<https://docs.aws.amazon.com/AmazonS3/latest/userguide/UsingBucket.html>

To upload your data (photos, videos, documents, etc.) to Amazon S3, you must first create an S3 bucket in one of the AWS Regions. You can then upload any number of objects to the bucket.

In terms of implementation, buckets and objects are AWS resources, and Amazon S3 provides APIs for you to manage them. For example, you can create a bucket and upload objects using the Amazon S3 API. You can also use the Amazon S3 console to perform these operations. The console uses the Amazon S3 APIs to send requests to Amazon S3.

Bucket configuration options

Amazon S3 supports various options for you to configure your bucket. For example, you can configure your bucket for website hosting, add a configuration to manage the lifecycle of objects in the bucket, and configure the bucket to log all access to the bucket. Amazon S3 supports subresources for you to store and manage the bucket configuration information. You can use the Amazon S3 API to create and manage these subresources. However, you can also use the console or the AWS SDKs.

These are referred to as subresources because they exist in the context of a specific bucket or object. The following table lists subresources that enable you to manage bucket-specific configurations.

| **Subresource** | **Description** |
| --- | --- |
| cors (cross-origin resource sharing) | You can configure your bucket to allow cross-origin requests.  For more information, see [Using cross-origin resource sharing (CORS)](https://docs.aws.amazon.com/AmazonS3/latest/userguide/cors.html). |
| event notification | You can enable your bucket to send you notifications of specified bucket events.  For more information, see [Amazon S3 Event Notifications](https://docs.aws.amazon.com/AmazonS3/latest/userguide/NotificationHowTo.html). |
| lifecycle | You can define lifecycle rules for objects in your bucket that have a well-defined lifecycle. For example, you can define a rule to archive objects one year after creation, or delete an object 10 years after creation.  For more information, see [Managing your storage lifecycle](https://docs.aws.amazon.com/AmazonS3/latest/userguide/object-lifecycle-mgmt.html). |
| location | When you create a bucket, you specify the AWS Region where you want Amazon S3 to create the bucket. Amazon S3 stores this information in the location subr |

Application

Description automatically generated with low confidence

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Go to bottom of page and click create bucket

Graphical user interface, text, application, email

Description automatically generated

We have initially made our Bucket and objects not public

Click on the blue hyperlink for the bucket

Graphical user interface, application

Description automatically generated

Select the Properties tab

**Graphical user interface

Description automatically generated with low confidence**

**Click edit**

**Graphical user interface, text, application

Description automatically generated**

**When you click Enable you will see some new attibutes**

**Graphical user interface, text, application, email

Description automatically generated**

**Index document needs to be set to your home page**

**A picture containing graphical user interface

Description automatically generated**

**Graphical user interface, text, application, Teams

Description automatically generated**

**Click Edit**

**Graphical user interface, text, application

Description automatically generated**

**Unclick Block *all* public access**

**This is the reverse of how the video shows you**

**Save changes**

**Graphical user interface, text, application

Description automatically generated**

**Confirm**

**Return to the Amazon S3 top level page**

**Graphical user interface, text, application, email

Description automatically generated**

**Note that Access is Objects can be public**

**Click on Bucket and upload**

**Graphical user interface, application

Description automatically generated**

**Click upload**

**Now drag and drop your desired files**

**Graphical user interface, application

Description automatically generated**

**Graphical user interface, text, application, email

Description automatically generated**

**Click Upload**

**Graphical user interface, application, Teams

Description automatically generated**

****

**What is the link to our website?**

**Go back to Bucket and properties**

**Graphical user interface, application

Description automatically generated**

**Graphical user interface, text, application, email

Description automatically generated**

**Click on our link there**

[**http://gtj-unt.s3-website-us-west-1.amazonaws.com/**](http://gtj-unt.s3-website-us-west-1.amazonaws.com/)

**Graphical user interface, text

Description automatically generated**

**Hmmmm…..**

**Access to the bucket but not the files**

**Go to the Objects tab. Select all files click action and select Make public**

**Actions make public**

**Graphical user interface, table

Description automatically generated**

**Graphical user interface, text, application, email

Description automatically generated**

**Table

Description automatically generated**

**Click make public**

**Graphical user interface, application

Description automatically generated**

**Click Close**

**Now let’s click on our link again**

**Or refresh the page.**

**We have a nice STATIC website again. Similar to what we did several weeks ago**

**Let’s add a Lambda functions**

Lambda is a compute service that lets you run code without provisioning or managing servers. Lambda runs your code on a high-availability compute infrastructure and performs all of the administration of the compute resources, including server and operating system maintenance, capacity provisioning and automatic scaling, code monitoring and logging. With Lambda, you can run code for virtually any type of application or backend service. All you need to do is supply your code in one of the [languages that Lambda supports](https://docs.aws.amazon.com/lambda/latest/dg/lambda-runtimes.html).

**Return to services page, find the compute block at the very top left and select Lambda**

**Graphical user interface, application

Description automatically generated with medium confidence**

**Graphical user interface, application, Teams

Description automatically generated**

**Graphical user interface, application

Description automatically generated**

**Explore the other tabs. We are going to create a function from scratch**

**Give it a name -- createQuote**

**Graphical user interface, text, application, email, Teams

Description automatically generated**

**Choose your runtime environment**

**Graphical user interface, text, application, email

Description automatically generated**

**We need an Execution role. What privileges will it have**

**Policy templates we just need one Basic Lambda@Edge.....**

**Just type in the word Basic and it will find it for you**

**Graphical user interface, text, application, email

Description automatically generated**

**Graphical user interface, text, application, email

Description automatically generated**

****

**Walk around the page and look at the tabs**

**First the database DynamoDB**

**Graphical user interface, application

Description automatically generated**

**Graphical user interface

Description automatically generated**

**Graphical user interface, text, application, Teams

Description automatically generated**

****

**Graphical user interface, application

Description automatically generated**

**Graphical user interface, application, Teams

Description automatically generated**

**Click on View items**

**Graphical user interface, application, Teams

Description automatically generated**

**Create item**

**Graphical user interface, text, application

Description automatically generated**

**Click Create item**

**Graphical user interface, application

Description automatically generated**

**Now check the box on our item and select Delete item from the Actions drop down**

**We need access to the table from the lambda function**

**Graphical user interface, application

Description automatically generated**

**Click on IAM (Identity and Access Management)**

**Graphical user interface, application

Description automatically generated**

**Click on Roles**

**Graphical user interface, text, application, email

Description automatically generated**

**Click on the role we created quoteRole**

**We have access to AWS Lambda but NOT Dynamo**

**Click on Attach policies**

**Graphical user interface, text, application, email

Description automatically generated**

**Search for Dynamo and select DB Full Access**

**Table

Description automatically generated with medium confidence**

**Check the box and press Attach policy**

**Go back to the Lambda service**

**Graphical user interface, application

Description automatically generated**

**Graphical user interface, text, application

Description automatically generated**

**Click on our createQuote Lambda function**

**Go to the source code panel**

**Before**

**Graphical user interface, application, website

Description automatically generated**

**We are going overhaul our original do-nothing handler. Once complete SAVE the code by clicking on the File -> Save menu item**

**Graphical user interface, application

Description automatically generated**

**Here is the final code to use**

var AWS = require('aws-sdk'),

documentClient = new AWS.DynamoDB.DocumentClient();

exports.createQuote = function (event, context, callback) {

var params = {

Item : {

"id" : new Date().getTime().toString(),

"Name": event.name

},

TableName: process.env.TABLE\_NAME

};

documentClient.put(params, function(err, data) {

callback(err, data);

});

};

**After**

**Make sure you save it**

**Graphical user interface, text, application

Description automatically generated**

**Need to modify the Runtime settings**

**Graphical user interface, text, application, Teams

Description automatically generated**

**Click Edit**

**Change the Handler name from index.handler to index.createQuote**

**Graphical user interface, text, application

Description automatically generated**

**Now we need an environment variable. You will create that in the Configuration panel, on the Environment variables tab. Click the Edit button**

**Graphical user interface, text, application, email

Description automatically generated**

**Graphical user interface, application, Teams

Description automatically generated**

**Click the Add environment variable button**

**Graphical user interface

Description automatically generated with medium confidence**

**Create the TABLE\_NAME environment variable as we saw referenced in the Lambda function code**

**Graphical user interface, application, Teams

Description automatically generated**

**Click Save**

**Graphical user interface, text, application, email, Teams

Description automatically generated**

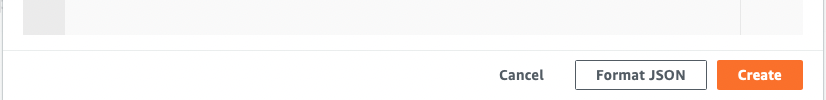
**We now have a completed function let’s create a test**

**Graphical user interface, application

Description automatically generated**

**Click on Test drop down and select Configure test event**

**Graphical user interface, text, application, email

Description automatically generated** 

Add in the Event name and body of the event and click Create

**Graphical user interface, text, application

Description automatically generated**

**Deploy the function**

**Graphical user interface, application

Description automatically generated**

**Click Deploy**

**Test your function**

**Click Test**

**Not very exciting all it does is return an empty JSON object**

**Now it is time for a new function**

**At this point I am beginning to see what my ‘favorite’ services are**

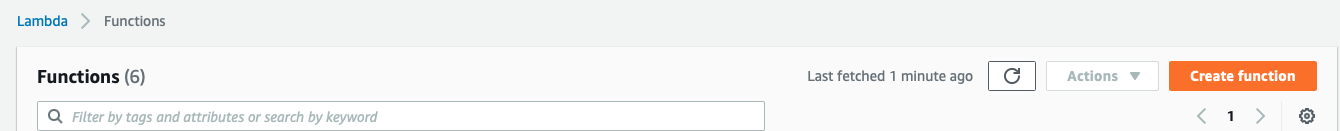
**Graphical user interface, application

Description automatically generated**

**Click on Lambda**

**And click on Create function**

**Graphical user interface

Description automatically generated **

**Author from scratch**

**Function name is getQuotes**

**Runtime Node.js 12.x**

**Graphical user interface, application

Description automatically generated**

**Open the Change default execution role**

**Select Use an existing role**

**Select service-role/quoteRole**

**Graphical user interface, text, application, email

Description automatically generated**

**And finally**

****

**Function is now created. It needs to be edited**

**Graphical user interface, text, application, chat or text message

Description automatically generated**

**Click the edit button on Runtime settings**

**Graphical user interface, application, Teams

Description automatically generated**

**Change the name of the handler to index.getQuotes**

**Graphical user interface, text, application

Description automatically generated**

**Also need the same environment variable as before**

**Click on the Configuration panel and the Environment variables tab**

**Graphical user interface, application

Description automatically generated**

**Then Edit**

**Graphical user interface, text, application, email, Teams

Description automatically generated**

**Graphical user interface, application, Teams

Description automatically generated**

**Click the Add environment variable button**

**Graphical user interface, application, Teams

Description automatically generated**

**Create the Key TABLE\_NAME with the Value of quotes**

**And click Save**

**Return to the Code panel**

**And create the getQuotes handler. This handler will respond with a list of quotes in our DynamoDB table quotes**

**Graphical user interface, text, application

Description automatically generated**

var AWS = require('aws-sdk'),

documentClient = new AWS.DynamoDB.DocumentClient();

exports.getQuotes = function (event, context, callback) {

var params = {

TableName : process.env.TABLE\_NAME

};

documentClient.scan(params, function(err, data) {

if (err) {

callback(err,null);

} else {

callback(null, data.Items)

}

});

};

**Finally, we need to create a test**

**Graphical user interface, application

Description automatically generated**

**Name the test getQuotes and the body of our test is empty. The function requires no parameters it just returns all rows from the table**

**Graphical user interface, text, application, email

Description automatically generated**

****

**Save the code**

**Graphical user interface, application

Description automatically generated**

**Deploy the code**

**Graphical user interface

Description automatically generated**

**And then click the Test button**

**Here is my Execution results**

**Graphical user interface, text, application, email

Description automatically generated**

**A quick glance back at our quotes table shows the same results**

**Graphical user interface, application

Description automatically generated**

**This is lovely to have it working inside of AWS. We need it to be accessible to the outside world**

**Graphical user interface, text, application

Description automatically generated**

**From the services options select API Gateway**

**Get Started**

**Graphical user interface, application, Teams

Description automatically generated**

**Choose the REST API**

**Graphical user interface, application, website

Description automatically generated**

**Graphical user interface, application

Description automatically generated**

**This is a REST API and we are creating a New API**

**Name the API quotesAPI**

**And click Create API**

**Graphical user interface, text, application

Description automatically generated**

**Graphical user interface, application, Word

Description automatically generated**

**We will select an action – Create Method**

**Graphical user interface, application

Description automatically generated**

**Then we choose the HTTP Method -- POST**

**Graphical user interface, text, application, chat or text message

Description automatically generated**

**Click the Check to the right of the POST**

**Graphical user interface, text, application, chat or text message

Description automatically generated**

**Graphical user interface, text, application

Description automatically generated**

**The Lambda function we want is createQuote**

**Graphical user interface, text, application

Description automatically generated**

**How do we know which region?**

**Look at your URL in the address bar**

****

**For me that is us-east-2**

**And click Save**

**Graphical user interface, application, Teams

Description automatically generated**

**And we see the Life Cycle of the POST method**

**Graphical user interface, application, Teams

Description automatically generated**

**Repeat the process of selecting an Action -> Create Method. Choose the GET method this time and the Lambda Function will be getQuotes**

**Graphical user interface, text, application, email

Description automatically generated**

**Click Save**

**Graphical user interface, text, application

Description automatically generated**

**Click OK**

**And we see the Life Cycle of the Get method**

**Graphical user interface, application, Teams

Description automatically generated**

**Next we will deploy our APIs**

**Graphical user interface, application

Description automatically generated**

**Graphical user interface, text, application

Description automatically generated**

**Select [New Stage]**

**Name your stage whatever you like**

**And Click Deploy**

**Graphical user interface, application

Description automatically generated**

**See the URL? We are going to copy that and use it in Postman!**

**Now we move over to Postman**

**Create a new request**

**Set the method to POST**

**Click on the Body panel and the raw radio button**

**In the body for our request put in a message just as we did over in the AWS DynamoDB pages**

**Here I just created a quote like this**

**{**

**“name”: “This is coming from Postman. So is this”**

**}**

**Watch out for the quotes those are grammatical quotes not programming quotes**

**Now click on send. The createQuote method is being requested. Note the Response below {}. Which indicates no errors. Make sure the text type is JSON. This on the same line as raw to the far right.**

**Graphical user interface, text, application, email

Description automatically generated**

**Now change the Method from POST to GET. No big reason to change the body as it is ignored with this request. Click SEND. This will invoke the getQuotes method**

**And lo and behold the response is**

**Graphical user interface, text, application, email

Description automatically generated**

**All the quotes we have been creating!**