

AWS Serverless Websites

AngularJS - API Gateway - Lambda - DynamoDB

Create a new S3 Bucket, enable static hosting

Static documents

Route 53 for DNS

Amazon CloudFront for CDN

Bucket: firumberrally
Region: Oregon
Creation Date: Wed Jan 13 00:01:11 GMT-600 2016
Owner: davidknaack

► Permissions

▼ Static Website Hosting

You can [host your static website](#) entirely on Amazon S3. Once you enable your bucket for static website hosting, your content is accessible to web browsers via the Amazon S3 website endpoint for your bucket.

Endpoint: `firumberrally.s3-website-us-west-2.amazonaws.com`

Each bucket serves a website namespace (e.g. "www.example.com"). Requests for your host name (e.g. "example.com" or "www.example.com") can be routed to the contents in your bucket. You can also redirect another host name (e.g. redirect "example.com" to "www.example.com"). See our [walkthrough](#) for how to : Amazon S3 static website with your host name.

☐ Do not enable website hosting

☒ Enable website hosting

Index Document:

Error Document:

► **Edit Redirection Rules:** You can set custom rules to automatically redirect web page requests for specific content.

Allow Public Access to Bucket

Bucket Properties, Permissions:

Add a bucket policy:

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Sid": "Allow Public Access to All Objects",
    "Effect": "Allow",
    "Principal": "*",
    "Action": "s3:GetObject",
    "Resource": "arn:aws:s3:::firlumberrally/*"
  }]
}
```

Bucket: firlumberrally

Bucket: firlumberrally
Region: Oregon
Creation Date: Wed Jan 13 00:01:11 GMT-600 2016
Owner: davidknaack

▼ Permissions

You can control access to the bucket and its contents using access policies. [Learn more](#)

Grantee:

☒ List

☒ Upload/Delete

☒ View Per



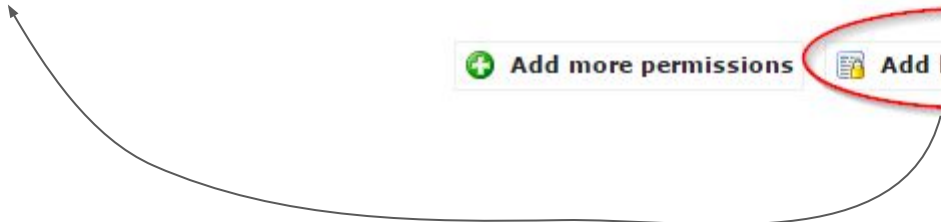
Add more permissions



Add bucket policy



Add CORS Config



Deploy Static Content

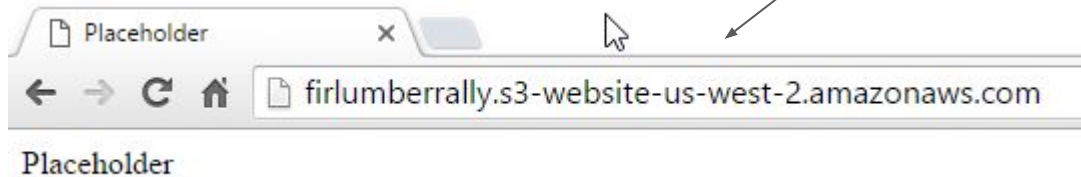
Hosting is now enabled, upload some content.

This can be done via the AWS console or AWS cli:

```
<html>
<head><title>Placeholder</title>
<body>Placeholder</body>
</html>
```

All Buckets / firlumberrally

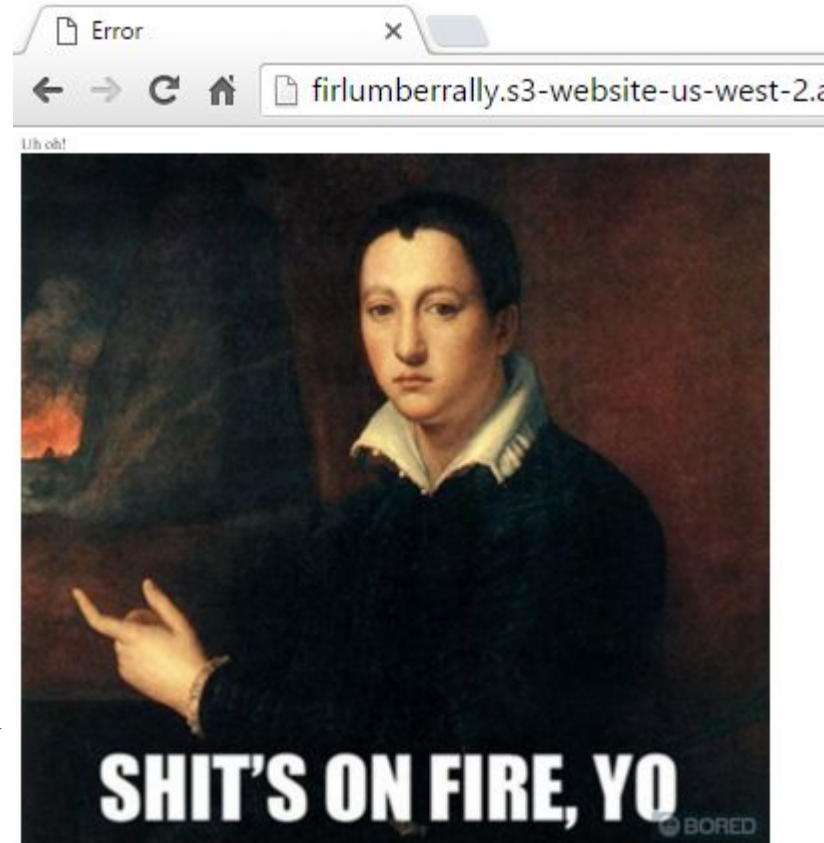
Name	
<input type="checkbox"/>	fir.html



And an error page

Just in case anything goes wrong,
add a useful error message and
upload the static files:

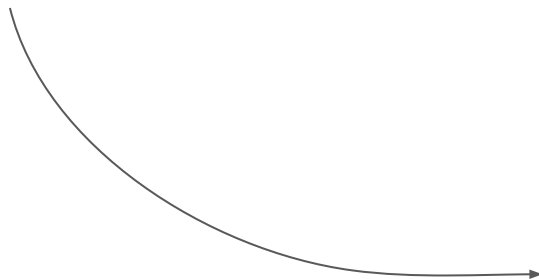
```
<html>
<head><title>Error</title>
<body>Uh oh!<br></body>
</html>
```



Create AWS Lambda RESTful API endpoint

Lambda blueprints make setting up common configurations easy.

Create a new Lambda function using the `microservice-http-endpoint` blueprint.



microservice-http-endpoint

A simple backend (read/write to DynamoDB) with a RESTful API endpoint using Amazon API Gateway.

nodejs · api-gateway

Configure the function

No coding required.

DynamoDB-backed CRUD operations are part of the blueprint function.

Function has cases for multiple operations, but each function only uses one.

Configure function

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

Name*	<input type="text" value="cars-list"/>
Description	<input type="text" value="Return list of cars in the database"/>
Runtime*	<input type="text" value="Node.js"/>

Lambda function code

Provide the code for your function. Use the editor if your code does not require libraries, you can upload your code and libraries as a .ZIP file. [Learn more](#) about

Code entry type ☒ Edit code inline ☐ Upload a .ZIP file

```
1 console.log('Loading function');
2
3 var doc = require('dynamodb-doc');
4 var dynamo = new doc.DynamoDB();
5
6 /**
7  * Provide an event that contains the following keys:
8  *
```

Configure Endpoint

Set or choose the API name and the security settings, if desired.

Add an access key in the API Gateway console.

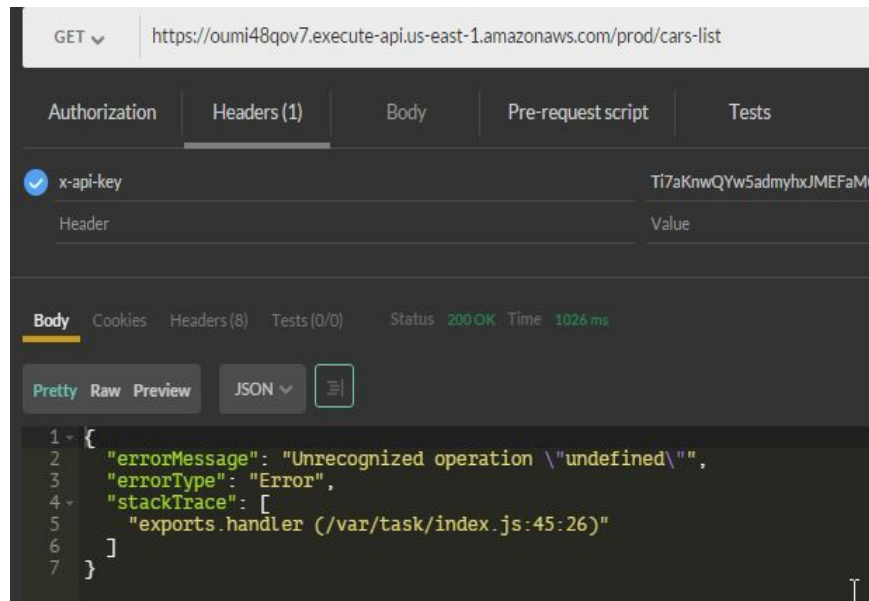
Configure endpoints

API endpoint type	API Gateway ▼	i
API name	FirLumberRally	i
Resource name	/cars-list	i
Method	GET ▼	i
Deployment stage	prod	i
Security	Open with access key ▼	i

API endpoint is now live and can be called with the API key in an x-api-key header, but the result is an error “Unrecognized operation “undefined””.

The Lambda function expects an ‘event’ parameter containing fields describing what to do with the DynamoDB.

```
6 ▾ /**
7   * Provide an event that contains the following keys:
8   *
9   *   - operation: one of the operations in the switch statement below
10  *   - tableName: required for operations that interact with DynamoDB
11  *   - payload: a parameter to pass to the operation being performed
12  */
13 ▾ exports.handler = function(event, context) {
14   //console.log('Received event:', JSON.stringify(event, null, 2));
15
16   var operation = event.operation;
17
```



Adding Mapping

In API Gateway console, select Resources, /cars-list, GET, Integration Request and add a Mapping Template.

No parameters required, so no mapping is performed, static template:



Now the Lambda function works, but the error indicates that the requested table does not exist.

```
{  "errorMessage": "Requested resource not found",  "errorType": "ResourceNotFoundException",  "stackTrace": [
```

In the DynamoDB console, create a 'car' table

Create DynamoDB table

DynamoDB is a schema-less database that only requires a table name and primary key attributes that uniquely identify items, partition the data, and sort data within each partition.

Table name* ⓘ

Primary key* Partition key

ⓘ

☐ Add sort key

Retest the API endpoint, and now we're getting JSON data back

```
{
  "Items": [],
  "Count": 0,
  "ScannedCount": 0
}
```

Do it all again to add a car-post endpoint.

Create the Lambda function, POST this time.



No template required, POST body directly supplies Lambda parameters.

POST <https://oumi48qov7.execute-api.us-east-1.amazonaws.com/prod/car-post>

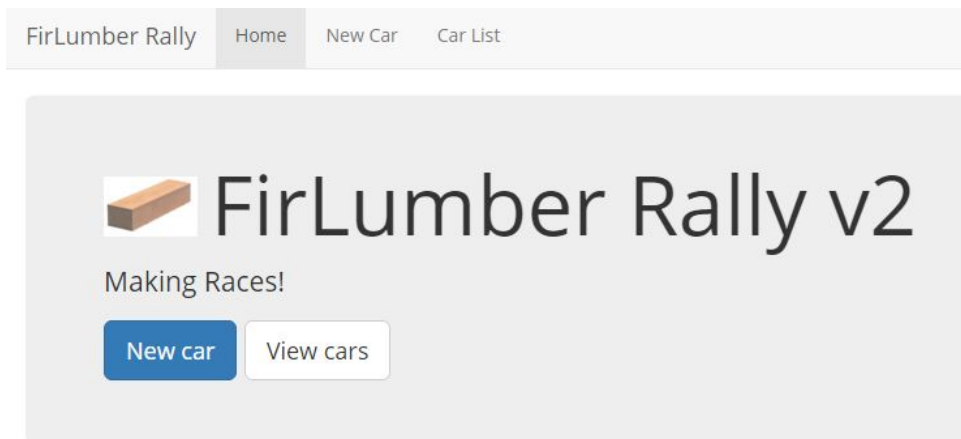
```
{
  "operation": "create",
  "tableName": "car",
  "payload": {
    "Item": {
      "Number": "1"
    }
  }
}
```

GET <https://oumi48qov7.execute-api.us-east-1.amazonaws.com/prod/cars-list>

```
{
  "Items": [
    {
      "Number": "1"
    }
  ],
  "Count": 1,
  "ScannedCount": 1
}
```

Drop a quick AngularJS website into S3

<input type="checkbox"/>	 editcar.html	Standard	729 bytes
<input type="checkbox"/>	 err.html	Standard	109 bytes
<input type="checkbox"/>	 firblock.jpg	Standard	6 KB
<input type="checkbox"/>	 flr.html	Standard	2.4 KB
<input type="checkbox"/>	 listcars.html	Standard	367 bytes
<input type="checkbox"/>	 main.css	Standard	1.8 KB
<input type="checkbox"/>	 main.js	Standard	2.2 KB
<input type="checkbox"/>	 root.html	Standard	335 bytes
<input type="checkbox"/>	 shifts-on-fire-yo__big_no.jpeg	Standard	76.5 KB



Controller calls the API gateway,
but with the defaults these are
different domains (use Route 53).

Use the API Gateway
console to add the
CORS header:

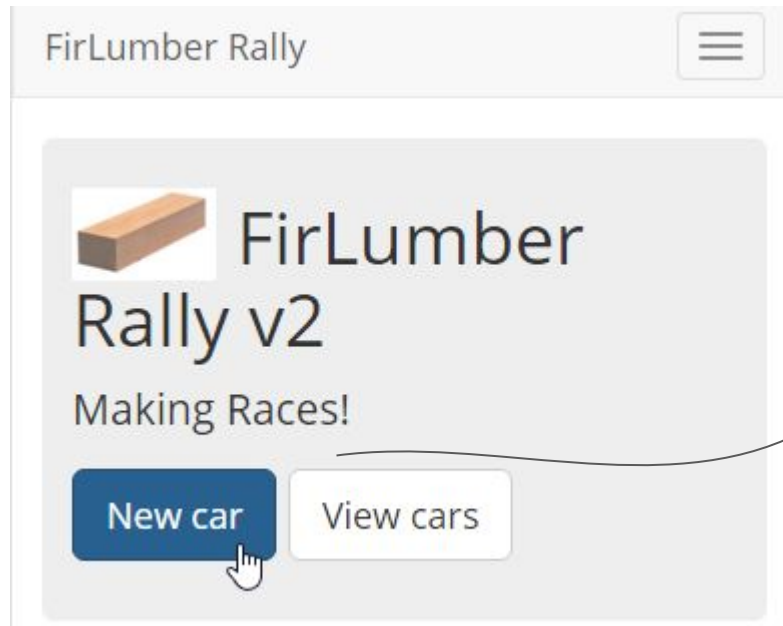


```
var ENDPOINT = 'https://oumi48gov7.execute-api.us-east-1.amazonaws.com/prod';

.controller('CarListController', ['$scope', '$location', '$http', function($scope, $location, $http) {
    $scope.carREST('get', ENDPOINT+'/cars-list')
    .then(function(response) {
        $scope.cars = response.data.Items;
    });
}]);
```


Enable CORS

Try it out:



This screenshot shows the 'New Car' form, which is accessed by clicking the 'New car' button. The form has a header bar with 'FirLumber Rally' and a hamburger menu icon. Below the header, the title 'New Car' is displayed. The form contains two input fields: 'Car Number' with the value '3' and 'Owner' with the value 'Dave'. A blue 'Submit' button is located below the 'Owner' field. A hand cursor is positioned over the 'Submit' button.

How about that, it worked!

FirLumber Rally	
	
All Cars	
Number	Owner
1	Sam
2	Bob
3	Dave



<http://firlumberrally.s3-website-us-west-2.amazonaws.com/#/>



<https://github.com/davidknaack/FirLumberRallyII>