**Getting started with AWS Projects**

**What is AWS?**  
Cloud computing with **Amazon Web Services (AWS)** is like renting a powerful computer via the internet. Businesses (and engineers like you!) use AWS services to store files, run applications, analyse data and do much more.

**Why does AWS ask for this information?**

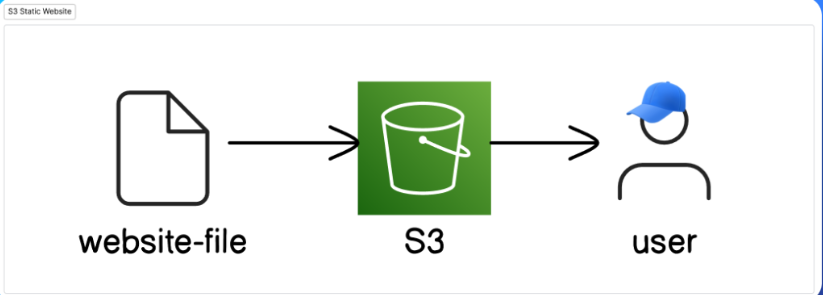
* AWS uses your **phone number** as a way to secure your account with two-factor authentication. This step adds an extra layer of protection for your account.
* Different parts of the world have different laws around tech and data, so AWS uses your **physical address** to make sure they don’t give you access to anything that’s not allowed where you live.

**why do I need to enter card information? I thought this was free!**  
Great question! AWS works on a pay-as-you-go system, so they'll ask for your credit card to get billing set up. But don’t worry, this does not actually mean you'll get charged anytime soon. AWS has 200+ free offerings, you only pay for what you actually use, and it's free to set up an AWS account.

**What is AWS Support?**  
AWS Support is AWS's tech support team that helps us with account and technical issues. The basic plan we're using is free with your AWS account and means you can reach out for help in their Support portal.  
  
The paid support plans give developers or businesses 24/7 direct access to Cloud Support Engineers, expert advice on AWS best practices, and faster troubleshooting help.

**1.Host a Website on Amazon S3**

You can use Amazon S3 to store and retrieve any amount of data at any time from anywhere on the web.



In this project, you'll use Amazon S3 (which stands for Amazon Simple Storage Service) to host a website.

**Websites** are run by pictures and files. You'll be hosting (which basically means make public on the Internet) a website on Amazon S3 by uploading website content into S3 and then making it publicly accessible.

**In this step, get ready to:**

* Open Amazon S3.
* Create a storage space for your website files.
* [**Log in to AWS.**](https://console.aws.amazon.com/console/home?nc2=h_ct&src=header-signin)
* In the AWS Management Console, search for **S3**.

1. Create a bucket in amazon s3
   1. In the AWS Management Console, search for **S3**.
   2. Choose create bucket
   3. For **Bucket name**, enter nextwork-website-project-name   
      “ Bucket name must not contain uppercase characters”
   4. Make sure to replace name with your name.

A screenshot of a computer

Description automatically generated

**Why can't I just keep 'name' in the bucket name?**  
An S3 bucket name is globally unique. After you create a bucket, **no other AWS account in the entire world** can use your bucket's name (unless you delete the bucket).

This also means that when you create your bucket, you need to make sure the bucket's name is unique too.

* Bucket names must be unique across all AWS accounts in all the AWS Regions within a partition. A partition is a grouping of Regions. AWS currently has three partitions: aws (Standard Regions), aws-cn (China Regions), and aws-us-gov (AWS GovCloud (US)).
* A bucket name cannot be used by another AWS account in the same partition until the bucket is deleted.**After you delete a bucket, be aware that another AWS account in the same partition can use the same bucket name.**

For **Object Ownership**, choose **ACLs enabled**.

A screenshot of a computer

Description automatically generated

It says disabled recommended, but we enable it because , making it enabled will eventually help us create public access for our bucket and its contents

**what are ACLs (Access Control Lists)?**  
An **ACL** = a set of rules that decides who can get access to a resource.

Enabling ACLs in this S3 setup lets you control who can access and do things with the objects (i.e. website files) you upload into your bucket.

With ACLs, different AWS accounts can own and control different files in your bucket.

**What is the yellow pop up saying?**  
A yellow warning banner will pop up when you enable ACLs. This banner tells you that it's **simpler** to use another tool called **bucket policies.**

It's true that bucket policies make it really easy to control access for an **entire** bucket (e.g. making the entire bucket and everything inside public), but ACLs are the way to go if you want to manage access for each object in your bucket individually.

In this project, we're using ACLs to show you how they work.

* Choose **Bucket owner preferred**.
* For **Block Public Access settings for this bucket**, clear the check box for **Block all public access**.
* **A yellow banner has popped up!**  
  This banner is telling us that the bucket and its objects might become public if we untick the checkbox. This is what we want to host a public website!
* Check the box that says **“I acknowledge that the current settings might result in this bucket and the objects within becoming public.”**
* For **Bucket Versioning**, choose **Enable**.
* Choose **Create bucket**.

A screenshot of a computer

Description automatically generated

**2.Upload website content to your bucket**

**What is HTML?**  
**HTML (HyperText Markup Language)** is used to create and design web pages. It's your way of telling the website where you want to display your text, images, videos and more. Think of HTML as the blueprint that shapes what you see when you visit a website.

* Return to the Amazon S3 console with your bucket page open. Choose the **Objects** tab.
* Choose **Upload**.
* Choose **Add files**.
* Choose **index.html**.
* Choose **Add folder**.
* Choose the unzipped folder - NOT the zip file itself!
* You might get a popup that tells you that all files in that folder will be uploaded.
* Choose **Upload**.
* S3 will get to work right away!

**A screenshot of a computer

Description automatically generated**

**3. Configure a static website on Amazon S3.**

S3 bucket? Created.

Website files? Uploaded.

Next up, let's make your website available on the internet by setting up **static website hosting!**

**In this step, get ready to:**

* Configure your S3 bucket for static website hosting (makes your website to go public on the internet)
* Visit your public website link.

**What does website hosting mean?**  
Website hosting is what makes your website public on the internet.

Even if you perfect an HTML file, no one else can see it when it's stored as a local file on your computer! Website hosting = storing your HTML file (and the other files for your website) on a web server, so it's accessible online.

By configuring your S3 bucket for hosting, we're telling this bucket: "please create a URL that will take anyone to a page that displays the HTML file I just uploaded."

* Choose the **Properties** tab.
* Scroll allllllllll the way down to the **Static website hosting** panel.
* Choose **Edit**.
* Configure the following settings:
  + **Static web hosting:** Choose **Enable**.
  + **Hosting type:** Choose **Host a static website**.
  + **Index document:** Enter index.html

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**What's a bucket website endpoint?**  
A bucket website endpoint is just like a regular website URL. It lets people visit your S3 bucket as a website.

A close-up of a white background

Description automatically generated

But Why?

**Why did I get this error?**  
Objects (in this case, the HTML and images files you uploaded) are private by default. This default setting helps keep your account's data secure.

The error message you're seeing is telling you that your static website is being hosted by S3, but the actual HTML/image files you've uploaded are still private. It's kind of like having a bucket on display, so everyone can see the bucket - but the contents are covered up, preventing anyone from seeing what's inside.

To solve this error, we need to set the permission of the objects to public - this is why we enabled ACLs in Task 1!

The only missing ingredient is to make your website files **publicly accessible,** so everyone has permission to view your website.

**In this step, get ready to:**

* Make your website files in S3 publicly accessible.
* See your website live on the internet!
* Keep the error message tab open and switch back to the Amazon S3 console tab.
* Would you still remember how to view your S3 bucket's objects? Try finding your bucket's **Objects** page and making your objects public using ACLs.

.... feeling a little stuck?

* No worries! If you're stuck, head to the **Objects** tab.
* Select the checkboxes next to your **index.html** file and the folder of website assets.
* In the **Actions** dropdown, choose **Make public using ACL**.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Now, we can see the website

A screenshot of a cell phone

Description automatically generated

Every time you upload a new file, you make ACL public using in Actions.

Pre-Signed URL:

Now, we can do a secret mission called securely sharing an object using pre-signed URL.

A pre-signed URL is basically a secret link that you will share directly to someone and tell them “Hey, I am gonna give just you the access to this object”. So, you can get access to what I am giving you but not everyone get access to it.

The cool thing about pre-signed AWS URL is that you can assign for how long this link can be active for.

A screenshot of a computer

Description automatically generated

A screenshot of a computer screen

Description automatically generated

A black text on a white background

Description automatically generated

Now, after 2 min:

A white rectangular object with black text

Description automatically generated

That is the pre-signed URL doing its job to make it secure. It cut off access to anyone you don’t want to give access to.

Now you can you bucket policy, to secure your entire bucket:

network-website-project-vikranth – bucket name , index.html – object name

Go to bucket policy in permission and type this:

{

"Version": "2012-10-17",

"Id": "MyBucketPolicy",

"Statement":[{

"Sid": "BucketPutDelete", //if someone tries to delete

"Effect": "Deny", // to not let them do it

"Principal": "\*",

"Action": "s3:DeleteObject", // an object

"Resource": "arn:aws:s3:::network-website-project-vikranth/index.html"}] //specify object which object they are going to delete

}

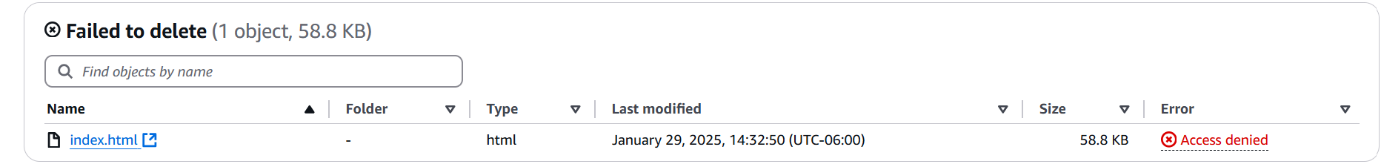
This is like a policy, my policy, please add this policy to my law book. If anyone tries to delete, I object and I am gonna specify exactly which object they are going to delete.

DENY IT !! Do not let them do that.

Now, lets try to delete that and see what happens:

A screenshot of a computer

Description automatically generated

It fails to delete objects. So, this is your policy doing its job. It is denying access to stoping anyone from trying to delete your index.html file. I really like that you can do this in S3. Its perfectly telling access is denied.

So, this is the policy proof that you made is working.

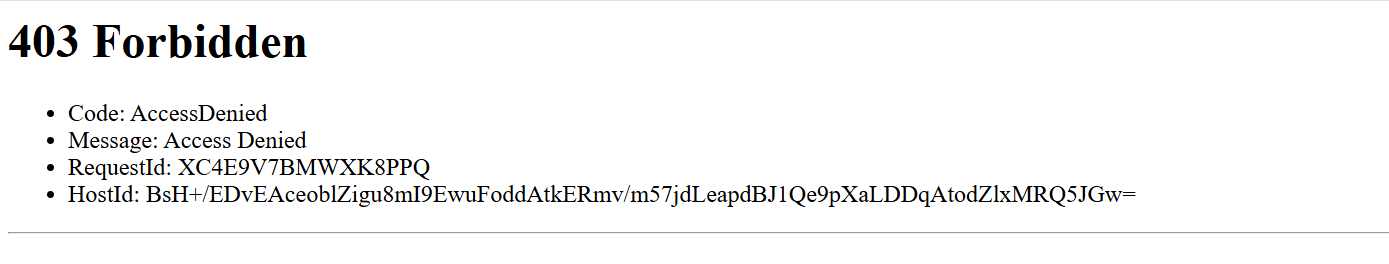
Now, we are into another secret mission, updating the website itself.

If you want to make few changes to the website, we are gonna change html file and we are gonna try to update it, so that the changes are actually gonna show on your official website.

Change anything in your index file in notepad++ , but how are we going to update it in S3.

We don’t have to delete the uploaded file, all we have to do is upload the updated file into the bucket. And just like that it replaces the old one for you.

How it recognizes is that it has the same name, but when you check the updated URL in the bucket. Its forbidden.



Its still forbidden because we still need to enable public access using ACL.

See, I just changed my name in the index file and updated. Now you can see the updated one in the S3 bucket.

A screenshot of a video game

Description automatically generated

Now, Bucket Versioning.

What’s the bucket versioning? (its pretty much removing undo button)

When bucket version is turned off, changes to the object can’t be undone.

For example, the old file is replaced with the new file. The original file is lost. If you delete a file, it is permanently deleted, and you can’t get it back.

Bucket version is turned off by default. However, when versioning is turned on, changed and deleted versions of files are saved. It’s also important to remember that once you turn on bucket versioning, you cannot turn it off.

By enabling a bucket versioning, you are making yourself by saying, hey lets have a backup copy for every single version of every file that I upload.

Just click on show versions in the bucket on top of the files you uploaded.

A screenshot of a computer

Description automatically generated

Now, we will see hosting your website on your special domain.

This is a new service which are in advance. When you enabled website hosting on S3, notice how URL for your website has already determined for you.

In this final mission you will use route 53 to purchase your own domain and host your website there, so you have a clean and customized URL.

* In your AWS management console search for Route 53
* Choose domain registration
* Choose register domain
* Now, you look for a name for your registered domains you want

(domain name is basically kind of a name that takes you to a specific website) you gonna buy a domain name, so that you could own one. So, its gonna take the website which you are hosting on S3. Domain name is almost like renting it, you don’t own it forever and ever. Usually by default if you are busying it for one year, and then after one year if you don’t own it, it goes back up in the market, someone else can buy it.

A screenshot of a computer

Description automatically generated

Now we will connect your bucket to your domain.

Inorder to connect your S3 bucket to your domain, once your domain is registered, head back to S3 and create a new bucket. Name of the S3 bucket needs to be the exact same name as your custom domain, and set it up as you have as the previous bucket(ACLs enabled, Disabled block access and enable static website hosting) Upload your HTML file and folder of image access.

Now, in Route 53 console, choose the hosted zones and select the hosted zone name that’s already been created for you.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

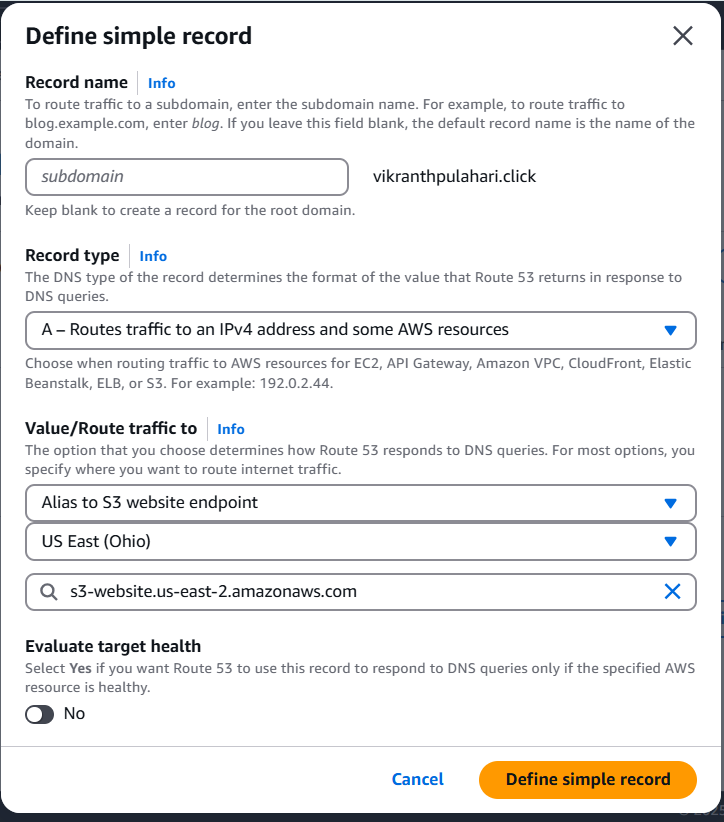
Description automatically generated

Now we need to create a new record to make sure your bucket is connected to your bucket.

* Record type: A – Routes traffic to an IPv4 address and some AWS resources.

(can you please setup this rule to my domain that if any one clicks on my domain which is vikranthpulahari.click, you are routing that, you are goanna direct them to an address that I know on AWS resource that I have, its your S3 bucket)

* Value/Route traffic to:
  + Choose endpoint: Alias to S3 website endpoint
  + Region: the region you hosted your S3 bucket
  + Your s3 bucket: make sure you are picking the new bucket you have just setup, and that is was the exact same name as your domain.
* Turn off evaluate target health.



A screenshot of a computer

Description automatically generated

So, this is how we direct the domain to a resource that you have setup in S3.

A screenshot of a computer

Description automatically generated

**HOSTING A WEBSITE ON S3**