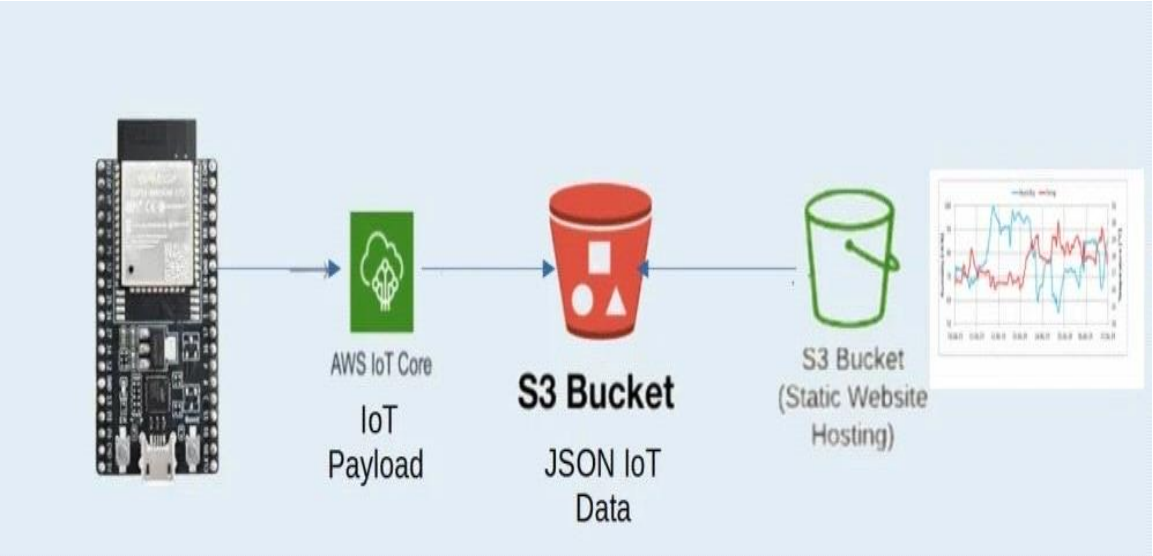
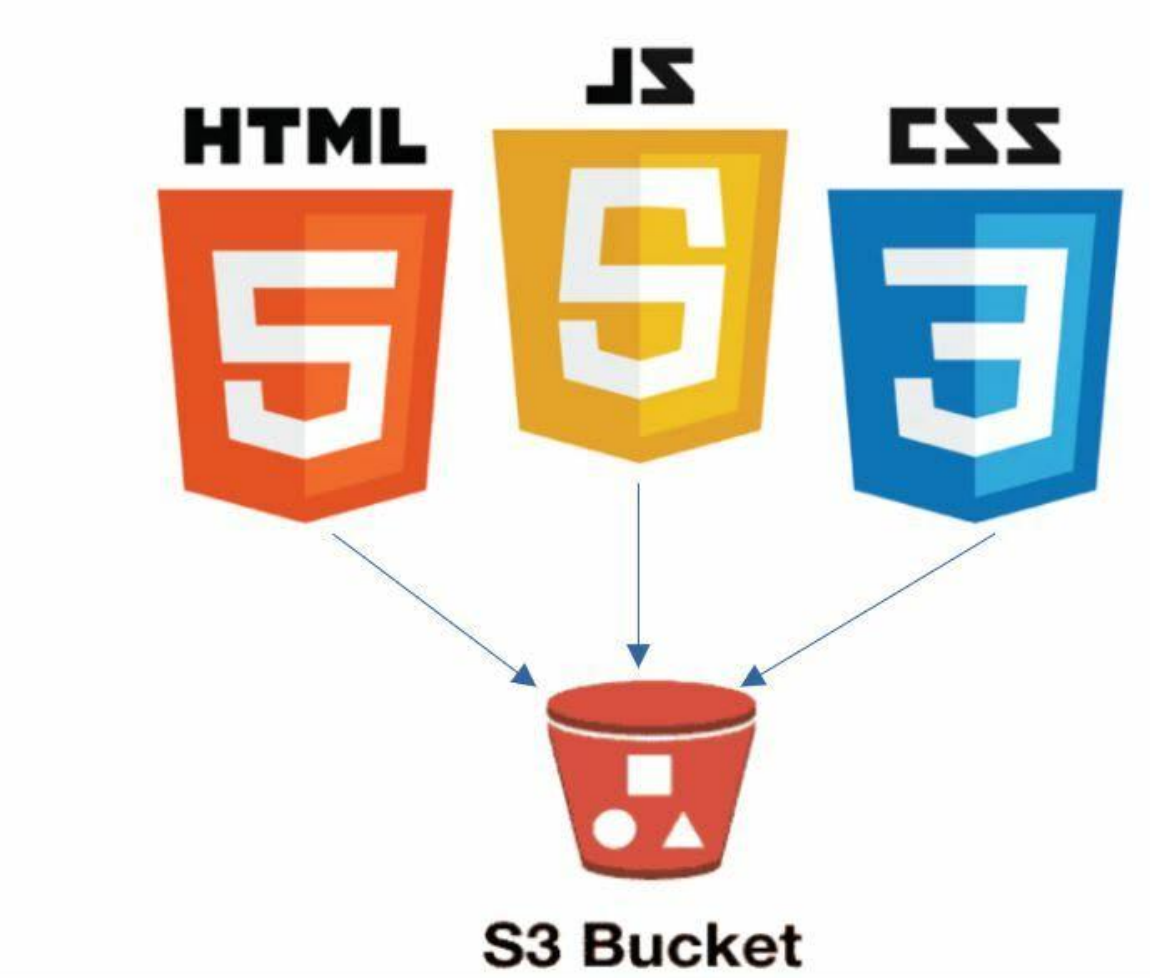
SERVERLESS IOT DATA PROCESSING DESING



When working with IoT devices, which transmit data to AWS, serverless IoT workflows can save the customer a tremendous amount of money. Instead of setting up an “always on” EC2 instance the client can engage individual AWS services only as needed. This multi-part IoT series will cover a variety of methods, with increasing levels of sophistication and functionality, to visualize IoT data on a static web host using various IoT centric services on AWS. The overall cost of using these AWS serverless services, even assuming you are off the free tier, will be

pennies for normal use.

Creating a public bucket in S3 Serverless IoT data



Making a Public S3 Bucket

The process of creating a public S3 bucket for website hosting

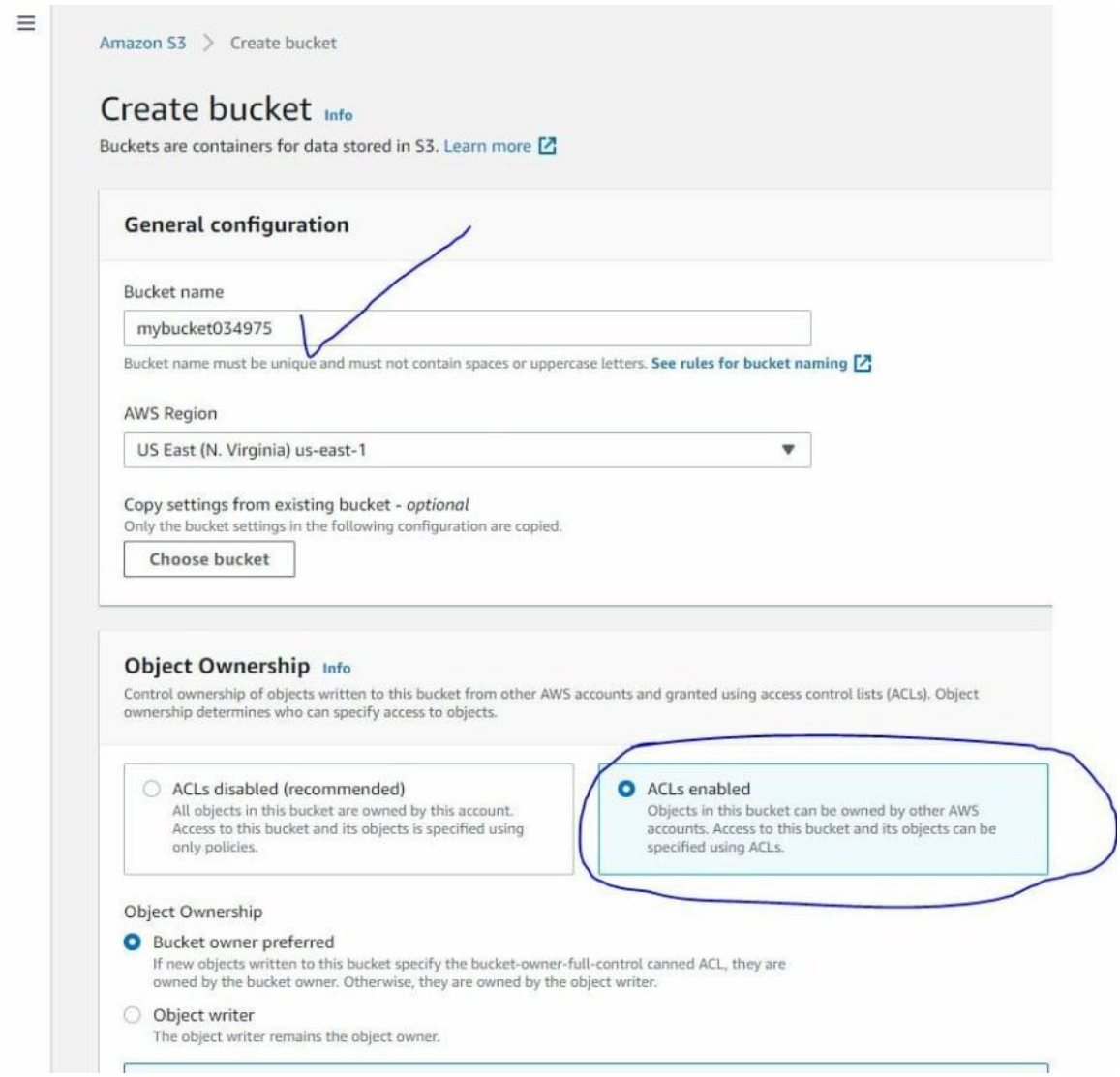
Go to AWS S3 and then select "Create bucket"

FOLLOW STEPS

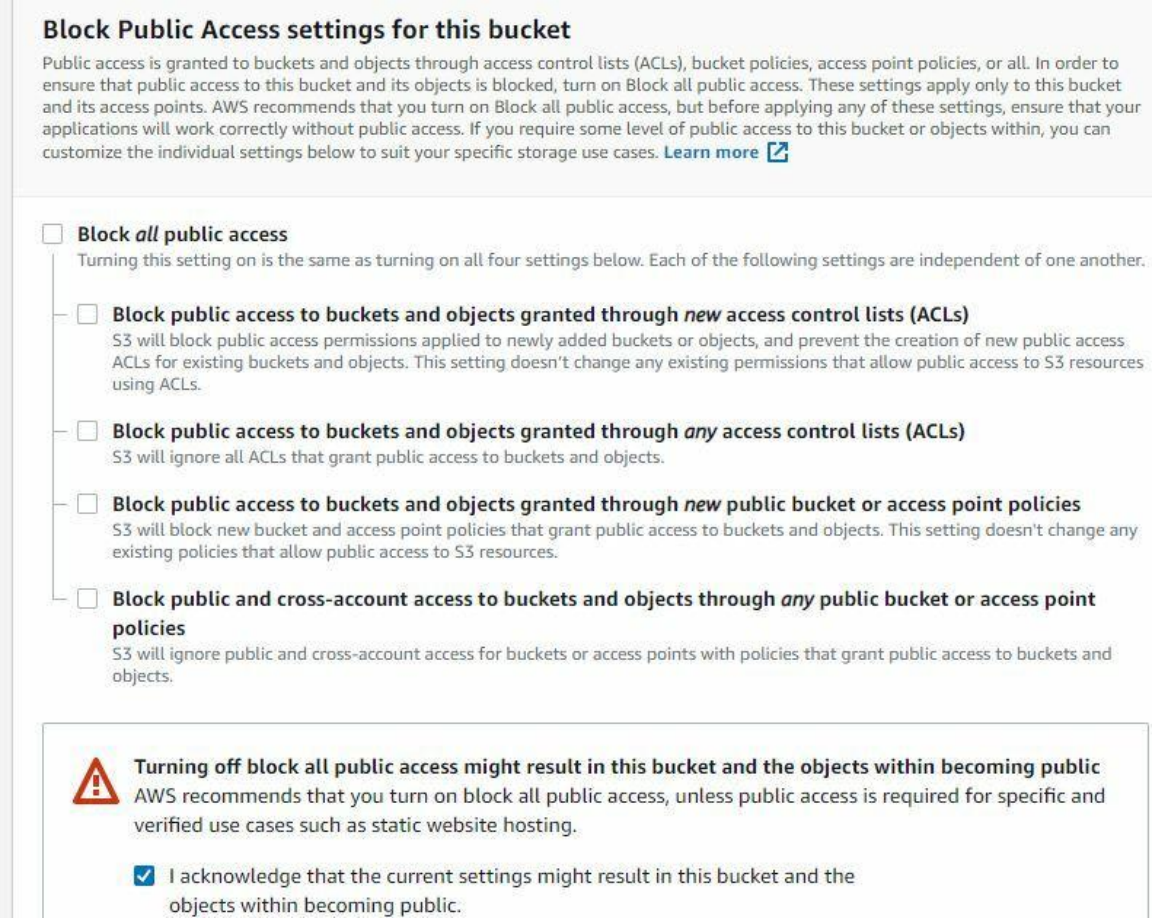
A) Give your bucket a globally unique name, here I call mine a catchy name: mybucket034975

B) Keep your S3 bucket in the same region as the rest of your AWS services for this lab.

C) Switch "Object Ownership" to "ACL's enabled", this is new for late 2021! We now must first enable our Access Control Lists to make them public.



*** Unblock your S3 bucket and acknowledge your really want to do this.**



***Finally, select the “Create bucket” button at the bottom of the screen. That's all you have to do for this page, but don’t worry, we are going to have more opportunity to make sure we really, really, and truly want to create a public bucket soon.**

Now go back into your newly created bucket and click on the “Permissions” tab.

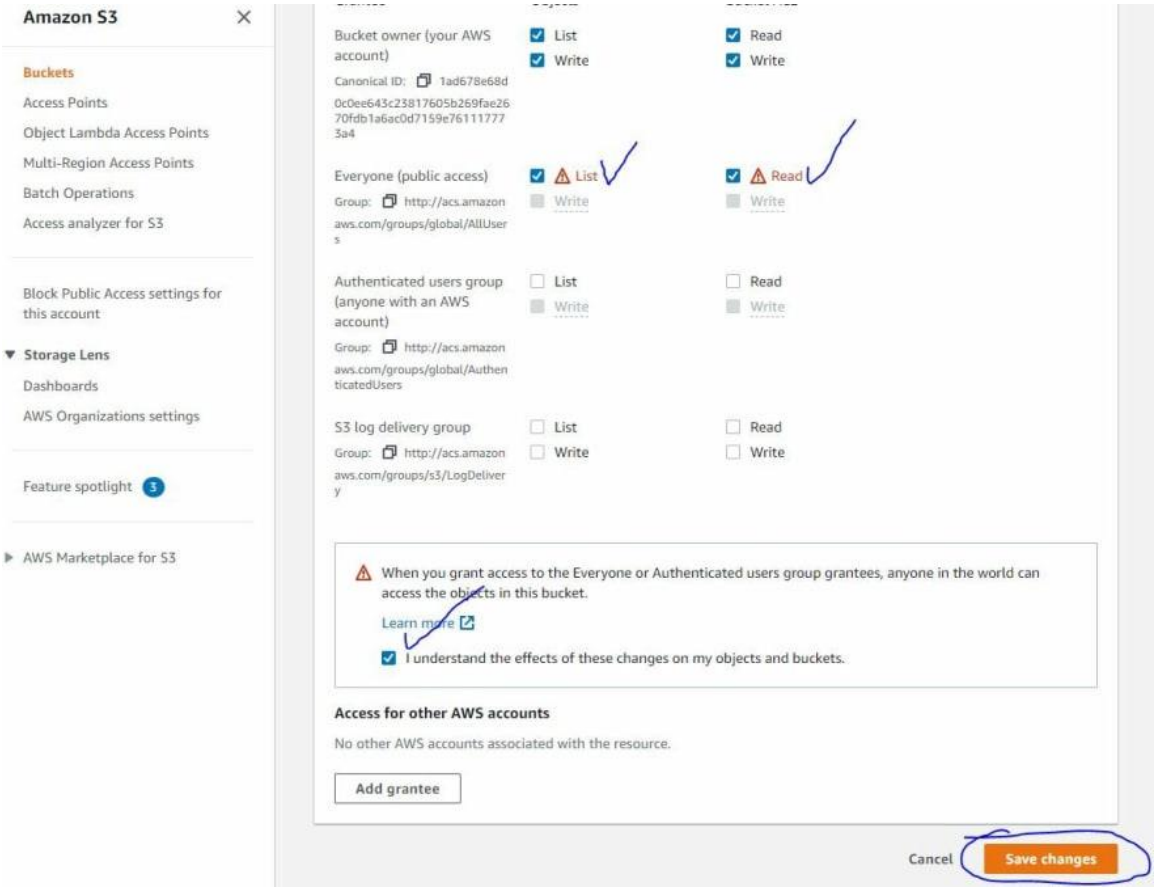
Go to Bucket Policy and choose “Edit.” We will paste and save a basic read-only policy.

```
[
    {
        "AllowedHeaders": [
            "Authorization"
        ],
        "AllowedMethods": [
            "GET"
        ],
        "AllowedOrigins": [
            "*"
        ],
        "ExposeHeaders": [],
        "MaxAgeSeconds": 6000
    }
]
```

You must paste the name of your bucket into the policy then follow it by ‘/*’ to allow access to all Get/Read partitions within the bucket. Also it's a good idea to change the “Sid” to something unique within your account.

*Now we get a chance to visit that ACL we enabled earlier in this process. Click “Edit” then

make the changes as shown below:



We are giving “Everyone,” or at least those know or can discover our unique bucket URL, permission to read our bucket info. Click on the 'List' and 'Read' buttons where shown and then acknowledge again that you are extra special certain that you want to do this . Then click “Save changes.”

*** Wow, we are at our last step in creating a**

public bucket. Now we should set the CORS policy so we don’t get any pesky “mixed use” access-control non-allowed origin issues for cross domain access – I hate those CORS rules used to be in XML only format and then AWS decided to keep everything consistent and switch the CORS format to JSON. Even though this change caused some legacy conflict issues with existing XML CORS rules it was the right choice as JSON is clearly better than XML despite what the SOAP fans on social media will tell you . Below is a generic CORS JSON document you can use in your own S3 bucket:

```
[
    {
        "AllowedHeaders": [
            "Authorization"
        ],
        "AllowedMethods": [
            "GET"
        ],
        "AllowedOrigins": [
            "*"
        ],
        "ExposeHeaders": [],
        "MaxAgeSeconds": 6000
    }
]
```


COMPLETE..