

AWS Lambda

Introduction

Let's build a simple serverless application using AWS Lambda.

This application will have an html front end hosted on AWS Amplify, where you can enter some text. On submitting the form, it will provide you with a response which is capitalized and reverse of your entered text.

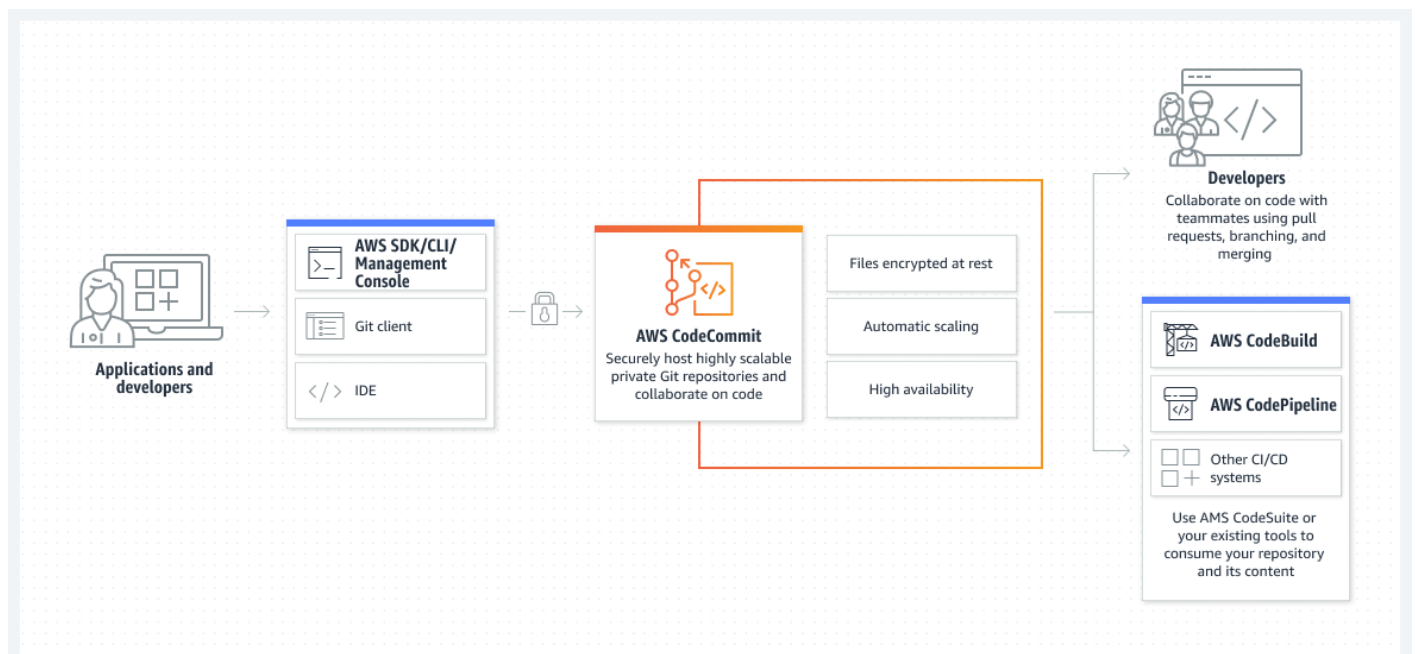
Capitalize and Reverse will be two separate Lambda functions to show you the chaining capabilities. Instead of accessing these functions directly, an API Gateway will be used to accept client requests and respond with the final output.

The components we use are:

AWS CodeCommit

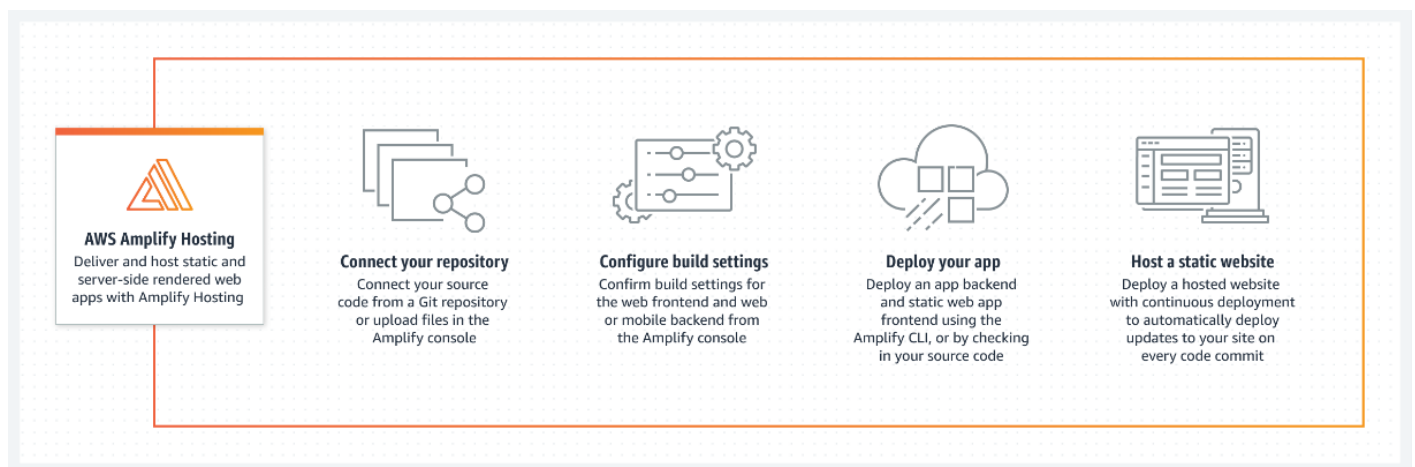
AWS CodeCommit is a secure, highly scalable, fully managed source control service that hosts private Git repositories.

As a Git-based service, CodeCommit is well suited to most version control needs. There are no arbitrary limits on file size, file type, and repository size.



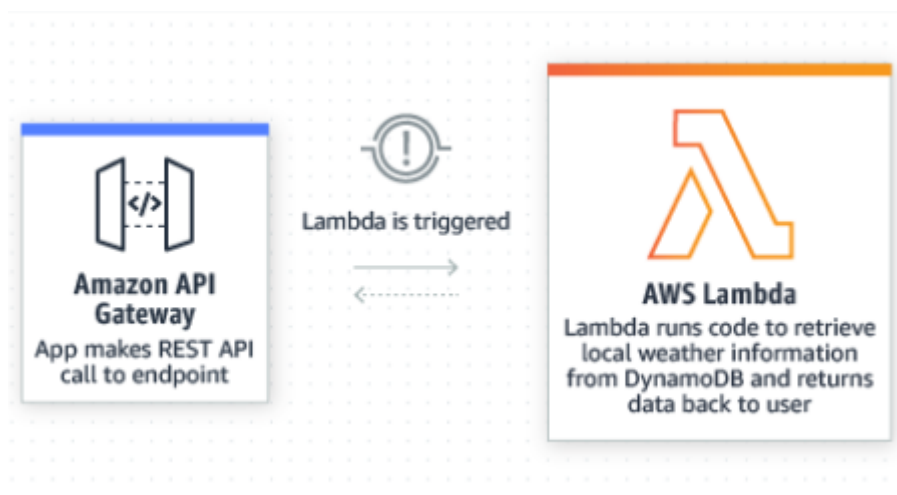
AWS Amplify

AWS Amplify is a complete solution that lets front end web and mobile developers easily build, ship, and host full-stack applications on AWS, with the flexibility to leverage the breadth of AWS services as use cases evolve.



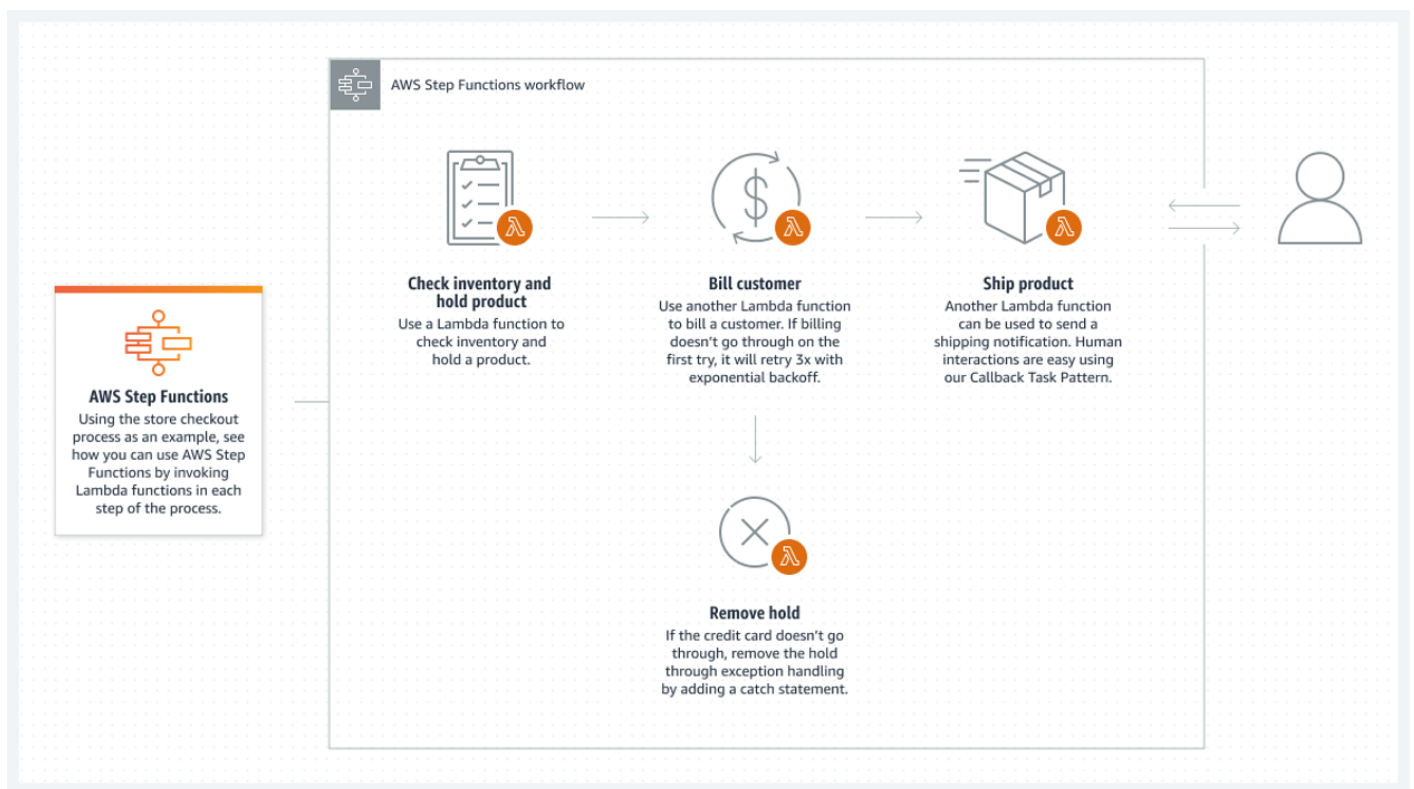
AWS Lambda Functions

AWS Lambda is a serverless, event-driven compute service that lets you run code for virtually any type of application or back end service without provisioning or managing servers. You can trigger Lambda from over 200 AWS services and software as a service (SaaS) application, and only pay for what you use.



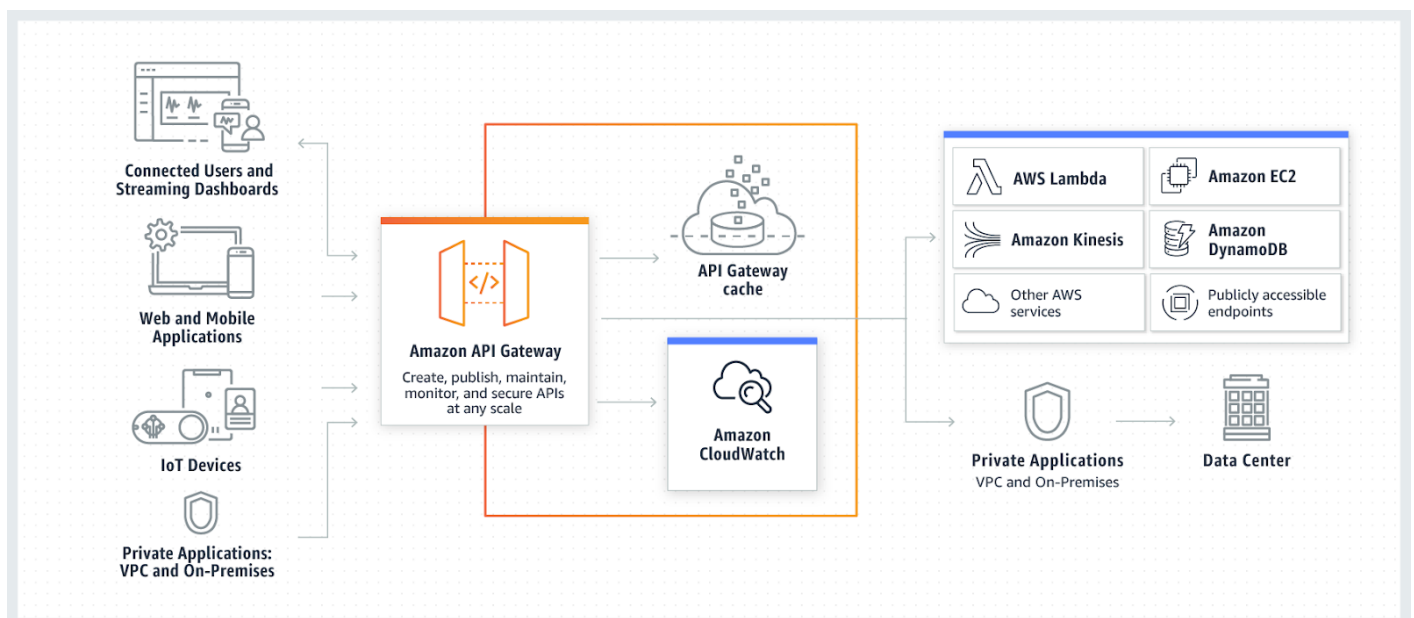
AWS Step Function

AWS Step Function is a visual workflow service that helps developers use AWS services to build distributed applications, automate processes, orchestrate microservices, and create data and machine learning (ML) pipelines.



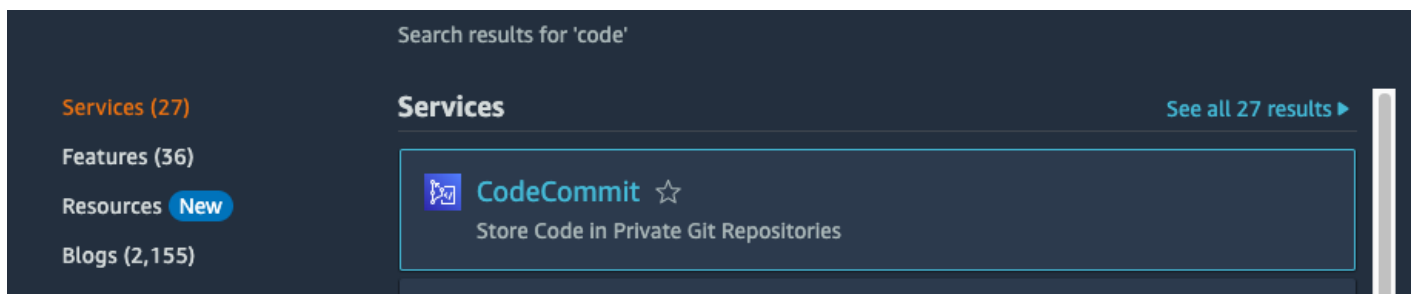
AWS API Gateway

Amazon API Gateway is a fully managed service that makes it easy for developers to create, publish, maintain, monitor, and secure APIs at any scale. APIs act as the "front door" for applications to access data, business logic, or functionality from your back end services.

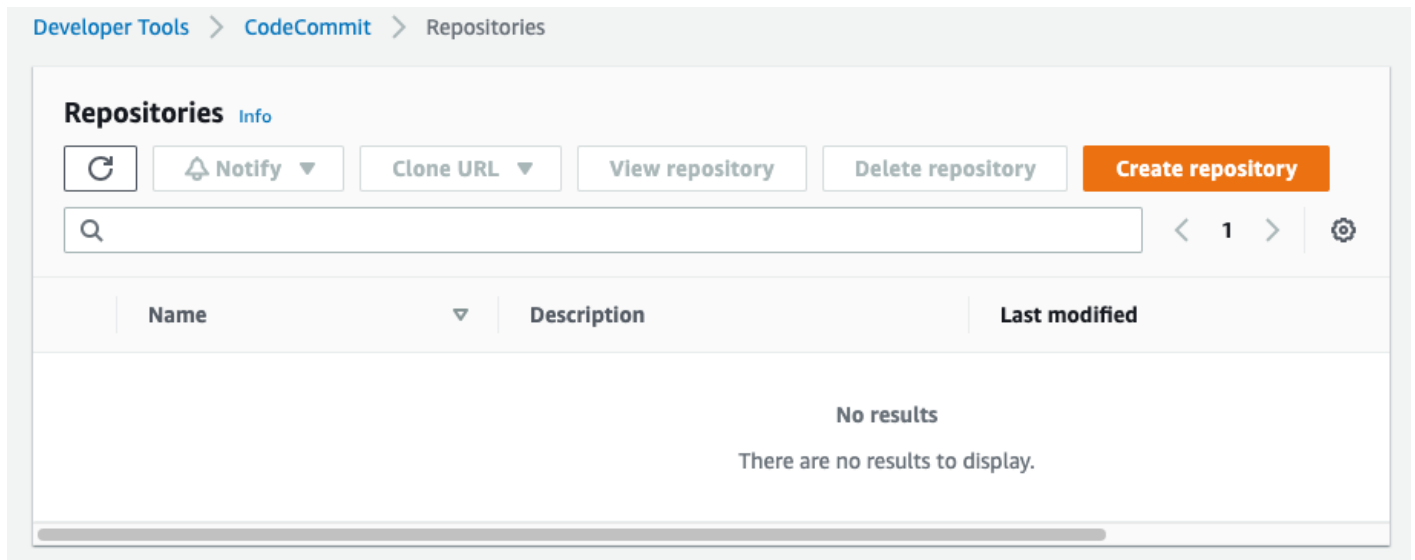


Process

Let's start with defining CodeCommit resource which you can use as your code repository.



You start with a blank repository. Click on **Create repository**.



Provide repository name and optional description.

Create repository

Create a secure repository to store and share your code. Begin by typing a repository name and a description for your repository. Repository names are included in the URLs for that repository.

Repository settings

Repository name

capitaliseAndReverseUI

100 characters maximum. Other limits apply.

Description - *optional*

Front end to let users add a text and see the response which is capitalised and reversed.

1,000 characters maximum

Tags

Add

☐ Enable Amazon CodeGuru Reviewer for Java and Python - *optional*

Get recommendations to improve the quality of the Java and Python code for all pull requests in this repository.

A service-linked role will be created in IAM on your behalf if it does not exist.

Cancel

Create

Now get the details of this repository to clone in your local environment.

✓ Success

Repository successfully created

Create a notification rule for this repository

×

Developer Tools > CodeCommit > Repositories > capitaliseAndReverseUI

capitaliseAndReverseUI

Clone URL ▲

Clone HTTPS

Clone SSH

Clone HTTPS (GRC)

▼ Connection steps

HTTPS

SSH

HTTPS (GRC)

Clone the repository on your computer to create the required html resources.

```
$ git clone https://git-codecommit.eu-west-2.amazonaws.com/v1/repos/capitaliseAndReverseUI
Cloning into 'capitaliseAndReverseUI'...
Username for 'https://git-codecommit.eu-west-2.amazonaws.com': 
Password for 'https://[redacted]@git-codecommit.eu-west-2.amazonaws.com': 
warning: You appear to have cloned an empty repository.
```

You then create a simple html page (that will contain the require JavaScript and CSS sections).

```
$ git add index.html
$ git status
On branch master

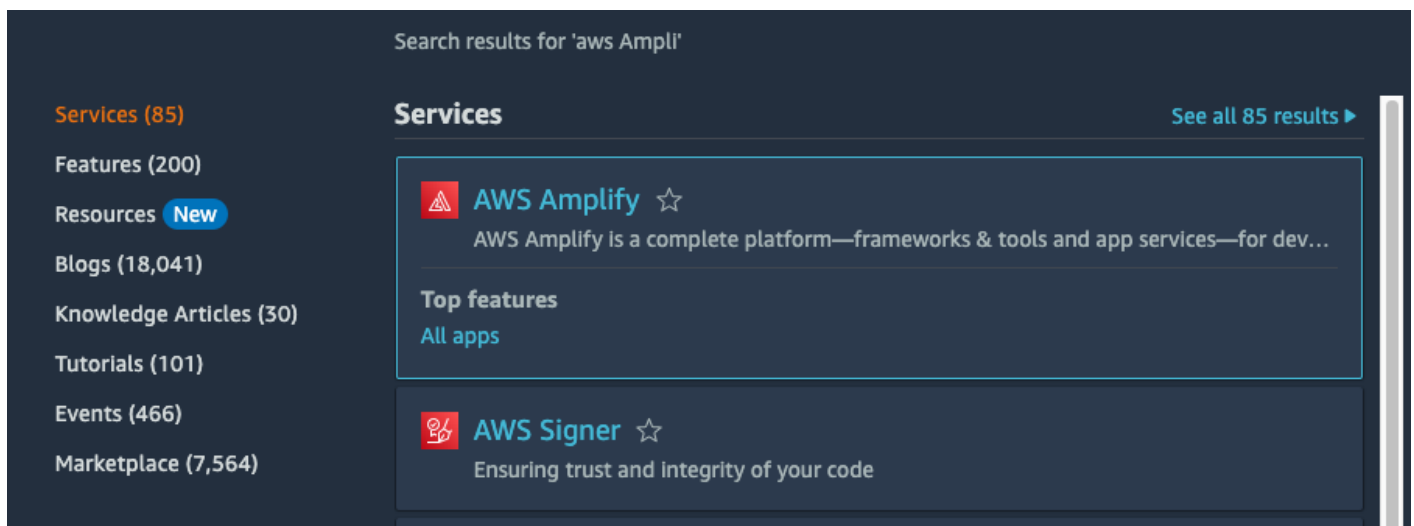
No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   index.html
```

Commit your changes and you can also push the changes to the remote repository on AWS CodeCommit.

```
$ git commit -m "index.html created which accepts a text value"
[master (root-commit) 14aa3ab] index.html created which accepts a text value
1 file changed, 49 insertions(+)
 create mode 100644 index.html
```

Now create AWS Amplify resource to host your static content (HTML).

The screenshot shows the AWS Management Console search results for 'aws Ampli'. On the left, there is a sidebar with navigation links: Services (85), Features (200), Resources (New), Blogs (18,041), Knowledge Articles (30), Tutorials (101), Events (466), and Marketplace (7,564). The main content area is titled 'Search results for 'aws Ampli'' and 'Services'. It lists two services: AWS Amplify and AWS Signer. AWS Amplify is highlighted with a red box and includes a star icon, a description 'AWS Amplify is a complete platform—frameworks & tools and app services—for dev...', and links for 'Top features' and 'All apps'. AWS Signer is listed below it with a description 'Ensuring trust and integrity of your code'. A 'See all 85 results' link is in the top right corner.

Start by creating the resource.



AWS Amplify

Fastest, easiest way to develop mobile and web apps that scale.

GET STARTED



AWS Amplify is a set of products and tools that enable mobile and front-end web developers to build and deploy secure, scalable full-stack applications, powered by AWS.

Choose **Host your web app.**

Amplify Hosting



Host your web app

Connect your Git repository to continuously deploy your frontend and backend. Host it on a globally available CDN.



Get started

Select AWS CodeCommit; this is where you have pushed changes from your local environment to the repository.

Get started with Amplify Hosting

Amplify Hosting is a fully managed hosting service for web apps. Connect your repository to build, deploy, and host your web app.

From your existing code

Connect your source code from a Git repository or upload files to host a web app in minutes.

☐ GitHub



☐ Bitbucket



☐ GitLab



☒ AWS CodeCommit



☐ Deploy without Git provider



Amplify Hosting requires read-only access to your repository.

Continue

You will now link the master branch with AWS Amplify. This will provide the continuous delivery for you whenever you push changes to master branch.

Add repository branch

AWS CodeCommit

✔ AWS CodeCommit authorization was successful.

Repository service provider



AWS CodeCommit

Recently updated repositories

If you don't see your repository below, please push a commit and then click the refresh button.

capitaliseAndReverseUI



Branch

Select a branch from your repository.

master

☐ Connecting a monorepo? Pick a folder.

Cancel

Previous

Next

Accept the default build settings.

Build settings

App build and test settings

App name

Pick a name for your app.

capitaliseAndReverseUI

Name cannot contain periods

Build and test settings

We've auto-detected your app's build settings. Please ensure your build command and output folder (baseDirectory) are correctly detected.

```
1 version: 1
2 frontend:
3   phases:
4     # IMPORTANT - Please verify your build commands
5     build:
6       commands: []
7   artifacts:
8     # IMPORTANT - Please verify your build output directory
9     baseDirectory: /
10    files:
11      - '**/*'
12    cache:
13      paths: []
14
```

Build and test settings

Download

Edit

☒ Allow AWS Amplify to automatically deploy all files hosted in your project root directory

► Advanced settings

IAM Role

IAM service role

Amplify requires read-only access to your CodeCommit repository. To create custom roles go to the [IAM console](#).

- ☒ Create and use a new service role
- ☐ Use an existing service role

Cancel

Previous

Next

Review and complete the process.

Review

Repository details

Repository service

AWS CodeCommit

Repository

capitaliseAndReverseUI

Branch

master

Branch environment

Application root

App settings

Edit

App name

capitaliseAndReverseUI

Framework

Web

Build image

Using default image

Build settings

Auto-detected settings will be used

Environment variables

None

Cancel

Previous

Save and deploy

Process takes some time to complete (provisioning, building, and deploying your changes).

capitaliseAndReverseUI

Actions ▼

The app homepage lists all deployed frontend and backend environments.

► Learn how to get the most out of Amplify Hosting

0 of 5 steps complete X

Hosting environments

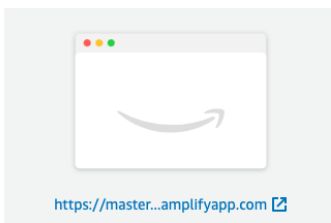
Backend environments

This tab lists all connected branches, select a branch to view build details.

Connect branch

master

Continuous deploys set up ([Edit](#))



Last deployment
23/11/2022, 22:05:05

Last commit
This is an autogenerated message | Auto-build | [AWS CodeCommit - master](#)

Previews
Disabled

Once completed, you can visit the URL to see your web application in action.


Submit

But this application is not complete, you are yet to build the back end to do the capitalization and reversal of the input string.

You start by defining the first AWS Lambda function to Capitalize the input text.

Services

See all 6 results ▶

 **Lambda** ☆

Run Code without Thinking about Servers

Compute

AWS Lambda

lets you run code without thinking about servers.

You pay only for the compute time that 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.

Get started

Author a Lambda function from scratch, or choose from one of many preconfigured examples.

Create a function

Provide the function name and runtime. Choose Python 3.9 for this.

Create function [Info](#)

AWS Serverless Application Repository applications have moved to [Create application](#).

Author from scratch

Start with a simple Hello World example.

Use a blueprint

Build a Lambda application from sample code and configuration presets for common use cases.

Container image

Select a container image to deploy for your function.

Basic information

Function name

Enter a name that describes the purpose of your function.

Use only letters, numbers, hyphens, or underscores with no spaces.

Runtime [Info](#)

Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.

Architecture [Info](#)

Choose the instruction set architecture you want for your function code.

☒ x86_64☐ arm64

Permissions [Info](#)

By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

► Change default execution role

► Advanced settings

Cancel

Create function

Defining the function will look like this:

capitaliseFunc

▼ Function overview [Info](#)



capitaliseFunc



Layers

(0)



API Gateway

+ Add trigger

Code

Test

Monitor

Configuration

Aliases

Versions

Code source [Info](#)

File Edit Find View Go Tools Window

Test

Deploy



Go to Anything (% P)

Environment

capitaliseFunc - /

lambda_function.py



lambda_function ×

Execution results ×



```
1 import json
2
3 def lambda_handler(event, context):
4     input_text = str(event['inputText'])
5     capitalised_input_text = input_text.upper()
6     return {"inputText": capitalised_input_text}
```

The code you have written is very basic, as it accepts input text as part of the body (it's a HTTP POST function). And returns the object again as input text with capitalized value (so you can chain this to the reverse function).

```
import json
```

```
def lambda_handler(event, context):
```

```
    input_text = str(event['inputText'])
```

```
    capitalised_input_text = input_text.upper()
```

```
    return {"inputText": capitalised_input_text}
```

You can also define a simple test event to validate your function logic.

Configure test event



A test event is a JSON object that mocks the structure of requests emitted by AWS services to invoke a Lambda function. Use it to see the function's invocation result.

To invoke your function without saving an event, modify the event, then choose Test. Lambda uses the modified event to invoke your function, but does not overwrite the original event until you choose Save changes.

Test event action

☐ Create new event

☒ Edit saved event

Event name

capitaliseText



Delete

Event JSON

Format JSON

```
1 {  
2   "inputText": "value1"  
3 }
```

And once you deploy your function, you can then test it and see the following outcome.

The screenshot shows the AWS Lambda console interface. At the top, there are tabs for 'Tools', 'Window', 'Test', and 'Deploy'. Below the tabs, there are tabs for 'lambda_function' and 'Execution result'. The 'Execution result' tab is active, showing the following details:

- Test Event Name:** capitaliseText
- Response:**

```
{  
  "inputText": "VALUE1"  
}
```
- Function Logs:**

```
START RequestId: b56e0c53-6fd2-4080-9800-ac42efdb43b4 Version: $LATEST  
END RequestId: b56e0c53-6fd2-4080-9800-ac42efdb43b4  
REPORT RequestId: b56e0c53-6fd2-4080-9800-ac42efdb43b4 Duration: 1.29 ms Billed Duration: 2 ms Memory Size: 128 MB Max Memory Used: 36 MB Init Duration: 106.76 ms
```
- Request ID:** b56e0c53-6fd2-4080-9800-ac42efdb43b4

At the top right of the console, there are status indicators: 'Status: Succeeded', 'Max memory used: 36 MB', and 'Time: 1.29 ms'.

Similarly, you create the reverse function.

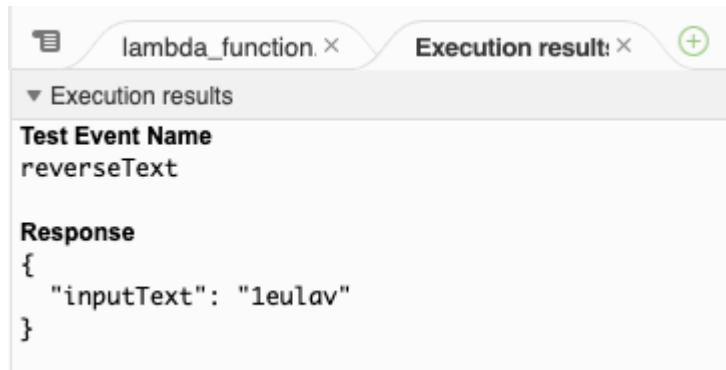
The screenshot shows the AWS Lambda console interface with the 'lambda_function' tab active. The code editor displays the following Python code:

```
1 import json  
2  
3 def lambda_handler(event, context):  
4     input_text = str(event['inputText'])  
5     reversed_input_text = input_text[::-1]  
6     return {"inputText": reversed_input_text}  
7
```

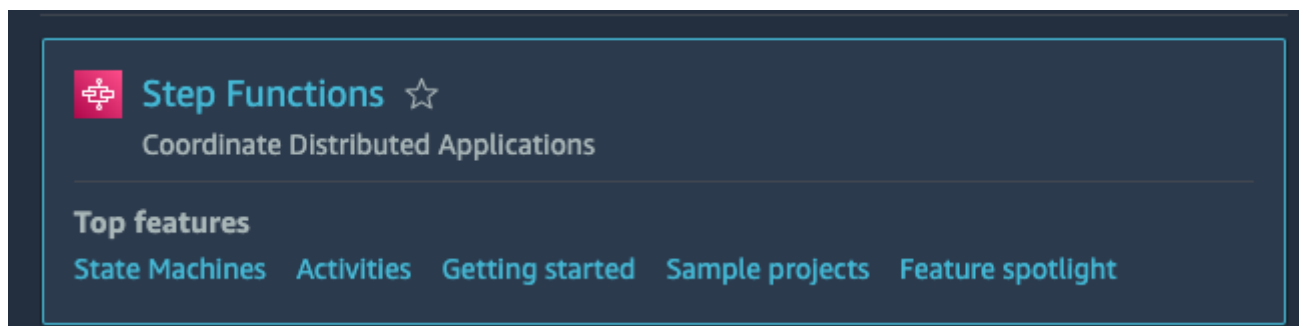
import json

```
def lambda_handler(event, context):
    input_text = str(event['inputText'])
    reversed_input_text = input_text[::-1]
    return {"inputText": reversed_input_text}
```

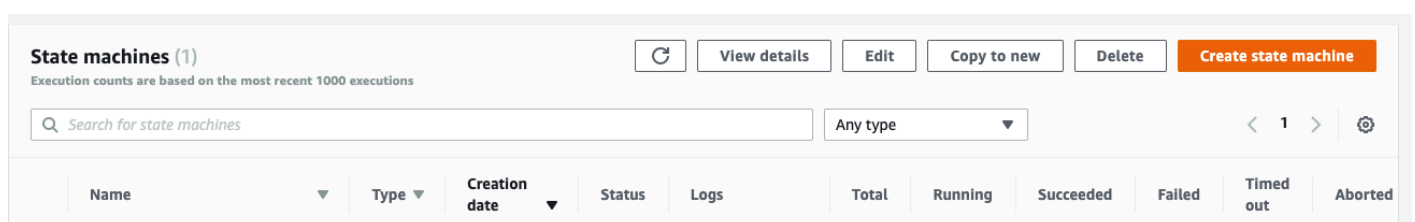
And test it



Now that you have two functions defined and created, you can chain them together using StepFunctions.



Start by creating a state machine.



You can choose to design workflow visually for ease and use Express to make your functions work synchronously.

Choose authoring method

Design your workflow visually

Drag and drop your workflow together with Step Functions Workflow Studio. New

Write your workflow in code

Author your workflow using Amazon States Language. You can generate code snippets to easily build out your workflow steps.

Run a sample project

Deploy and run a fully functioning sample project in minutes using CloudFormation.

Type

☐ Standard

Durable, checkpointed workflows for machine learning, order fulfillment, IT/DevOps automation, ETL jobs, and other long-duration workloads.

☒ Express

Event-driven workflows for streaming data processing, microservices orchestration, IoT data ingestion, mobile backends, and other short duration, high-event-rate workloads.

► Help me decide

Cancel

Next

Start

Lambda: Invoke

Invoke Capitalise

Lambda: Invoke

Invoke Reverse

End

Invoke Capitalise

Configuration

Input

Output

Error handling

State name

Invoke Capitalise

API

Lambda: Invoke

Integration type Info

The type of service integration to use. [Learn more](#)

Optimized

API Parameters

Function name

The Lambda function to invoke

Enter function name

arn:aws:lambda:eu-west-2::function:capitaliseFunc:\$LATEST

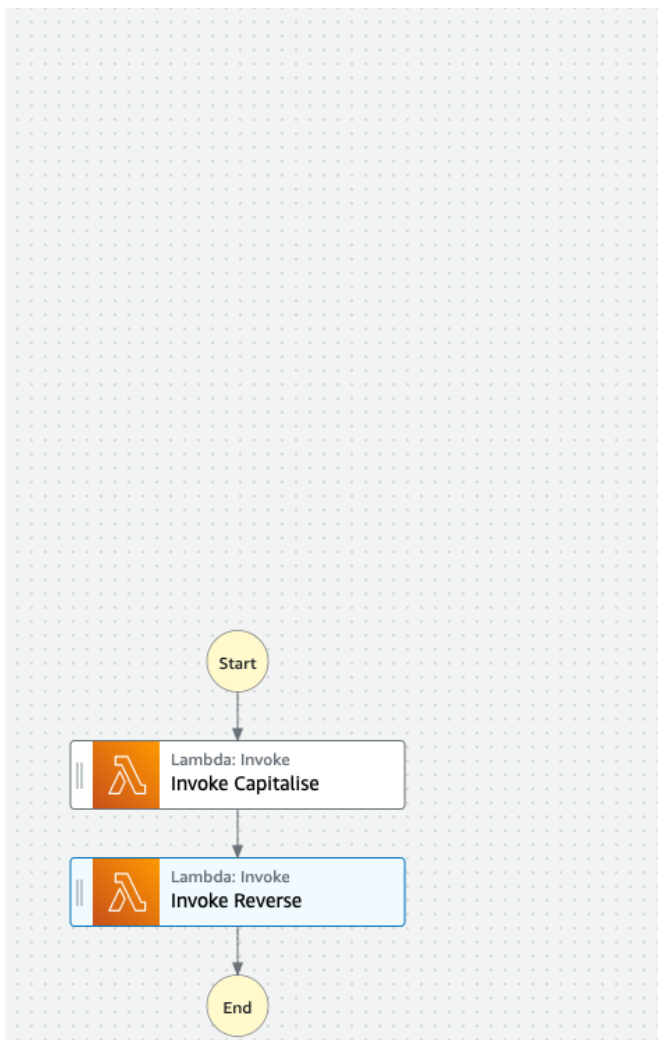
Must be a valid function name.

View function

Payload

The JSON that you want to provide to your Lambda function.

Use state input as payload



Invoke Reverse

Configuration

Input

Output

Error handling

State name

Invoke Reverse

API

Lambda: Invoke

Integration type [Info](#)

The type of service integration to use. [Learn more](#)

Optimized

API Parameters

☐ Edit as JSON

Function name

The Lambda function to invoke

Enter function name

arn:aws:lambda:eu-west-2

:function:reverseFunc:\$LATEST

Must be a valid function name.

[View function](#)

Payload

The JSON that you want to provide to your Lambda function.

Use state input as payload

Edit CapitaliseAndReverseStateMachine

Cancel

Start execution

Save

Definition

Define your workflow using [Amazon States Language](#). Test your data flow with the new [Data Flow Simulator](#).

Generate code snippet

Format JSON

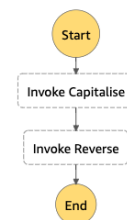
Export

Layout

Workflow Studio **New**

```

1 {
2   "Comment": "A description of my state machine",
3   "StartAt": "Invoke Capitalise",
4   "States": {
5     "Invoke Capitalise": {
6       "Type": "Task",
7       "Resource": "arn:aws:states:::lambda:invoke",
8       "OutputPath": "$$.Payload",
9       "Parameters": {
10        "Payload.$": "$",
11        "FunctionName": "arn:aws:lambda:eu-west-2: :function:capitaliseFunc:$LATEST"
12      },
13       "Retry": [
14        {
15          "ErrorEquals": [
16            "Lambda.ServiceException",
17            "Lambda.AWSLambdaException",
18            "Lambda.SdkClientException",
19            "Lambda.TooManyRequestsException"
20          ],
21          "IntervalSeconds": 2,
22          "MaxAttempts": 6,
23          "BackoffRate": 2
24        }
25      ],
26       "Next": "Invoke Reverse"
27     },
28     "Invoke Reverse": {
29       "Type": "Task",
30       "Resource": "arn:aws:states:::lambda:invoke",
31       "OutputPath": "$$.Payload",
32       "Parameters": {
33        "Payload.$": "$",
34        "FunctionName": "arn:aws:lambda:eu-west-2: :function:reverseFunc:$LATEST"
35      },
36       "Retry": [
37        {
  
```



Click on New execution to test your State machine.

Start execution

Start an execution using the latest definition of the state machine. [Learn more](#)

Name - optional

testint_state_machine

Input - optional

Enter input values for this execution in JSON format

Format JSON

Export

Import

1 { "inputText": "this is an example of anagram radar" }

Execution: testing_state_machine:d0f5e86c-20c6-45b9-985b-eadd2dc30b03

Edit state machine

Export

New execution

Details

Execution input and output

Definition

Input

1 {
2 "inputText": "this is an example of anagram radar"
3 }


Output

1 {
2 "inputText": "RADAR MARGANA FO ELPMAXE NA SI SIHT"
3 }

Search results for 'API Gate'

Services

See all 37 results



API Gateway ☆

Build, Deploy and Manage APIs

REST API

Develop a REST API where you gain complete control over the request and response along with API management capabilities.

Works with the following:
Lambda, HTTP, AWS Services

Import

Build



Amazon API Gateway

APIs > Create

Show all hints



Choose the protocol

Select whether you would like to create a REST API or a WebSocket API.

☒ REST ☐ WebSocket

Create new API

In Amazon API Gateway, a REST API refers to a collection of resources and methods that can be invoked through HTTPS endpoints.

☒ New API ☐ Import from Swagger or Open API 3 ☐ Example API

Settings

Choose a friendly name and description for your API.

API name*

capitaliseAndReverseAPI

Description

Endpoint Type

Edge optimized



* Required

Create API

APIs

Custom Domain Names

VPC Links

API: **capitaliseAndR...**

Resources

Stages

Authorizers

Resources

Actions ▾

/ Methods

RESOURCE ACTIONS

Create Method

Create Resource

Enable CORS

Edit Resource Documentation

API ACTIONS

Deploy API

Import API

Edit API Documentation

Delete API

Resources Actions ▾ New Child Resource

/

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

Configure as [proxy resource](#)

☐

Resource Name*

capitaliseAndReverse

Resource Path*

/ capitaliseandreverse

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

Resources

Actions ▾

/capitalise Methods

RESOURCE ACTIONS

Create Method

Create Resource

Enable CORS

Edit Resource Documentation

Delete Resource

None

Not required

Provide information about the target backend that this method will call and whether the incoming request data should be modified.

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

AWS Region eu-west-2 ⓘ

AWS Service Step Functions ⓘ

AWS Subdomain ⓘ

HTTP method POST ⓘ

Action StartSyncExecution ⓘ

Execution role arn:aws:iam:: /APIGatewayToStepFunctions ⓘ

Credentials cache Do not add caller credentials to cache key ⓘ

Content Handling Passthrough ⓘ ⓘ

Use Default Timeout ☒ ⓘ

▼ Mapping Templates

- Request body passthrough**
- ☐ When no template matches the request Content-Type header ⓘ
 - ☐ When there are no templates defined (recommended) ⓘ
 - ☒ Never ⓘ

Content-Type	
application/json	⊖

 [Add mapping template](#)

application/json

Generate template:

```
1 #set($input = $input.json('$'))
2 {
3     "input": "$util.escapeJavaScript($input)",
4     "stateMachineArn": "arn:aws:states:eu-west-2::stateMachine
      :CapitaliseAndReverseStateMachine"
5 }
```

You then define the Stage. A Stage is a named reference to a deployment, which is a snapshot of the API. You use a Stage to manage and optimize a particular deployment. For example, you can configure Stage settings to enable caching, customize request throttling, configure logging, define stage variables, or attach a canary release for testing.

Invoke URL: <https://execute-api.eu-west-2.amazonaws.com/prod>

Settings

Logs/Tracing

Stage Variables

SDK Generation

Export

Deployment History

Documentation History

Canary

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 [Create Web ACL](#)

Client Certificate

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

Certificate

Save Changes

Generate the SDK, so you can use the generated code in your web app and call this API Gateway.

Settings

Logs/Tracing

Stage Variables

SDK Generation

Export

Deployment History

Documentation History

Canary

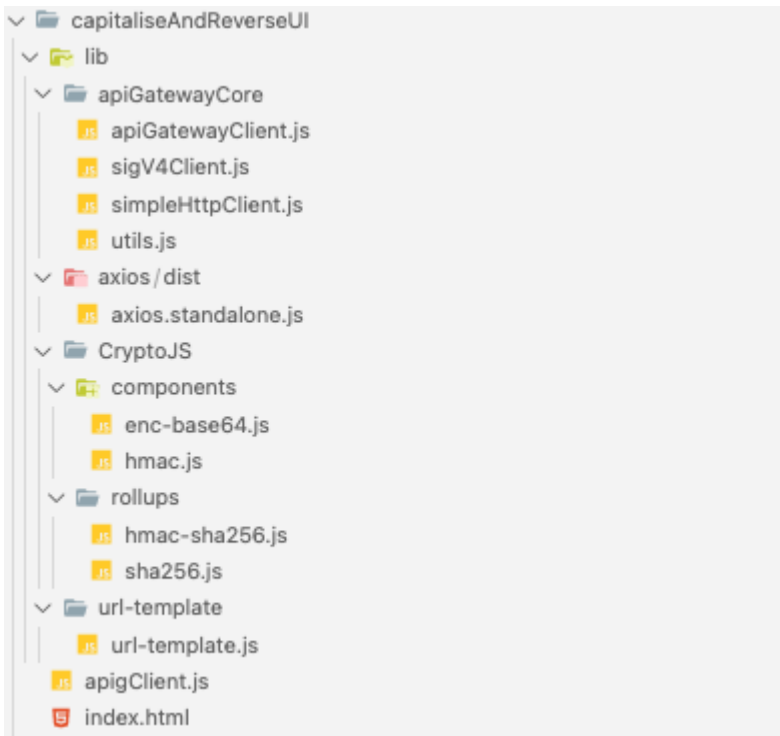
Choose a platform and provide the settings for the SDK you will generate.

Platform*


* Required

Generate SDK

You then extract the generated JavaScript code as below:



And finally deploy the API (back in the AWS API Gateway section).

Deploy API 

Choose a stage where your API will be deployed. For example, a test version of your API could be deployed to a stage named beta.

Deployment stage	<input type="text" value="[New Stage]"/>
Stage name*	<input type="text" value="prod"/>
Stage description	<input type="text"/>
Deployment description	<input type="text"/>

Cancel **Deploy**

Your final HTML will look like below; do notice that you have introduced a field to display your output.

```
<!DOCTYPE html>
```

```
<html lang="en">
```

```
<head>
```

```
<meta charset="UTF-8" />
```



```
<meta http-equiv="X-UA-Compatible" content="IE=edge" />
```

```
<meta name="viewport" content="width=device-width, initial-scale=1.0" />
```

```
<title>Reverse and Capitalise with AWS Lambda</title>
```

```
<style>
```

```
  body {
```

```
    font-family: Verdana;
```

```
    text-align: center;
```

```
  }
```

```
  form {
```

```
    max-width: 500px;
```

```
    margin: 50px auto;
```

```
    padding: 30px 20px;
```

```
    box-shadow: 2px 5px 10px rgba(0, 0, 0, 0.5);
```

```
  }
```

```
  .form-control {
```

```
    text-align: left;
```

```
    margin-bottom: 25px;
```

```
  }
```

```
  .form-control input {
```

```
    padding: 10px;
```

```
    display: block;
```

```
    width: 95%;
```

```
  }
```

```
</style>
```

```
</head>
```

```
<body>
```

```
<form id="form" onsubmit="callLambdaFunction(); return false;">

  <div class="form-control">

    <input type="text" id="inputText" placeholder="Enter some text here" />

  </div>

  <div class="form-control">

    <button type="submit" value="submit">Submit</button>

  </div>

  <div class="form-control">

    <input type="text" readonly id="outputText" placeholder="Output will appear here" />

  </div>

</form>

<script type="text/javascript" src="lib/axios/dist/axios.standalone.js"></script>

<script type="text/javascript" src="lib/CryptoJS/rollups/hmac-sha256.js"></script>

<script type="text/javascript" src="lib/CryptoJS/rollups/sha256.js"></script>

<script type="text/javascript" src="lib/CryptoJS/components/hmac.js"></script>

<script type="text/javascript" src="lib/CryptoJS/components/enc-base64.js"></script>

<script type="text/javascript" src="lib/url-template/url-template.js"></script>

<script type="text/javascript" src="lib/apiGatewayCore/sigV4Client.js"></script>

<script type="text/javascript" src="lib/apiGatewayCore/apiGatewayClient.js"></script>

<script type="text/javascript" src="lib/apiGatewayCore/simpleHttpClient.js"></script>

<script type="text/javascript" src="lib/apiGatewayCore/utls.js"></script>

<script type="text/javascript" src="apigClient.js"></script>

<script type="text/javascript">

function callLambdaFunction() {

  try {

    var inputTextValue = document.getElementById("inputText").value;
```

```
var apigClient = apigClientFactory.newClient();

var params = {};

var body = {

    inputText: inputTextValue,

};

apigClient

    .capitaliseandreversePost(params, body)

    .then(function (result) {

        document.getElementById("outputText").value = JSON.parse(result.data.output).inputText;

    })

    .catch(function (result) {

        console.log(result);

    });

} catch (error) {

    console.log(error);

}

return false;

}

</script>

</body>

</html>
```

You then commit and push the changes to AWS CodeCommit repository and wait for it to be deployed.

capitaliseAndReverseUI

Actions ▾

The app homepage lists all deployed frontend and backend environments.

► Learn how to get the most out of Amplify Hosting

1 of 5 steps complete

Hosting environments

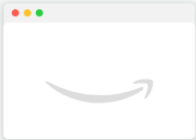
Backend environments

This tab lists all connected branches, select a branch to view build details.

Connect branch

master

Continuous deploys set up [\(Edit\)](#)



<https://master...amplifyapp.com>

✓

Provision

✓

Build

✓

Deploy

Last deployment	Last commit	Previews
24/11/2022, 00:39:02	Please visit AWS CodeCommit Co... 74aedc0 AWS CodeCommit - master	Disabled

And you can now test your web app by visiting the URL provided to you by AWS Amplify.

this is an example of anagram radar

Submit

RADAR MARGANA FO ELPMAXE NA SI SIHT

Conclusion

Services provided by AWS, more specifically around Lambda can be used to create sophisticated applications providing both front and back end. And you can build the whole eco system for your app from code repository to deployed as a serverless application.