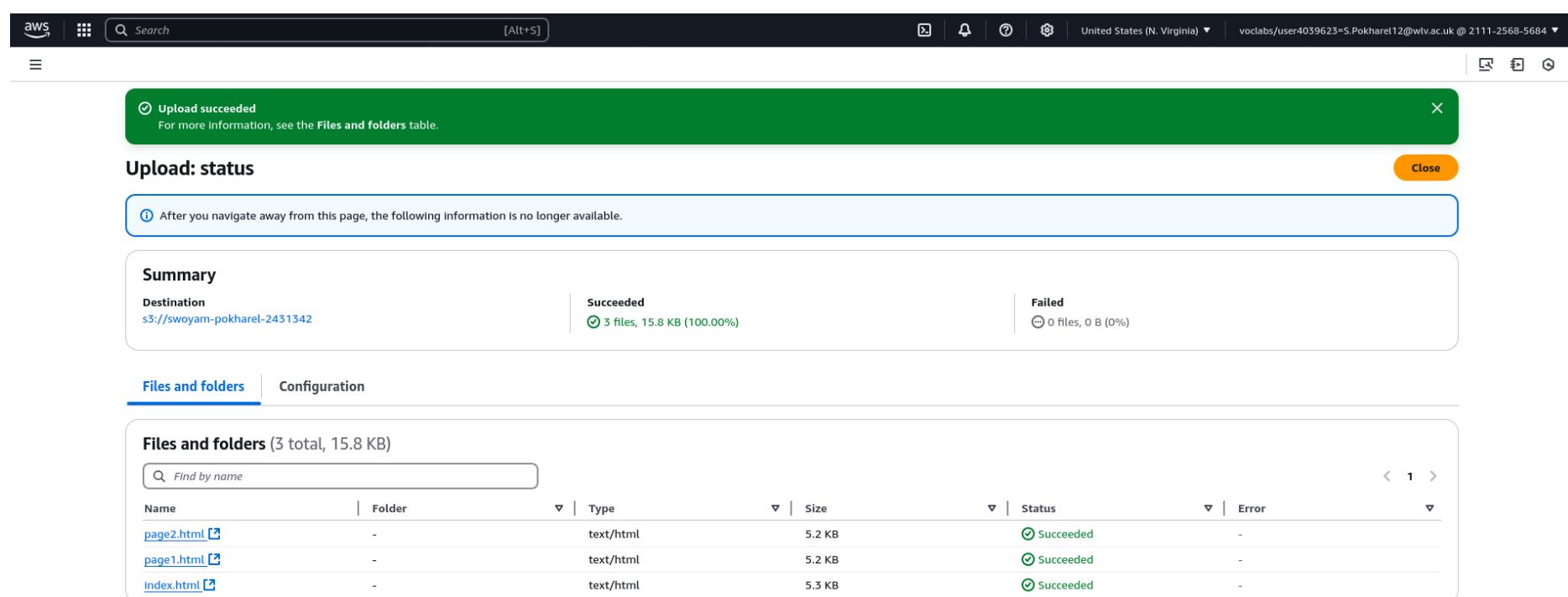
IV: Uploading Changes



The screenshot shows the AWS S3 'Upload' interface. At the top, there's a search bar and navigation links for 'Amazon S3 > Buckets > swoyam-pokharel-2431342 > Upload'. The main area is titled 'Upload Info' with a note about uploading files larger than 160GB. A large dashed box allows dragging and dropping files. Below it, a table lists 'Files and folders (3 total, 15.8 KB)'. The table has columns for Name, Folder, Type, and Size. The files listed are page2.html, page1.html, and index.html, all of which are text/html type and around 5.2-5.3 KB. There are 'Remove', 'Add files', and 'Add folder' buttons. On the left, sections for 'Destination' (set to s3://swoyam-pokharel-2431342), 'Destination details', 'Permissions', and 'Properties' are visible. At the bottom right are 'Cancel' and 'Upload' buttons.

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Screenshot Of Uploading Changes



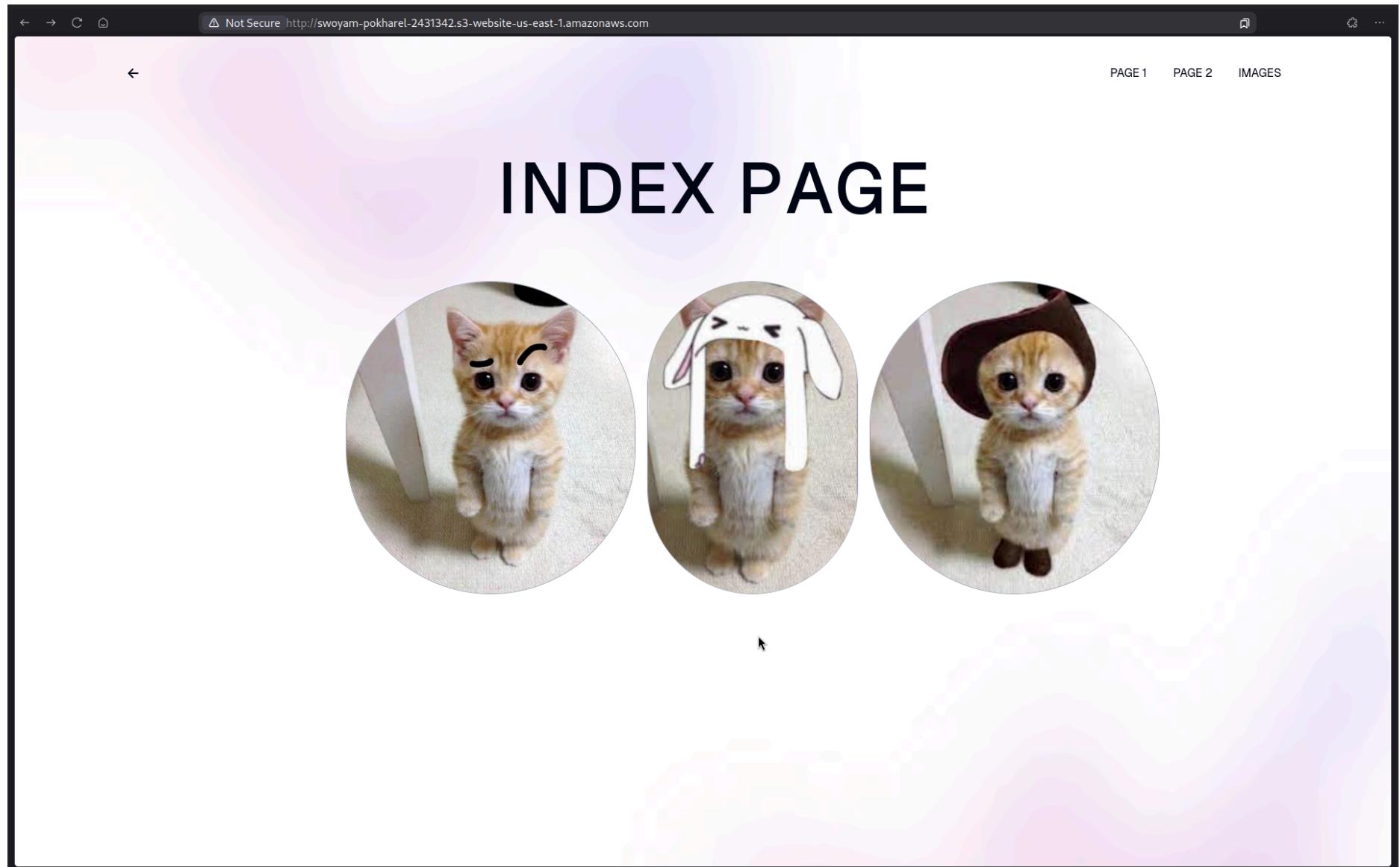
The screenshot shows the AWS S3 'Upload: status' summary. It displays a green success message: 'Upload succeeded. For more information, see the Files and folders table.' Below this, a summary table shows 'Succeeded' (3 files, 15.8 KB (100.00%)) and 'Failed' (0 files, 0 B (0%)). The 'Files and folders' tab is selected, showing a table of the same three files: page2.html, page1.html, and index.html, all marked as 'Succeeded'. The 'Configuration' tab is also present.

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Screenshot Confirming Upload

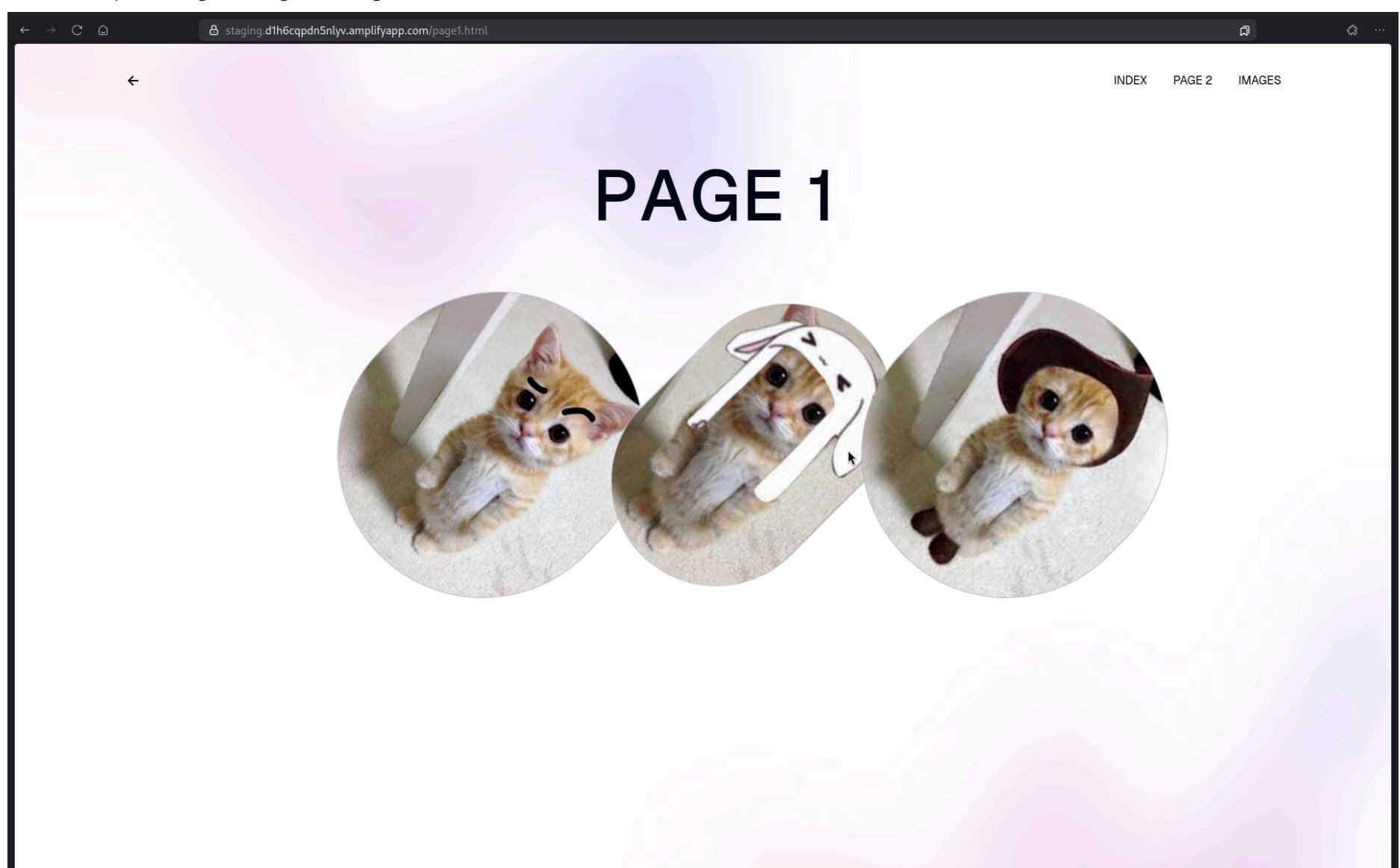
Here, we simply upload the changes we've made and now the navigation happens according to the question.

V: Screenshots



Screenshot Of Static Page Hosted On S3

From here, clicking on Page 1 navigates us to:



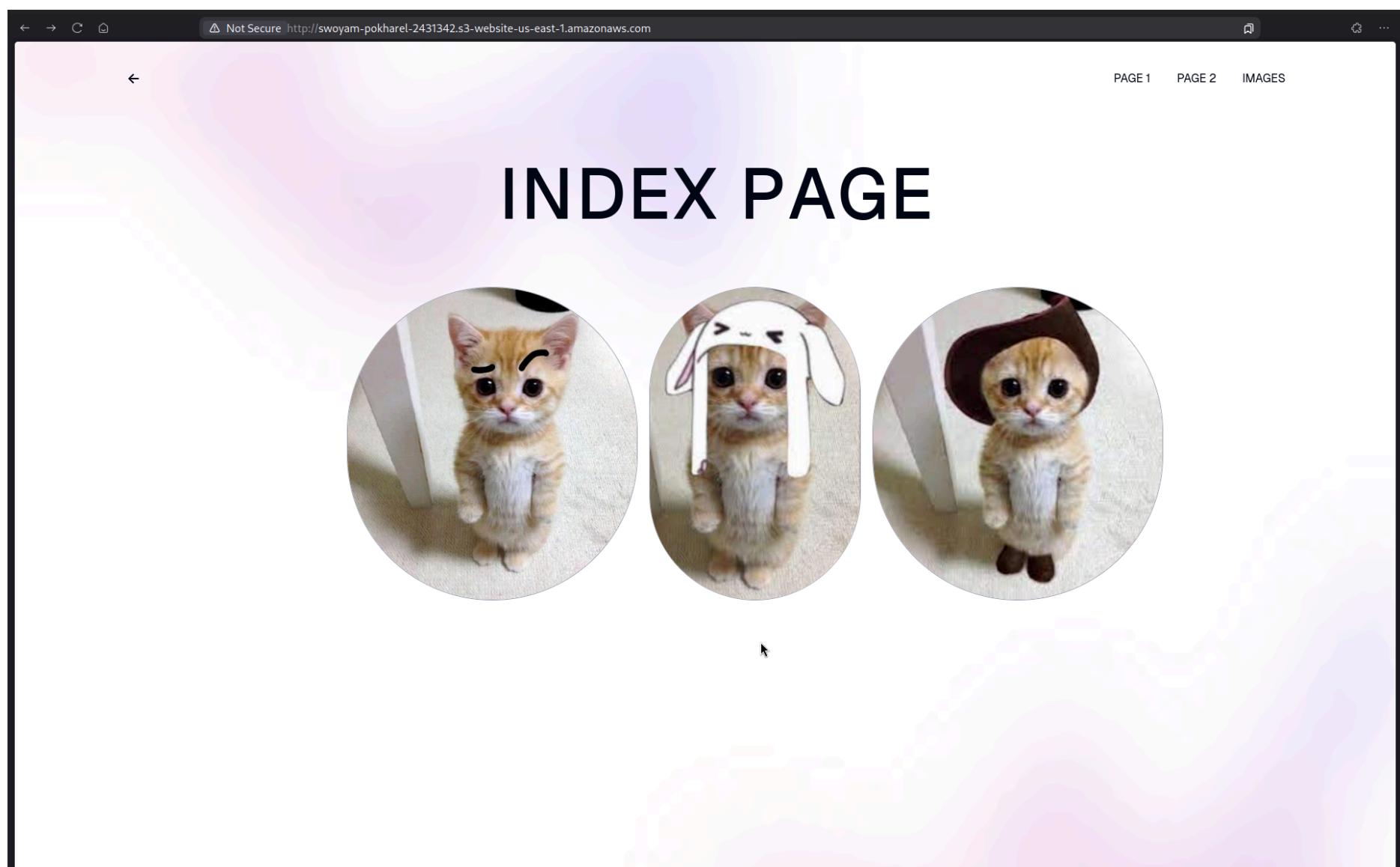
Screenshot Of Amplify Hosted Page 1

From here, clicking on page 2 navigates us to:



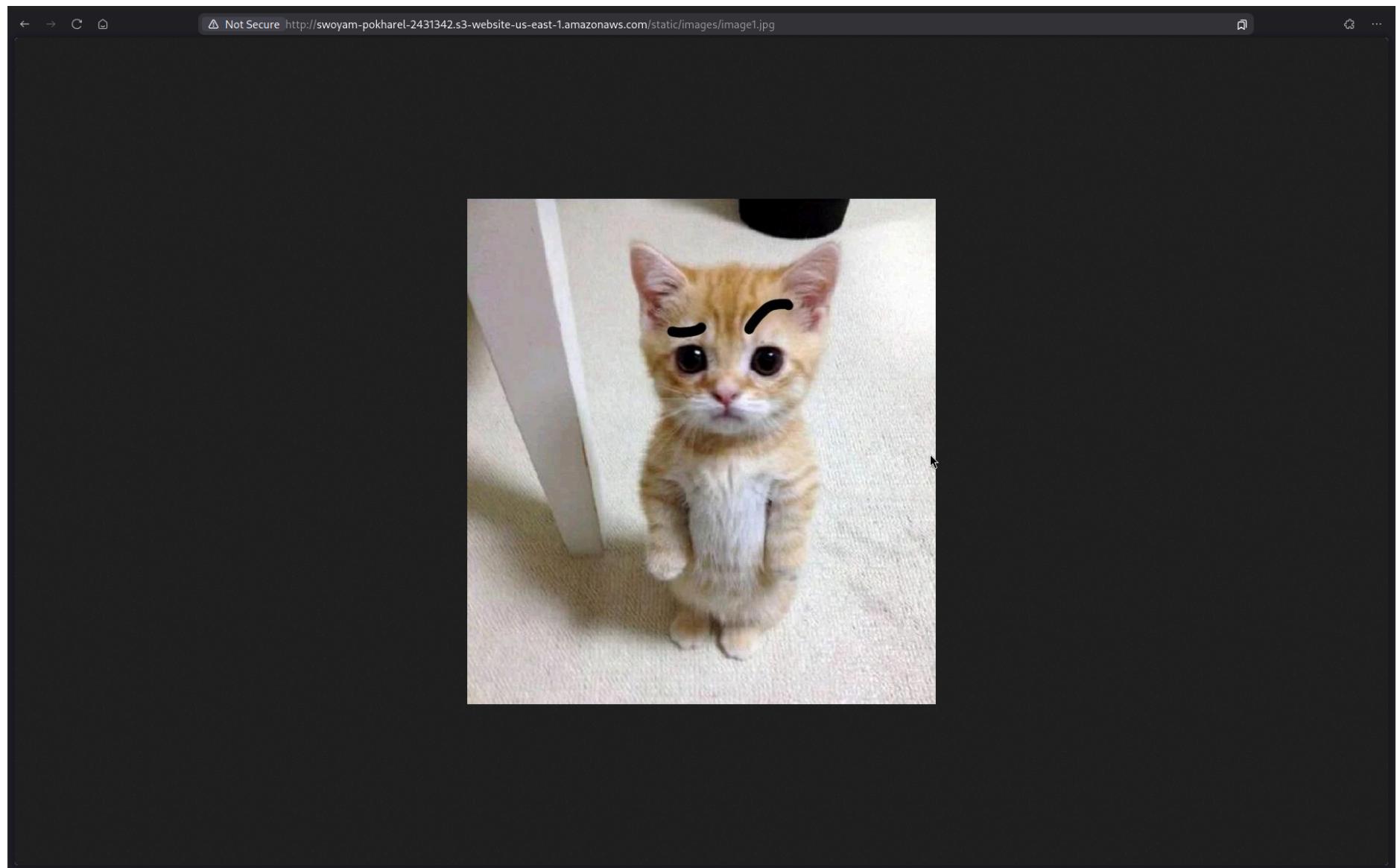
Screenshot Of Amplify Hosted Page 2

If we navigate back to index from here, we get redirected back to:



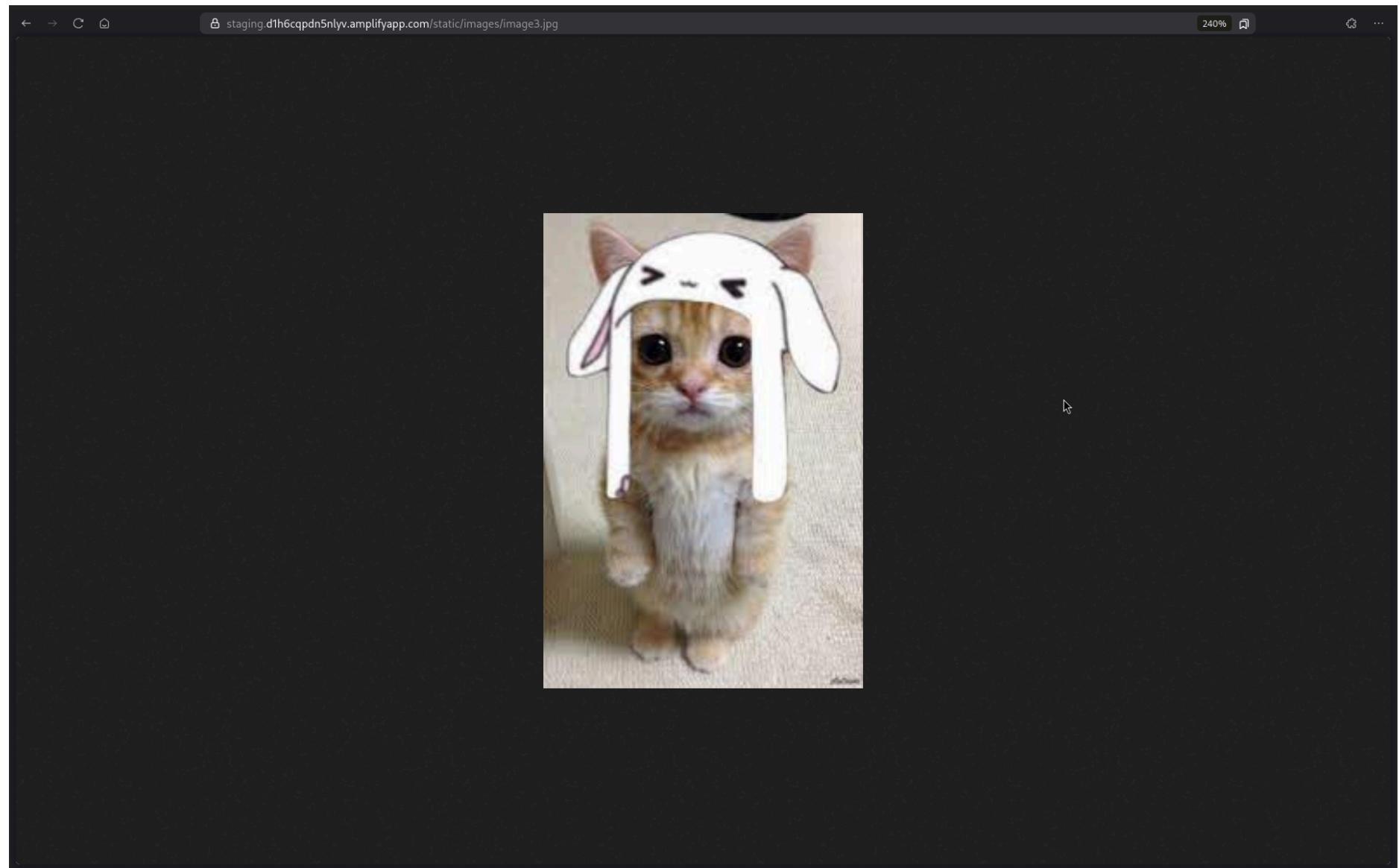
Screenshot Of Redirection Back to Index Hosted On S3

Navigating to any image through the navigation on pages under the S3 domain results in the image being loaded from an S3 URL.



Example Screenshot of the image when accessed via index.html (loaded from the S3 domain).

Navigating to any image through the navigation on pages under the Amplify domain results in the image being loaded from an S3 URL:



Example Screenshot of the image when accessed via page1.html (loaded from the Amplify domain).