

ECV-SVR-310

Build Serverless Environment with AWS Lambda

2017.08.28

Version 1.1

Agenda

Introduction.....	3
Overview	3
Topics covered	3
Introducing the Technologies	3
Amazon S3?	3
Amazon DynamoDB	3
AWS Lambda.....	3
Amazon API Gateway	4
About this lab	4
Scenario	4
Architecture Diagram.....	5
Create a website hosting through S3	5
Create S3 bucket.....	5
Upload files to S3 Bucket	6
Enable Static website hosting through S3	8
View S3 static website hosting.....	11
Working with Amazon DynamoDB.....	12
Create DynamoDB Table	12
Create Items in DynamoDB Table	13
Working with Lambda Function.....	15
Create Lambda Function.....	15
Working with API Gateway.....	18
Create APIs in API Gateway	18
Deploy APIs	22
Conclusion	25

Introduction

Overview

This lab demonstrated how to create a serverless environment which leverage AWS Serverless service. To build an environment which offload maintain loading from customer side.

Topics covered

By the end of this lab, you will be able to:

- Create a static website hosting through S3
- Create a Lambda function
- Create DynamoDB Table
- Create API though API Gateway

Introducing the Technologies

Amazon S3?

Amazon S3 is storage for the Internet. It's a simple storage service that offers software developers a highly-scalable, reliable, and low-latency data storage infrastructure at very low costs.

Amazon DynamoDB

Amazon DynamoDB is a fully managed NoSQL database service that provides fast and predictable performance with seamless scalability. Amazon DynamoDB enables customers to offload the administrative burdens of operating and scaling distributed databases to AWS, so they don't have to worry about hardware provisioning, setup and configuration, throughput capacity planning, replication, software patching, or cluster scaling.

AWS Lambda

AWS Lambda lets you run code without provisioning or managing servers. You pay only for the compute time you consume - there is no charge when your code

is not running. With Lambda, you can run code for virtually any type of application or backend service - all with zero administration. Just upload your code and Lambda takes care of everything required to run and scale your code with high availability. You can set up your code to automatically trigger from other AWS services or call it directly from any web or mobile app.

Amazon API Gateway

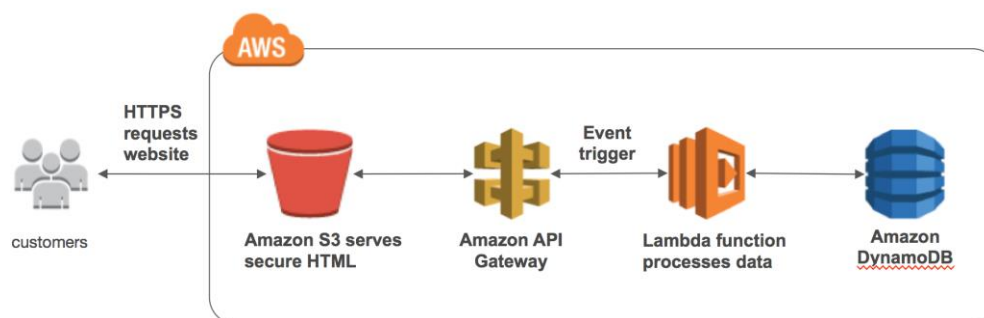
Amazon API Gateway is a fully managed service that makes it easy for developers to publish, maintain, monitor, and secure APIs at any scale. With a few clicks in the AWS Management Console, you can create an API that acts as a “front door” for applications to access data, business logic, or functionality from your back-end services, such as applications running on Amazon Elastic Compute Cloud (Amazon EC2), code running on AWS Lambda, or any web application. Amazon API Gateway handles all of the tasks involved in accepting and processing up to hundreds of thousands of concurrent API calls, including traffic management, authorization and access control, monitoring, and API version management. Amazon API Gateway has no minimum fees or startup costs. You pay only for the API calls you receive and the amount of data transferred out.

About this lab

Scenario

First, you will hosting a static website in S3. You will use DyanmoDB as a database which will record the data. All we need to do is retrieve the data from DynamoDB and show the record in S3 web page.

Architecture Diagram



The workshop's region will be in 'Oregon'

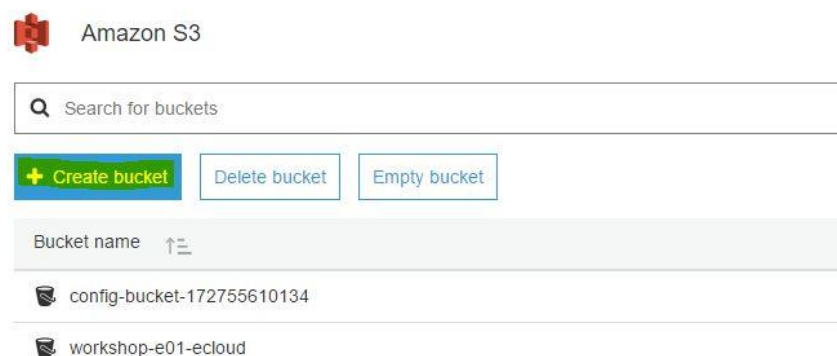
Create a website hosting through S3

Create S3 bucket

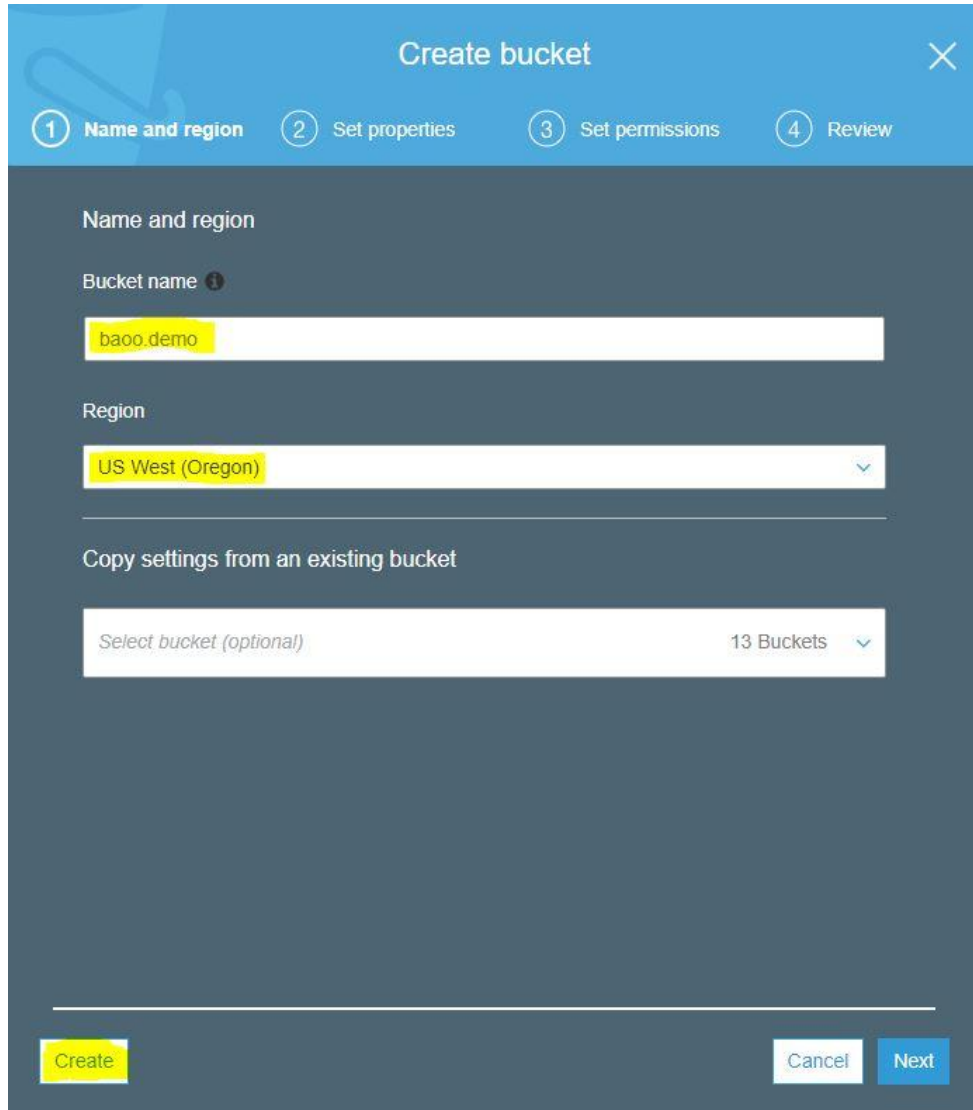
1.1. On the service menu, click 'S3'



1.2. Click 'Create Bucket'

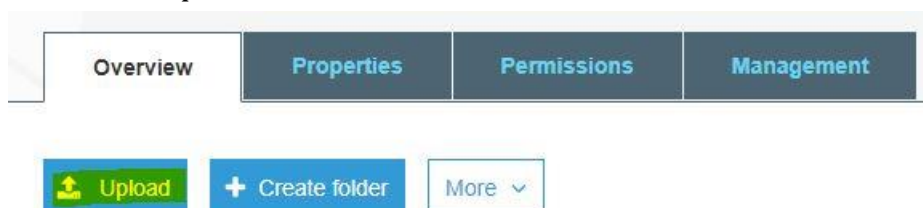


- 1.3. For Bucket Name, type 'Unique Name'
- 1.4. For Region, choose 'US West(Oregon)
- 1.5. Click "Create" to create S3 bucket without any setting

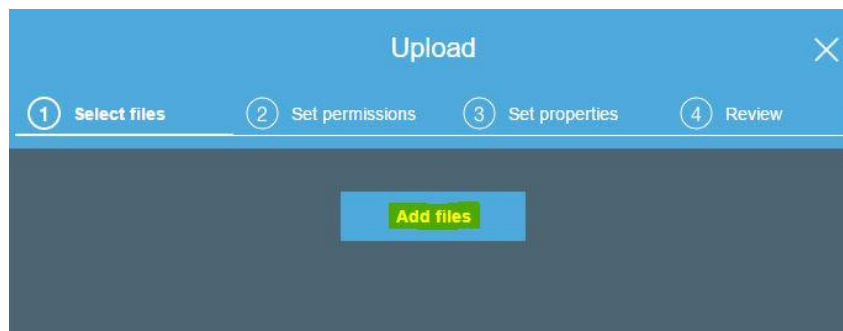


Upload files to S3 Bucket

- 1.6. Select the bucket which you created before
- 1.7. Click 'Upload'

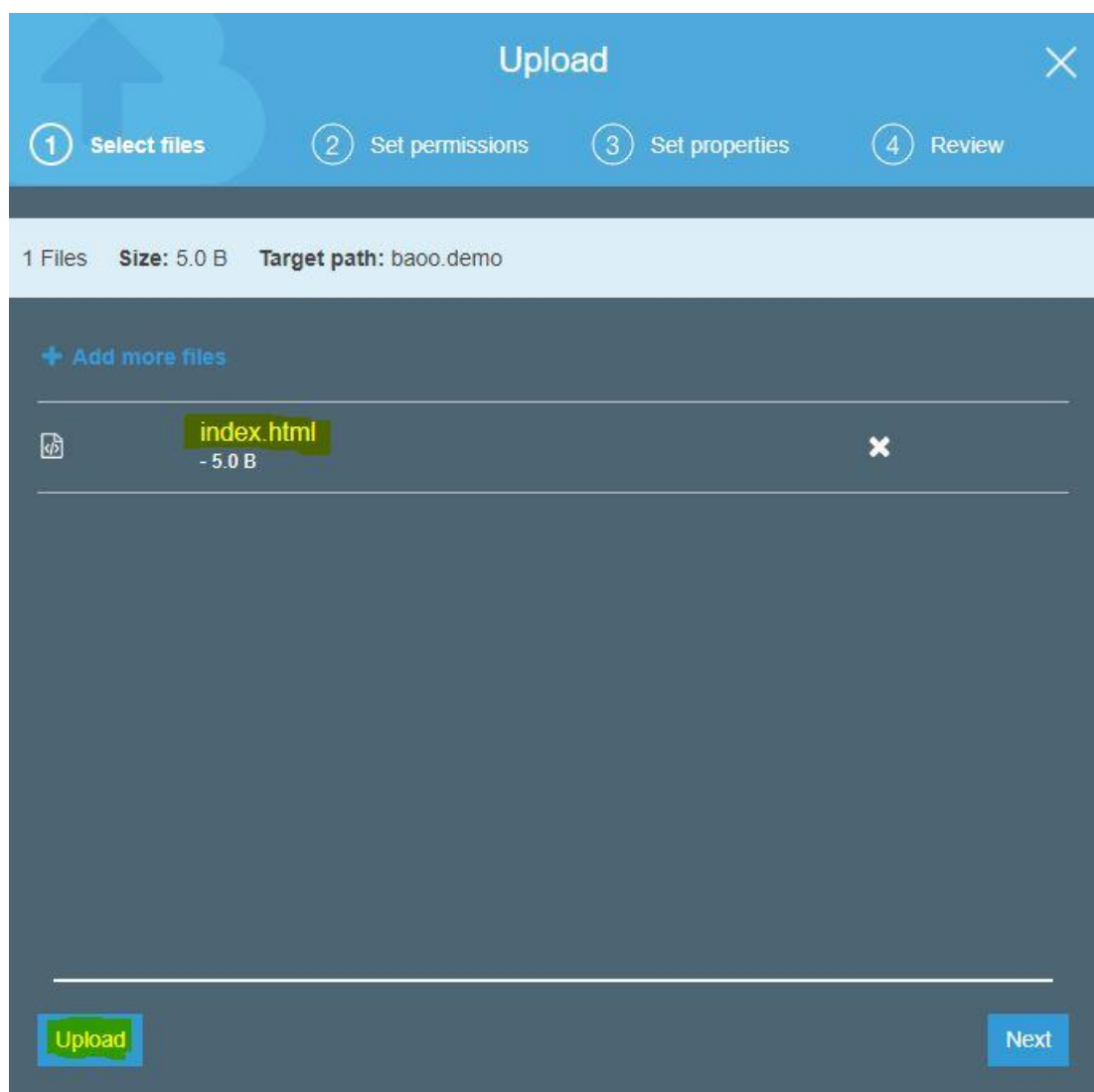


1.8. Click 'Add files'

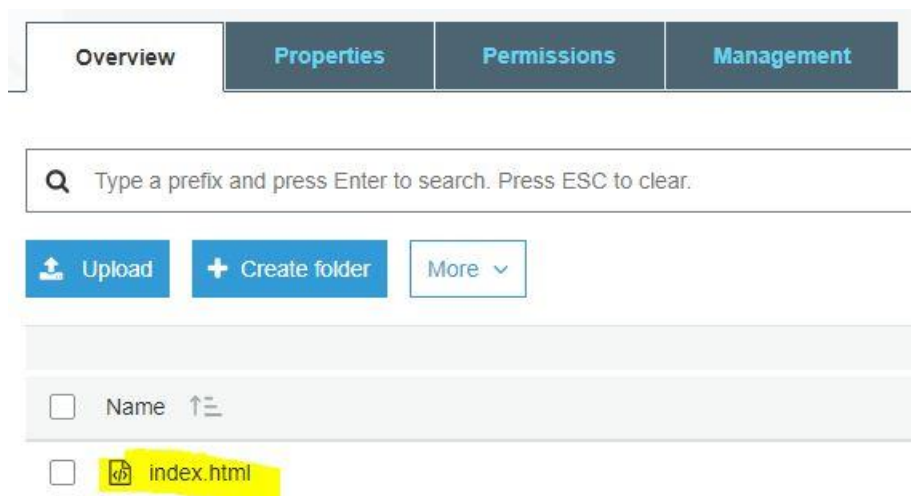


1.9. Select the html file which in the download folder, then choose

1.10. Click 'Upload' without any setting.

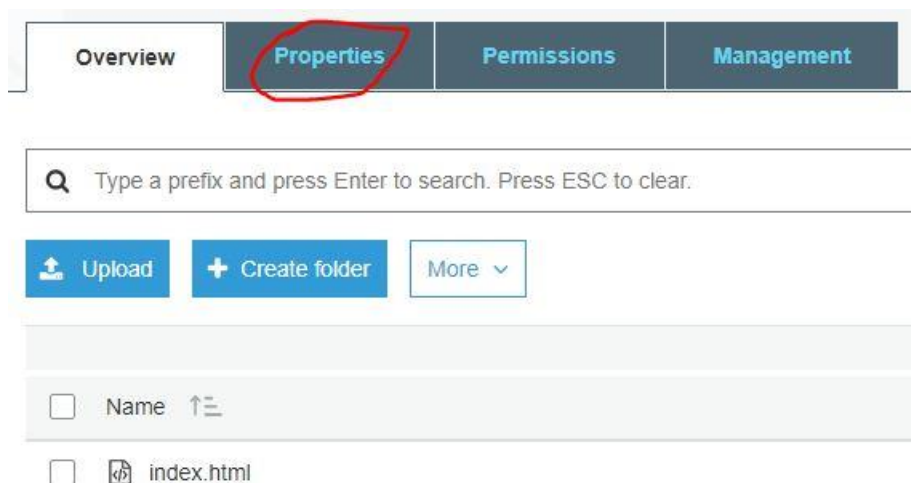


1.11. Check the status of the object which exists in S3 bucket

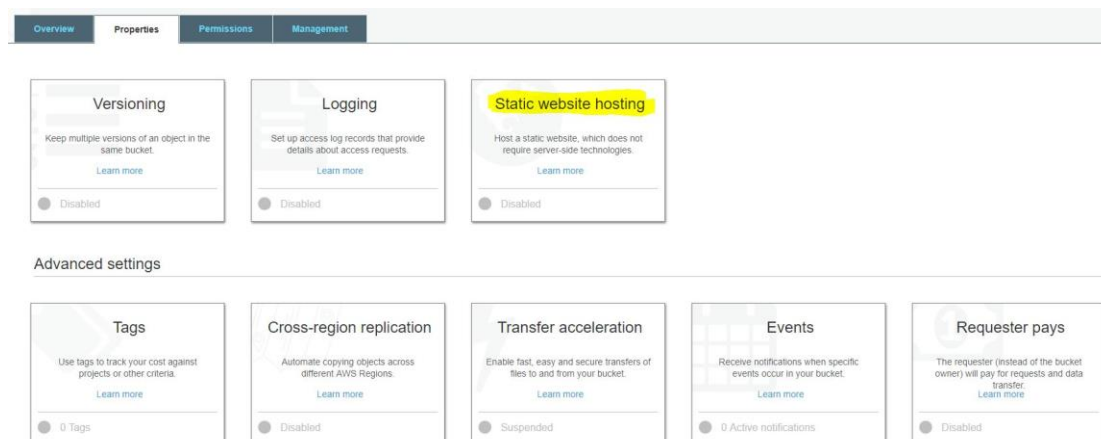


Enable Static website hosting through S3

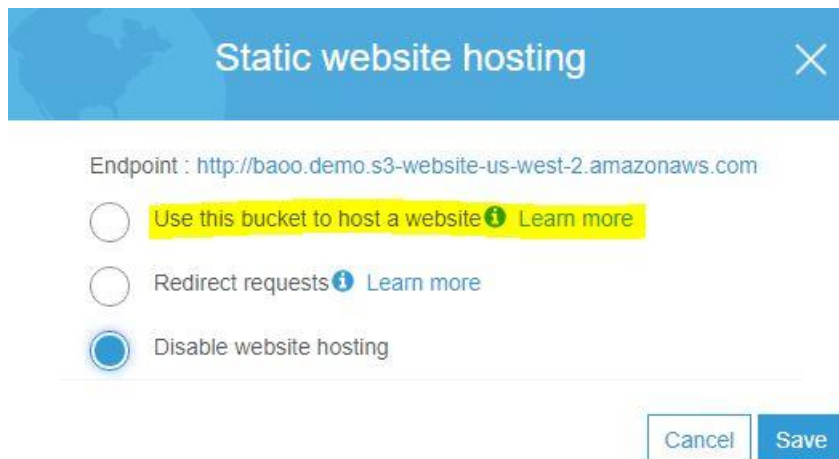
1.12. Select 'Properties' tab



1.13. Select 'Static website hosting'



1.14. Select 'Use this bucket to host a website'



Static website hosting

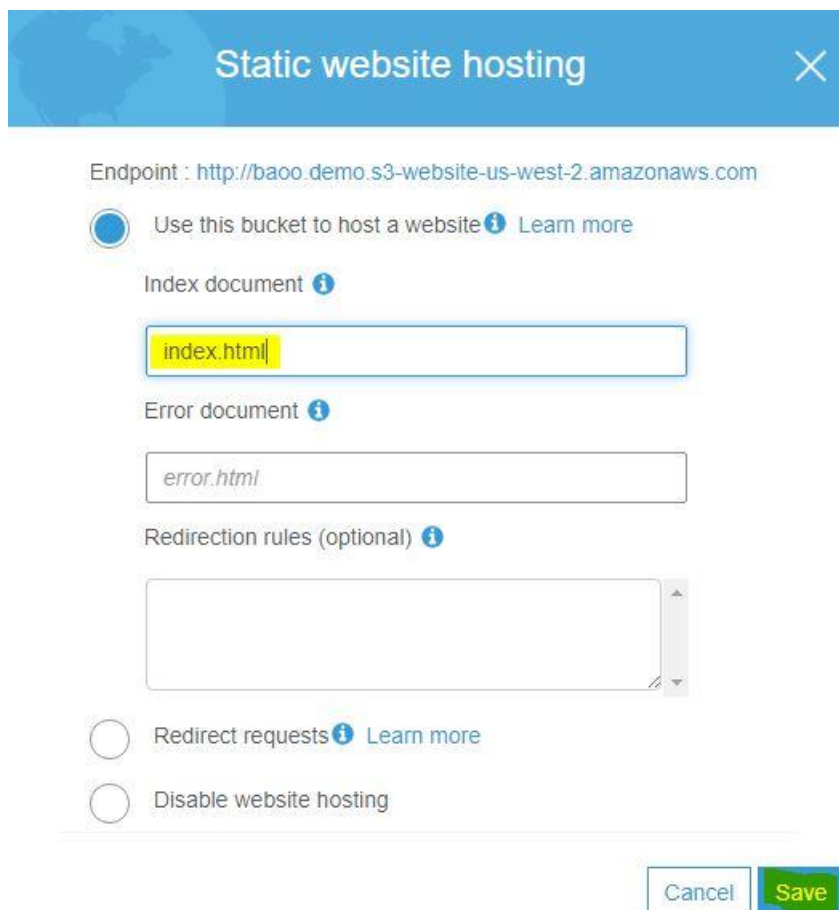
Endpoint : `http://baoo.demo.s3-website-us-west-2.amazonaws.com`

☐ Use this bucket to host a website [Learn more](#)

☐ Redirect requests [Learn more](#)

☒ Disable website hosting

1.15. Type the file name for the index document, then click 'Save'



Static website hosting

Endpoint : `http://baoo.demo.s3-website-us-west-2.amazonaws.com`

☒ Use this bucket to host a website [Learn more](#)

Index document [i](#)

Error document [i](#)

Redirection rules (optional) [i](#)

☐ Redirect requests [Learn more](#)

☐ Disable website hosting

1.16. Verify status of 'Bucket hosting'



1.17. Choose 'Bucket Policy' tab in 'Permissions'



1.18. Update the bucket policy by giving example policy, and modify the resource to your own S3 ARN. Then click "Save"

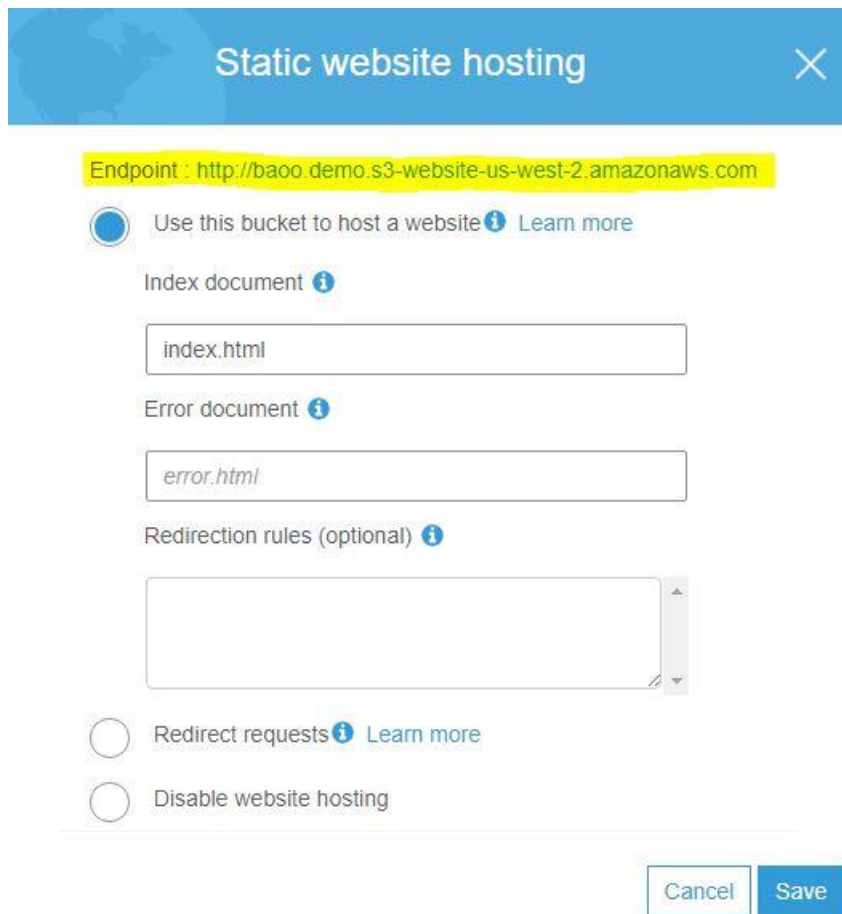
Bucket policy editor ARN: `arn:aws:s3:::baoo.demo`
Type to add a new policy or edit an existing policy in the text area below.

```
1 {
2   "Version": "2012-10-17",
3   "Statement": [{
4     "Sid": "PublicReadGetObject",
5     "Effect": "Allow",
6     "Principal": "*",
7     "Action": ["s3:GetObject"],
8     "Resource": ["arn:aws:s3:::baoo.demo/*"]
9   }]
10 }
11 ]
12 }
13
```



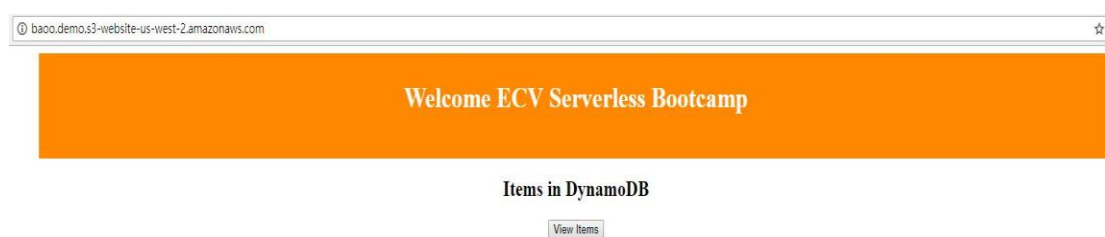
View S3 static website hosting

1.19. Select the 'Endpoint' in the 'Static website hosting' tab



The image shows a 'Static website hosting' configuration dialog. At the top, the title bar says 'Static website hosting' with a close button. Below the title bar, the 'Endpoint' is set to 'http://baoo-demo.s3-website-us-west-2.amazonaws.com'. There are two radio button options: 'Use this bucket to host a website' (selected) and 'Redirect requests'. Under 'Use this bucket to host a website', there are fields for 'Index document' (set to 'index.html') and 'Error document' (set to 'error.html'). There is also a text area for 'Redirection rules (optional)'. At the bottom, there are 'Cancel' and 'Save' buttons.

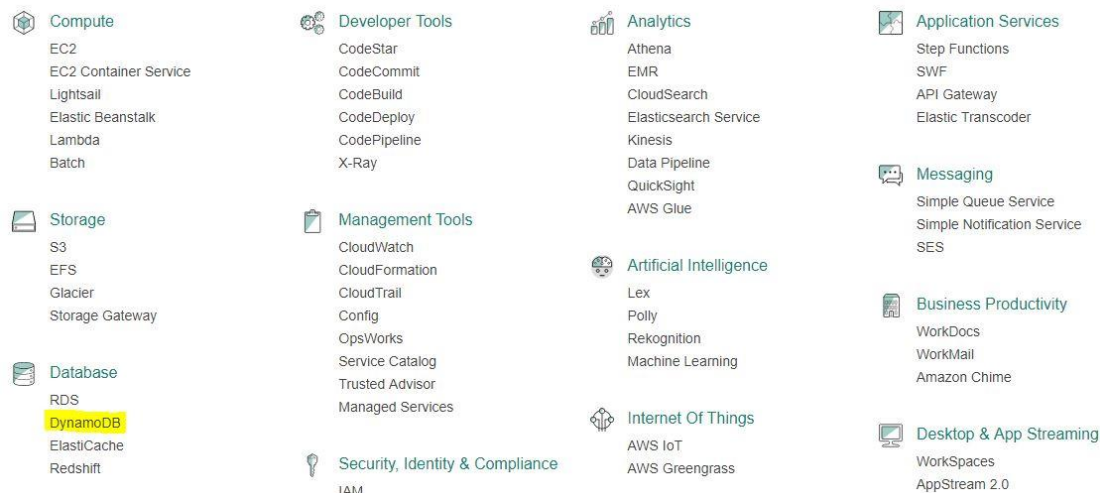
1.20. You will see the website as below:



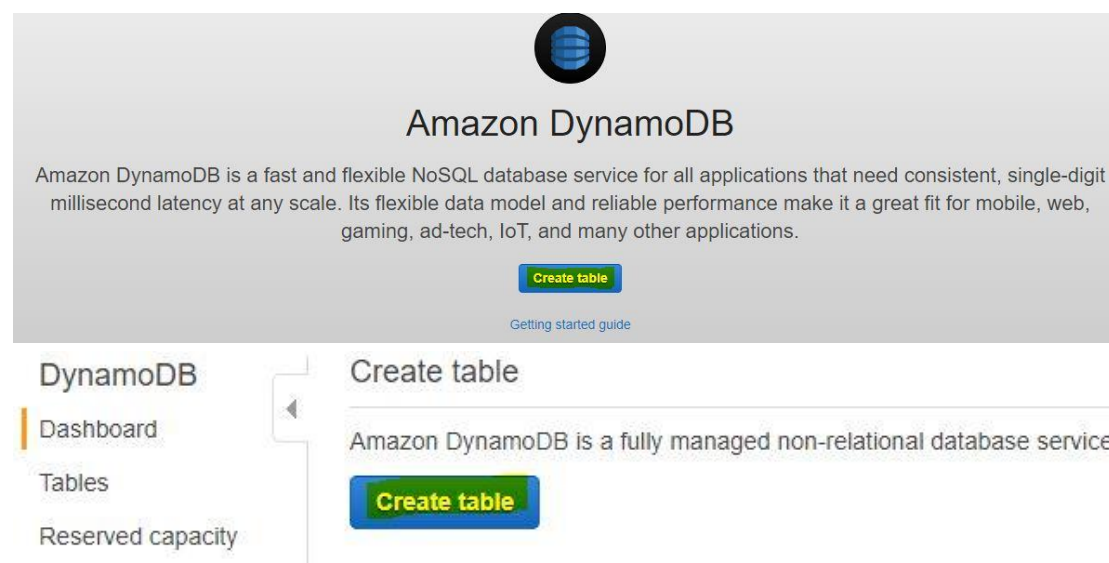
Working with Amazon DynamoDB

Create DynamoDB Table

2.1. On the service menu, click 'DynamoDB'



2.2. Click 'Create table' in the welcome page

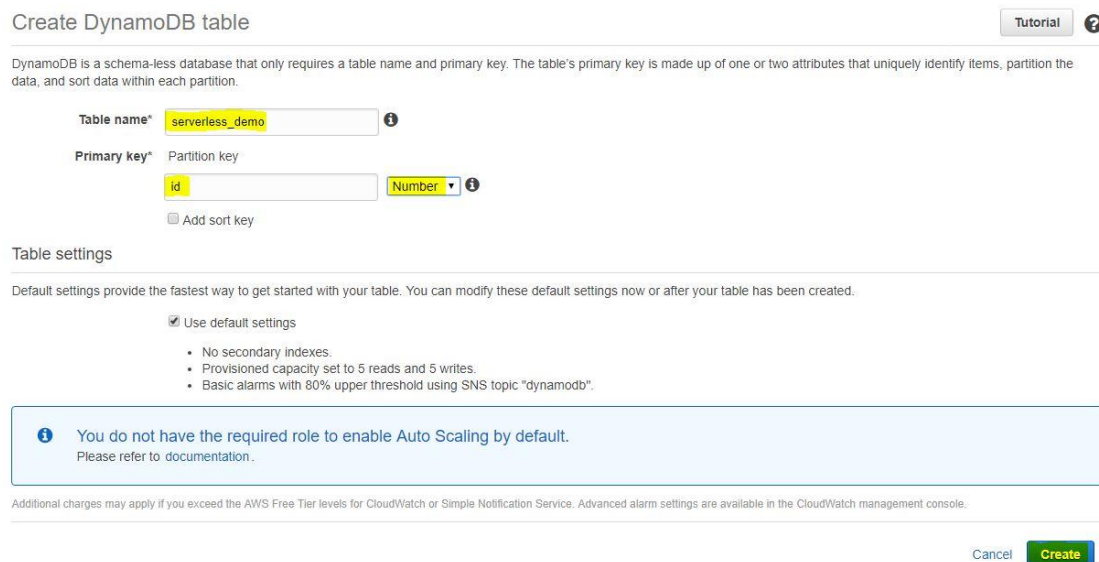


2.3. In the “Create DynamoDB table” section, do the following as below:

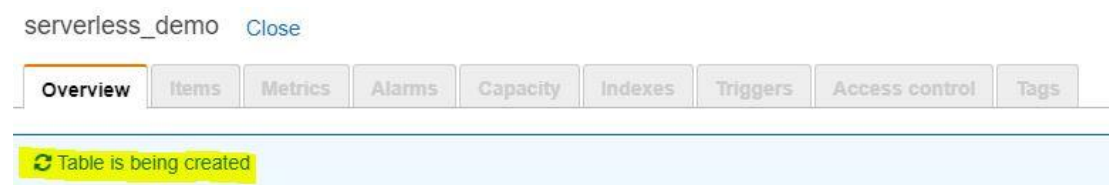
2.4. For ‘Table name’, type <YOUR_TABLE_NAME>

2.5. For ‘Primary key’, in the ‘Primary key’, type “id”. Set data type as ‘Number’.

2.6. Click ‘Create’.



2.7. Waiting for the table create successful.



Create Items in DynamoDB Table

2.8. In the DynamoDB console, choose the “Items” tab.



2.9. Select ‘Create item’.

serverless_demo [Close](#)

Overview **Items** Metrics Alarms Capacity

Create item Actions ▾

Scan: [Table] serverless_demo: id ^

Scan ▾ [Table] serverless_demo: id

+ Add filter

Start search Cancel changes

<input type="checkbox"/>	id
--------------------------	----

2.10. In the prompt console, type data type & data value which you want.

2.11. Then click 'Save'.

Create item

Tree ▾

Item {7}

- + id Number : 0
- + season String : spring
- + weather String : cloudy
- + weekday String : tuesday
- + workingday Boolean : true
- + month Number : 2
- + count Number : 2591

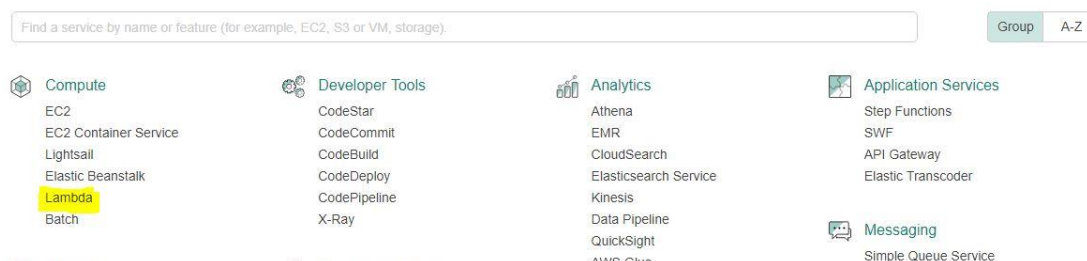
2.12. You can review the item which just type-in via console

<input type="checkbox"/>	id	count	month	season	weather	weekday	workingday
<input type="checkbox"/>	0	2591	2	spring	cloudy	tuesday	true

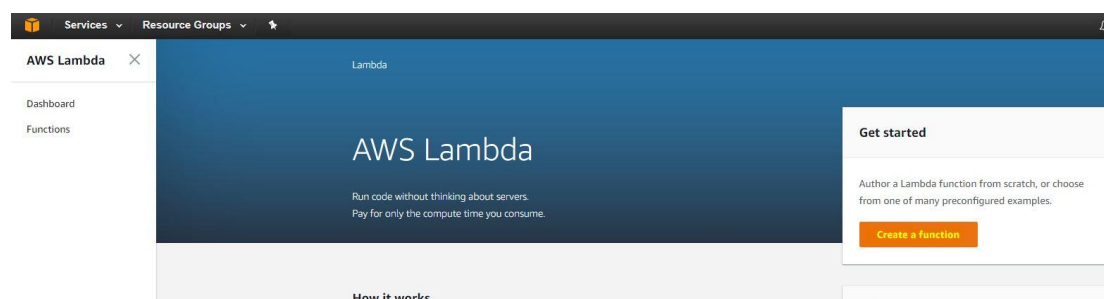
Working with Lambda Function

Create Lambda Function

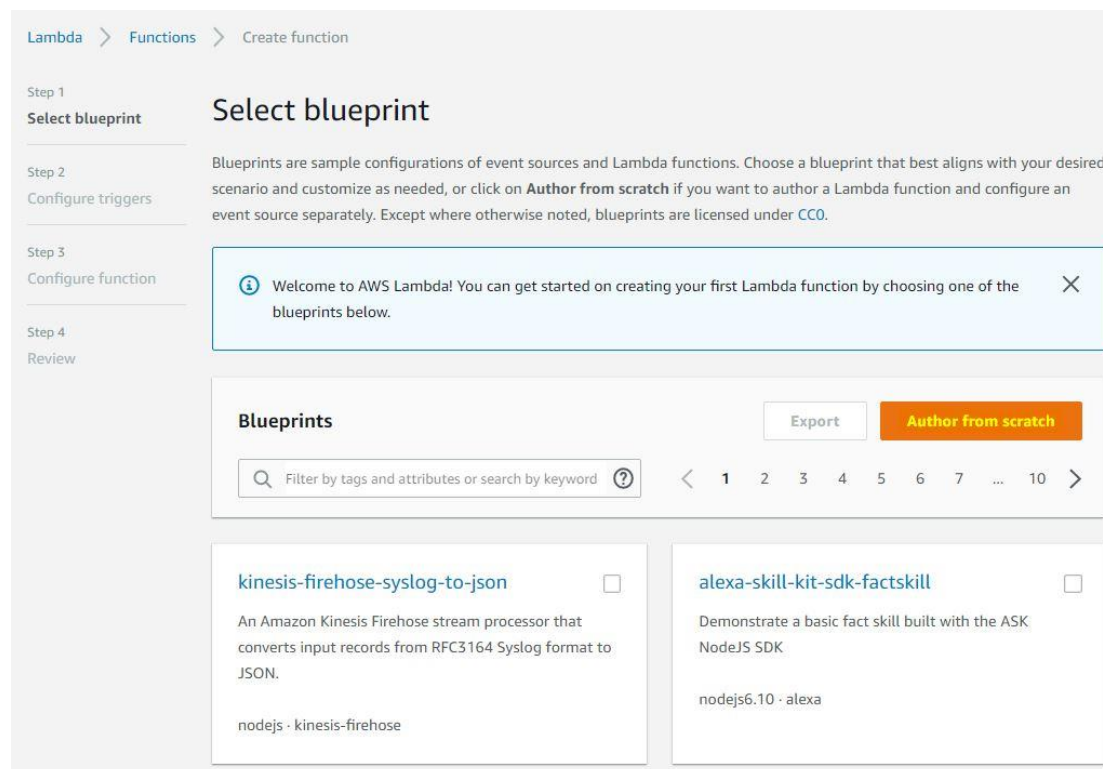
3.1. In the service menu, select 'Lambda'



3.2. Select 'Create a function' on the right panel.



3.3. Select 'Author from scratch' for a blank function



3.4. Select 'Next', without setting trigger.



Configure triggers

You can choose to add a trigger that will invoke your function.

Welcome to AWS Lambda! You can get started on creating your first Lambda function by choosing one of the blueprints below. ×

Add trigger

Remove

  Lambda

Cancel

Previous

Next

3.5. For the 'Basic Information' section, type the function name and select 'Python 3.6' for Runtime.

Configure function

A Lambda function consists of the custom code you want to execute. [Learn more](#) about Lambda functions.

Basic information

Name*

Description

Runtime*

Python 3.6 ▼

3.6. In the “Lambda function code” section, copy the code from download materials which named ‘lambda_function’

3.7. Modify the “TableName” to **<YOUR_TABLE_NAME>** in DynamoDB in the code.


Lambda function code

Provide the code for your function. Use the editor if your code does not require custom libraries (other than boto3). If you need custom libraries, you can upload your code and libraries as a .ZIP file.

Code entry type

Edit code inline ▼

```
1 def lambda_handler(event, context):
2     # TODO: implement
3     return 'Hello from Lambda'
```



Lambda function code

Provide the code for your function. Use the editor if your code does not require custom libraries (other than boto3). If you need custom libraries, you can upload your code and libraries as a .ZIP file.

Code entry type

Edit code inline ▼

```
1 import boto3
2
3 def lambda_handler(event, context):
4
5     print ("-----Lambda Start-----")
6
7     client_dynamodb = boto3.client('dynamodb')
8
9     response_dynamodb = client_dynamodb.scan(
10         TableName = 'serverless_demo'
11     )['Items']
12
13     print (response_dynamodb)
```

3.8. In the “Lambda function handler and role” section

3.9. Select “Choose an existing role” in the Role field and “lambda-dynamodb” in the Existing role field.

Lambda function handler and role

Handler*

The filename.handler-method value in your function. For example, "main.handler" would call the handler method defined in main.py.

lambda_function.lambda_handler

Role*

Defines the permissions of your function. Note that new roles may not be available for a few minutes after creation. [Learn more about Lambda execution roles.](#)

Choose an existing role ▼

Existing role*

You may use an existing role with this function. Note that the role must be assumable by Lambda and must have Cloudwatch Logs permissions.

lambda-dynamodb ▼

3.10. After complete the configure in this section, click 'Next' at the bottom.



3.11. Take a review and click 'Create function' to create lambda function.



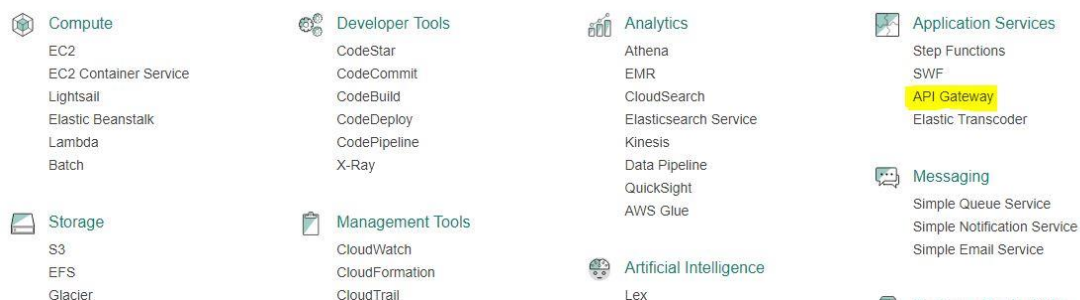
3.12. You will see the message as below which means create lambda function successful.



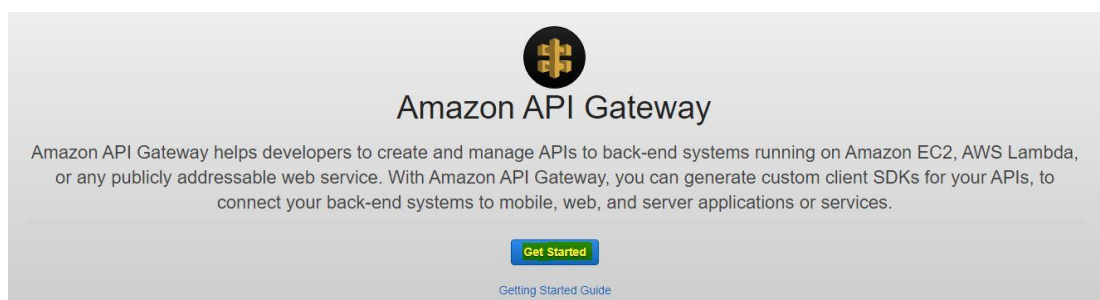
Working with API Gateway

Create APIs in API Gateway

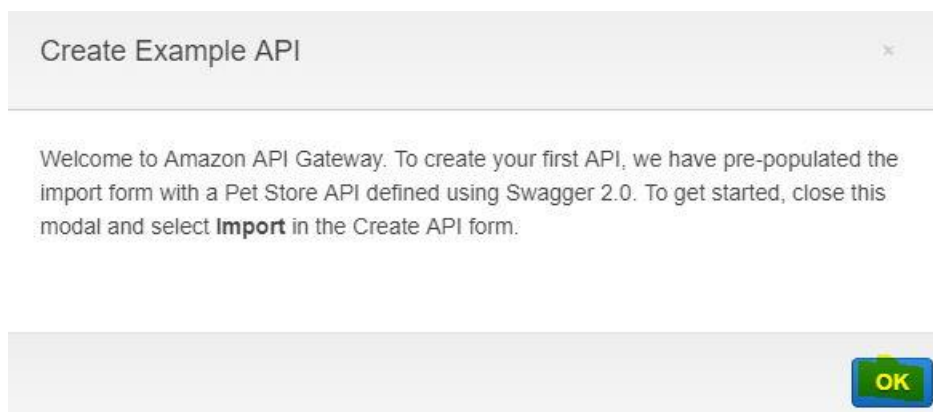
4.1. On the service menu, click 'API Gateway'.



4.2. In the welcome page, click 'Get Started'.



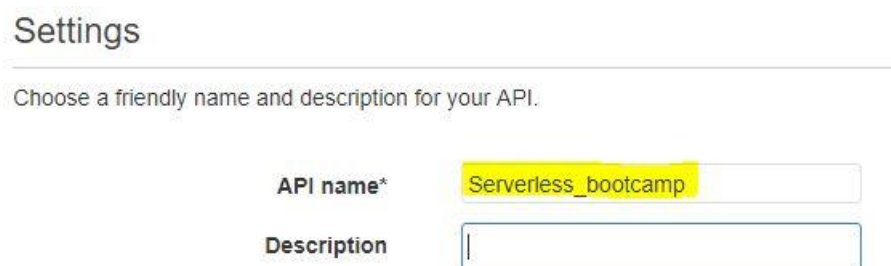
4.3. In the prompt, click 'OK' to create a new API



4.4. Select 'New API' in the 'Create new API'

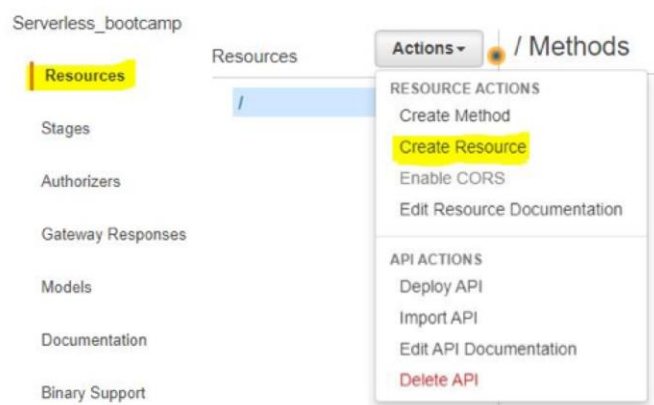


4.5. Type-in API name in 'API name', then click 'Create API'



4.6. The console will show the information as below:


4.7. In the "Resources" tab, choose the root "/", click "Actions" and select "Create Resource".




4.8. Type a name in the 'Resource Name', for example: 'get-item'.

4.9. Make sure 'Enable API Gateway CORS' is enabled.

New Child Resource


Use this page to create a new child resource for your resource. 

Configure as [proxy resource](#) 

Resource Name*

Resource Path*

You can add path parameters using brackets. For example, the resource path **{username}** represents a path parameter called 'username'. Configuring **/(proxy+)** as a proxy resource catches all requests to its sub-resources. For example, it works for a GET request to /foo. To handle requests to /, add a new ANY method on the / resource.


Enable API Gateway CORS ☒ 

* Required

[Cancel](#) [Create Resource](#)

4.10. After resource being created, click "Actions" and select "Create Method" to add method

Resources

Actions  /get-items Methods

RESOURCE ACTIONS

- Create Method
- Create Resource
- Enable CORS
- Edit Resource Documentation
- Delete Resource

API ACTIONS


- Deploy API
- Import API
- Edit API Documentation
- Delete API

None

Not required



4.11. In the drop-down list, select 'GET' method and click 'yes'.

Resources

Actions 

/get-items

OPTIONS

GET  

4.12. In the setup page as below:

- Select "Lambda Function" for "Integration type"
- Choose "us-west-2" in the "Lambda Region"
- Type-in the Lambda function created in previous chapter in the "Lambda Function"
- Click "Save"

/get-items - GET - Setup

Choose the integration point for your new method.

Integration type ☒ Lambda Function ⓘ
☐ HTTP ⓘ
☐ Mock ⓘ
☐ AWS Service ⓘ

Use Lambda Proxy integration ☐ ⓘ

Lambda Region us-west-2 ▼

Lambda Function serverless-bootcamp ⓘ

4.13. Click 'OK' to give API Gateway permission to invoke Lambda function

Add Permission to Lambda Function ×

You are about to give API Gateway permission to invoke your Lambda function:
arn:aws:lambda:us-west-2:471856162574:function:serverless-bootcamp

Cancel **OK**

4.14. Click 'Actions' within Resource layer, and select 'Enable CORS'

Resources

Actions ▼ /get-items Methods

- RESOURCE ACTIONS
 - Create Method
 - Create Resource
 - Enable CORS**
 - Edit Resource Documentation
 - Delete Resource
- API ACTIONS
 - Deploy API
 - Import API
 - Edit API Documentation
 - Delete API

GET
OPTIONS

None
Not required

4.15. Select “Enable CORS and replace existing CORS headers”

Enable CORS

Gateway Responses for *Serverless_bootcamp API* ☐ DEFAULT 4XX ☐ DEFAULT 5XX ⓘ

Methods ☒ GET ☒ OPTIONS ⓘ

Access-Control-Allow-Methods GET, OPTIONS ⓘ

Access-Control-Allow-Headers 'Content-Type,X-Amz-Date,Authorization' ⓘ

Access-Control-Allow-Origin* '*' ⓘ

▶ Advanced

Enable CORS and replace existing CORS headers

4.16. Review and click “Yes, replace existing values”

Confirm method changes

The following modifications will be made to this resource's methods and will replace any existing values. Are you sure you want to continue?

- Add **Access-Control-Allow-Headers, Access-Control-Allow-Methods, Access-Control-Allow-Origin Method Response Headers** to **OPTIONS** method
- Add **Access-Control-Allow-Headers, Access-Control-Allow-Methods, Access-Control-Allow-Origin Integration Response Header Mappings** to **OPTIONS** method
- Add **Access-Control-Allow-Origin Method Response Header** to **GET** method
- Add **Access-Control-Allow-Origin Integration Response Header Mapping** to **GET** method

Cancel **Yes, replace existing values**

4.17. Waiting the steps all be checked

Enable CORS

- ✓ Add **Access-Control-Allow-Headers, Access-Control-Allow-Methods, Access-Control-Allow-Origin Method Response Headers** to **OPTIONS** method
- ✓ Add **Access-Control-Allow-Headers, Access-Control-Allow-Methods, Access-Control-Allow-Origin Integration Response Header Mappings** to **OPTIONS** method
- ✓ Add **Access-Control-Allow-Origin Method Response Header** to **GET** method
- ✓ Add **Access-Control-Allow-Origin Integration Response Header Mapping** to **GET** method

Your resource has been configured for CORS. If you see any errors in the resulting output above please check the error message and if necessary attempt to execute the failed step manually via the Method Editor.

Deploy APIs

4.18. Click ‘Actions’ and select ‘Deploy API’

Resources

Actions **Enable CORS**

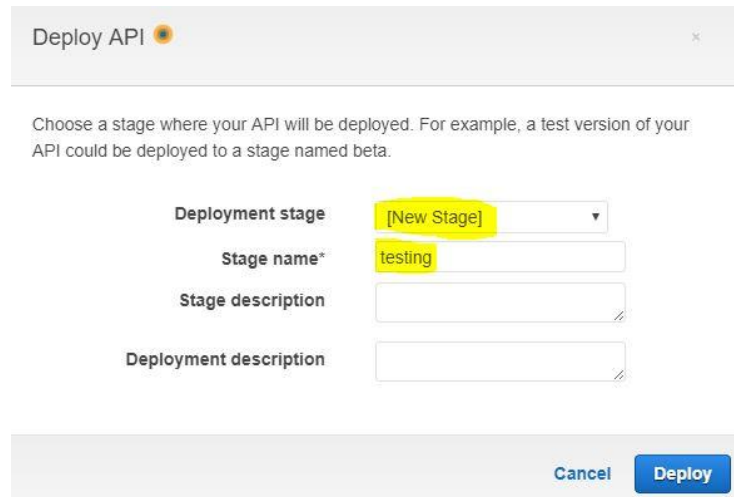
RESOURCE ACTIONS

- Create Method
- Create Resource
- Enable CORS
- Edit Resource Documentation
- Delete Resource

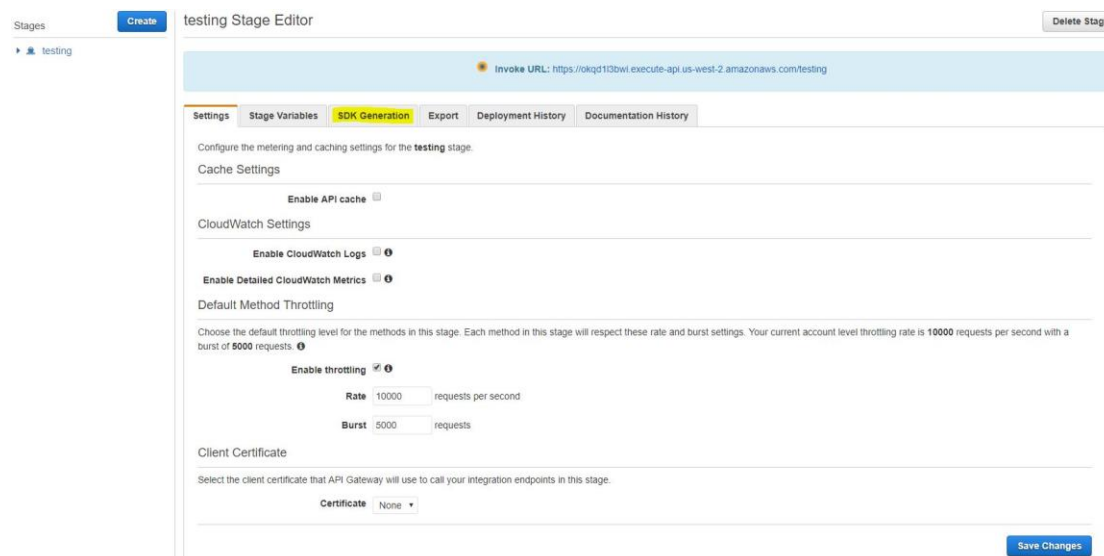
API ACTIONS

- Deploy API**
- Import API
- Edit API Documentation
- Delete API

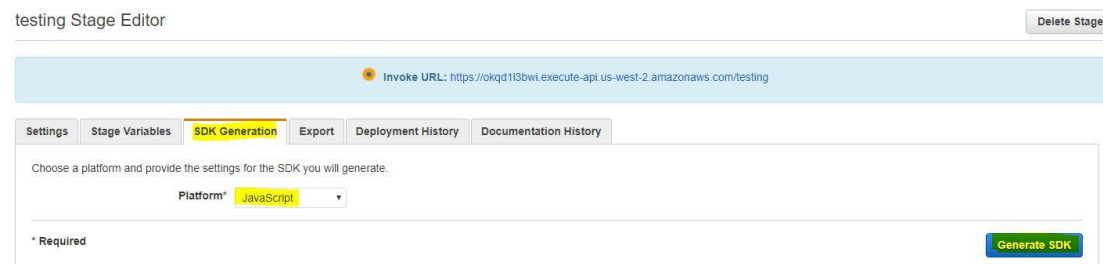
- 4.19. In the prompt console,
- Select '[New Stage]' in Deployment stage
 - Give the name for "Stage name"
 - Click 'Deploy'



- 4.20. The console would jump out as below, select the 'SDK Generation' tab



- 4.21. In the 'SDK Generation' tab, select 'JavaScript' in the 'Platform' field and click 'Generate SDK'

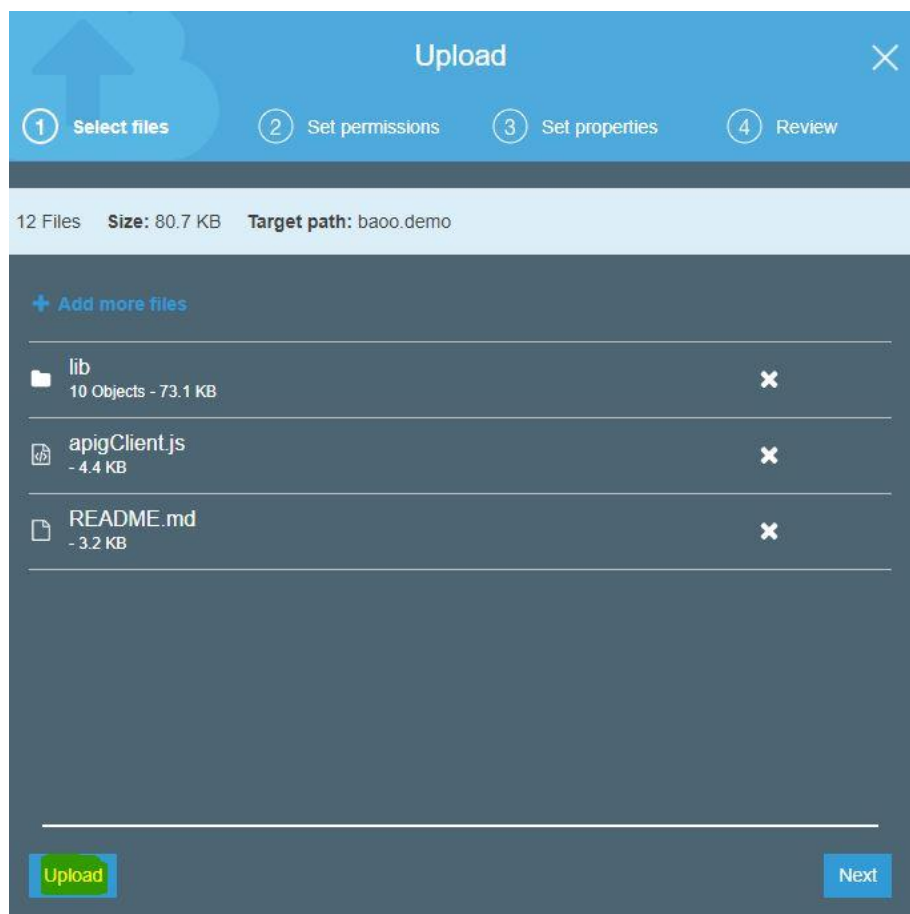


4.22. The browser would ask you to confirm download a zip file named as
'javascript_<TIMESTAMP>.zip'

4.23. Save and unzip the zip file, after entering the folder, there would be files
show as below:



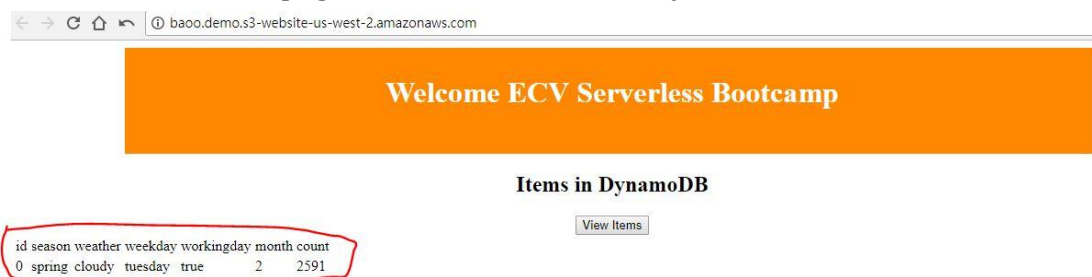
4.24. Upload those files to the S3 Bucket created in Chapter S3, must to be the
same layer as the html.



4.25. Modify the javascript in the html file to your APIs & data in DynamoDB

```
apigClient.getItemsGet(params, body, additionalParams)
.then(function(result){
    var txt = '';
    myObj = JSON.parse(this.responseText);
    txt += "<table border='1'"
    txt += "<tr>";
    txt += "<td>" + "id" + "</td>";
    txt += "<td>" + "season" + "</td>";
    txt += "<td>" + "weather" + "</td>";
    txt += "<td>" + "weekday" + "</td>";
    txt += "<td>" + "workingday" + "</td>";
    txt += "<td>" + "month" + "</td>";
    txt += "<td>" + "count" + "</td>";
    txt += "</tr>";
    for (x in result.data) {
        console.log(result.data[x]);
        txt += "<tr>";
        txt += "<td>" + result.data[x].id.N + "</td>";
        txt += "<td>" + result.data[x].season.S + "</td>";
        txt += "<td>" + result.data[x].weather.S + "</td>";
        txt += "<td>" + result.data[x].weekday.S + "</td>";
        txt += "<td>" + result.data[x].workingday.BOOL + "</td>";
        txt += "<td>" + result.data[x].month.N + "</td>";
        txt += "<td>" + result.data[x].count.N + "</td>";
        txt += "</tr>";
    }
    txt += "</table>"
    document.getElementById("items_list").innerHTML = txt;
    return result;
}).catch(function(result){
    console.log(result);
});
```

4.26. Reload the web page and click button to test your API



Conclusion

Congratulations! You now have learned how to:

- Create a static website hosting through S3
- Create a Lambda function
- Create DynamoDB Table
- Create API though API Gateway