// Chapter 3

Learn S3 While Cutting Your Hosting Cost To A Dollar

The AWS Service You Hear About Most

In Spring 2006, Amazon introduced Simple Storage Service (S3). S3 would disrupt the status quo of on-premise storage at the time and go on to be the foundational service of AWS. Buckets are high availability and high durability flat object stores.

Since 2006, the service has evolved to be much more. Competitors to AWS like Azure, Google Cloud, and IBM all have alternative offerings to S3. Yet it is clear that S3 is much more than a place to store objects.

S3 is the most popular AWS service. It comes up in almost any architecture discussion. There are countless conference talks and blog posts revolving around leveraging it. Marketed as a flat object store that can hold billions of objects at any given time at 11 nines of durability. Chew on that for a second, S3 has 99.999999999% durability. You have a better chance of getting struck by lightning than S3 losing one of your files.

It meets a large number of use cases. The website hosting option we are going to cover in this chapter is only one. Here are a few other use cases you can leverage S3 for:

- Multi petabyte storage for data lakes and big data analytics.
- Media streams, downloads, and uploads.
- Querying petabytes of information with Athena, Hadoop, or other MapReduce tools.

For all the great use cases there are for S3, it is important to remember a few key constraints it has. Knowing these ahead of time allows you to account for them before you get hit by them:

- Eventual consistency. A PUT or DELETE operation that overrides an existing key has eventual consistency. If your application is frequently updating the same file `foo.txt` in a bucket while another operation is doing a GET for that file at the same time, it is possible for the GET operation to grab an outdated copy.

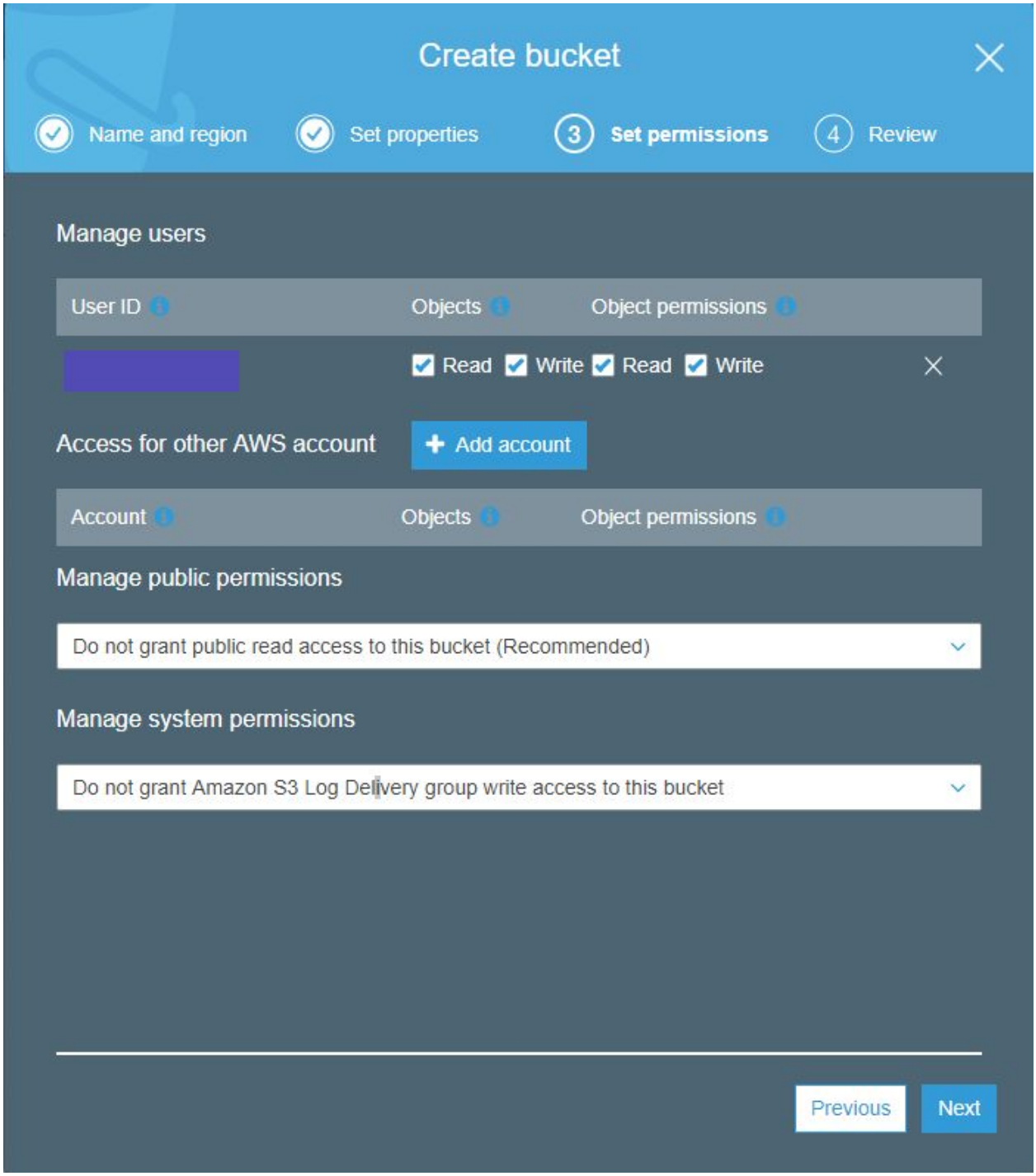
- It goes to 11 for durability, but not availability. Many a developer have fallen victim to thinking the 11 9's of durability applies to availability as well. This is wrong. In fact S3 only has two 9's of availability, still good, but it does not go to 11 (🎸). An example occurred recently when S3 experienced an outage in us-east-1 (N. Virginia). It took down half the internet for many hours. AWS actually had to credit some customers as they violated their availability SLA.
- Global names. Bucket names in S3 are globally namespaced. There can be no bucket across the AWS ecosystem with the same name. We will dive a bit deeper into why this is the case in this section. *Hint: Can you and I own the same domain name?*

Security Notice

Simple Storage Service can provide developers many great things. Unfortunately it has made the news for other reasons as well. Improperly configured bucket permissions and access control lists have led to the leaking of billions of files. Several major leaks have happened from the likes of Verizon, Groupize, and even the Dow Jones.

So I am going to state this once for everyone to refer to.

S3 buckets by default are not publicly readable. You must change the access control list (ACL) to provide public read access to the world. By default when you create a new S3 bucket you administer the public permissions for that bucket.



By default the public permissions deny public read access. Making a bucket publicly readable makes it accessible from anywhere. **Do not store information in public buckets that you do not want to share with the world.** There are plenty of tools out there that scan random namespaces to find open S3 buckets and siphon data out.

To summarize, making a bucket publicly readable means anyone can read data from that bucket. If you don't want to share your data, don't do it.

Ok that is out of the way. We are creating a website bucket. The bucket needs to be public so browsers are able to load the resources. This is a public bucket. If you have information that is private, do not put it in this bucket.

Under A Dollar For Website Hosting

Back in the day you would have to buy website hosting through a hosting provider. You would often buy your domain name and then FTP your website files to a server given to you by the provider. Configuration for the server happened through an admin panel you logged into.

These servers were often limited on space. They had vulnerabilities and questionable configurations that even the NSA questioned your decision for using them. All for about **$10/month**.

In this chapter we say au revoir to the old days. You can buy your domain name via Amazon DNS provider, Route53. Create an S3 bucket, and with a few clicks turn it into a web server. You don't have to maintain or configure anything and space is never limited. There is no backdoor admin panel to administer your site and you can upload content to your site with one CLI call.

All for less than **$1/month**.

Every Bucket Must Have A Name

Before you begin hosting your awesome static website out of S3, you need a bucket first. It is critical that your bucket has the same name as your domain name.

If your website domain is `www.my-awesome-site.com`, then your bucket name must be `www.my-awesome-site.com`.

This has to do with how S3 routes requests for static website hosting buckets. A request comes into the bucket, and then S3 uses the Host header in the request to route to the appropriate bucket.

```
Host: www.my-awesome-site.com
```

This is why S3 buckets are special. They must have names that are DNS-compliant in order for static website hosting to work. Static website hosting enabled buckets must map from a DNS name to an S3 bucket with the same name. This is the reason for the documented constraints of S3 bucket names:

- Names must be unique across all bucket names in S3.
- Names cannot be changed after initial creation.

When you know how S3 website hosting is going to route traffic these constraints make a lot more sense. Imagine if these constraints were not in place. How would S3 know which bucket to send requests for your static website to? It wouldn’t be able to.

Let's go ahead and create the S3 bucket for our static website.

1. Navigate to S3 in the AWS Console.
2. Click Create Bucket.
3. Enter the url of your static site for the name.
4. Click Create in the bottom left.

Or if you are a command line fan, you can create your bucket with the AWS CLI.

```
C:\Users\Kyle\>aws s3api create-bucket --bucket 'www.my-awesome-site.com'
```

Configuring Your S3 Bucket for Static Website Hosting

Alright, you have your bucket. It has the same name as your domain name, yes? Time to configure the bucket for static website hosting.

Turning on static website hosting for your bucket is as simple as a few clicks in the AWS Console.

1. Navigate to S3 in the AWS Console.
2. Click into your bucket.
3. Click the "Properties" section.
4. Click the "Static website hosting" option.
5. Select "Use this bucket to host a website".
6. Enter "index.html" as the Index document.

Or if you are all about command lines and would rather not have a graphical user interface (GUI) in your way, this AWS CLI command turns website hosting on for your bucket.

```
C:\Users\Kyle>aws s3 website s3://www.my-awesome-site.com/ --index-document index.html --error-document
error.html
```

Your bucket is configured for static website hosting, and you now have an S3 website url: `www.my-awesome-site.com.s3-website-us-east-1.amazonaws.com`

Your bucket serves your static website, so it must be accessible to anyone in the world. This is referred to as *anonymous access* to the bucket.

To do this, you must update the Bucket Policy of your bucket to have public read access to anyone in the world. The steps to update the policy of your bucket in the AWS Console are as follows:

1. Navigate to S3 in the AWS Console.
2. Click into your bucket.
3. Click the “Permissions” section.
4. Select “Bucket Policy”.
5. Add the following Bucket Policy and then Save

```
{
    "Version": "2008-10-17",
    "Statement": [
        {
            "Sid": "PublicReadGetObject",
            "Effect": "Allow",
            "Principal": {
                "AWS": "*"
            },
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::www.my-awesome-site.com/*"
        }
    ]
}
```

Or for the command line fans out there, if policy.json is the above bucket policy, then use:

```
C:\Users\Kyle>aws s3api put-bucket-policy --bucket www.my-awesome-site.com --policy file://policy.json
```

It is important to note the principal section of the bucket policy. This part of the policy opens up your bucket to anyone in the world. Any object in this bucket is available to the public via the S3 website url.

Don't put anything in this bucket that you're not willing to share with the world.

Configuring Your Naked Domain Redirect

You configured your `www.my-awesome-site.com` bucket to have static website hosting turned on. You updated your bucket policy to allow access from anywhere. Now is a good time to talk about naked domains.

A naked domain is the term used to talk about domain names without the www. So in the running example our naked domain is `my-awesome-site.com`.

Remember that S3 is going to use a `Host` header on requests to route traffic to the appropriate bucket. Currently, requests to `my-awesome-site.com` will not route to your S3 website bucket. You need to create a bucket for the naked domain route and configure it to redirect requests to the `www.my-awesome-site.com` bucket.

1. Navigate to S3 in the AWS Console.
2. Click "Create Bucket".

3. Enter “my-awesome-site.com” as the name.
4. Click through to “Create Bucket”.
5. Click into your new bucket.
6. Click the “Properties” section.
7. Click the “Redirect requests” option.
8. In “Target bucket or domain” enter “www.my-awesome-site.com”.
9. Click “Save”.

You now have two S3 buckets one for your www domain and one for your naked domain. The second bucket will receive requests for `my-awesome-site.com`. Those requests get redirected to your www bucket `www.my-awesome-site.com`. Before that is possible we now need to configure a few DNS records.

DNS Records For Your Bucket Urls

You have a bucket that is configured for static website hosting and it has a S3 website url. There is a bucket to redirect naked domain requests to the www domain. You understand that this bucket is accessible to the world? Awesome, you are cruising right through this.

In order for a user to load your S3 website you’ll need to provide mapping from your root domain name to your S3 website url. This mapping is often referred to as a CNAME (Canonical Name) record inside of your Domain Name Servers (DNS) records.

```
www.my-awesome-site.com
--- maps to ---
www.my-awesome-site.com.s3-website-us-east-1.amazonaws.com
```

The process to complete this step varies depending on who your DNS provider is. In general this is what you are looking for within your DNS provider:

- Create a record for a host like `www`
- The record type must be `CNAME`
- The value must be your S3 website url `www.my-awesome-site.com.s3-website-us-east-1.amazonaws.com`

The naked domain bucket must have a DNS record as well to route requests for `my-awesome-site.com`.

```
my-awesome-site.com
--- maps to ---
my-awesome-site.com.s3-website-us-east-1.amazonaws.com
```

This will depend on your DNS provider as well. These are the high level steps you want to take when configuring your DNS records for mapping your naked domain.

- Create a record for your naked domain (my-awesome-site.com)
- The record type will be `A`, `ALIAS`, or `ANAME`
- Configure the mapping to your naked domain S3 website url after the bucket name `s3-website-us-east-1.amazonaws.com`

Once your DNS records have been updated then your domain can serve the static website in S3. This is can be verified by navigating to your domain in a browser and seeing it load.

Uploading Your Static Website

Your bucket is configured for static website hosting. You have configured DNS records for your domain to map to your S3 website url? Great! It is time to deploy your static website to S3.

Remember, S3 is a flat object store, which means each object in the bucket represents a key without any hierarchy. While the AWS S3 Console makes you believe there is a directory structure, there isn't. Everything stored in S3 is keys with prefixes.

This is important to note because if you have a website structure like this:

```
about/
    index.html
contact/
    index.html
css/
    styles.css
.....
index.html
```

It is easy to assume that this is a traditional directory structure. In fact, the AWS S3 Console makes you believe this as well.

But in actuality `about` is not a directory. It is a *prefix* for the `index.html` key. It is important when first learning S3 to keep this detail in mind. Objects in S3 are stored as flat keys that contain prefixes, not directories.

With that out of the way, let's upload your static website into your newly configured S3 website bucket!

If you are a GUI person, then you upload your static website to S3 via the AWS Console by completing these steps:

1. Navigate to S3 in the AWS Console.
2. Click into your bucket.
3. Click the "Upload" button.
4. Drag and drop or select "Add files", and add the entire static website directory.
5. Click "Next".
6. Leave the default permissions S3 offers.
7. Click "Next".

8. Leave the default permissions for “Set properties”.
9. Click “Next”.
10. Click “Upload”.

For the command line gurus out there, those 10 steps are reduced to one command line operation.

```
C:\Users\Kyle>aws s3 cp personal-blog/src/_site/ s3://www.my-awesome-site.com/ --recursive
```

Just like that your static website has been uploaded to S3. Navigating to your site in a browser now loads your static website.

Benefits

In five simple and easy steps, you have learned how to host your static website with AWS S3. Not to mention you scored some pretty sweet benefits.

💵 Low Cost

Hosting a website in S3 does not incur extra charges. You are paying standard S3 prices on GET requests and Data Transfer out of the bucket when a user visits your site.

- GET Requests cost $0.004 per 10,000 requests
- Data Transfer Out cost $0.090 per GB (up to 10 TB / month)

A Cost breakdown example: Let’s say that `www.my-awesome-site.com` loads 20 resources. The total size of those resources per visit is 1MB. The average total monthly visits is 20,000. Then we estimate the total cost of S3 on a monthly basis at around $1.96 per month.

Not long ago, you paid $10/month, so $2 is worth it.

🚧 Maintenance

Your static website now resides in S3. There is no longer any server side code to maintain and no web servers to configure and keep up to date.

⚖️ Scale

S3 is a high availability and durable service that AWS maintains. If your website goes from 10 users a day to 10 million, S3 scales your website automatically.

🔐 Security

There is no server running that you maintain. Thus you avoid making configuration errors that make you vulnerable to attacks. You are still responsible for the security of your bucket. Remember your website bucket is public!

Low cost, low maintenance, high scale, and high security. Those are some serious wins under your belt for hosting a static website and learning S3 at the same time. You now have a solid foundation for your problem that you can extend to leverage and learn even more AWS services.

In the upcoming chapters will extend the problem of hosting static websites on AWS to encompass low latency delivery as well. By taking advantage of the AWS Content Delivery Network, CloudFront, we can achieve millisecond latency for any user in the world. Combine that with AWS Certificate Manager and we can also get the cherry on top, free SSL communication.

Low latency, high performance, and free SSL. Does that peak your interest? Let’s get started.