



CI/CD using GitHub Action

Deploy Lambda Functions from GitHub Actions

What is CI/CD?

CI/CD (Continuous Integration/Continuous Delivery) is a methodology that streamlines software development through collaboration and automation and is a critical component of implementing DevOps.



Continuous Integration

Continuous Integration is a practice where development teams frequently commit application code changes to a shared repository. These changes automatically trigger new builds which are then validated by automated testing to ensure that they do not break any functionality.



Continuous Delivery

Continuous Delivery is an extension of that process. It's the automation of the release process so that new code is deployed to target environments - typically to test or staging environments - in a repeatable and automated fashion.

→ [Learn more](#)



Continuous Deployment

CD is also used to describe Continuous Deployment which focuses on the automation process to release what is now a fully functional build into production.

src : <https://www.docker.com/solutions/cicd>



What are GitHub Actions ?

- ❑ GitHub Actions makes it easy to automate all your software workflows, now with world-class CI/CD.
- ❑ Build, test, and deploy your code right from GitHub.
- ❑ Make code reviews, branch management, and issue triaging work the way you want.

Recommended Github Actions tutorial - <https://lab.github.com/github/hello-github-actions!>



What is AWS CloudFormation ?

The easiest way to describe what CloudFormation is that it is a tool from AWS that allows you to spin up resources effortlessly. You define all the resources you want AWS to spin up in a blueprint document, click a button, and then AWS magically creates it all. This blueprint is called a template in CloudFormation speak.

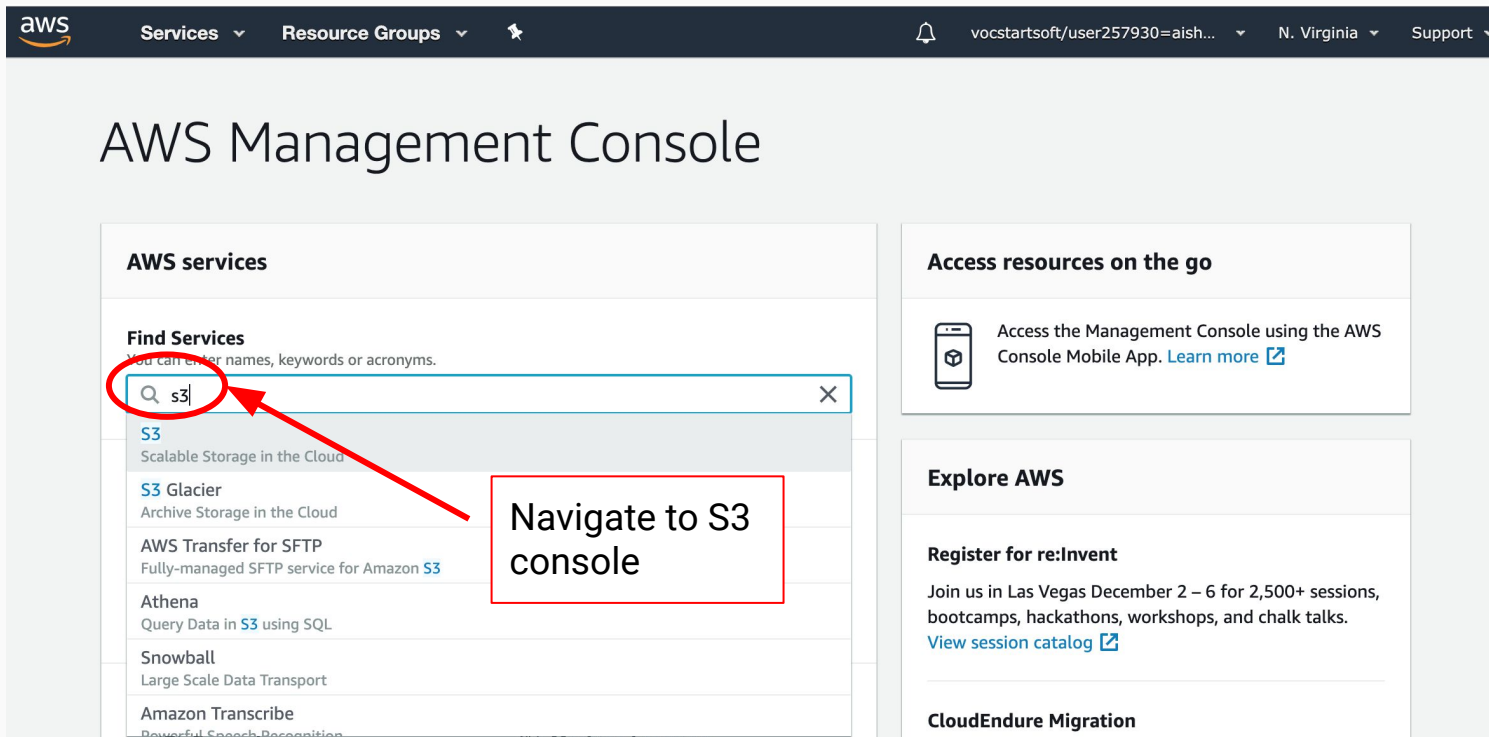
So instead of having to write script with a bunch of AWS API calls, wait loops, and retry logic, you just tell describe what you want and tell CloudFormation to do it for you. Beautiful.

Ref : <https://aws.amazon.com/cloudformation/>

Prerequisite to Deploy your Lambda Function

Prerequisite

1. Create an AWS S3 bucket with permissions to access objects publically.



The screenshot displays the AWS Management Console interface. At the top, the navigation bar includes the AWS logo, 'Services', 'Resource Groups', and a user profile section with a notification bell, the text 'vocstartsoft/user257930=aish...', the region 'N. Virginia', and a 'Support' link. The main heading 'AWS Management Console' is centered. On the left, the 'AWS services' section features a 'Find Services' search bar. A red circle highlights the search bar, which contains the text 's3'. A red arrow points from a text box to the search bar. The search results list 'S3 Scalable Storage in the Cloud' as the top result, followed by 'S3 Glacier Archive Storage in the Cloud', 'AWS Transfer for SFTP Fully-managed SFTP service for Amazon S3', 'Athena Query Data in S3 using SQL', 'Snowball Large Scale Data Transport', and 'Amazon Transcribe Powerful Speech Recognition'. On the right, the 'Access resources on the go' section promotes the AWS Console Mobile App. Below it, the 'Explore AWS' section includes a 'Register for re:Invent' announcement for December 2-6 in Las Vegas, with a link to the 'View session catalog', and a 'CloudEndure Migration' link.

AWS services

Find Services
You can enter names, keywords or acronyms.

S3
Scalable Storage in the Cloud

S3 Glacier
Archive Storage in the Cloud

AWS Transfer for SFTP
Fully-managed SFTP service for Amazon **S3**

Athena
Query Data in **S3** using SQL

Snowball
Large Scale Data Transport

Amazon Transcribe
Powerful Speech Recognition

Access resources on the go

Access the Management Console using the AWS Console Mobile App. [Learn more](#)

Explore AWS

Register for re:Invent
Join us in Las Vegas December 2 – 6 for 2,500+ sessions, bootcamps, hackathons, workshops, and chalk talks. [View session catalog](#)

CloudEndure Migration

Navigate to S3 console

Prerequisite [Contd.]

1. Create an AWS S3 bucket with permissions to access objects publically.

The screenshot shows the AWS Management Console interface for Amazon S3. The left sidebar contains navigation links for Amazon S3, Buckets, Batch operations, and Block public access (account settings). The main content area is titled 'S3 buckets' and includes a search bar, a dropdown for 'All access types', and a summary showing '5 Buckets' and '1 Regions'. A red circle highlights the '+ Create bucket' button, and a red arrow points from this button to a text box that says 'Click on create bucket'.

Amazon S3

Buckets

Batch operations

Block public access (account settings)

Amazon S3 Block Public Access lets you to enforce a no public access policy for your accounts & buckets. [Learn more »](#)
[Documentation](#)

S3 buckets

Discover the console

Search for buckets

All access types

+ Create bucket

Edit public access settings

Empty

Delete

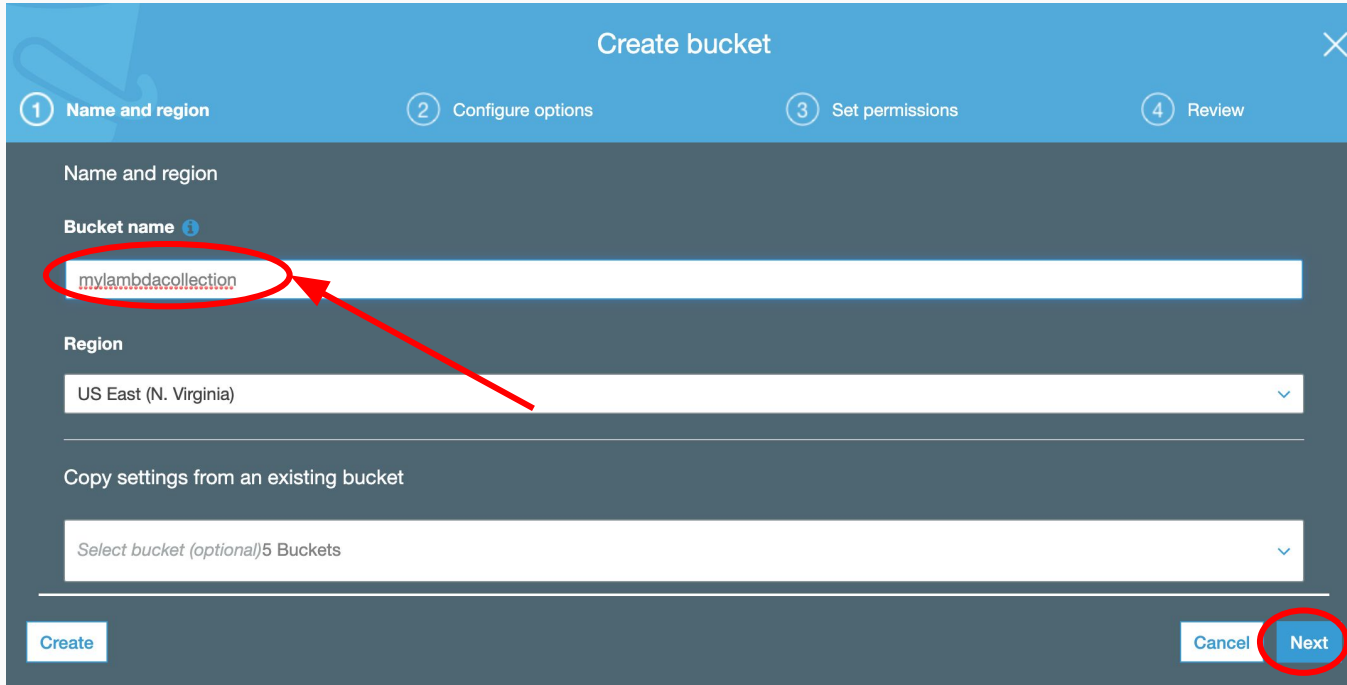
5 Buckets

1 Regions

Click on create bucket

Prerequisite [Contd.]

1. Create an AWS S3 bucket with permissions to access objects publically.



Create bucket

1 Name and region 2 Configure options 3 Set permissions 4 Review

Name and region

Bucket name ⓘ

mylambdacollection

Region

US East (N. Virginia)

Copy settings from an existing bucket

Select bucket (optional) 5 Buckets

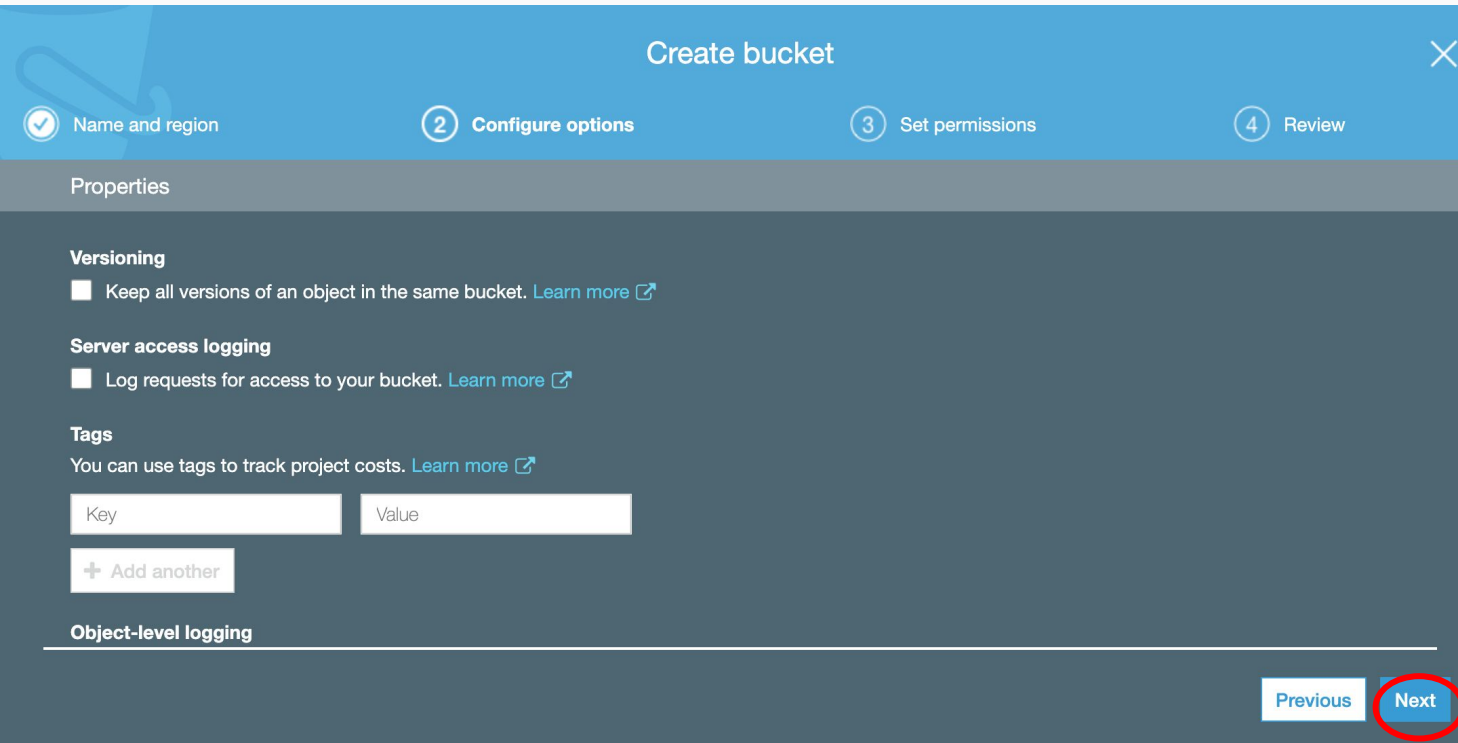
Create Cancel Next

Enter a DNS
complaint name for
your bucket

Click Next

Prerequisite [Contd.]

1. Create an AWS S3 bucket with permissions to access objects publically.



The screenshot shows the 'Create bucket' wizard in the AWS Management Console. The title bar is blue with the text 'Create bucket' and a close button. Below the title bar is a progress bar with four steps: 1. Name and region (checked), 2. Configure options (active), 3. Set permissions, and 4. Review. The main content area is dark grey and contains the following sections:

- Versioning**: A checkbox labeled 'Keep all versions of an object in the same bucket.' with a 'Learn more' link and an external link icon.
- Server access logging**: A checkbox labeled 'Log requests for access to your bucket.' with a 'Learn more' link and an external link icon.
- Tags**: A heading followed by the text 'You can use tags to track project costs.' with a 'Learn more' link and an external link icon. Below this are two input fields labeled 'Key' and 'Value', and a button labeled '+ Add another'.
- Object-level logging**: A heading at the bottom of the main content area.

At the bottom right of the wizard, there are two buttons: 'Previous' and 'Next'. The 'Next' button is highlighted with a red circle.

For the purpose of this exercise, we do not require additional configuration.

Click Next

Prerequisite [Contd.]

1. Create an AWS S3 bucket with permissions to access objects publically.

Create bucket

✓ Name and region

✓ Configure options

3 Set permissions

4 Review

☐ Block *all* public access

Turning this setting on is the same as turning on all four settings below. Each of the following settings are independent of one another.

☐ Block public access to buckets and objects granted through *new* access control lists (ACLs)

S3 will block public access permissions applied to newly added buckets or objects, and prevent the creation of new public access ACLs for existing buckets and objects. This setting doesn't change any existing permissions that allow public access to S3 resources using ACLs.

☐ Block public access to buckets and objects granted through *any* access control lists (ACLs)

S3 will ignore all ACLs that grant public access to buckets and objects.

☐ Block public access to buckets and objects granted through *new* public bucket policies

S3 will block new bucket policies that grant public access to buckets and objects. This setting doesn't change any existing policies that allow public access to S3 resources.

☒ Block public and cross-account access to buckets and objects through *any* public bucket policies

S3 will ignore public and cross-account access for buckets with policies that grant public access to buckets and objects.

Previous

Next

Set your bucket configuration as shown

Click Next

Prerequisite [Contd.]

1. Create an AWS S3 bucket with permissions to access objects publically.

Create bucket

✓ Name and region

✓ Configure options

✓ Set permissions

4 Review

Name and region

Edit

Bucket name mylambdacollection **Region** US East (N. Virginia)

Options

Edit

Versioning	Disabled
Server access logging	Disabled
Tagging	0 Tags
Object-level logging	Disabled
Default encryption	None
CloudWatch request metrics	Disabled
Object lock	Disabled

Previous

Create bucket

Review your changes and click on create bucket

Prerequisite [Contd.]

1. Create an AWS S3 bucket with permissions to access objects publically.

S3 buckets

 [Discover the console](#)

 Search for buckets

All access types 

 Create bucket

Edit public access settings



Empty

Delete

6 Buckets

1 Regions



<input type="checkbox"/>	Bucket name ▾	Access  ▾	Region ▾	Date created ▾
<input checked="" type="checkbox"/>	 mylambdacollection	Objects can be public	US East (N. Virginia)	Oct 1, 2019 11:17:51 AM GMT+0800

Your S3 bucket with publicly accessible object has been created.

Use this bucket name while setting up the configuration details of the project.

Steps to Deploy your Lambda Function

Steps to Deploy your Lambda Function

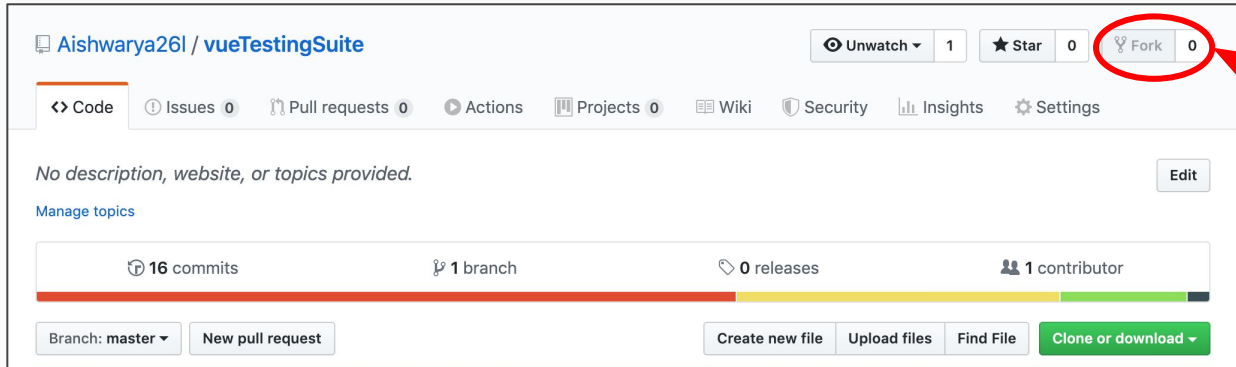
Repository link : <https://github.com/Aishwarya26l/vueTestingSuite>

Step 1:

- **Fork this repository** : <https://github.com/Aishwarya26l/vueTestingSuite>
- **Navigate to your Forked repository**
- **Click on the Actions tab and “Enable” Github actions**

Note :

Navigate to Actions tab on the Github console and click on “Enable” Actions on your forked repository



Click here to fork the repository

Steps to Deploy your Lambda Function

Step 2:

Navigate to your Github console. Click on settings and then click on “Secrets”

A screenshot of the GitHub repository 'Aishwarya26I / vueTestingSuite'. The 'Settings' tab is selected, and the 'Secrets' section is visible. The 'Secrets' section lists four secrets: AWS_ACCESS_KEY_ID, AWS_SECRET_ACCESS_KEY, AWS_SESSION_TOKEN, and BUCKET_NAME. A red circle highlights the first three secrets. The 'Secrets' section also includes a description: 'Secrets are environment variables that are encrypted and only exposed to selected actions.' and a link to 'Add a new secret'.

Add the following secrets -

- **AWS_ACCESS_KEY_ID**
- **AWS_SECRET_ACCESS_KEY**
- **AWS_SESSION_TOKEN**
- **BUCKET_NAME** - The S3 bucket name with publically accessible objects

For AWS Educate users, credentials can be accessed as shown in the following slides

Steps to Deploy your Lambda Function

Welcome to AWS Educate Starter Account

Use your Starter Account to access to a wide variety of AWS Services and start building!
Click on the AWS Console button to sign in and get started.

- [What AWS services can I use in my Starter Account?](#)

You can use the following services in your Starter Account: apigateway, athena, cloud9, cloudformation, cloudfront, cloudtrail, cloudwatch, codecommit, codedeploy, codepipeline, cognito-identity, cognito-idp, cognito-sync, comprehend, deeplens, dynamodb, ec2, ecs, elasticache, elasticfilesystem, elasticloadbalancing, elasticmapreduce, events, execute-api, glue, iam, inspector, iot, kinesis, kinesisanalytics, firehose, kms, lambda, lex, logs, machinelearning, mobilehub, opsworks, polly, rds, rekognition, route53 (other than domain name purchasing), s3, sns, sqs, swf, sagemaker, translate, transcribe

Your Starter Account Status



Active
full access 0



\$57.73
credits (estimated)



150d 23:03:11
remaining term



2:60
session time

[Account Details](#)

[AWS Console](#)

NOTE: For AWS Educate users, user credentials can be found here -

The credentials expire every 3 hours and must be overwritten in the GitHub console before every push for successful auto-deployment.

Steps to Deploy your Lambda Function

```
Credentials
AWS Access
  Session started at: 2019-09-29T23:46:14-0700
  Session to end at: 2019-09-30T02:46:14-0700
  Remaining session time: 1h22m33s
AWS Starter account
  Term: 151 days 17:41:49
AWS CLI:
  Copy and paste the following into ~/.aws/credentials
  [default]
  aws_access_key_id=[REDACTED]
  aws_secret_access_key=[REDACTED]
  aws_session_token=[REDACTED]
```

NOTE:

Copy the values corresponding to the keys required and add them in your Github console.

Steps to Deploy your Lambda Function

Step 3:

Make changes to the code and push your changes to trigger the actions.

1. From the github console, click on the file you would like to modify. Example - src/index.html

The screenshot shows the GitHub repository page for 'Aishwarya261 / vueTestingSuite'. The repository has 1 Unwatch, 0 Stars, and 28 Forks. The 'Code' tab is selected, showing the file list for the 'src/' directory. The file 'index.html' is highlighted with a red circle. The file list includes:

File	Description	Time
..		
bin	Added github action to deploy lambda function	2 days ago
deploy.sh	Added github action to deploy lambda function	2 days ago
executeShellCommand.js	Added github action to deploy lambda function	2 days ago
getContents.js	Added github action to deploy lambda function	2 days ago
index.html	Added github action to deploy lambda function	2 days ago
index.js	Added github action to deploy lambda function	2 days ago
jest.config.js	Added github action to deploy lambda function	2 days ago
package-lock.json	Added github action to deploy lambda function	2 days ago
package.json	Added github action to deploy lambda function	2 days ago
testRunner.js	Added github action to deploy lambda function	2 days ago

Navigate to src/ and click on index.html or any other file you wish to modify.

Steps to Deploy your Lambda Function

Step 3[Contd]:

Aishwarya261 / vueTestingSuite

Unwatch 1

Star 0

Fork 28

Code

Issues 0

Pull requests 1

Actions

Projects 0

Wiki

Security

Insights

Settings

Branch: master

vueTestingSuite / src / index.html

Find file

Copy path



Aishwarya261 Added github action to deploy lambda function

4a1c2e2 2 days ago

1 contributor

343 lines (341 sloc) 15.5 KB

Raw

Blame

History



Edit this file



```
1 <!-- Shown Block - Test - layoutItems[1]
2 Editable Block - JS Source code - layoutItems[0] -->
3 <html>
4 <head>
5   <meta charset="utf-8" />
6   <meta
7     content="width=device-width,initial-scale=1,minimal-ui"
8     name="viewport"
9   />
10  <link
11    rel="shortcut icon"
12    href="data:image/x-icon;base64,AAABAAEABAAAAEAIABoBAAAFgAAACgAAAAQAAAAIAAAAAEIAAAAAAAAAQAAAAAAAAAAAAAAAAAAAA
13  />
14  <link
15    rel="stylesheet"
```

Click on Edit this file

Steps to Deploy your Lambda Function

Step 3[Contd]:

```
335     r
336     .CodeMirror {
337       height: 150px;
338     }
339     .CodeMirror-overlayscroll-vertical {
340       display: block !important;
341     }
342   </style> |
343 </html>
```



Commit changes

Updated index.html

Add an optional extended description...

☒ Commit directly to the `master` branch.

☐ Create a new branch for this commit and start a pull request. [Learn more about pull requests.](#)

Commit changes

Cancel

After modifying the file, scroll down and enter a message for the changes you've made.

Click on commit changes.

Your changes will be directly committed to the master branch. This will trigger the action that is setup.

Steps to Deploy your Lambda Function

Step 4:

Navigate to “Actions” on your Github console to view the deployment status -

 **Aishwarya26l** / **vueTestingSuite**

 Unwatch ▾

1

★ Star

0

 Fork

0

<> Code

! Issues 0

 Pull requests 0

▶ Actions

 Projects 0

 Wiki

 Security

 Insights

 Settings

No description, website, or topics provided.

Edit

[Manage topics](#)

 16 commits

 1 branch

 0 releases

 1 contributor

Aishwarya26I / vueTestingSuite

Unwatch

1

Star

0

Fork

0

Code

Issues0

Pull requests0

Actions

Projects0

Wiki

Security

Insights

Settings

Update README.md

master

ad41f49

Deploy Lambda Function on push

on: push

Test action

GitHub Actions / Test action

successful 19 hours ago in 3m 8s

Search logs

<

>

...

Set up job

2s

Run actions/checkout@master

3s

Run ./

3m 3s

1 ▶ Run ./

13 Dockerfile for action: '/home/runner/work/vueTestingSuite/vueTestingSuite/./Dockerfile'.

14 /usr/bin/docker build -t 04bb81:102267eac6f850e5e2ae674a50bf9349

15 "/home/runner/work/vueTestingSuite/vueTestingSuite"

16 Sending build context to Docker daemon 2.655MB

17 Step 1/11 : FROM python:alpine

18 alpine: Pulling from library/python

19 9d48c3bd43c5: Already exists

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

96

97

98

99

100

101

102

103

104

105

106

107

108

109

110

111

112

113

114

115

116

117

118

119

120

121

122

123

124

125

126

127

128

129

130

131

132

133

134

135

136

137

138

139

140

141

142

143

144

145

146

147

148

149

150

151

152

153

154

155

156

157

158

159

160

161

162

163

164

165

166

167

168

169

170

171

172

173

174

175

176

177

178

179

180

181

182

183

184

185

186

187

188

189

190

191

192

193

194

195

196

197

198

199

200

201

202

203

204

205

206

207

208

209

210

211

212

213

214

215

216

217

218

219

220

221

222

223

224

225

226

227

228

229

230

231

232

233

234

235

236

237

238

239

240

241

242

243

244

245

246

247

248

249

250

251

252

253

254

255

256

257

258

259

260

261

262

263

264

265

266

267

268

269

270

271

272

273

274

275

276

277

278

279

280

281

282

283

284

285

286

287

288

289

290

291

292

293

294

295

296

297

298

299

300

301

302

303

304

305

306

307

308

309

310

311

312

313

314

315

316

317

318

319

320

321

322

323

324

325

326

327

328

329

330

331

332

333

334

335

336

337

338

339

340

341

342

343

344

345

346

347

348

349

350

351

352

353

354

355

356

357

358

359

360

361

362

363

364

365

366

367

368

369

370

371

372

373

374

375

376

377

378

379

380

381

382

383

384

385

386

387

388

389

390

391

392

393

394

395

396

397

398

399

400

401

402

403

404

405

406

407

408

409

410

411

412

413

414

415

416

417

418

419

420

421 "Description": "AWS API Gateway with a Lambda Integration"

422 }

423

424 Uploading to 15b53c421393363d5684842a2658171d 175204 / 175204.0 (100.00%)

425 Successfully packaged artifacts and wrote output template to file packaged.yaml.

426 Execute the following command to deploy the packaged template

427 aws cloudformation deploy --template-file /github/workspace/packaged.yaml --stack-name <YOUR STACK NAME>

428

429 Waiting for changeset to be created..

430 Waiting for stack create/update to complete

431 Successfully created/updated stack - vueTestingSuite

Complete job

0s

Step 4 [Contd]:

Successfully deployed.

You can now view your created lambda function on the AWS console and navigate to its API Gateway to view the page.

Deployed Lambda function

The screenshot displays the AWS Lambda console for a function named 'vueTestingSuite'. The breadcrumb navigation shows 'Lambda > Functions > vueTestingSuite'. The function's ARN is 'arn:aws:lambda:us-east-1:216730220204:function:vueTestingSuite'. The 'Configuration' tab is active, showing the 'Designer' section. A red arrow points to the 'vueTestingSuite' function card, which indicates it has 0 layers. Below the function card, there is a section for 'API Gateway' and a dashed box labeled 'Resources that the function's role has access to appear here'.

aws Services Resource Groups

Lambda > Functions > vueTestingSuite

ARN - arn:aws:lambda:us-east-1:216730220204:function:vueTestingSuite

vueTestingSuite

Throttle Qualifiers Actions Select a test event Test Save

This function belongs to the AWS CloudFormation stack **vueTestingSuite**. [Manage this stack](#) on the CloudFormation console.

Configuration Monitoring

▼ Designer

[Go back to application vueTestingSuite](#)

vueTestingSuite

Layers (0)

API Gateway

Resources that the function's role has access to appear here

Step 4 [Contd]:

Successfully deployed

Lambda Function :
vueTestingSuite

Deployed CloudFormation stack

The screenshot displays the AWS CloudFormation console interface. At the top, the navigation bar includes the AWS logo, 'Services', 'Resource Groups', and user information. The left sidebar shows the 'CloudFormation' section with options like 'Stacks', 'Stack details', 'Drifts', 'StackSets', 'Exports', 'Designer', 'Previous console', and 'Feedback'. The main content area is titled 'vueTestingSuite' and shows a list of stacks under the heading 'Stacks (14)'. The 'vueTestingSuite' stack is highlighted, showing its creation time (2019-09-30 14:53:15 UTC+0800) and status (UPDATE_COMPLETE). Below this, other stacks like 'pythonTest' and 'cloud9-pythonTests' are listed, also in 'UPDATE_COMPLETE' status. On the right, the 'vueTestingSuite' stack details are shown, including buttons for 'Delete', 'Update', 'Stack actions', and 'Create stack'. The 'Stack info' tab is active, displaying the 'Overview' section with the Stack ID (arn:aws:cloudformation:us-east-1:216730220204:stack/vueTestingSuite/fa58d230-e34e-11e9-b592-0eb40de15aba) and its description (AWS API Gateway with a Lambda function). A red arrow points to the Stack ID. The 'Status' is 'UPDATE_COMPLETE' and the 'Status reason' is empty.

CloudFormation X

CloudFormation > Stacks > vueTestingSuite

Stacks (14)

Filter by stack name

Active

☒ View nested < 1 >

vueTestingSuite
2019-09-30 14:53:15 UTC+0800
 UPDATE_COMPLETE

pythonTest
2019-09-30 13:32:01 UTC+0800
 UPDATE_COMPLETE

cloud9-pythonTests
2019-09-03 11:34:13 UTC+0800
 UPDATE_COMPLETE

vue cloud9 Test lambda code c70f30

vueTestingSuite

Delete Update Stack actions Create stack

Stack info Events Resources Outputs Parameters

Change sets

Overview

Stack ID	Description
arn:aws:cloudformation:us-east-1:216730220204:stack/vueTestingSuite/fa58d230-e34e-11e9-b592-0eb40de15aba	AWS API Gateway with a Lambda
Status	Status reason
UPDATE_COMPLETE	-

Step 4 [Contd]:

Successfully deployed

Cloudformation stack :
vueTestingSuite

Additional Information



Actions and Workflows

There are two components to using GitHub Actions:

- the **action** itself
- a **workflow** that uses the action

A workflow can contain many actions, but each action has its own purpose.

Step 1: Creating a Dockerfile

- Actions come in two types: **container** actions and **JavaScript** actions. Our action will use a Docker container so it will require a **Dockerfile**.

```
FROM python:alpine
```

Docker linux image
with python

```
LABEL "com.github.actions.name"="lambda-github-actions"
```

```
LABEL "com.github.actions.description"="Deploy Lambda through GitHub"
```

```
LABEL "com.github.actions.icon"="upload-cloud"
```

```
LABEL "com.github.actions.color"="purple"
```

Github Action name
and other
configurations

```
RUN apk add --no-cache --virtual .build-deps gcc musl-dev \  
&& pip install cython \  
&& apk del .build-deps
```

```
RUN pip3 install awscli
```

```
RUN apk add zip
```

```
RUN apk add --update nodejs npm
```

```
RUN apk add --update npm
```

```
ADD entrypoint.sh /entrypoint.sh
```

```
RUN chmod +x /entrypoint.sh
```

```
ENTRYPOINT ["/entrypoint.sh"]
```

Add necessary packages needed in your docker image here. For this project the libraries/packages included are-

1. Cython
2. Awscli
3. Zip
4. Node and NPM

The entrypoint.sh script will be run in Docker, and it will define what the action is really going to be doing.

```
1  #!/bin/sh -l
2
3  export AWS_ACCESS_KEY_ID=$AWS_ACCESS_KEY_ID
4  export AWS_SECRET_ACCESS_KEY=$AWS_SECRET_ACCESS_KEY
5  export AWS_DEFAULT_REGION=$AWS_DEFAULT_REGION
6  export AWS_SESSION_TOKEN=$AWS_SESSION_TOKEN
7  export APINAME="$LAMBDA_FUNC_NAME-API"
8  export OVERLAY_S3URL="s3://${BUCKET_NAME}/${LAMBDA_FUNC_NAME}/lambda-
9
10 # Remove zipped contents if it exists
11 rm -f lambda-deploy.zip
12 rm -f lambda-deploy-overlay.tgz
13
14 # Move to src directory
15 cd src
16
17 echo
18 echo "*****"
19 echo "*****NPM INSTALL*****"
20 echo "*****"
21
22 # npm install node modules
23 if npm install
24 then
25     echo "$(tput setaf 2)npm successfully installed$(tput sgr 0)"
26 else
27     echo "$(tput setaf 2)*****ERROR*****$(tput sgr 0)"
28 fi
29
30 # zip project contents
31 zip -qr ../lambda-deploy.zip *
```

Step 2: Add the action's script

An entrypoint script must exist in our repository so that Docker has something

Export AWS configuration variables which will be picked up by AWS-CLI

Zip up and upload your project in the src/ folder to the AWS S3 cloud

Validate your template structure and then package template.yaml and have it generate packaged.yaml file with S3 reference link to the uploaded code.

Deploy the cloudformation stack with the parameters required.

```

33 # Move back to main project
34 cd ..
35
36 # Create a tarball of the entire project structure
37 tar -czf lambda-deploy-overlay.tgz ./
38
39 # Upload tar to AWS S3
40 aws s3 cp --acl public-read lambda-deploy-overlay.tgz "$OVERLAY_S3URL"
41
42 # Validate Cloudformation template
43 aws cloudformation validate-template \
44     --template-body file://template.yaml
45
46 # Package template
47 aws cloudformation package \
48     --template-file template.yaml \
49     --output-template-file packaged.yaml \
50     --s3-bucket "${BUCKET_NAME}"
51
52 # Deploy stack
53 if aws cloudformation deploy \
54     --stack-name ${LAMBDA_FUNC_NAME} \
55     --template-file packaged.yaml \
56     --capabilities CAPABILITY_IAM \
57     --region ${AWS_DEFAULT_REGION} \
58     --parameter-overrides LambdaFuncName=${LAMBDA_FUNC_NAME} \
59     LambdaRuntime=${LAMBDA_RUNTIME} \
60     LambdaHandler=${LAMBDA_HANDLER} \
61     LambdaMemory=${LAMBDA_MEMORY} \
62     LambdaTimeout=${LAMBDA_TIMEOUT}
63 then
64     exit 0
65 else
66     exit 1
67 fi
68
69
70 exit 0

```

Step 2[Part 2]: Add the action's script

Upload the the entire contents of the project onto AWS S3

Package the template file to fill in the S3 link in the generated output file called packaged.yaml

Deploy the cloudformation stack

Workflow Files

Workflows are defined in special files in the `.github/workflows` directory, named `main.yml`. Workflows can execute based on your chosen event. For this project, we'll be using the push event.

Step 3: Add a workflow file

```
{.} main.yml x
.github > workflows > {.} main.yml > {} jobs > {} build > [ ] steps > {} 1 > {} env > abc BUCKET_NAME
1  name: Deploy Lambda Function on push
2  on: push
3  jobs:
4    build:
5      name: Test action
6      runs-on: ubuntu-latest
7      steps:
8        - uses: actions/checkout@master
9        - uses: ./
10     env:
11       AWS_DEFAULT_REGION: "us-east-1"
12       LAMBDA_FUNC_NAME: "vueTestingSuite"
13       LAMBDA_RUNTIME: "nodejs10.x"
14       LAMBDA_HANDLER: "index.handler"
15       LAMBDA_MEMORY: 1024
16       LAMBDA_TIMEOUT: 40
17       BUCKET_NAME: ${ secrets.BUCKET_NAME }
18       AWS_SESSION_TOKEN: ${ secrets.AWS_SESSION_TOKEN }
19       AWS_SECRET_ACCESS_KEY: ${ secrets.AWS_SECRET_ACCESS_KEY }
20       AWS_ACCESS_KEY_ID: ${ secrets.AWS_ACCESS_KEY_ID }
21
```

This name appears on any pull request or in the Actions tab. The name is especially useful when there are multiple workflows in your repository..

Workflow will execute anytime code is pushed to your repository, using the push event.

Step 4: Use an action in your workflow

Workflows piece together jobs, and jobs piece together steps. We'll now create a job that runs an action. Actions can be used from within the same repository, from any other public repository, or from a published Docker container image. We'll use an action that we'll define in this repository.

```
{.} main.yaml x
.github > workflows > {.} main.yaml > {} jobs > {} build > [ ] steps > {} 1 > {} env > abc BUCKET_NAME
1  name: Deploy Lambda Function on push
2  on: push
3  jobs:
4    build:
5      name: Test action
6      runs-on: ubuntu-latest
7      steps:
8        - uses: actions/checkout@master
9        - uses: ./
10       env:
11         AWS_DEFAULT_REGION: "us-east-1"
12         LAMBDA_FUNC_NAME: "vueTestingSuite"
13         LAMBDA_RUNTIME: "nodejs10.x"
14         LAMBDA_HANDLER: "index.handler"
15         LAMBDA_MEMORY: 1024
16         LAMBDA_TIMEOUT: 40
17         BUCKET_NAME: ${ secrets.BUCKET_NAME }
18         AWS_SESSION_TOKEN: ${ secrets.AWS_SESSION_TOKEN }
19         AWS_SECRET_ACCESS_KEY: ${ secrets.AWS_SECRET_ACCESS_KEY }
20         AWS_ACCESS_KEY_ID: ${ secrets.AWS_ACCESS_KEY_ID }
21
```

- **jobs**: is the base component of a workflow run
- **build**: is the identifier we're attaching to this job
- **name**: is the name of the job, this is displayed on GitHub when the workflow is running
- **steps**: the linear sequence of operations that make up a job
- **uses**: `actions/checkout@master` uses an action called `checkout` to use a copy of github actions code repository
- **env**: is used to specify the environment variables that will be available to your action in the runtime environment.

Step 5 [Part 1]: Cloudformation template

{...} template.yaml ×

{...} template.yaml > {} Parameters > {} LambdaTimeout

```
1  AWSTemplateFormatVersion: "2010-09-09"
2  Transform: AWS::Serverless-2016-10-31
3  Description: AWS API Gateway with a Lambda Integration
4  Parameters:
5    LambdaFuncName:
6      Type: String
7      Default: "pythonTest"
8    LambdaRuntime:
9      Type: String
10     Default: "python3.7"
11    LambdaHandler:
12      Type: String
13      Default: "index.lambda_handler"
14    LambdaMemory:
15      Type: Number
16      Default: 128
17    LambdaTimeout:
18      Type: Number
19      Default: 40
```

Ref :

<https://docs.aws.amazon.com/serverless-application-model/latest/developerguide/serverless-sam-template-basics.html>

AWS SAM (Serverless Application Model) Template

Parameters -

Specify all the variables needed by the template. Default values and datatype for the same can be specified.

For this project -

The environment variables are passed in to the template as parameters. These parameters will then be used to configure the lambda.


```

20 Resources:
21   FunctionRole:
22     Properties:
23       AssumeRolePolicyDocument:
24         Statement:
25           - Action:
26               - sts:AssumeRole
27             Effect: Allow
28             Principal:
29               Service:
30                 - lambda.amazonaws.com
31       ManagedPolicyArns:
32         - arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole
33         - arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess
34       Path: /
35     Type: AWS::IAM::Role
36   LambdaFunc:
37     Properties:
38       CodeUri: ./lambda-deploy.zip
39       Description: Vue Testing Suite
40       FunctionName:
41         Ref: "LambdaFuncName"
42     Events:
43       ExecuteFunc:
44         Properties:
45           Method: any
46           Path: /
47           Type: Api
48     Handler:
49       Ref: "LambdaHandler"
50     MemorySize:
51       Ref: "LambdaMemory"
52     Role:
53       Fn::GetAtt:
54         - FunctionRole
55         - Arn
56     Runtime:
57       Ref: "LambdaRuntime"
58     Timeout:
59       Ref: "LambdaTimeout"
60     Type: AWS::Serverless::Function

```

Step 5 [Part 2]: Cloudformation template

Resources -

- **FunctionRole -**
 - a. An AWS IAM role with lambda basic execution and S3 read only access permissions
- **LambdaFunc**
 - a. Lambda function as a resource.
 - b. Assign the IAM role created before.
 - c. The parameters are being referenced to configure the lambda function.
 - d. Set up an event type API which will act as the API gateway for the lambda function.
 - e. The CodeUri points to a local resource - the zipped project contents in this case. On packaging this template this CodeUri will be replaced with the link to the uploaded files on S3. This can be found in the output-template packaged.yaml file in our case.