Reading: 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.

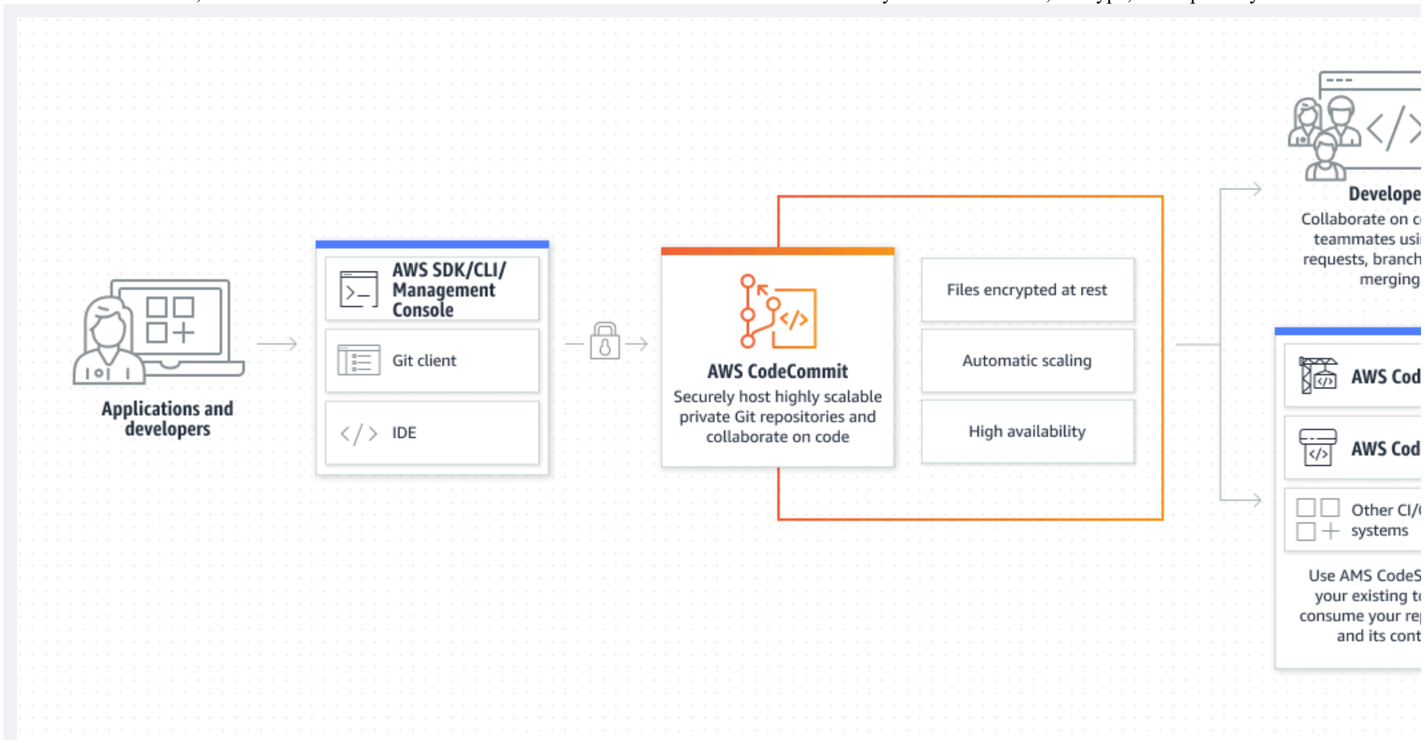
Capitalize and Reverse will be two separate Lambda functions to show you the chaining capabilities. Instead of accessing these functions directly, an API Gateway will be used to accept client requests and respond with the final output.

The application will include two separate Lambda functions: **Capitalize** and **Reverse**. These functions will be chained together using AWS Step Functions. Instead of accessing these functions directly, an API Gateway will be used to accept client requests and respond with the final output.

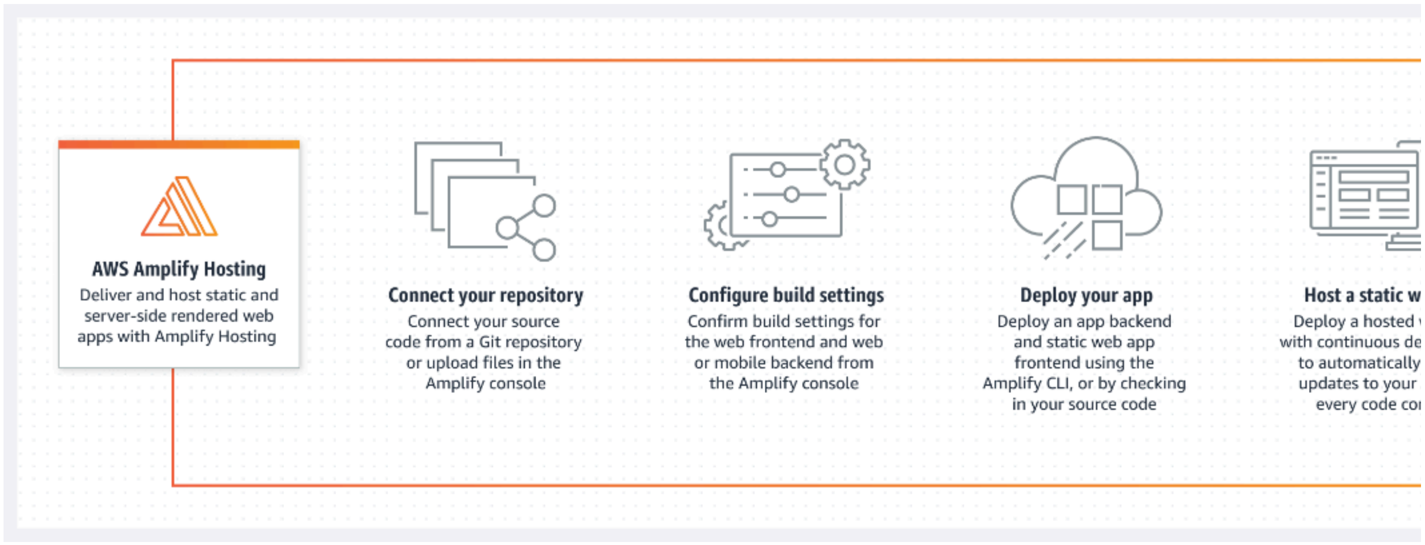
Components Used:

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

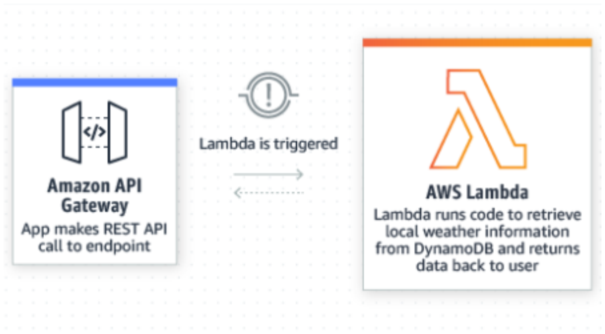
  As a Git-based service, CodeCommit is well suited to most version control needs. There are no arbitrary limits on file size, file type, and repository size.


Applications and developers
AWS SDK/CLI/ Management Console
Git client
IDE
AWS CodeCommit
Securely host highly scalable private Git repositories and collaborate on code
Files encrypted at rest
Automatic scaling
High availability
Develope
Collaborate on c teammates usi requests, branch merging
AWS Cod
AWS Cod
Other CI/ systems
Use AMS CodeS your existing t consume your re and its cont

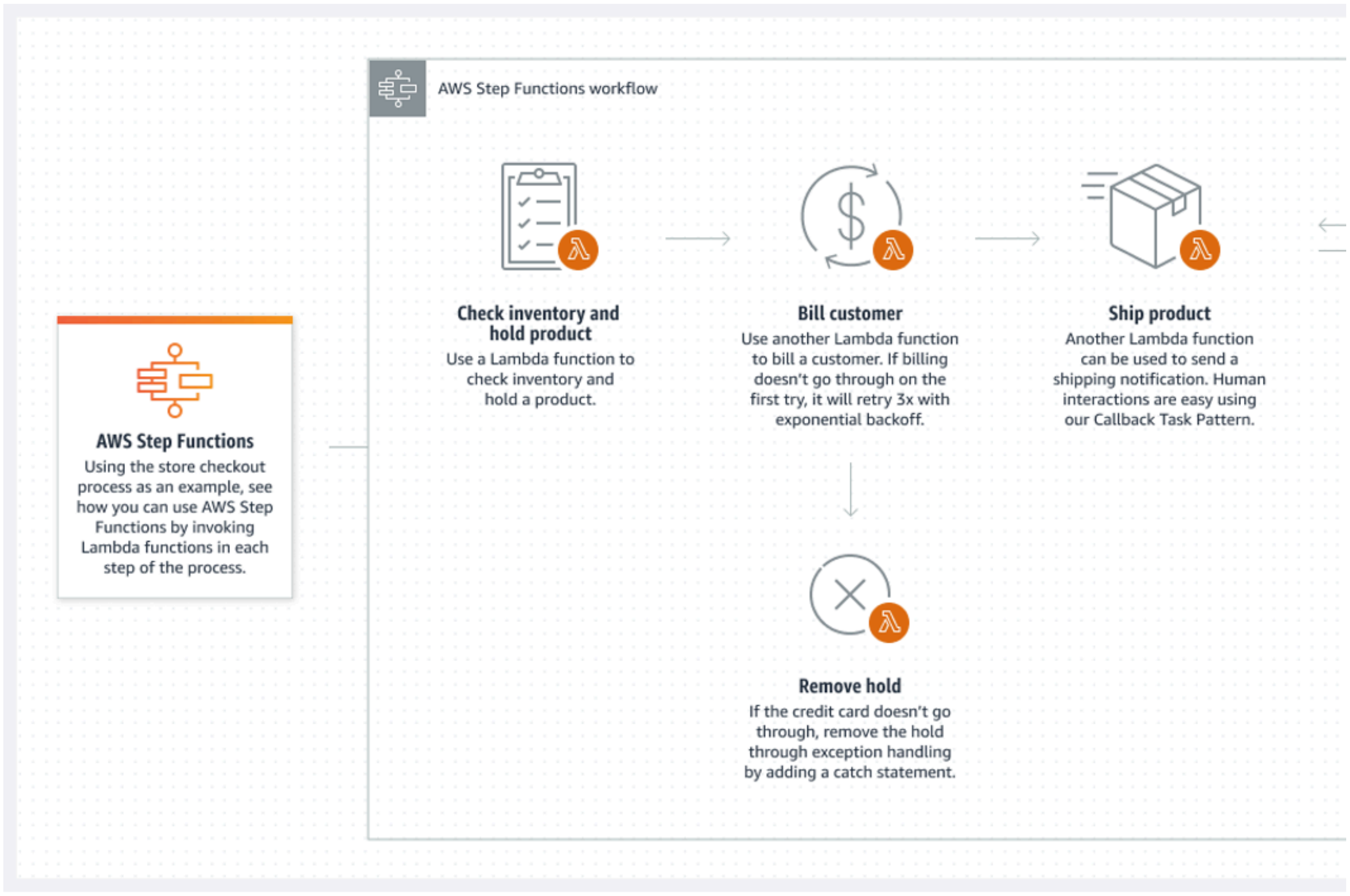
- **AWS Amplify**: AWS Amplify is a complete solution that lets front end web and mobile developers easily build, ship, and host full-stack applications on AWS, with the flexibility to leverage the breadth of AWS services as use cases evolve.


AWS Amplify Hosting
Deliver and host static and server-side rendered web apps with Amplify Hosting
Connect your repository
Connect your source code from a Git repository or upload files in the Amplify console
Configure build settings
Confirm build settings for the web frontend and web or mobile backend from the Amplify console
Deploy your app
Deploy an app backend and static web app frontend using the Amplify CLI, or by checking in your source code
Host a static w
Deploy a hosted with continuous de to automatically updates to your every code co

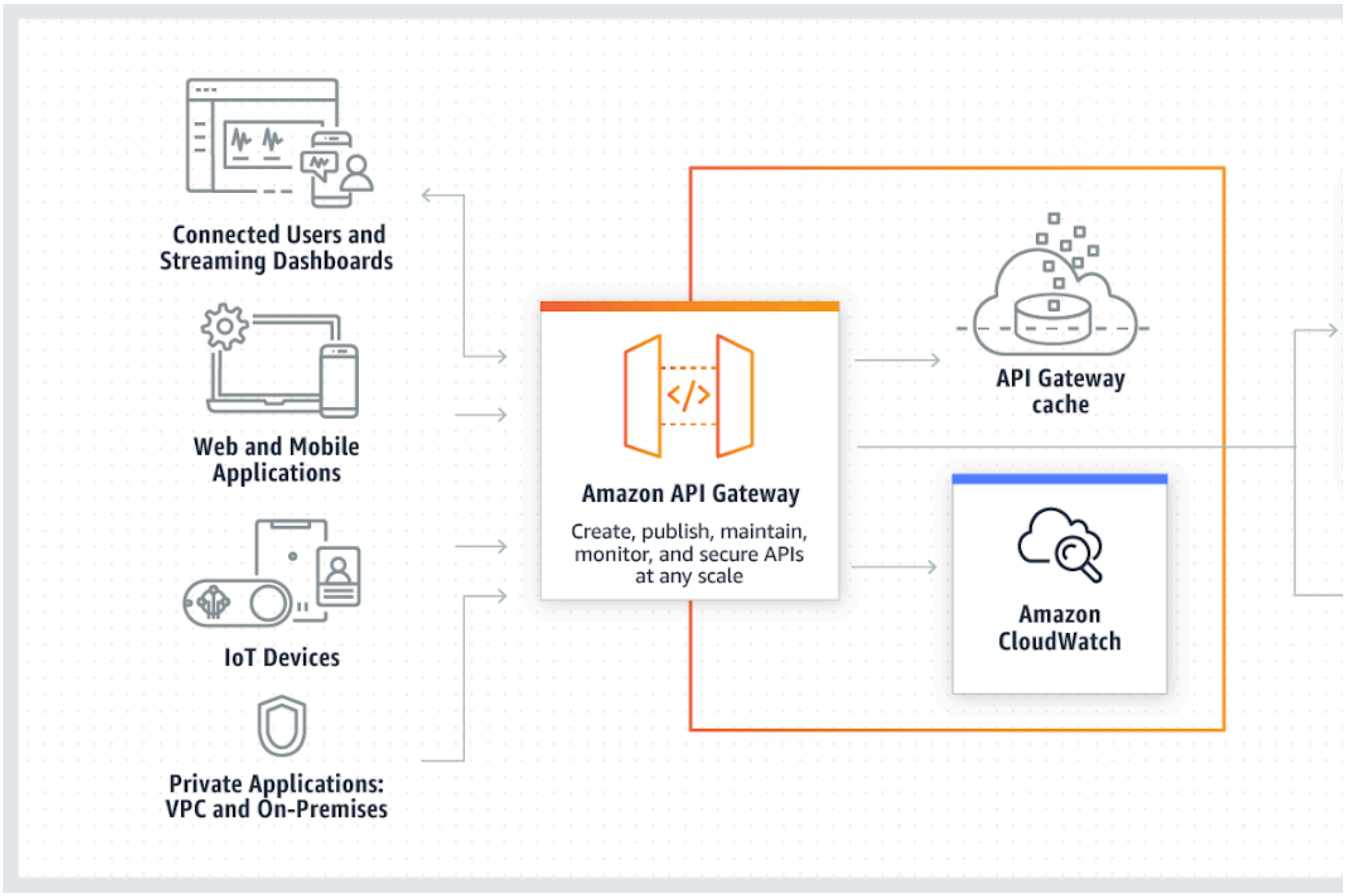
- **AWS Lambda**: AWS Lambda is a serverless, event-driven compute service that lets you run code for virtually any type of application or back end service without provisioning or managing servers. You can trigger Lambda from over 200 AWS services and software as a service (SaaS) application, and only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


Amazon API Gateway
App makes REST API call to endpoint
Lambda is triggered
AWS Lambda
Lambda runs code to retrieve local weather information from DynamoDB and returns data back to user

- **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.

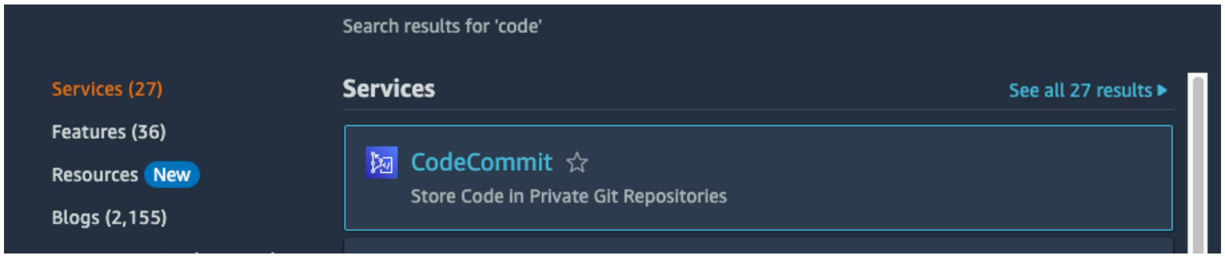

AWS Step Functions workflow
AWS Step Functions
Using the store checkout process as an example, see how you can use AWS Step Functions by invoking Lambda functions in each step of the process.
Check inventory and hold product
Use a Lambda function to check inventory and hold a product.
Bill customer
Use another Lambda function to bill a customer. If billing doesn't go through on the first try, it will retry 3x with exponential backoff.
Ship product
Another Lambda function can be used to send a shipping notification. Human interactions are easy using our Callback Task Pattern.
Remove hold
If the credit card doesn't go through, remove the hold through exception handling by adding a catch statement.

- **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.

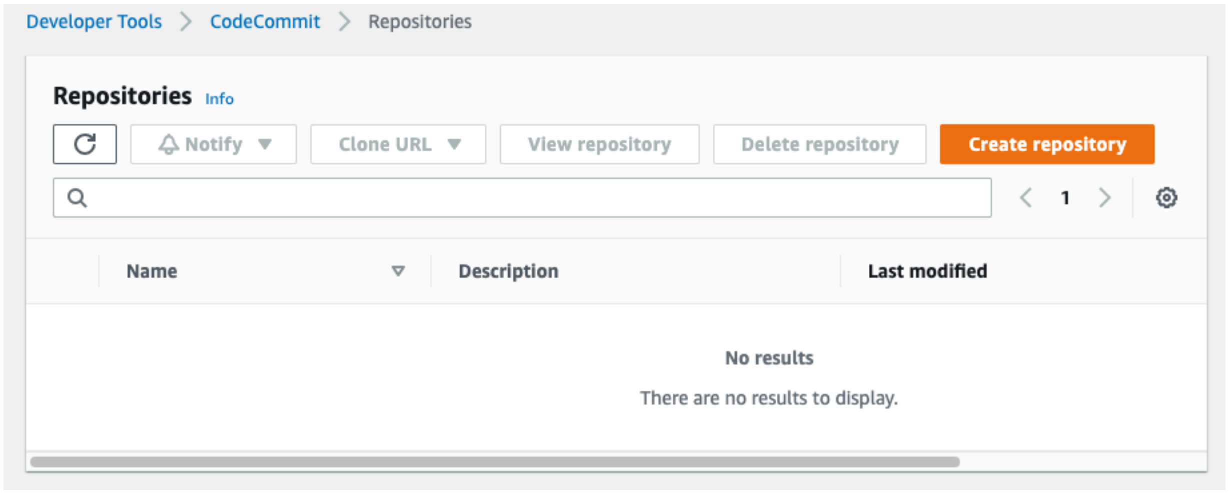

Connected Users and Streaming Dashboards
Web and Mobile Applications
IoT Devices
Private Applications: VPC and On-Premises
Amazon API Gateway
Create, publish, maintain, monitor, and secure APIs at any scale
API Gateway cache
Amazon CloudWatch

Process

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



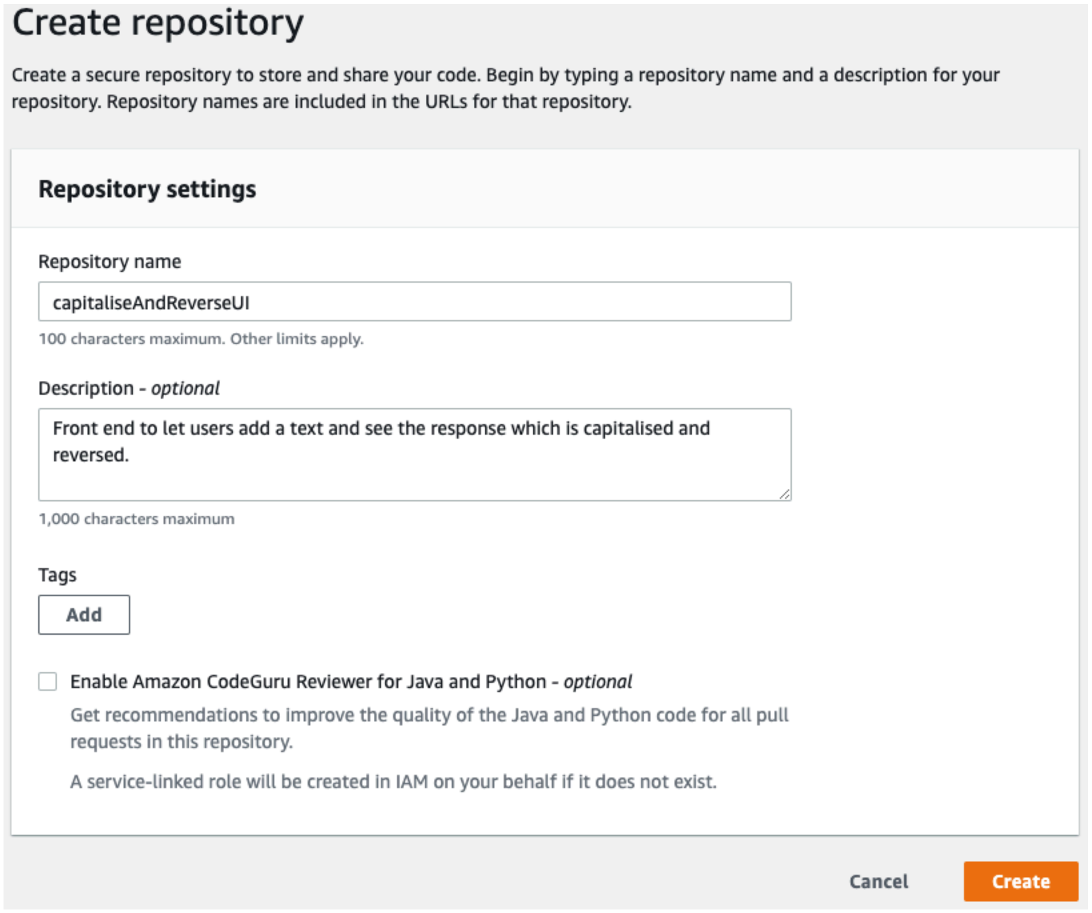
1. You start with a blank repository. Click on Create repository.



No results

There are no results to display.

2. Provide a repository name and an 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.

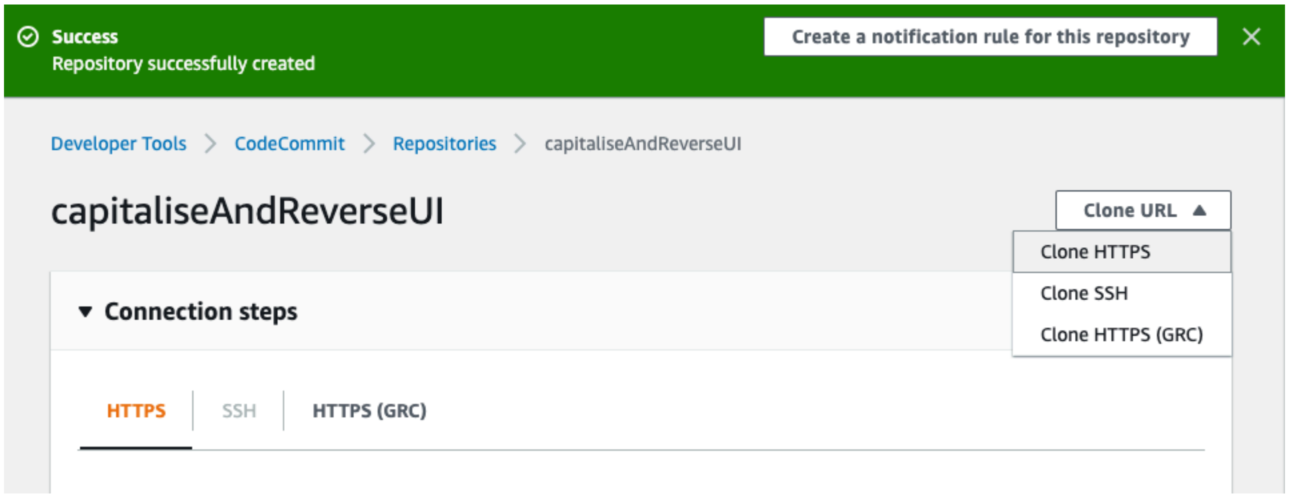
Repository settings

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

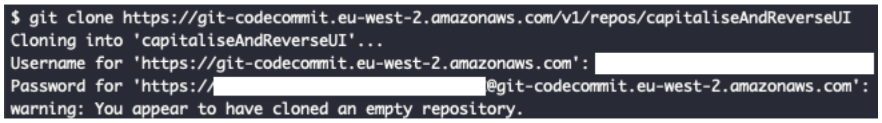
3. Now get the details of this repository to clone in your local environment.



Success
Repository successfully created

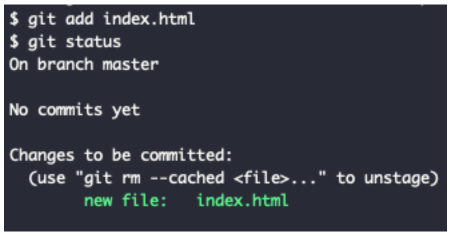
Create a notification rule for this repository
Developer Tools > CodeCommit > Repositories > capitaliseAndReverseUI
capitaliseAndReverseUI
Clone URL
Clone HTTPS
Clone SSH
Clone HTTPS (GRC)
Connection steps
HTTPS
SSH
HTTPS (GRC)

4. 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': [redacted]
Password for 'https://[redacted]@git-codecommit.eu-west-2.amazonaws.com':
warning: You appear to have cloned an empty repository.
```

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

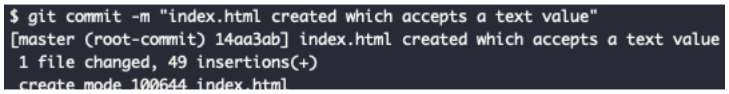


```
$ git add index.html
$ git status
On branch master

No commits yet

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

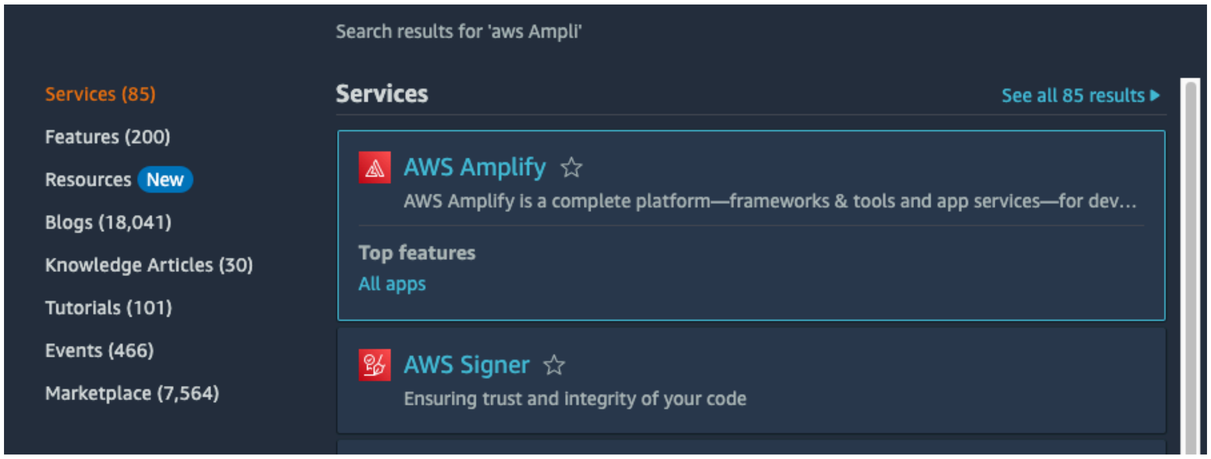
6. Commit your changes and you can also push the changes to the remote repository on AWS CodeCommit.



```
$ 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
```

Host Front-End with AWS Amplify

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



Search results for 'aws Ampli'

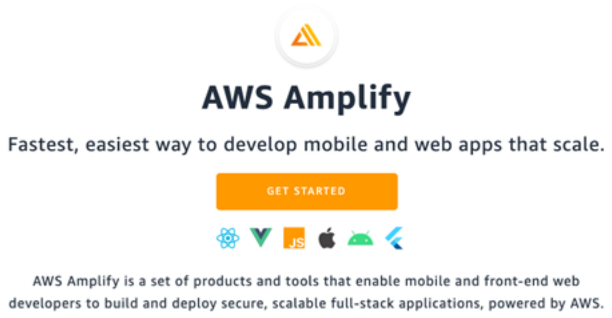
Services (85)
Features (200)
Resources New
Blogs (18,041)
Knowledge Articles (30)
Tutorials (101)
Events (466)
Marketplace (7,564)

Services — See all 85 results

AWS Amplify
AWS Amplify is a complete platform—frameworks & tools and app services—for dev…
Top features
All apps

AWS Signer
Ensuring trust and integrity of your code

2. Start by creating the resource.



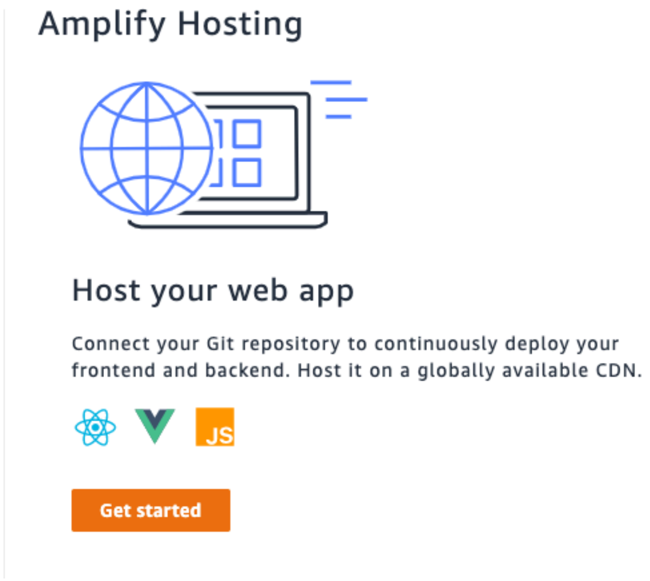
AWS Amplify

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.

3. Choose Host your web app.



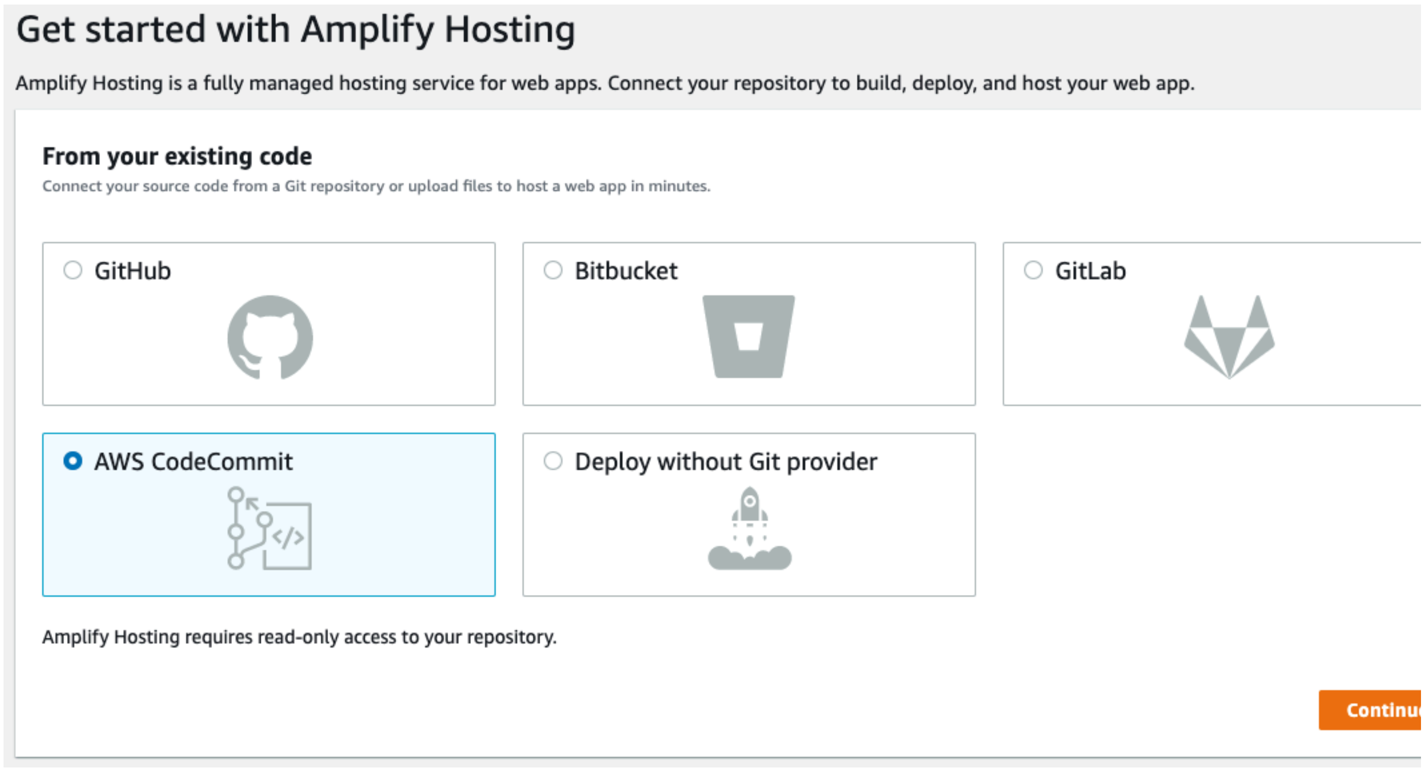
Amplify Hosting

Host your web app

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

Get started

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



Get started with Amplify Hosting

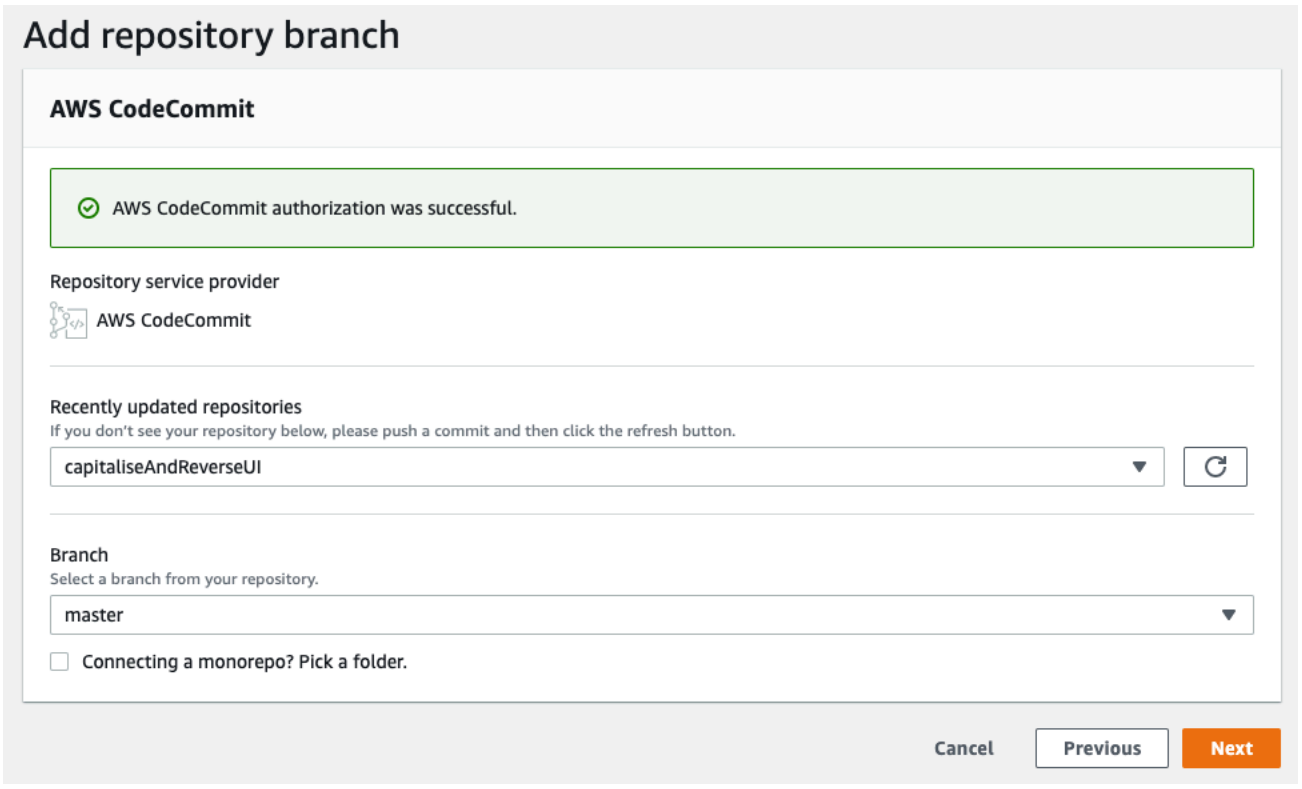
Amplify Hosting is a fully managed hosting service for web apps. Connect your repository to build, deploy, and host your web app.

From your existing code

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

Amplify Hosting requires read-only access to your repository.

5. 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

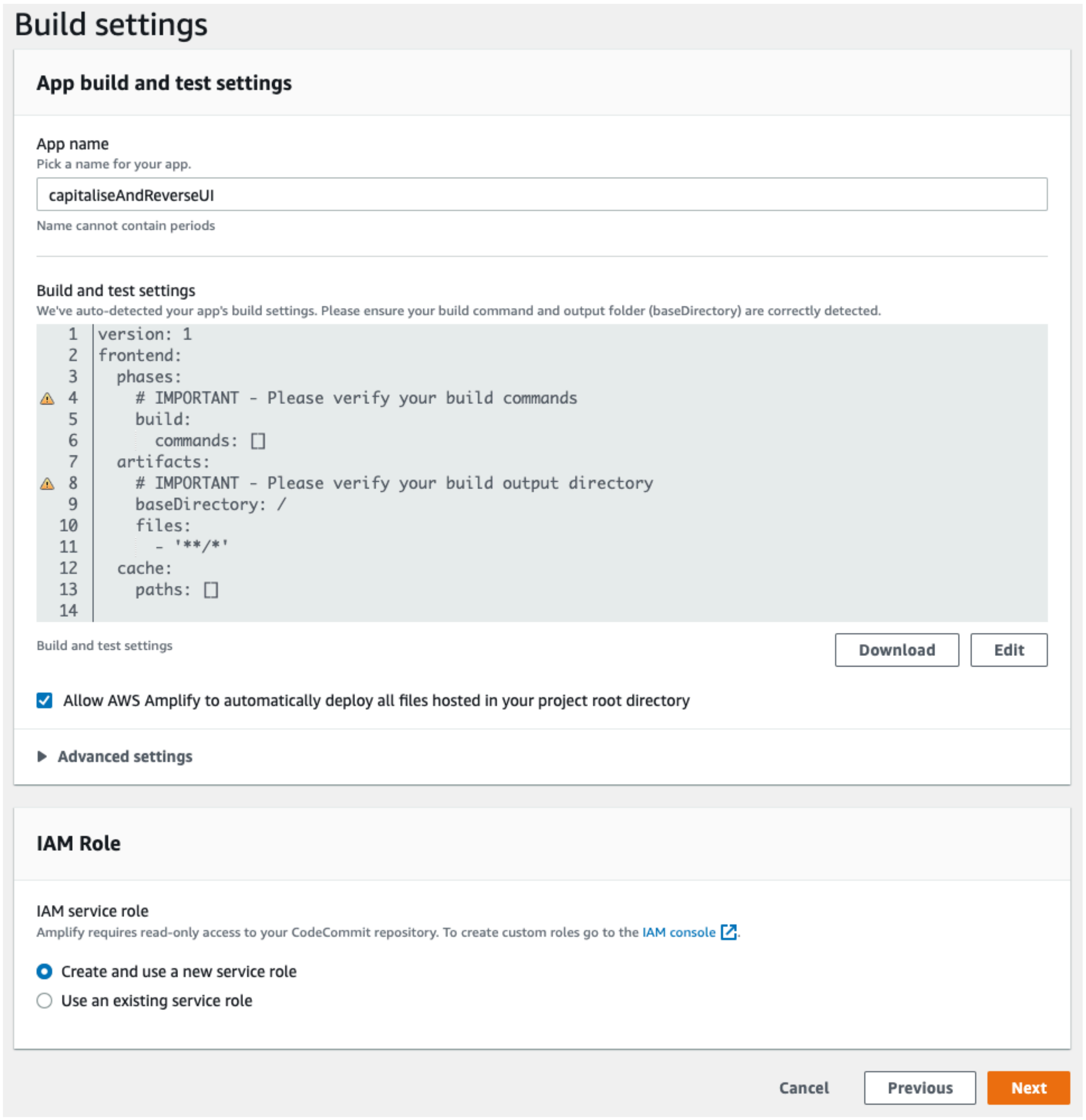
Branch

Select a branch from your repository.

master

Connecting a monorepo? Pick a folder.

6. 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.

```
version: 1
frontend:
  phases:
    # IMPORTANT - Please verify your build commands
    build:
      commands: []
  artifacts:
    # IMPORTANT - Please verify your build output directory
    baseDirectory: /
    files:
      - '**/*'
  cache:
    paths: []

```

Build and test settings

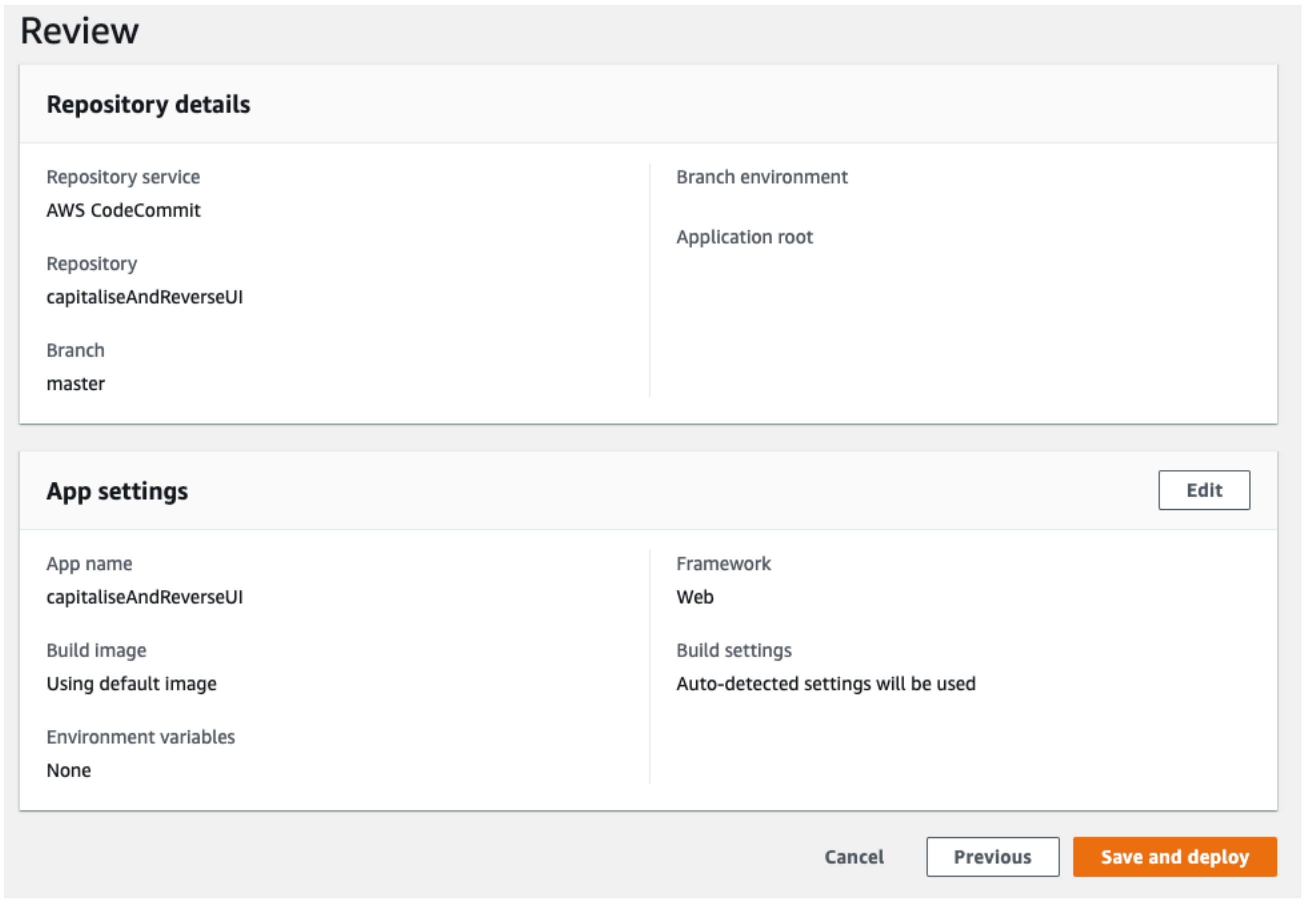
[✓] Allow AWS Amplify to automatically deploy all files hosted in your project root directory

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

7. Review and complete the process.



Review

Repository details

Repository service
AWS CodeCommit

Repository
capitaliseAndReverseUI

Branch
master

Branch environment

Application root

App settings

App name
capitaliseAndReverseUI

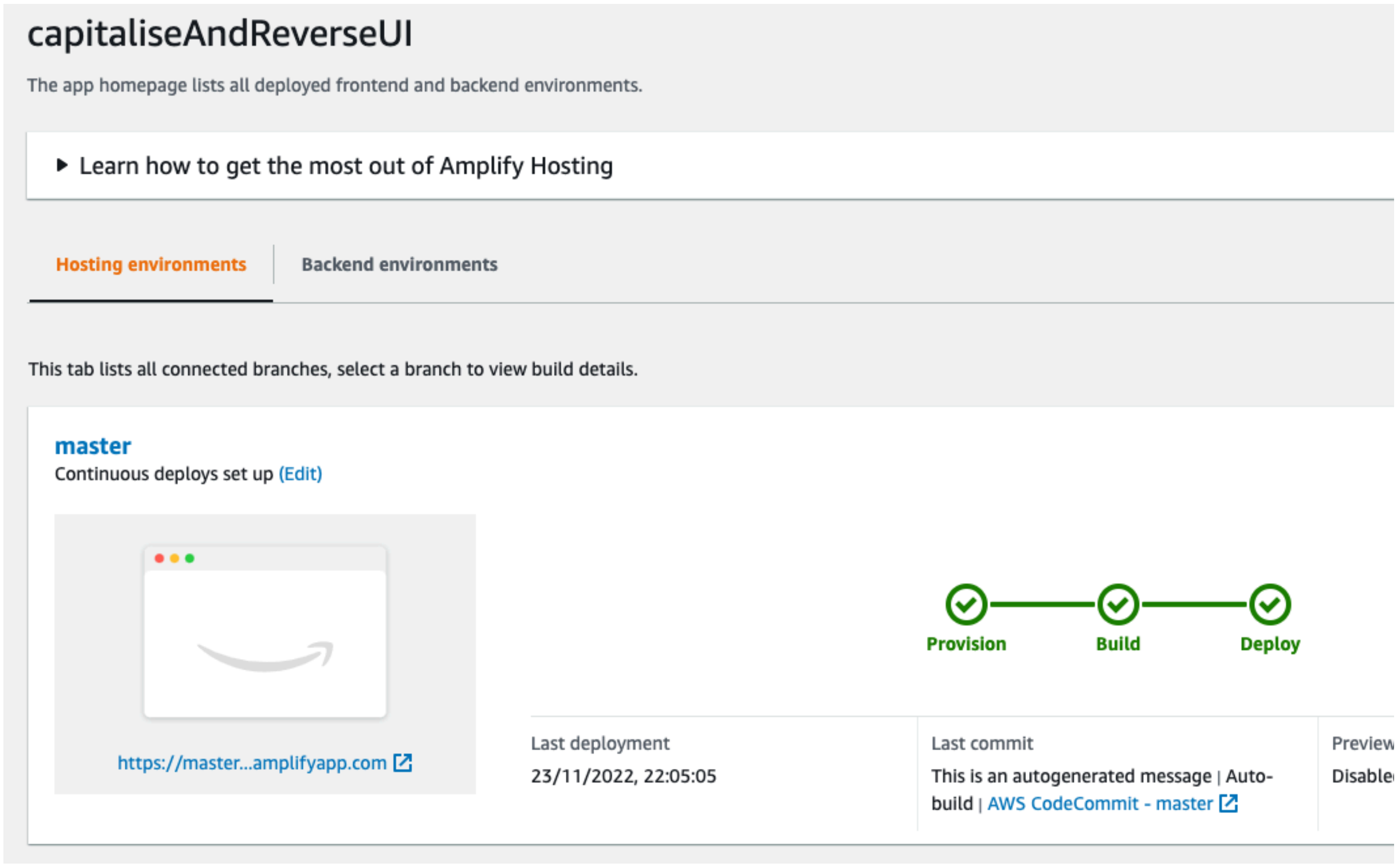
Build image
Using default image

Environment variables
None

Framework
Web

Build settings
Auto-detected settings will be used

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



capitaliseAndReverseUI

The app homepage lists all deployed frontend and backend environments.

▶ Learn how to get the most out of Amplify Hosting

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

master
Continuous deploys set up (Edit)

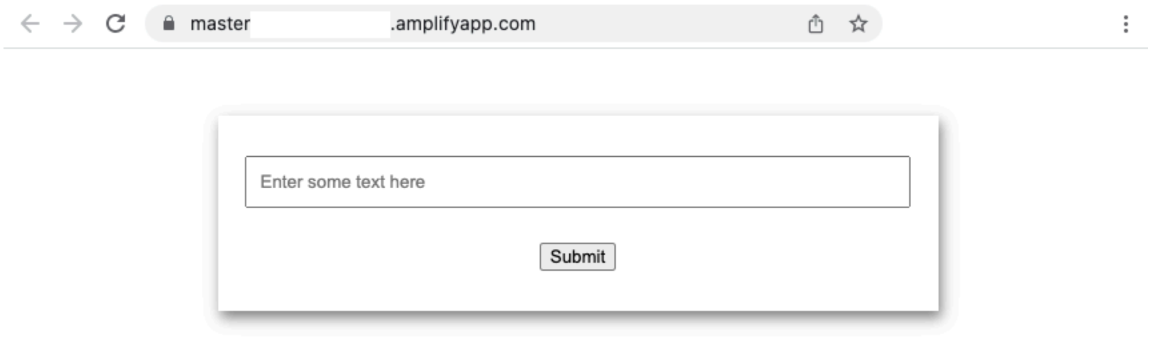
https://master...amplifyapp.com

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

Last commit
This is an autogenerated message | Auto-build | AWS CodeCommit - master

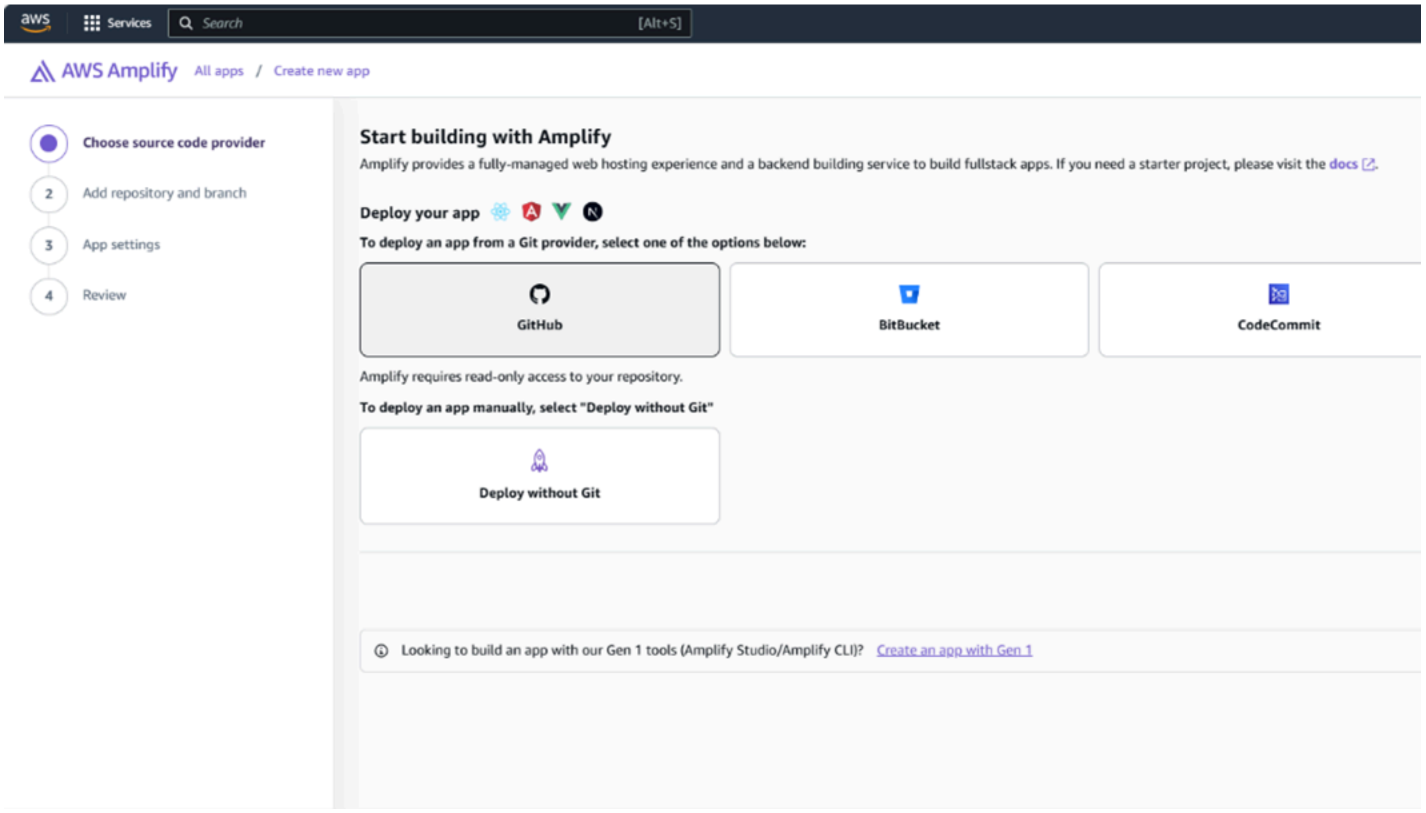
Preview
Disable

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



Deploying Your Application on AWS Amplify using GitHub

1. **Select GitHub** and click **Next**. You'll need to define the GitHub repository you want to use as your code repository.



Start building with Amplify

Amplify provides a fully-managed web hosting experience and a backend building service to build fullstack apps. If you need a starter project, please visit the docs.

Deploy your app

To deploy an app from a Git provider, select one of the options below:

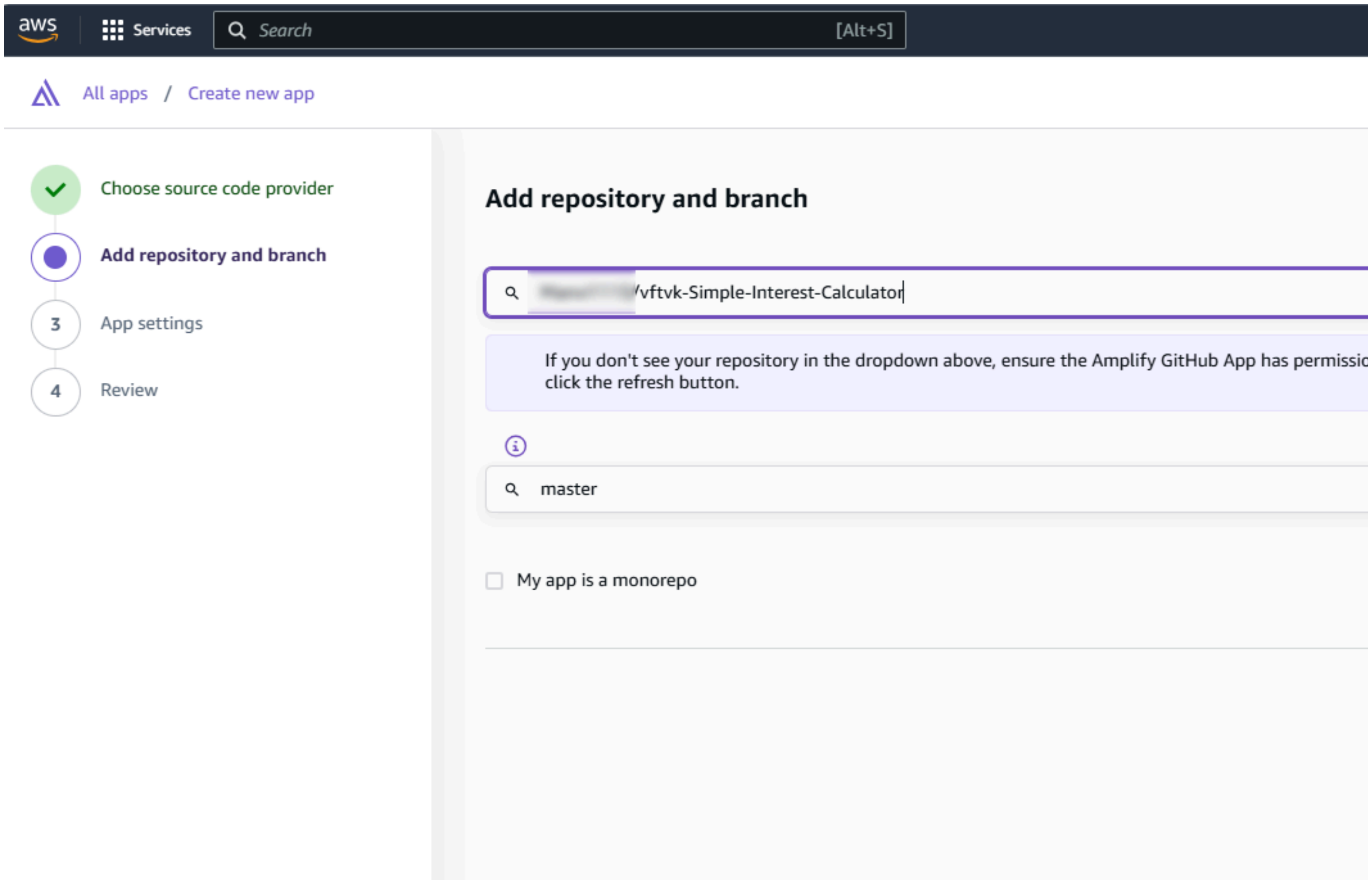
Amplify requires read-only access to your repository.

To deploy an app manually, select "Deploy without Git"

Looking to build an app with our Gen 1 tools (Amplify Studio/Amplify CLI)? Create an app with Gen 1

Note: After selecting GitHub, You'll be prompted to authorize access.

2. Choose the project you wish to deploy, then connect its working branch (main/master) to GitHub. This setup will enable continuous delivery, automatically deploying updates whenever you push changes to the working branch.


aws
Services
Search
[Alt+S]
All apps / Create new app
Choose source code provider
Add repository and branch
3 App settings
4 Review

Add repository and branch

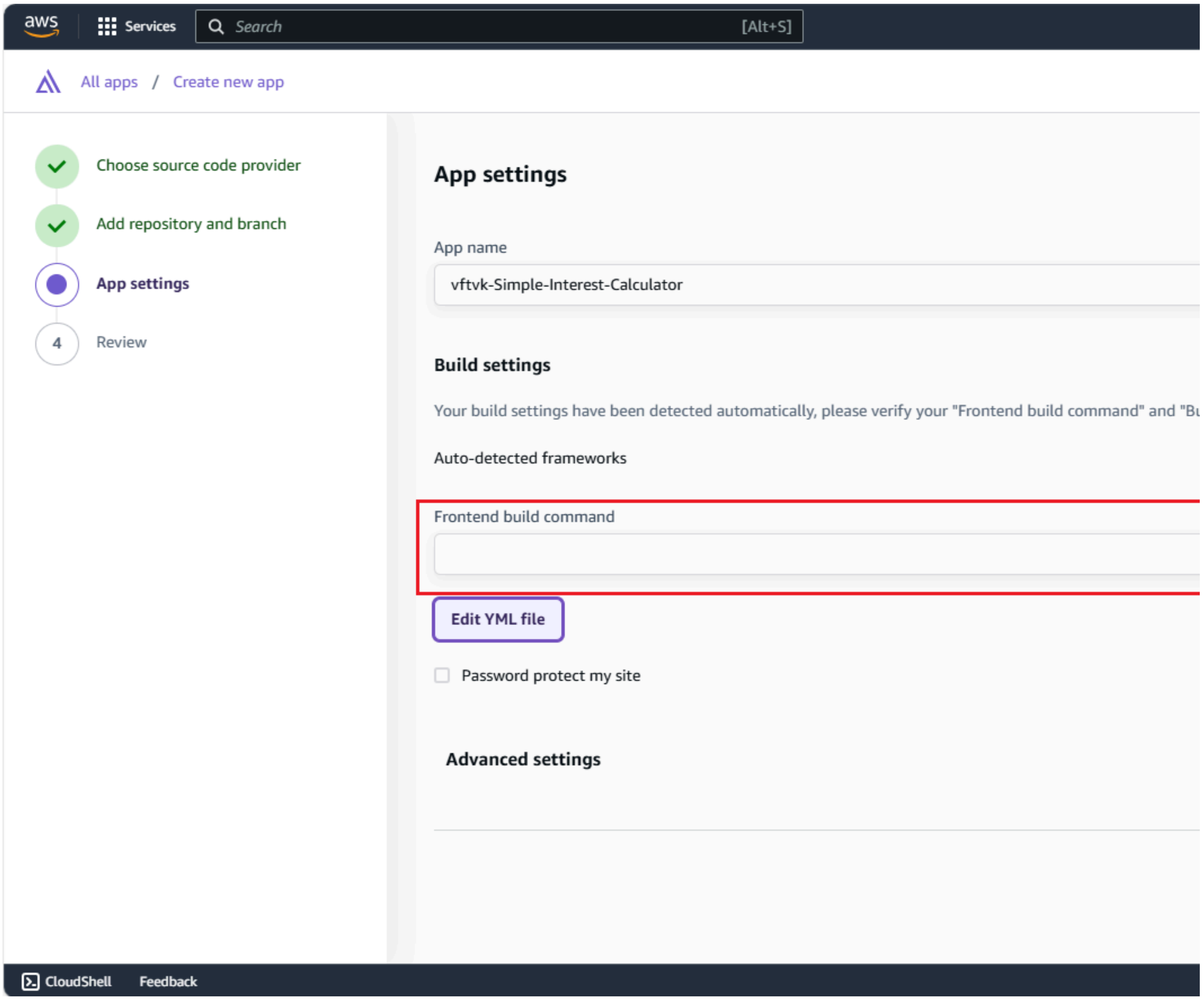
[redacted]/vftvk-Simple-Interest-Calculator

If you don't see your repository in the dropdown above, ensure the Amplify GitHub App has permissic
click the refresh button.

master

☐ My app is a monorepo

Note: Specify the Frontend build command and the build directory according to your project.


Services
Search
[Alt+S]
All apps / Create new app
Choose source code provider
Add repository and branch
App settings
4 Review

App settings

App name

vftvk-Simple-Interest-Calculator

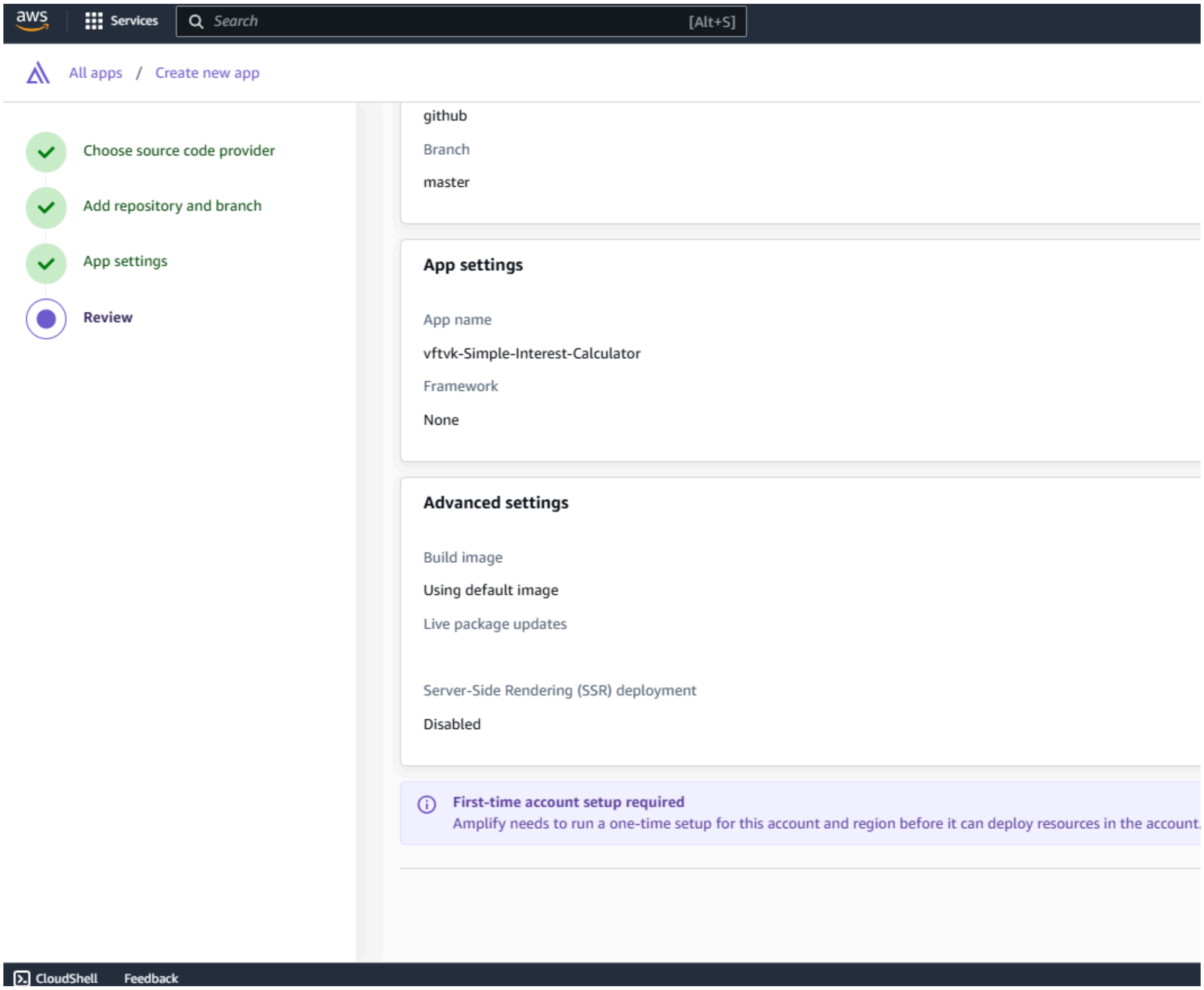
Build settings

Your build settings have been detected automatically, please verify your "Frontend build command" and "Bu

Auto-detected frameworks

Frontend build command

3. Review and complete the process.


aws
Services
Search
[Alt+S]
All apps / Create new app
Choose source code provider
Add repository and branch
App settings
Review
CloudShell
Feedback

github

Branch

master

App settings

App name

vftvk-Simple-Interest-Calculator

Framework

None

Advanced settings

Build image

Using default image

Live package updates

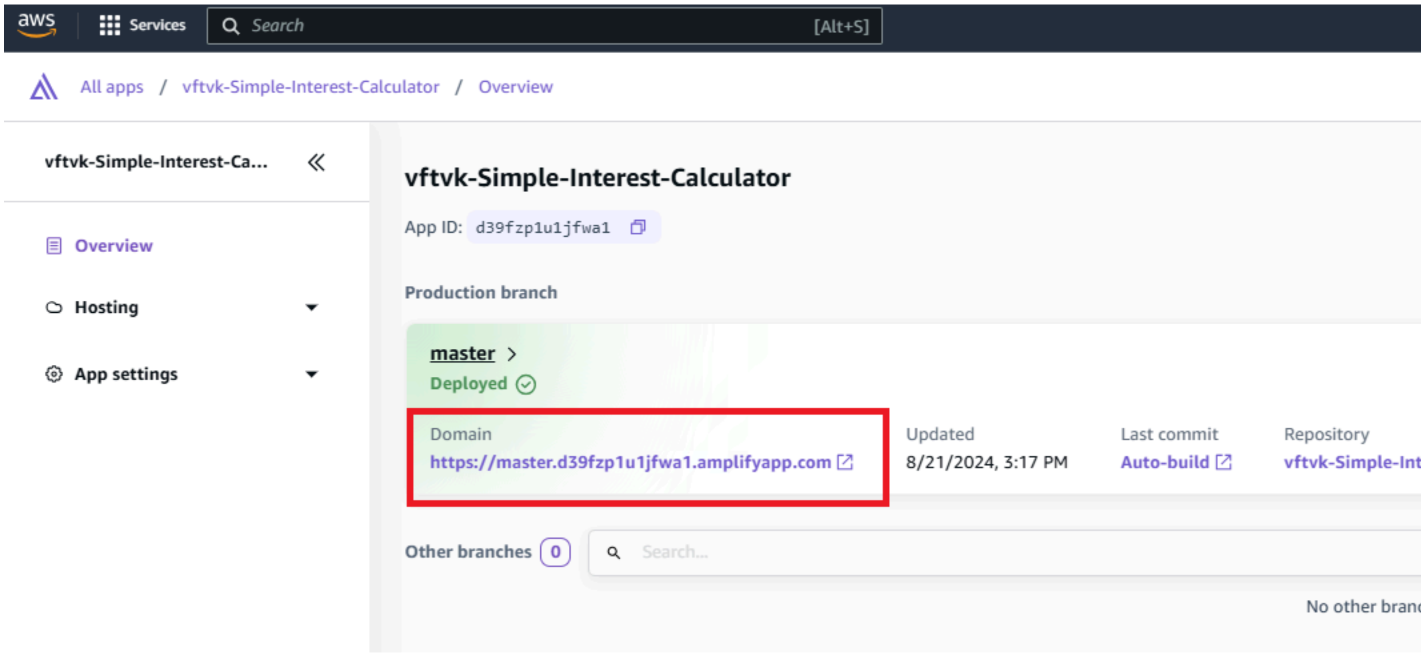
Server-Side Rendering (SSR) deployment

Disabled

First-time account setup required

Amplify needs to run a one-time setup for this account and region before it can deploy resources in the account.

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


aws
Services
Search
[Alt+S]
All apps / vftvk-Simple-Interest-Calculator / Overview
vftvk-Simple-Interest-Ca...
Overview
Hosting
App settings
Search...

vftvk-Simple-Interest-Calculator

App ID: d39fzp1u1jfwa1

Production branch

master

Deployed

Domain

https://master.d39fzp1u1jfwa1.amplifyapp.com

Updated

8/21/2024, 3:17 PM

Last commit

Auto-build

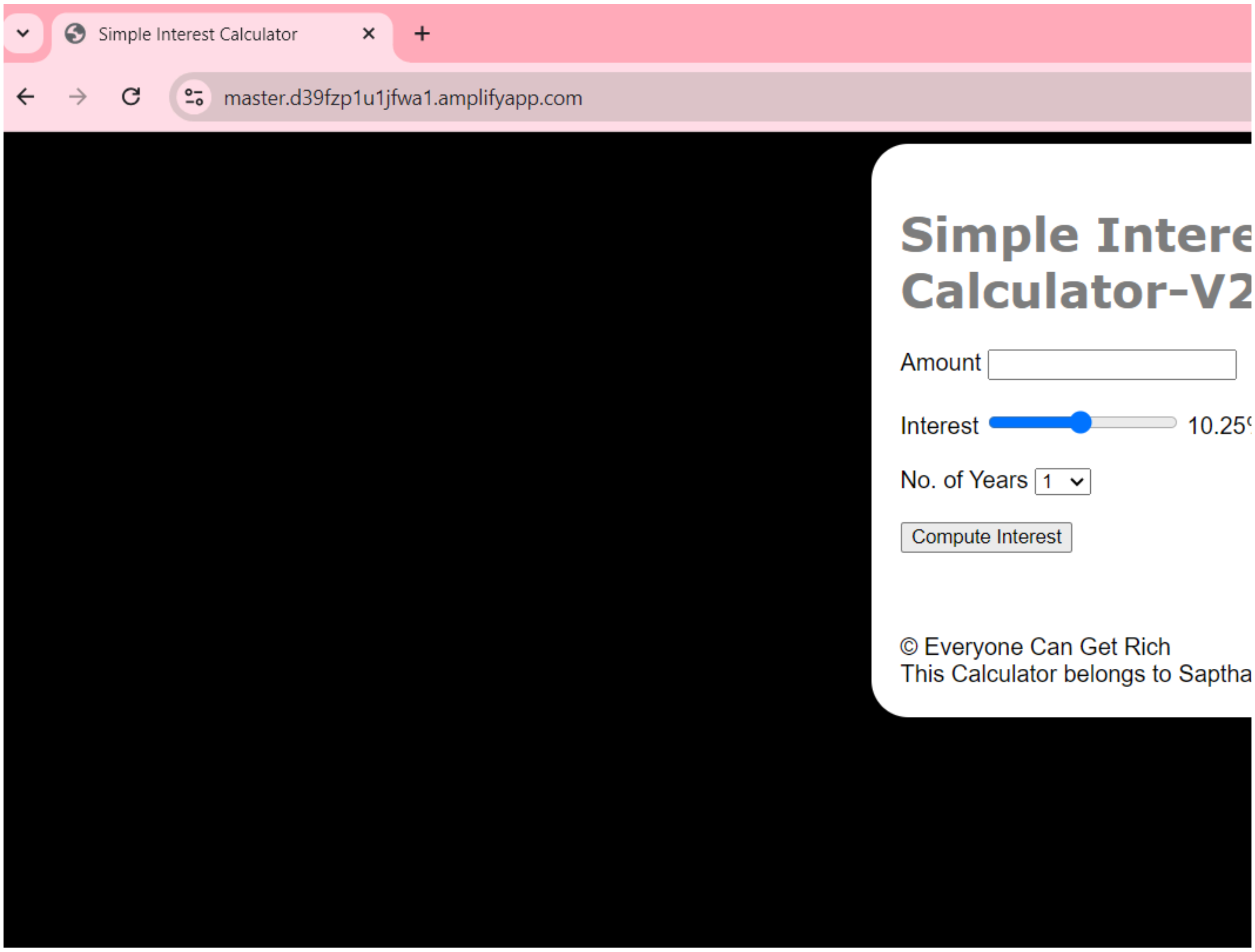
Repository

vftvk-Simple-Int

Other branches 0

No other branc

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



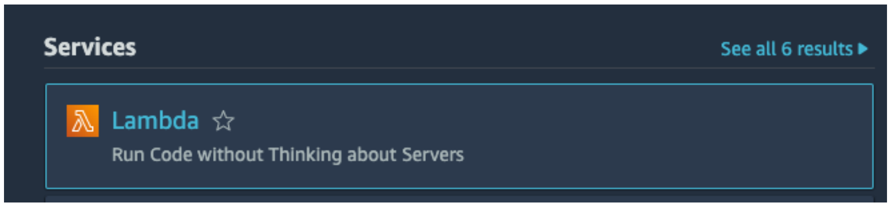
Note: If the deployment fails, click on the Hosting from the left pannel and choose build settings and update the YML file.

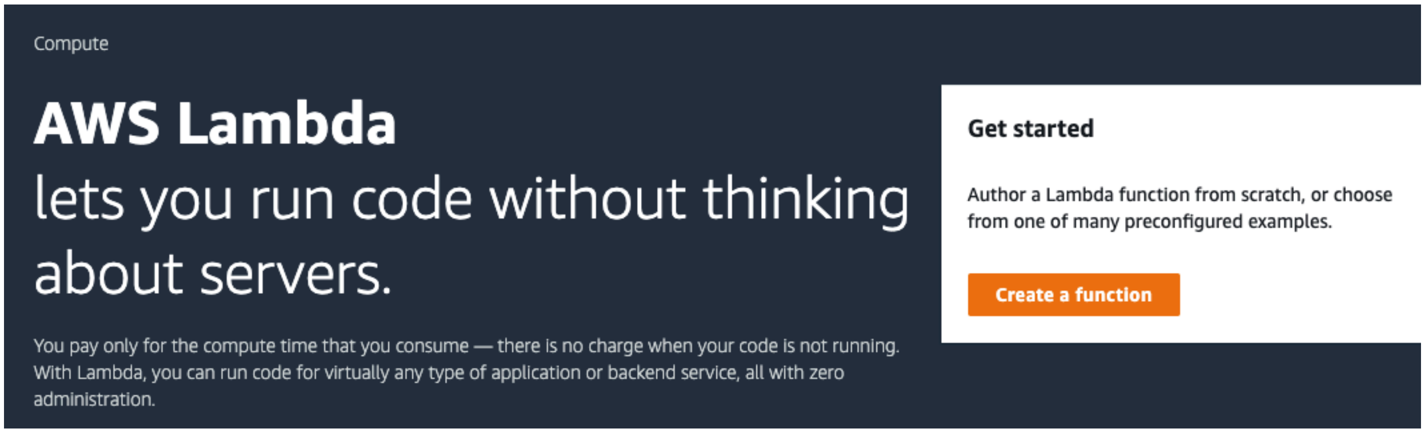
But this application is not complete, you are yet to build the back end to do the capitalization and reversal of the input string.

Create AWS Lambda Functions

Capitalize Function:

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




Compute

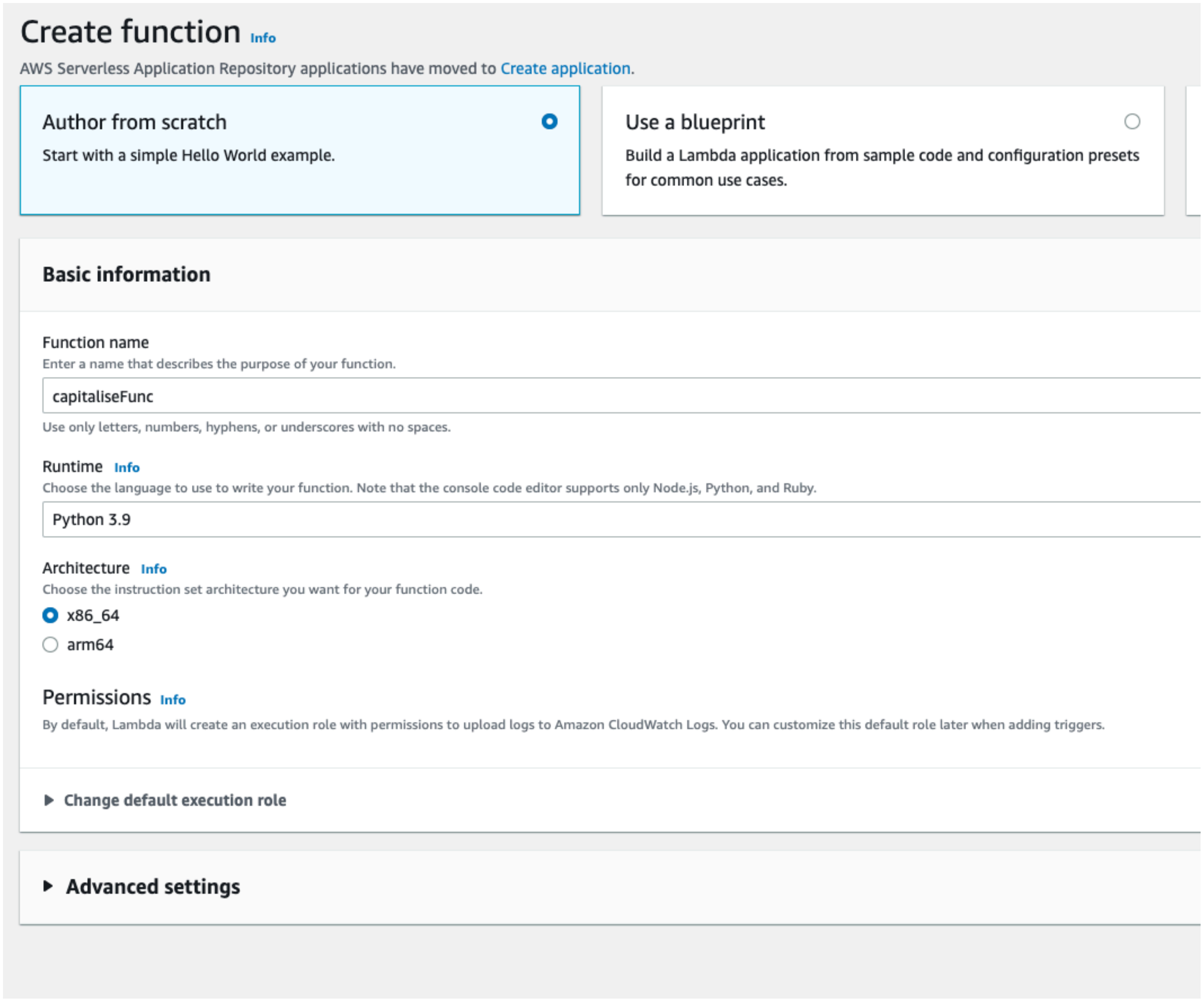
AWS Lambda
lets you run code without thinking about servers.

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.

2. 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.

Basic information

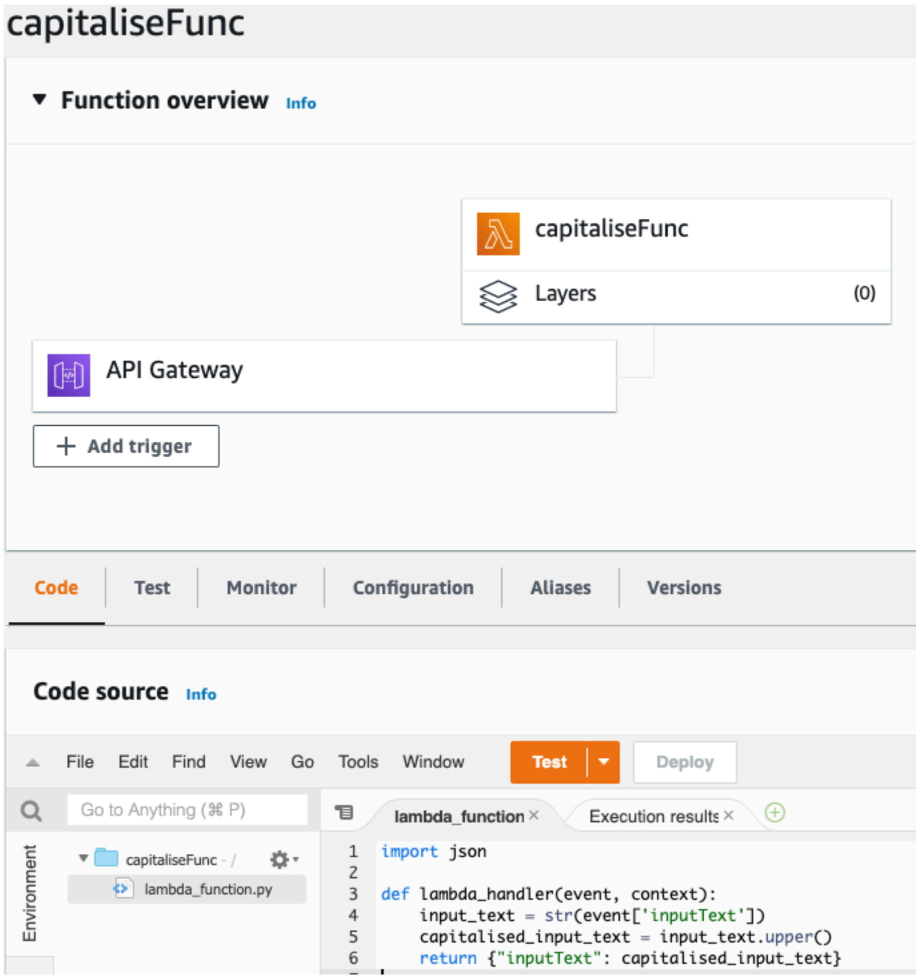
Function name
Enter a name that describes the purpose of your function.
capitaliseFunc
Use only letters, numbers, hyphens, or underscores with no spaces.

Runtime Info
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.
Python 3.9

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.

3. Defining the function will look like this:

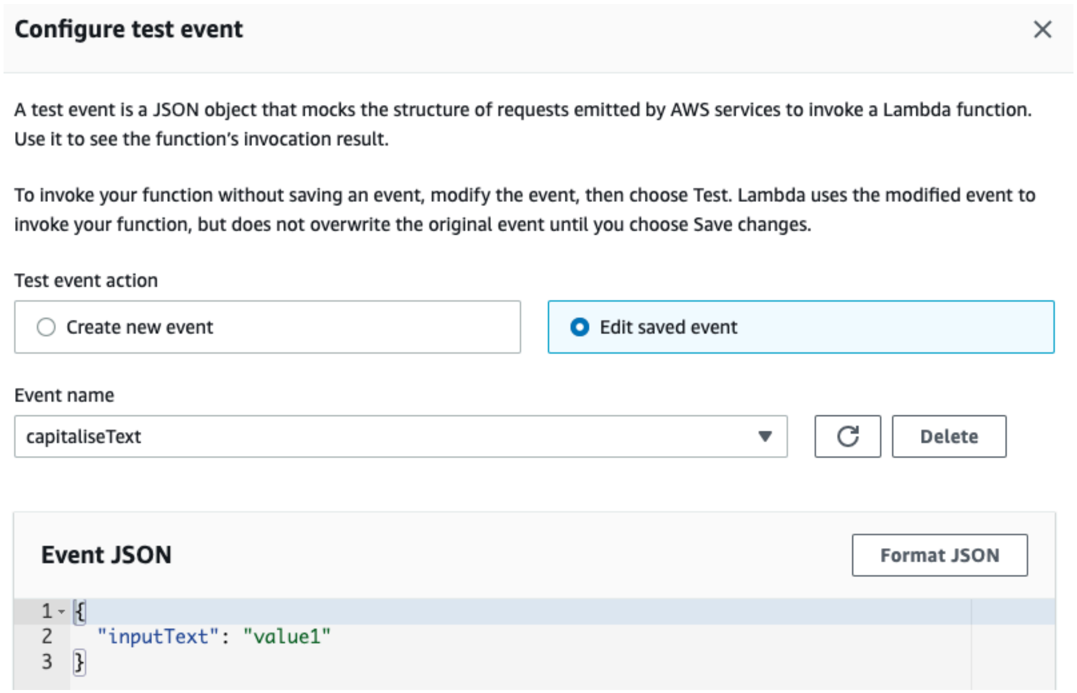


```
import json

def lambda_handler(event, context):
    input_text = str(event['inputText'])
    capitalised_input_text = input_text.upper()
    return {"inputText": capitalised_input_text}
```

4. 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}
```



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.

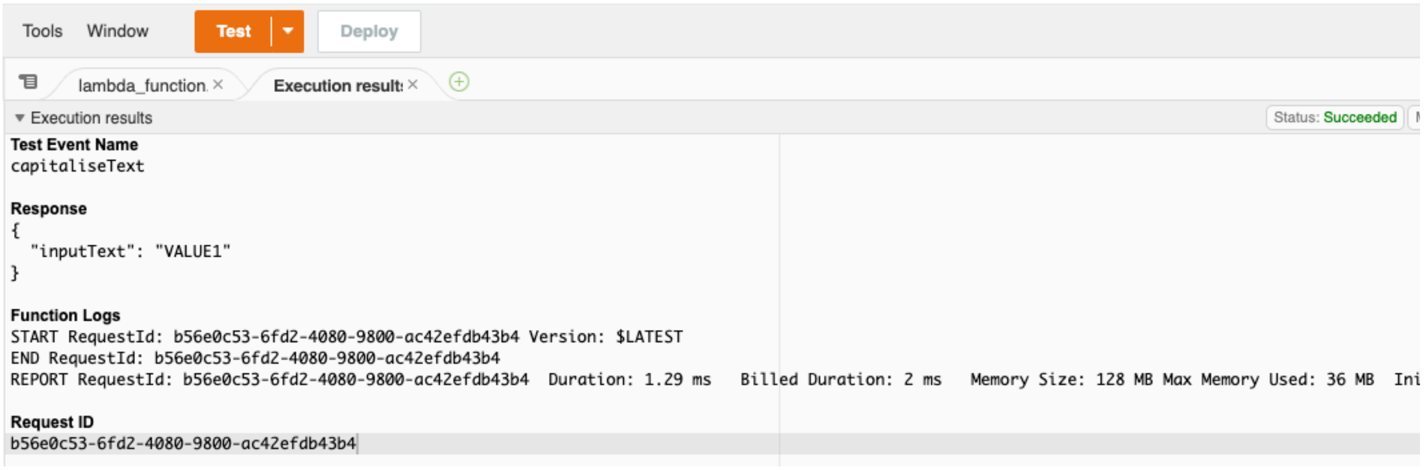
To invoke your function without saving an event, modify the event, then choose Test. Lambda uses the modified event to invoke your function, but does not overwrite the original event until you choose Save changes.

Test event action: Create new event [✓ Edit saved event]

Event name: capitaliseText

```
{
  "inputText": "value1"
}
```

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



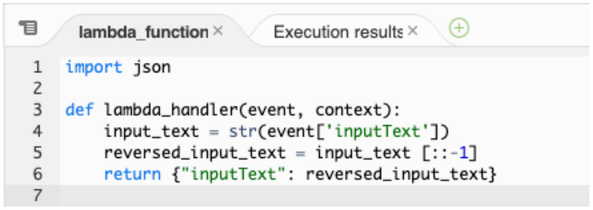
```
Test Event Name
capitaliseText

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  Ini

Request ID
b56e0c53-6fd2-4080-9800-ac42efdb43b4
```

6. Similarly, you create the reverse function.

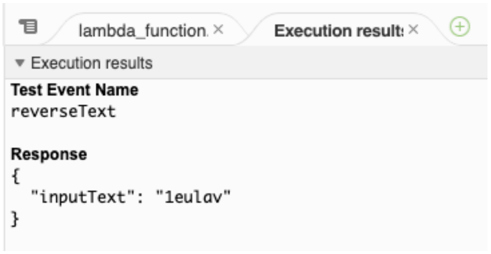


```python
import json

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

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

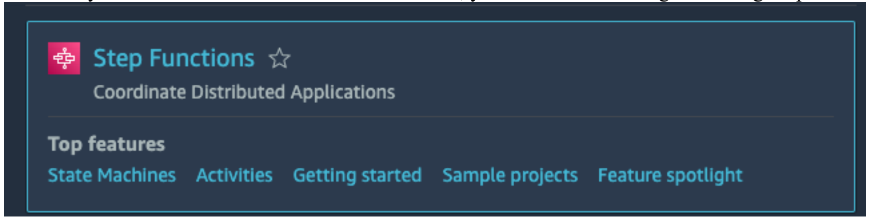
7. Deploy and test the function.



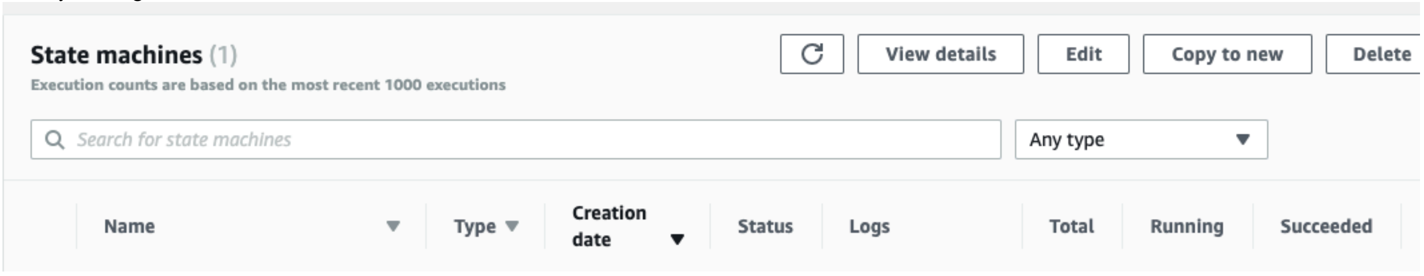
```
Test Event Name
reverseText

Response
{
  "inputText": "1eulav"
}
```

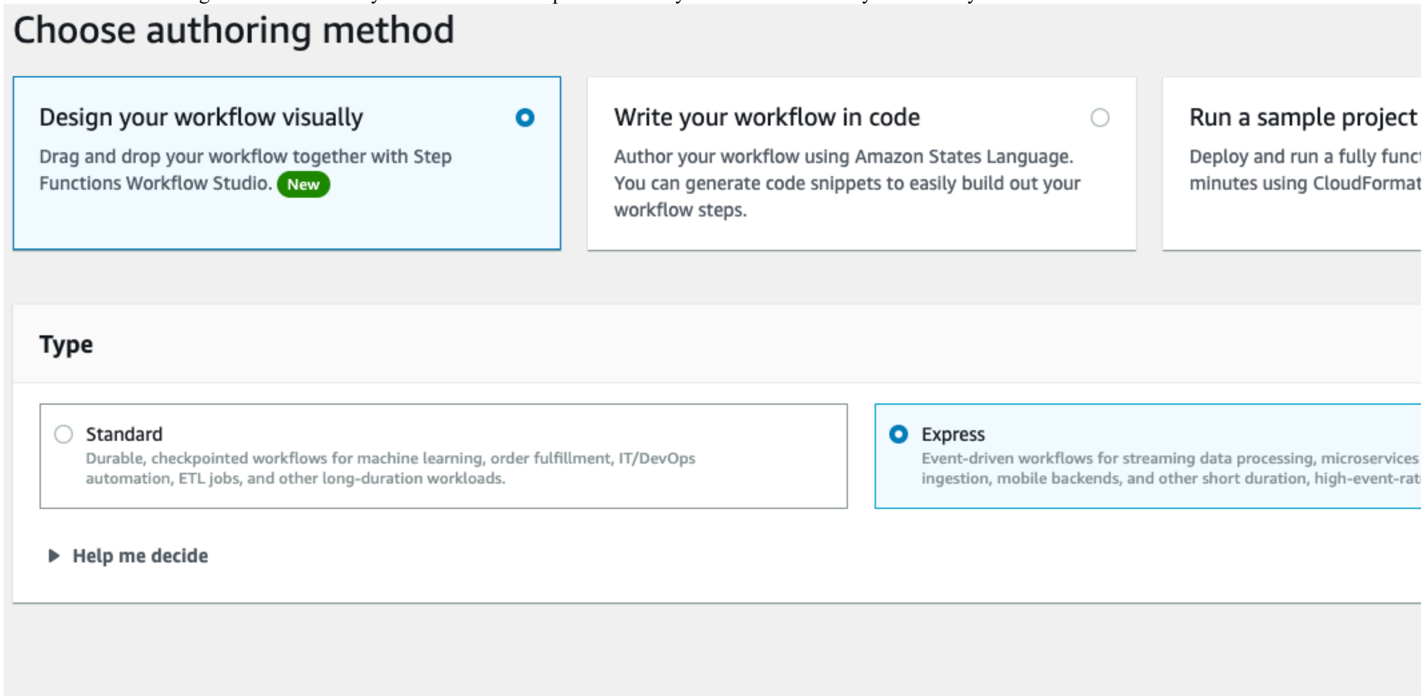
8. Now that you have two functions defined and created, you can chain them together using StepFunctions.



9. Start by creating a state machine.



10. 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.

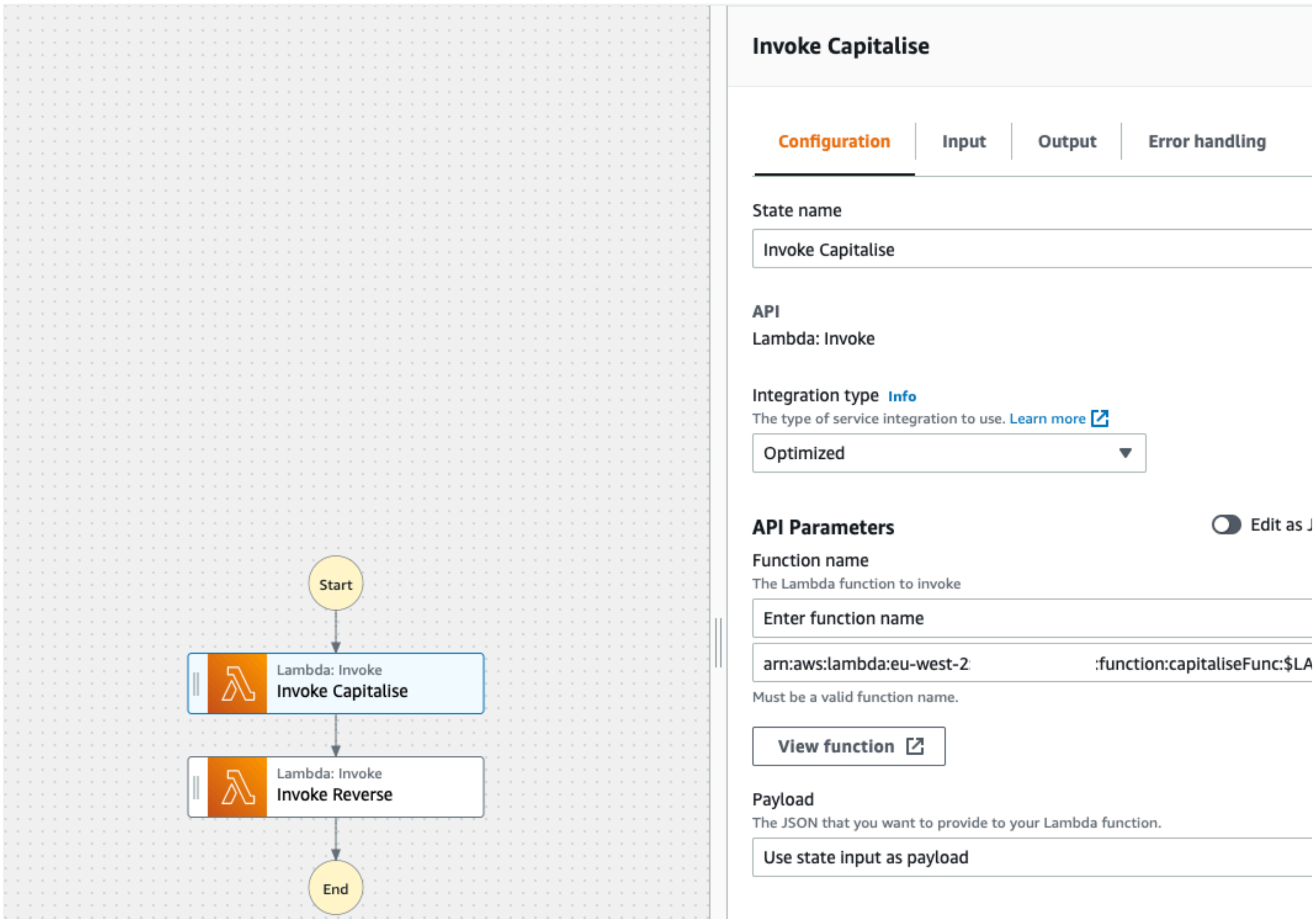
Run a sample project
Deploy and run a fully func
minutes using CloudFormat

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
ingestion, mobile backends, and other short duration, high-event-rat

Help me decide



Invoke Capitalise

Configuration | Input | Output | Error handling

State name: Invoke Capitalise

API
Lambda: Invoke

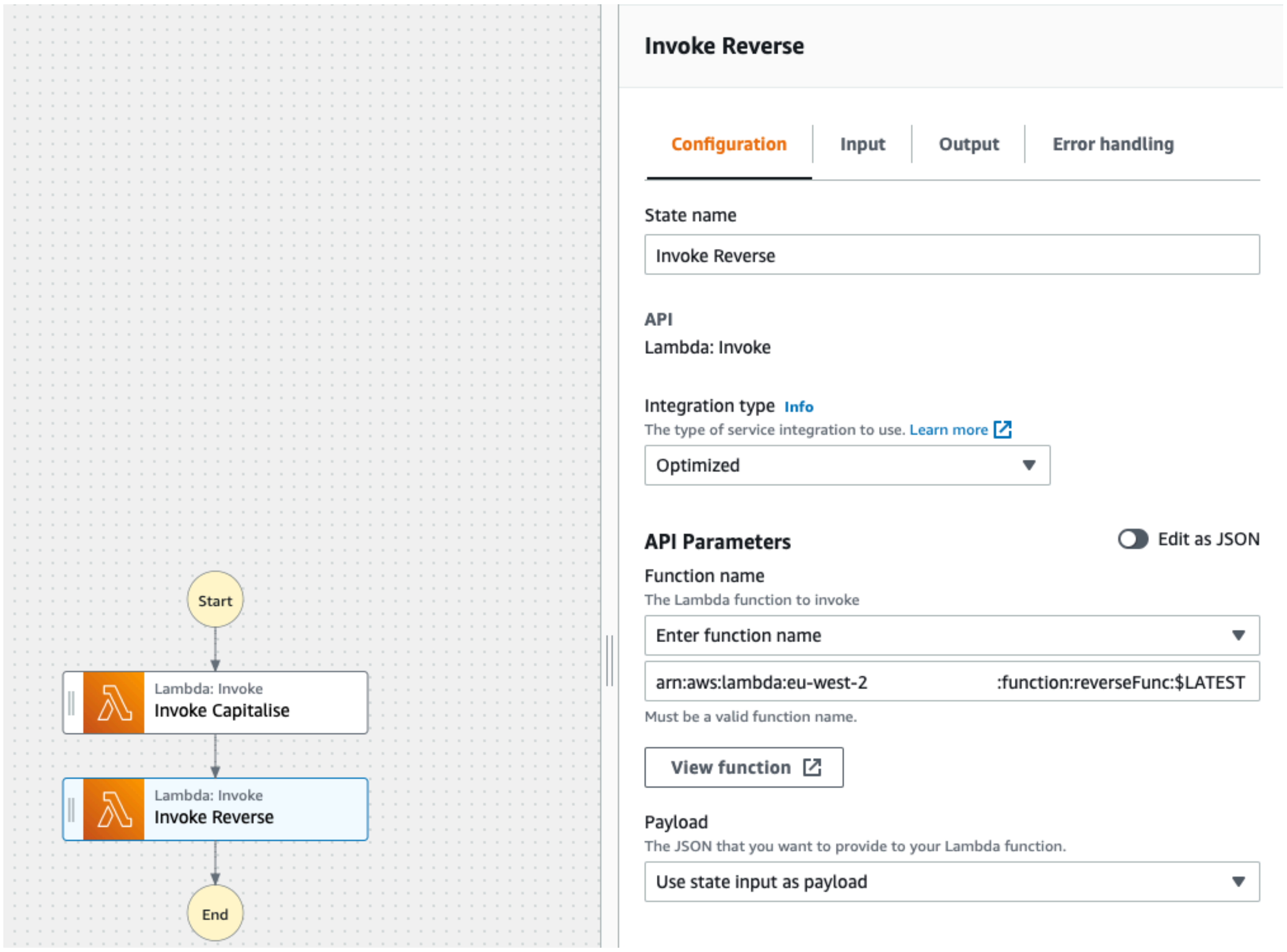
Integration type Info
The type of service integration to use. Learn more
Optimized

API Parameters — Edit as J

Function name
The Lambda function to invoke
Enter function name
arn:aws:lambda:eu-west-2: :function:capitaliseFunc:$LA
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

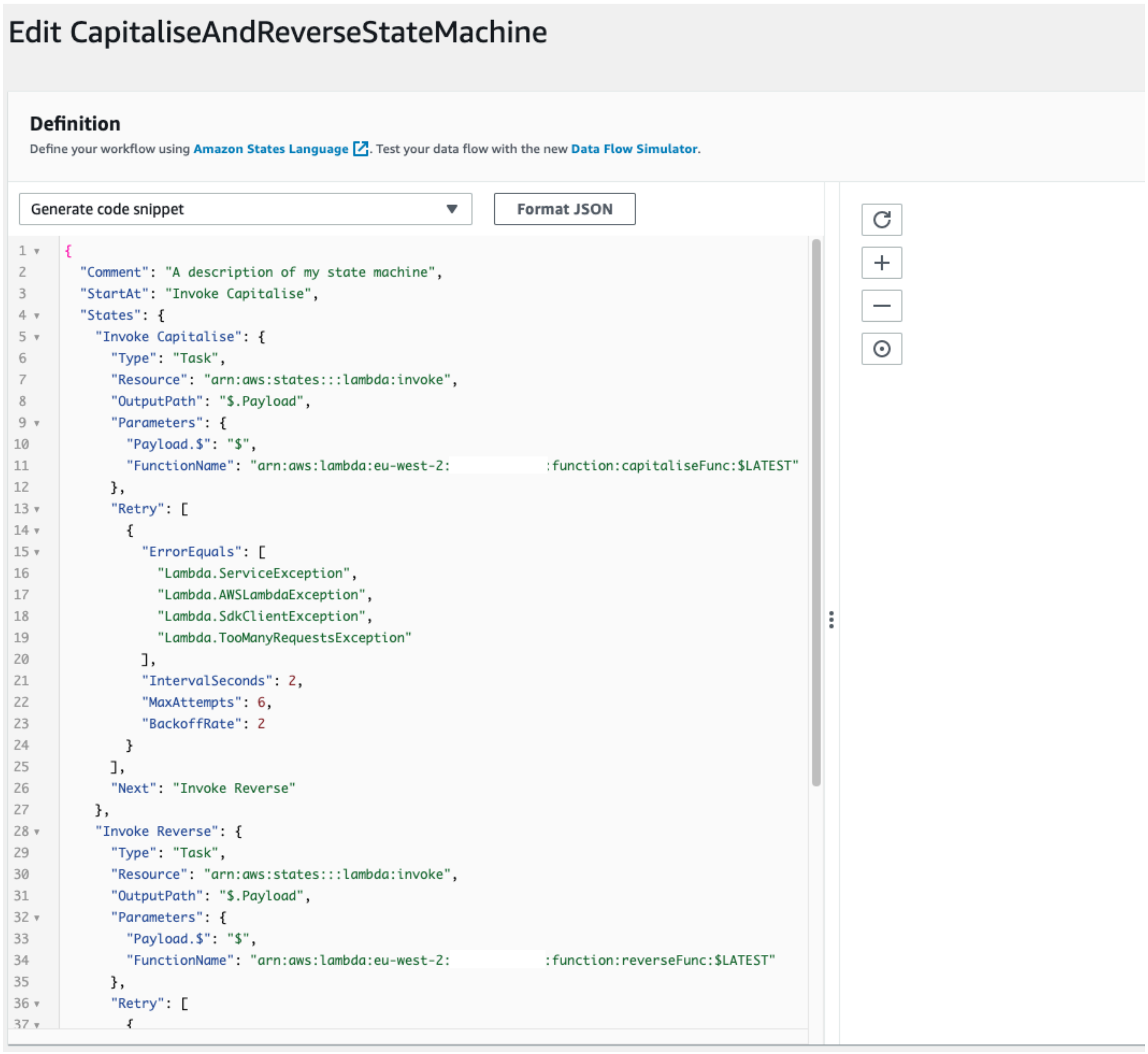
Must be a valid function name.

View function

Payload

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

Use state input as payload



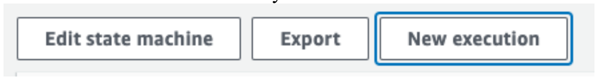
Edit CapitaliseAndReverseStateMachine

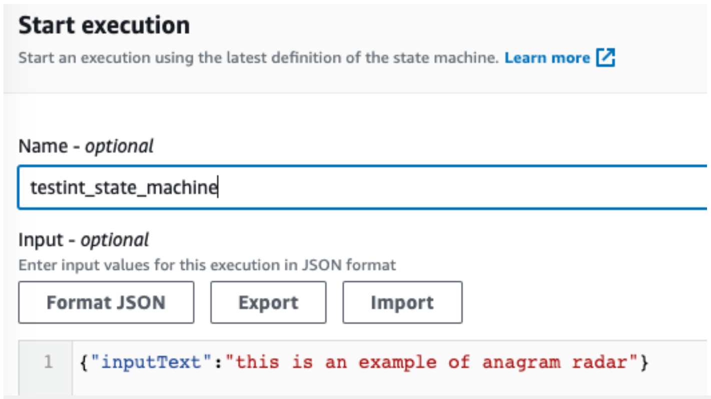
Definition

Define your workflow using **Amazon States Language**. Test your data flow with the new **Data Flow Simulator**.

```
{
  "Comment": "A description of my state machine",
  "StartAt": "Invoke Capitalise",
  "States": {
    "Invoke Capitalise": {
      "Type": "Task",
      "Resource": "arn:aws:states:::lambda:invoke",
      "OutputPath": "$.Payload",
      "Parameters": {
        "Payload.$": "$",
        "FunctionName": "arn:aws:lambda:eu-west-2:[redacted]:function:capitaliseFunc:$LATEST"
      },
      "Retry": [
        {
          "ErrorEquals": [
            "Lambda.ServiceException",
            "Lambda.AWSLambdaException",
            "Lambda.SdkClientException",
            "Lambda.TooManyRequestsException"
          ],
          "IntervalSeconds": 2,
          "MaxAttempts": 6,
          "BackoffRate": 2
        }
      ],
      "Next": "Invoke Reverse"
    },
    "Invoke Reverse": {
      "Type": "Task",
      "Resource": "arn:aws:states:::lambda:invoke",
      "OutputPath": "$.Payload",
      "Parameters": {
        "Payload.$": "$",
        "FunctionName": "arn:aws:lambda:eu-west-2:[redacted]:function:reverseFunc:$LATEST"
      },
      "Retry": [
        {
```

11. 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*

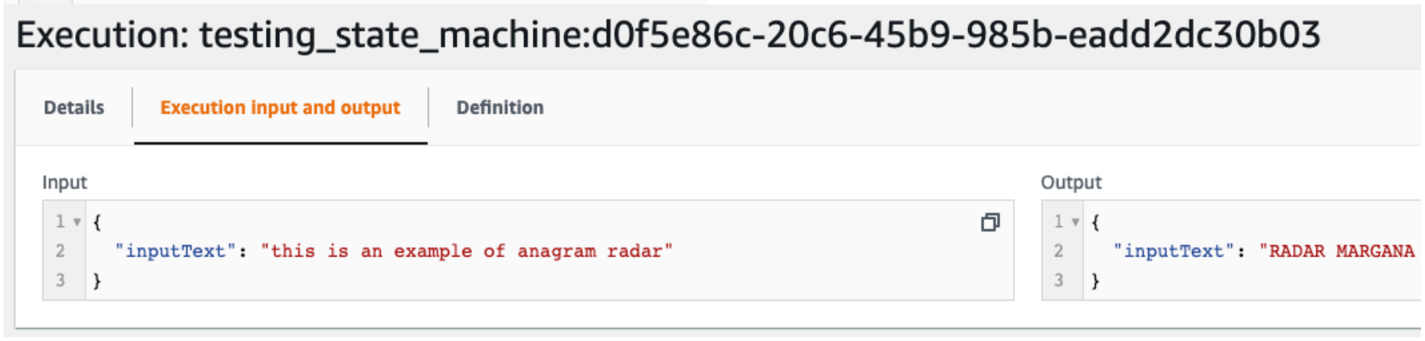
testint_state_machine

Input - *optional*

Enter input values for this execution in JSON format

Format JSON | Export | Import

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



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

Details | Execution input and output | Definition

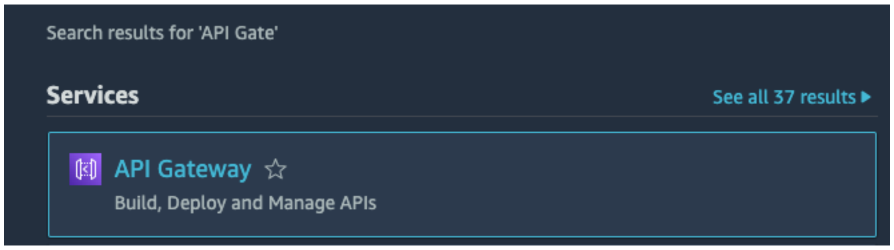
Input

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

Output

```
{
  "inputText": "RADAR MARGANA
}
```

Create API Gateway

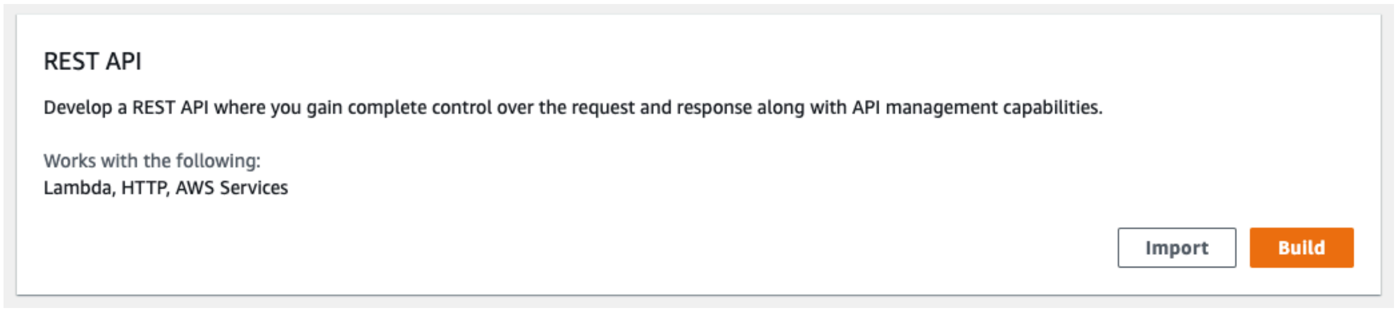


Search results for 'API Gate'

Services — See all 37 results

API Gateway

Build, Deploy and Manage APIs

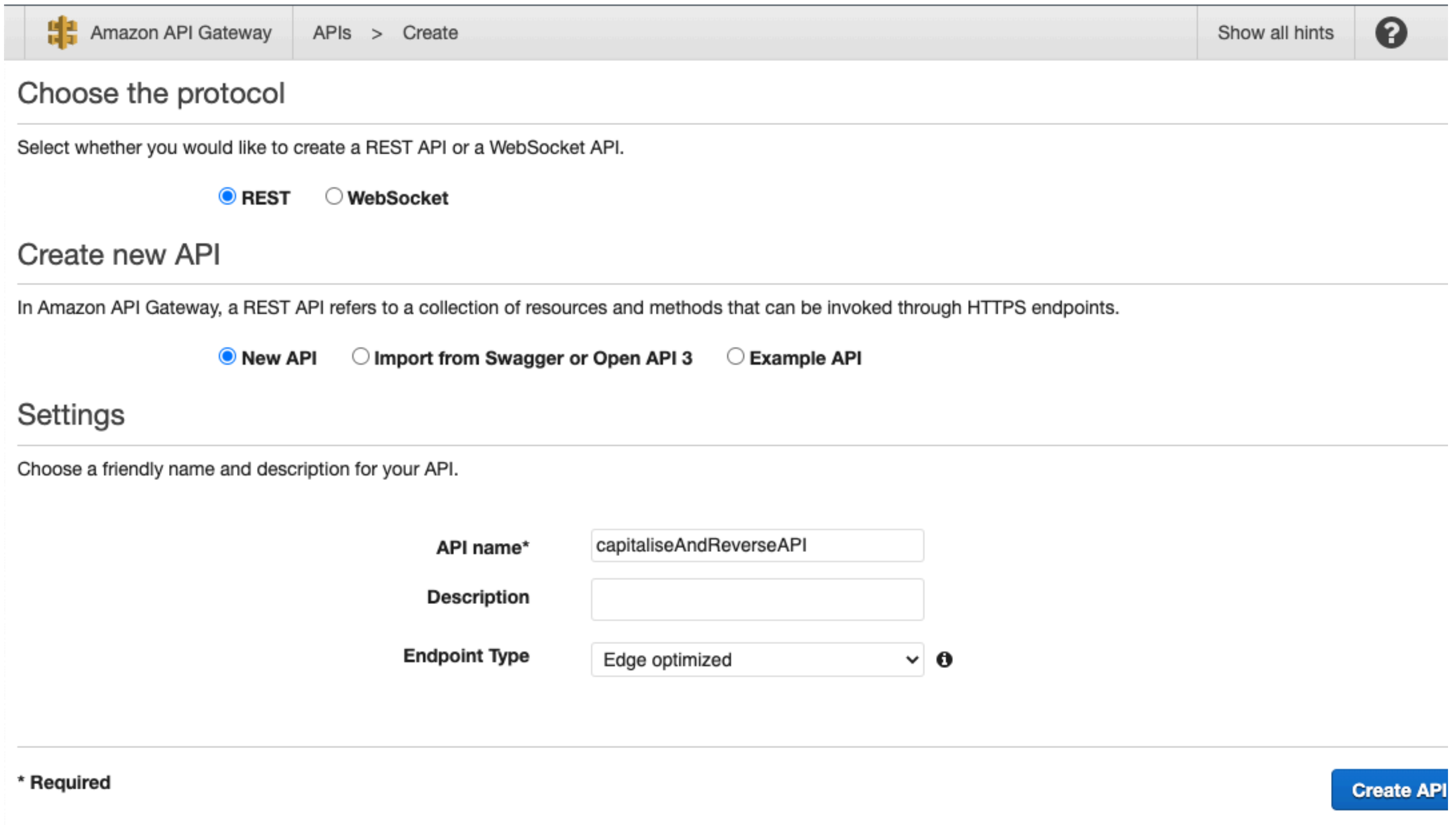


REST API

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



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.

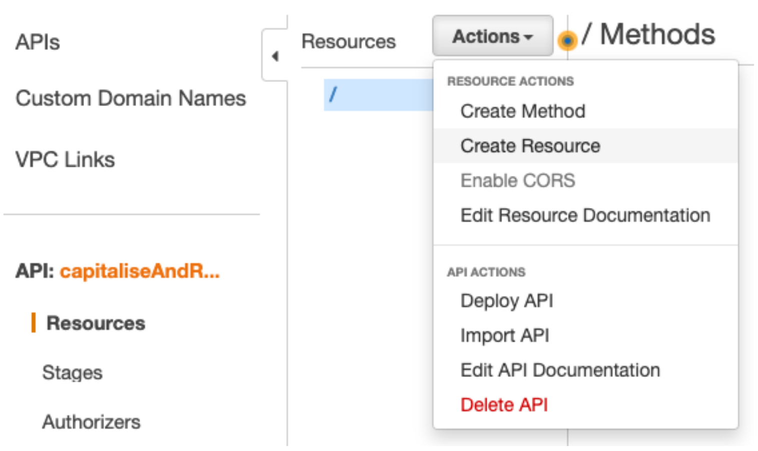
[✓ New API] Import from Swagger or Open API 3 Example API

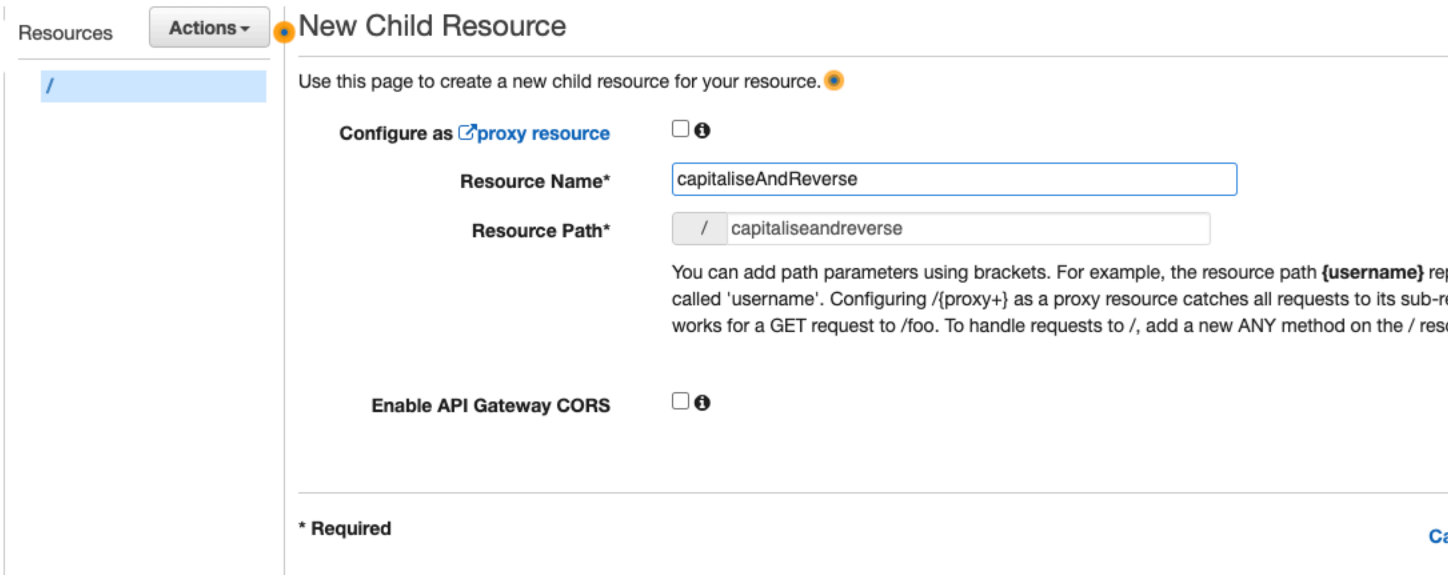
Settings

Choose a friendly name and description for your API.

API name*	capitaliseAndReverseAPI
Description	
Endpoint Type	Edge optimized

* Required





New Child Resource

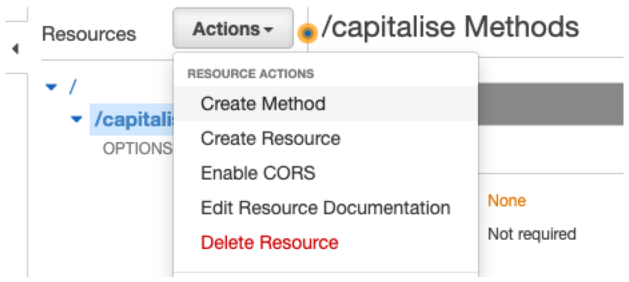
Use this page to create a new child resource for your resource.

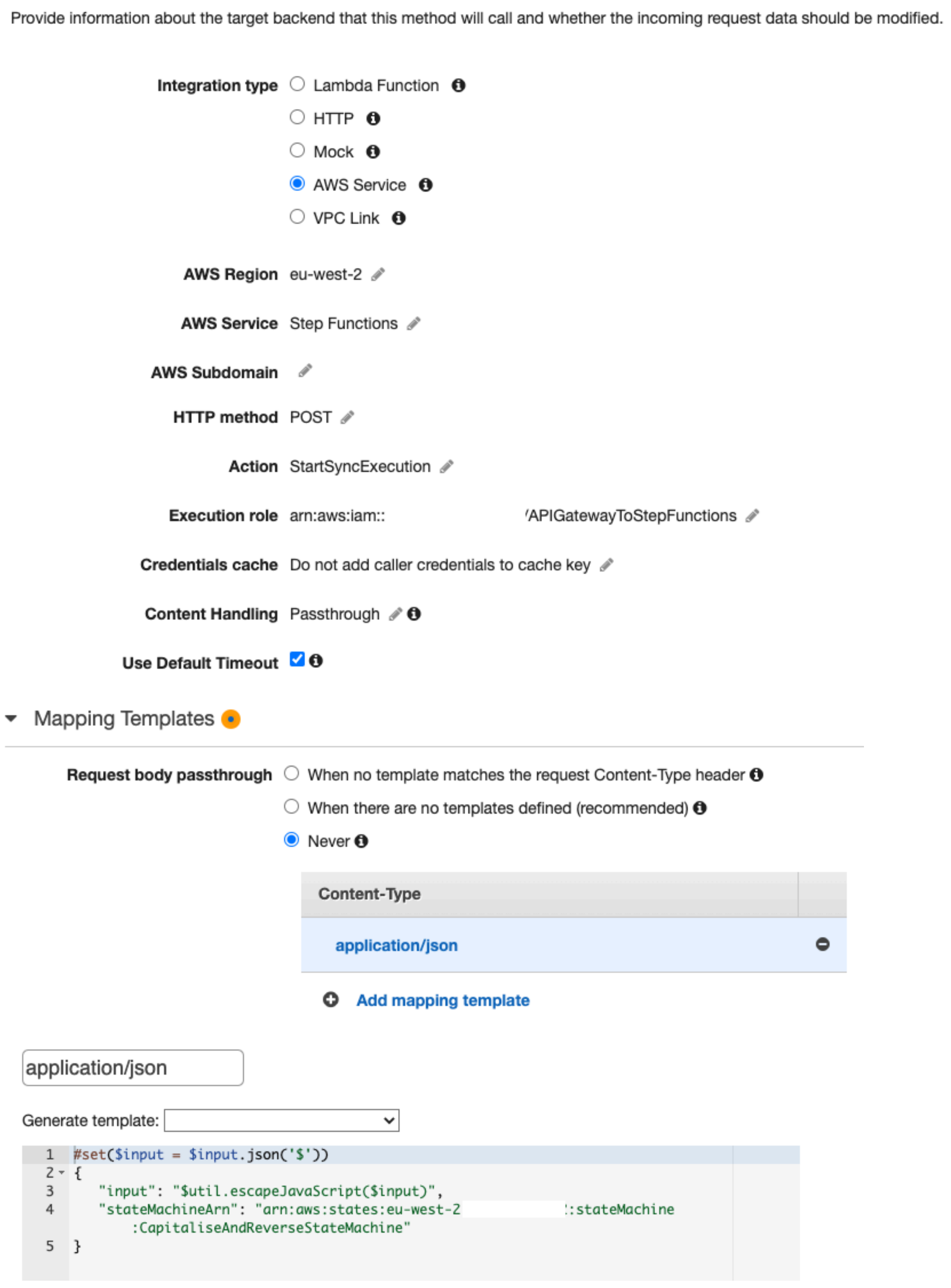
Configure as proxy resource	☐
Resource Name*	capitaliseAndReverse
Resource Path*	/ capitaliseandreverse

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

Enable API Gateway CORS ☐

* 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

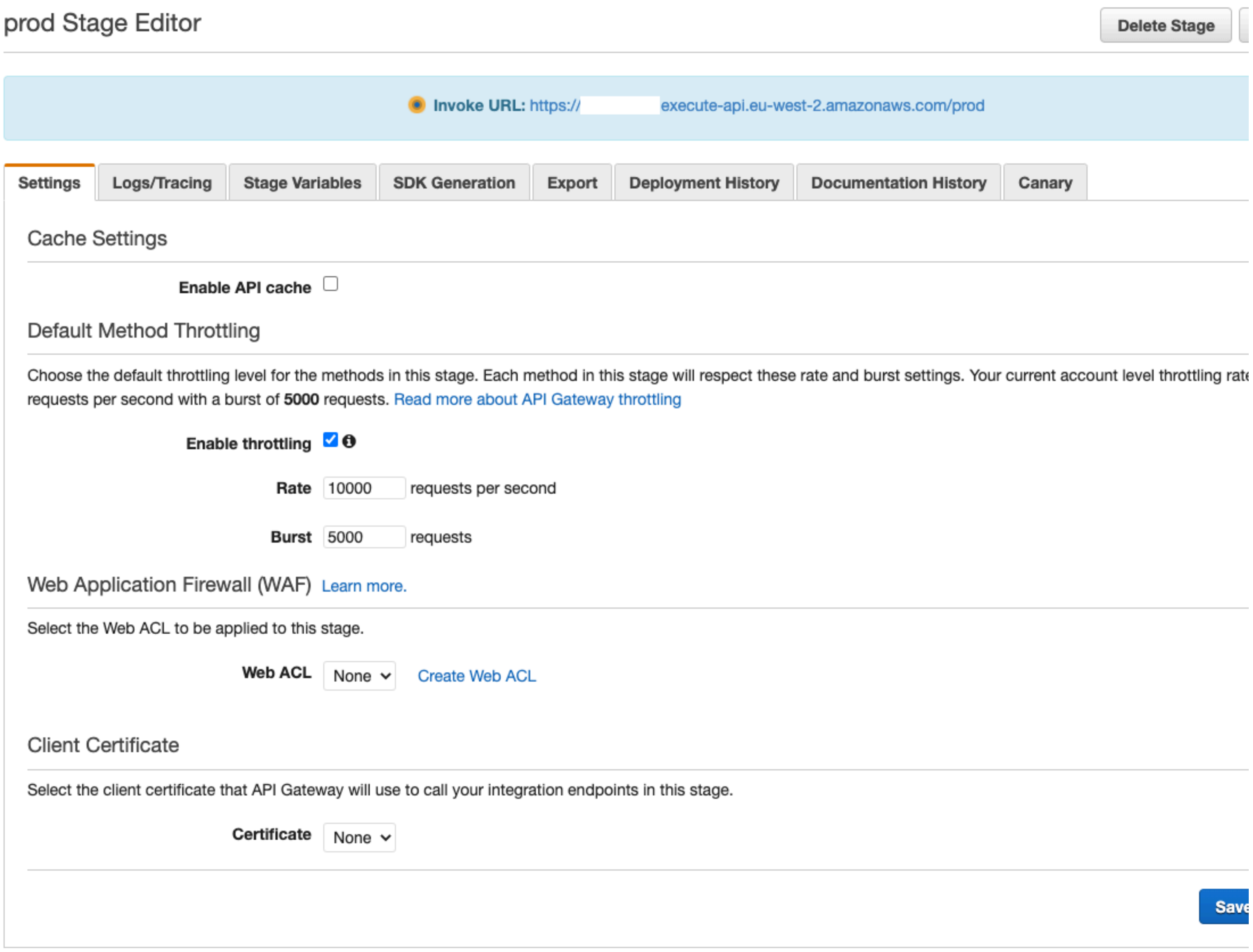
Add mapping template

application/json

Generate template:

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

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.



prod Stage Editor

Invoke URL: https:// execute-api.eu-west-2.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
requests per second with a burst of **5000** requests. Read more about API Gateway throttling

Enable throttling ✓

Rate 10000 requests per second

Burst 5000 requests

Web Application Firewall (WAF) Learn more.

Select the Web ACL to be applied to this stage.

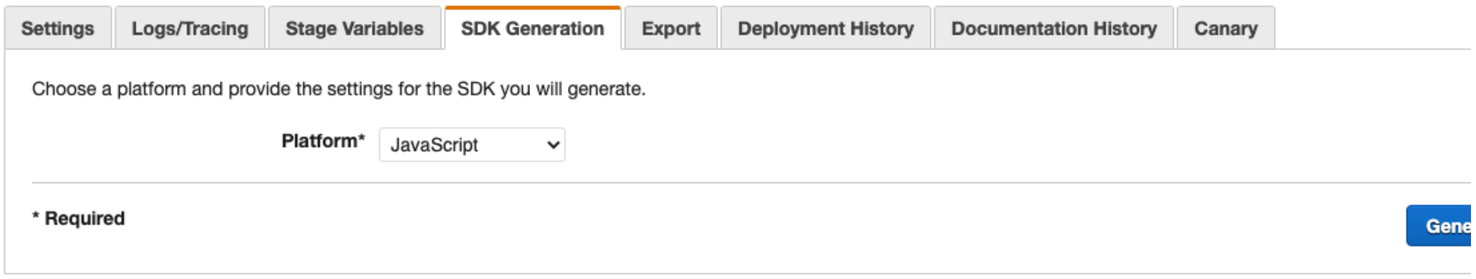
Web ACL None Create Web ACL

Client Certificate

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

Certificate None

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

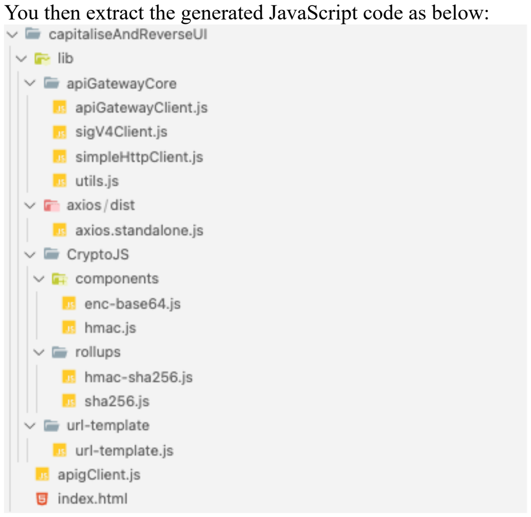


Choose a platform and provide the settings for the SDK you will generate.

Platform* JavaScript

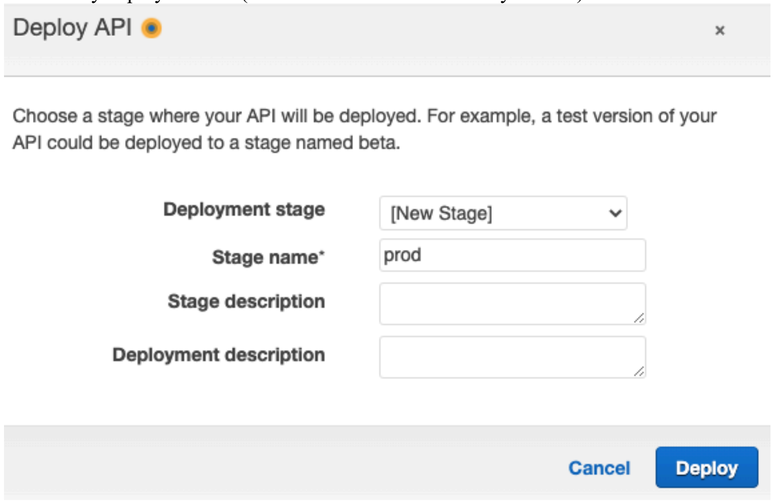
* Required

You then extract the generated JavaScript code as below:



```
capitaliseAndReverseUI
  lib
    apiGatewayCore
      apiGatewayClient.js
      sigV4Client.js
      simpleHttpClient.js
      utils.js
    axios/dist
      axios.standalone.js
    CryptoJS
      components
        enc-base64.js
        hmac.js
      rollups
        hmac-sha256.js
        sha256.js
    url-template
      url-template.js
  apigClient.js
  index.html
```

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	[New Stage]
Stage name*	prod
Stage description	
Deployment description	

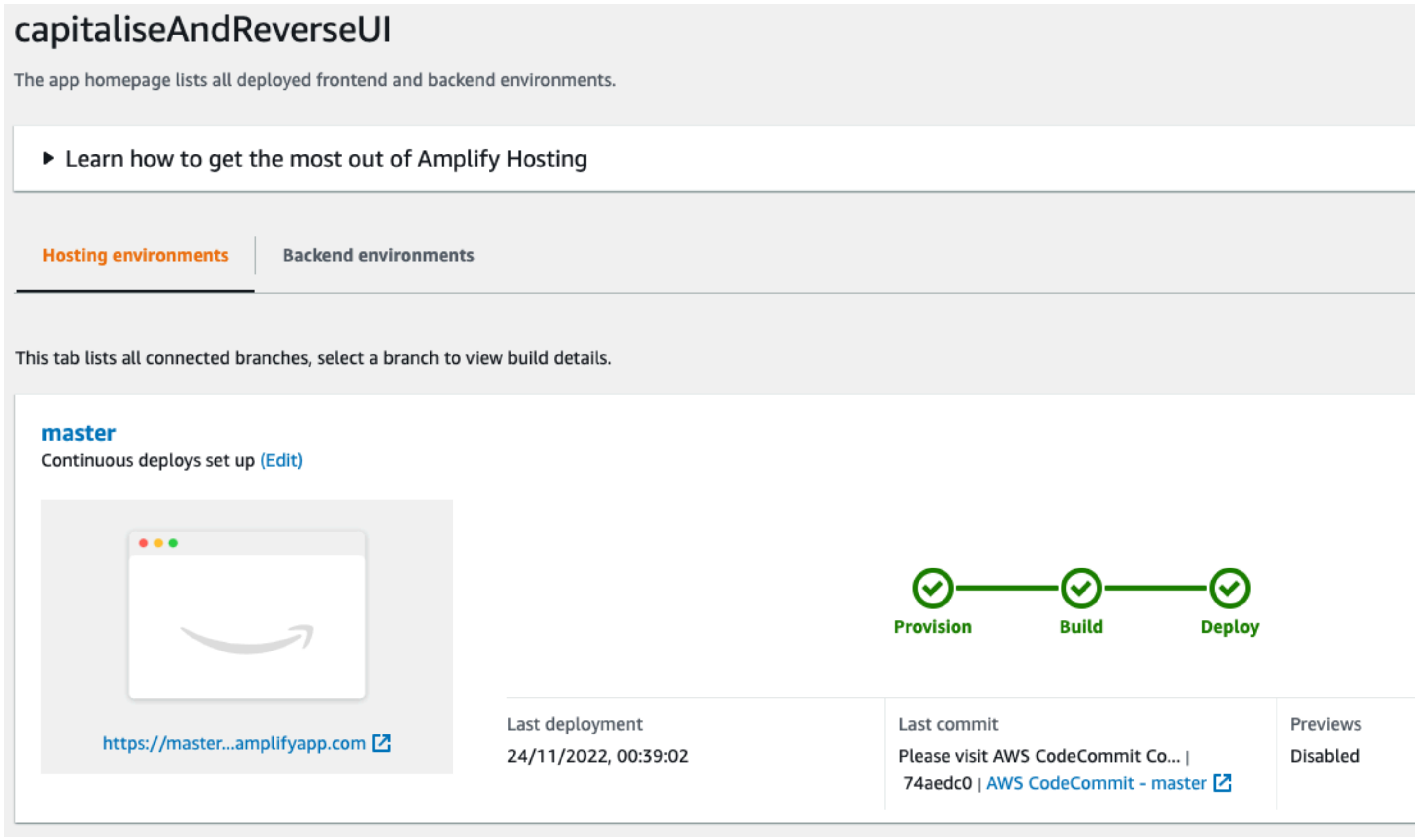
Finalize Front-End

Your final HTML will looks 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/utils.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

The app homepage lists all deployed frontend and backend environments.

▶ Learn how to get the most out of Amplify Hosting

Hosting environments | Backend environments

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

master

Continuous deploys set up (Edit)

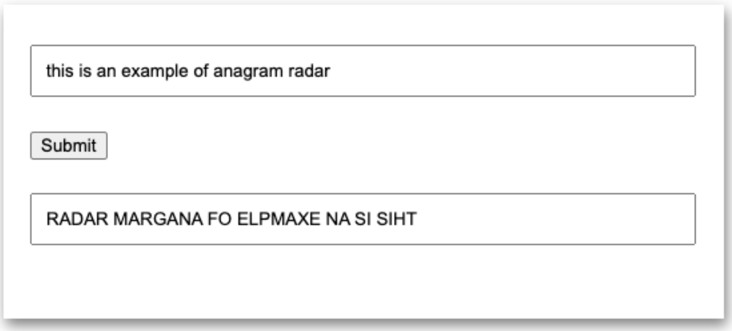
https://master...amplifyapp.com

Last deployment: 24/11/2022, 00:39:02

Last commit: Please visit AWS CodeCommit Co... | 74aedc0 | AWS CodeCommit - master

Previews: 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

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.

Skills Network