

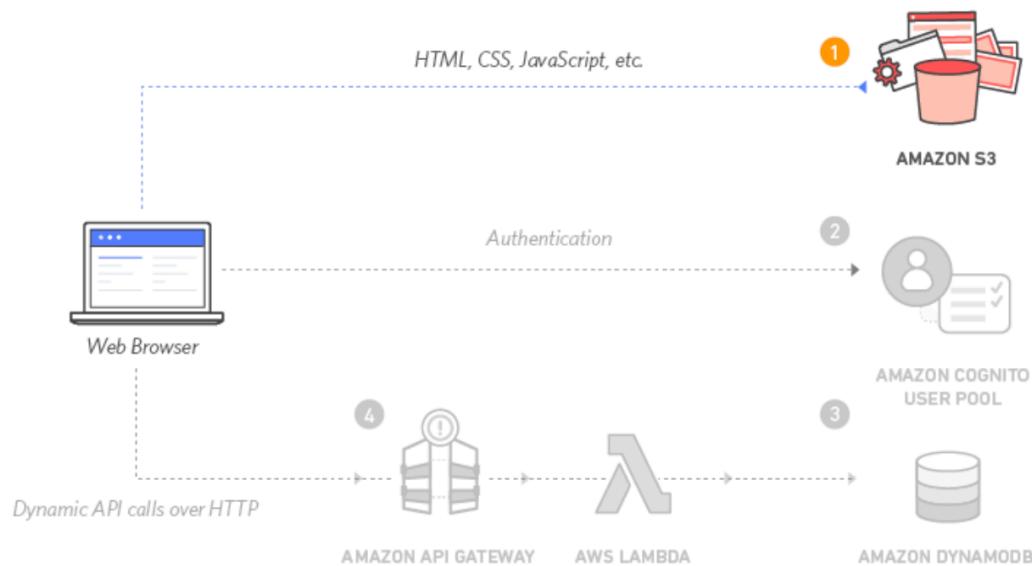
Build a Serverless Web Application

Time Spent: 120 min

Overview

In this module you'll configure Amazon Simple Storage Service (S3) to host the static resources for your web application. In subsequent modules you'll add dynamic functionality to these pages using JavaScript to call remote RESTful APIs built with AWS Lambda and Amazon API Gateway.

Architecture Overview



Related components:

- AWS Cognito
- AWS API Gateway
- AWS S3
- AWS DynamoDB
- AWS Lambda

What I learned:

- How to use AWS S3 bucket to host the static frontend webpage
- Use AWS Cognito to create user pool and ID, to implement the sign-in authentication function

- Implement register and verification function by Cognito
- How to build and deploy the new Gateway API
- How to build the authorization by Cognito token

What I built:

- Use AWS S3 bucket to host the static frontend webpage



- New Config file

 A screenshot of a code editor window titled 'config.js'. The code is written in JavaScript and defines a configuration object for AWS Cognito. The code is as follows:


```

1  window._config = {
2    cognito: {
3      userPoolId: 'us-east-1_WTEqSrko4', // e.g. us-east-2_uXboG5pAb
4      userPoolClientId: '56hs1jufjb5ublsdut1f5f2m1h', // e.g. 25ddkmj4v6hfsfvruhpf17n4hv
5      region: 'us-east-1' // e.g. us-east-2
6    },
  
```

- Authentication success

Your verification code Inbox x

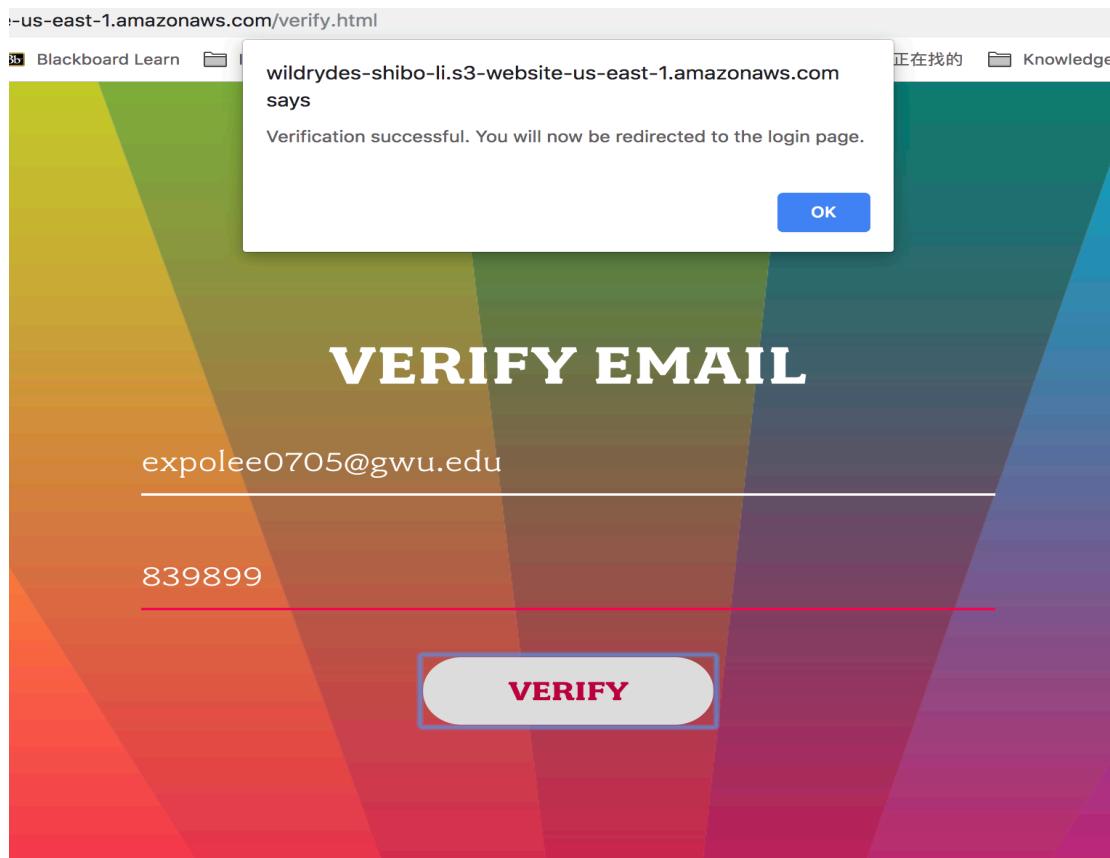
 no-reply@verificationemail.com via amazoneses.com
to expolee0705 ▾

Your confirmation code is 839899

 Reply

 Reply all

 Forward



- Lambda function test successfully

The screenshot shows the AWS Lambda function editor for the 'RequestUnicorn' function. At the top, there are tabs for 'Throttle', 'Qualifiers', 'Actions', 'TestRequestEvent', 'Test', and 'Save'. Below the tabs, there are sections for 'Function code' (with a 'Info' link), 'Code entry type' (set to 'Edit code inline'), 'Runtime' (set to 'Node.js 6.10'), and 'Handler' (set to 'index.handler'). The main area contains the 'index.js' file code:

```

65     headers: {
66       'Access-Control-Allow-Origin': '*',
67     },
68   });
69 }).catch((err) => {
70   console.error(err);
71
72   // If there is an error during processing, catch it and return
73   // from the function successfully. Specify a 500 HTTP status
74   // code and provide an error message in the body. This will provide a
75   // more meaningful error response to the end client.
76   errorResponse(err.message, context.awsRequestId, callback)
77 });
78 };
79 }

```

Below the code editor is an 'Execution Result' panel. It shows the 'Execution results' tab, which displays the response object:

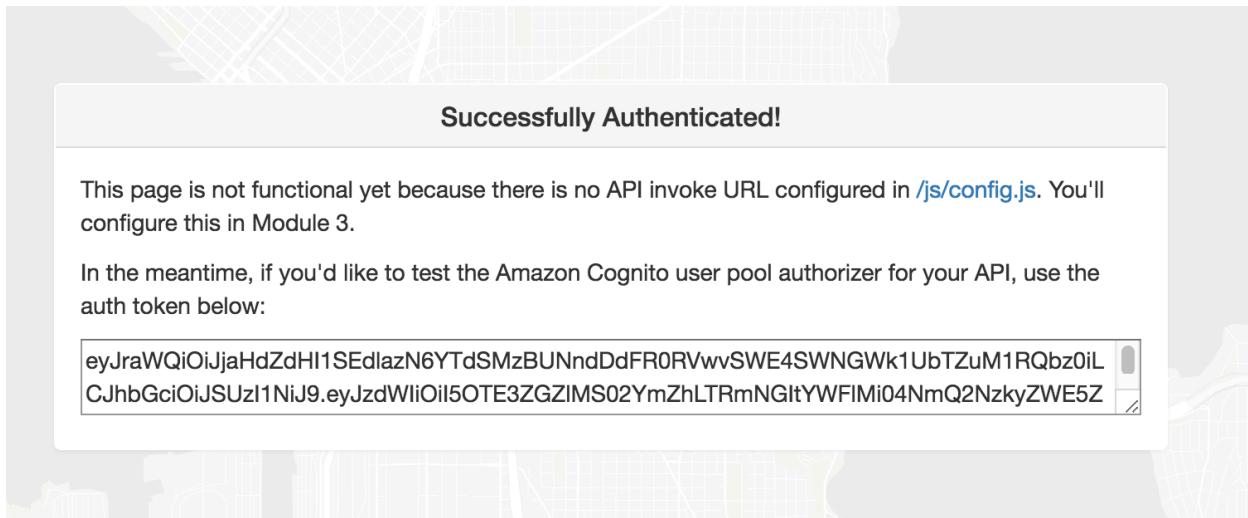
```

{
  "statusCode": 201,
  "body": "{\"RideId\":\"xWzUoi1GdsLbhr9qdA9A\", \"Unicorn\":{\"Name\":\"Bucephalus\", \"Color\": \"Golden\", \"Gender\": \"Male\"}, \"UnicornName\":\"Bucephalus\", \"Eta\": \"2025-01-01T12:00:00Z\"}",
  "headers": {
    "Access-Control-Allow-Origin": "*"
  }
}

```

The execution result summary at the bottom indicates 'Status: Succeeded', 'Max Memory Used: 38 MB', and 'Time: 1103.60 ms'.

- Authorization by token



WildRydes - Test Authorizer

You can test your authorizer by providing values that will be used to invoke your Lambda function or make a call to your Cognito User Pool.

Authorization Token ⓘ

Authorization (header) eyJraWQiOiJjaHdHI1SEdlazN6YTdSMzBUNndDdFR0RVvvSWE-

Test

Response

Response Code: 200

Latency 79

Claims

```
{  
  "aud": "56hsljufjb5ublsdut1f5f2m1h",  
  "auth_time": "1544410612",  
  "cognito:username": "expolee0705@gwu.edu",  
  "email": "expolee0705@gwu.edu",  
  "email_verified": "true",  
  "event_id": "3f077f07-fc27-11e8-b4d7-f330073449a8",  
  "exp": "Mon Dec 10 03:56:52 UTC 2018",  
  "iat": "Mon Dec 10 02:56:52 UTC 2018",  
  "iss": "https://cognito-idp.us-east-1.amazonaws.com/us-east-1_WTEqSrko4",  
  "sub": "9917dfe1-6bfa-4f4b-aae2-86d6792ea9db",  
  "token_use": "id"  
}
```

Close

- Finish deploy the API

Amazon API Gateway APIs > WildRydes (lxkurlbmif) > Stages > prod

prod Stage Editor

Invoke URL: <https://lxkurlbmif.execute-api.us-east-1.amazonaws.com/prod>

Cache Settings

Enable API cache

Default Method Throttling

Choose the default throttling level for the methods in this stage. Each method in this stage will respect these rate and burst settings. Your current account level throttling rate is **10000** requests per second with a burst of **5000** requests. [Read more about API Gateway throttling](#)

Enable throttling [?](#)

Rate requests per second

Burst requests

Web Application Firewall (WAF) [Learn more.](#)

Select the Web ACL to be applied to this stage.

Web ACL [None](#) [Create Web ACL](#)

Client Certificate

Select the client certificate that API Gateway will use to call your integration endpoints in this stage.

Certificate [None](#) [+](#)

- Implement the whole project

