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!

src : https://github.com/features/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.

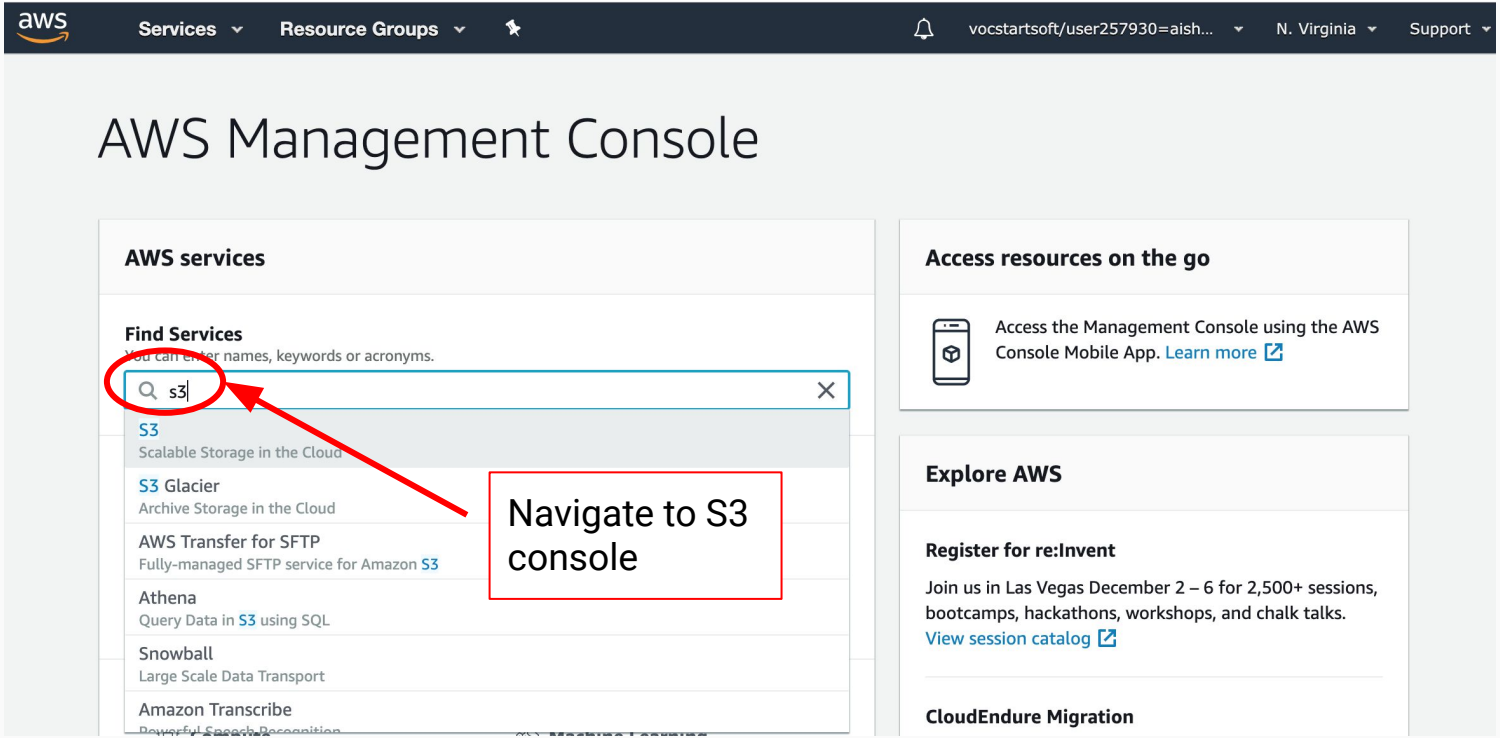
So instead of having to write script with a bunch of AWS API calls, wait loops, and retry logic, you just tell describe what you want and tell CloudFormation to do it for you. Beautiful.

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

Prerequisite to Deploy your Lambda Function

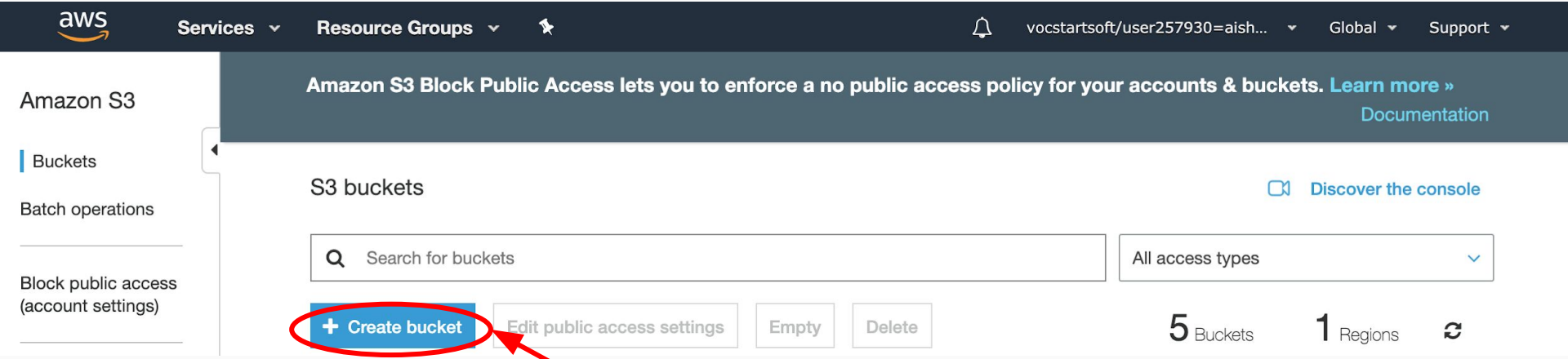
Prerequisite

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



Prerequisite [Contd.]

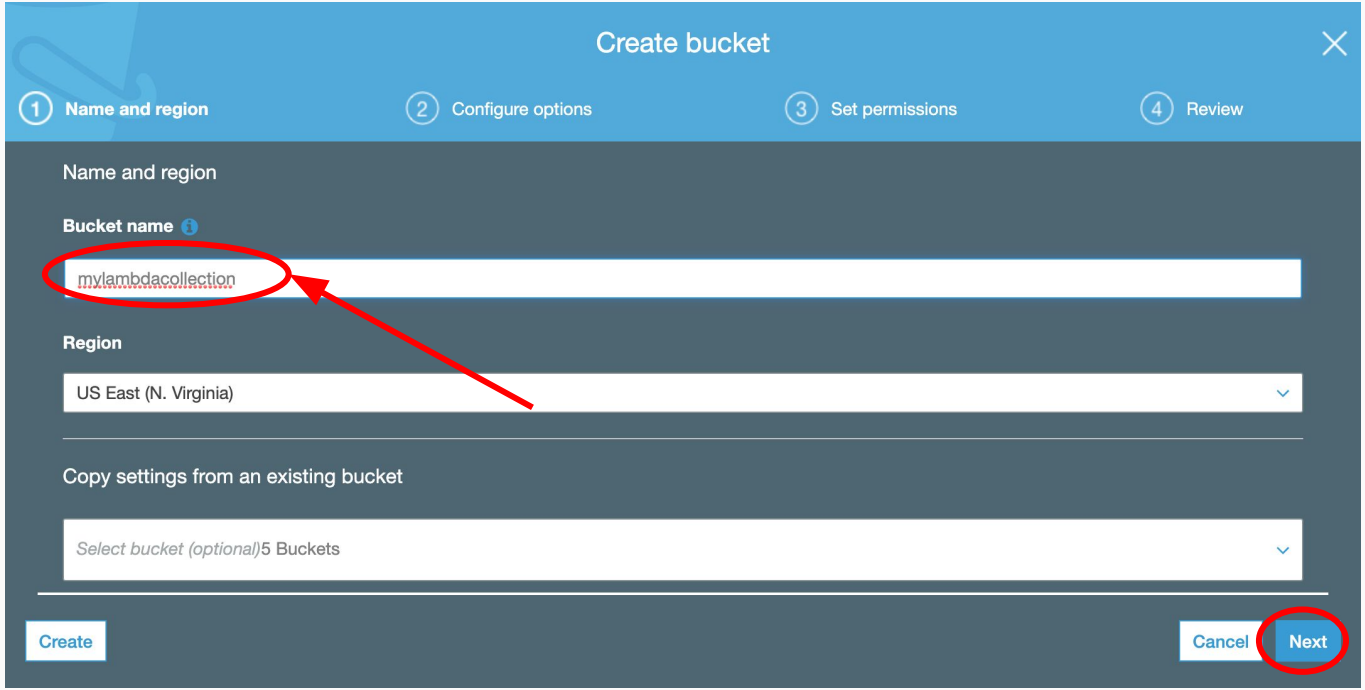
1. **Create an AWS S3 bucket with permissions to access objects publically.**



Click on create bucket

Prerequisite [Contd.]

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

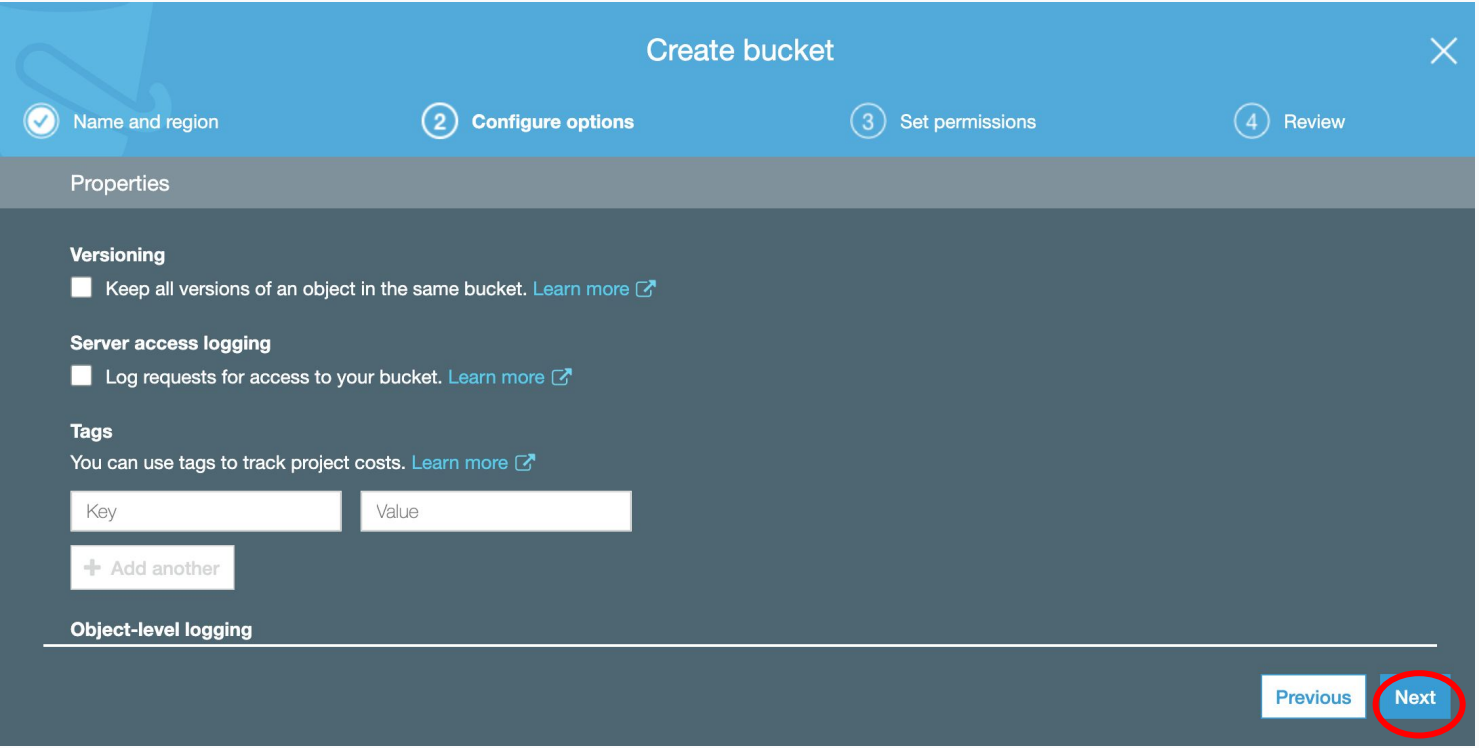


Enter a DNS complaint name for your bucket

Click Next

Prerequisite [Contd.]

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

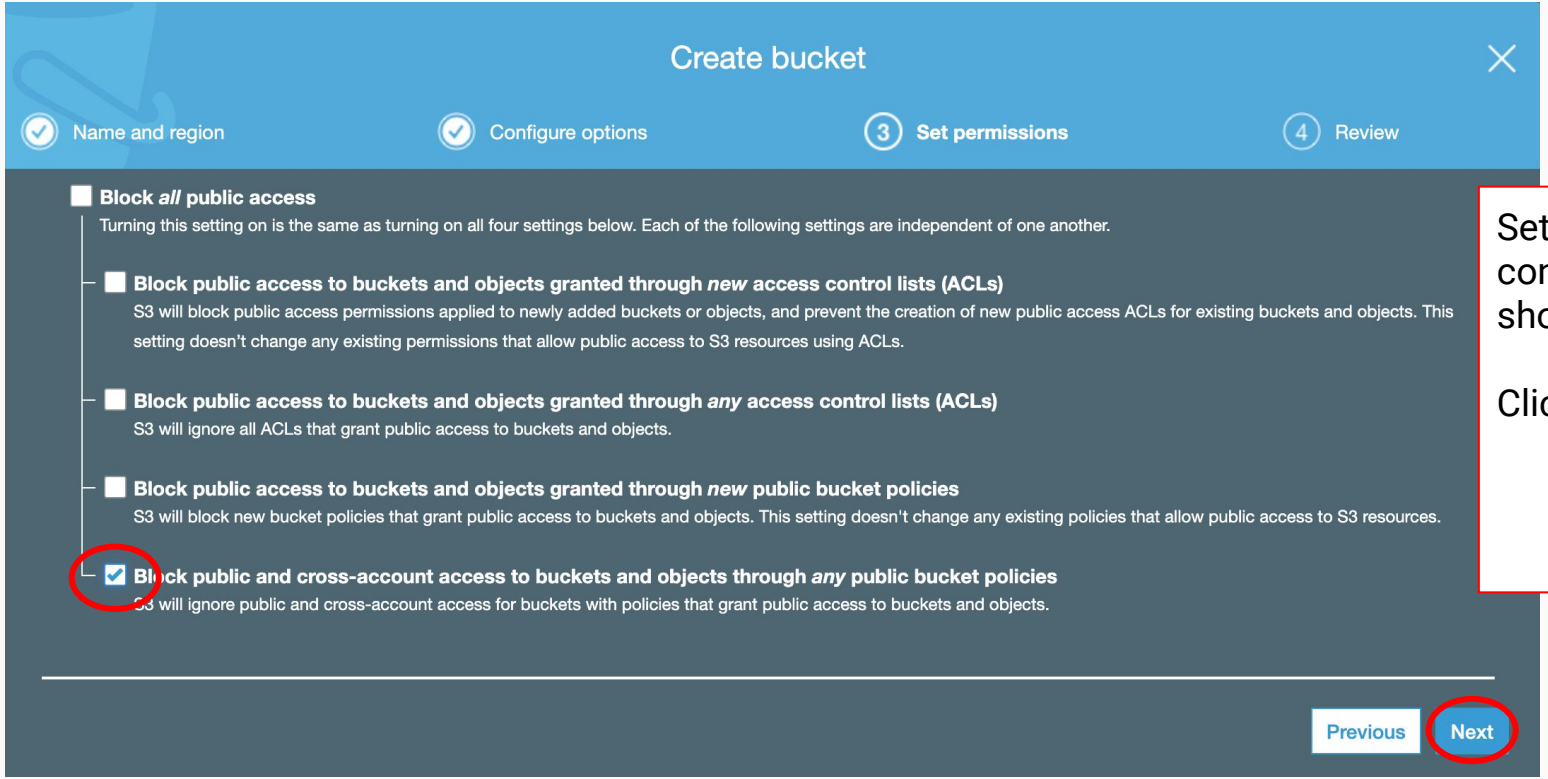


For the purpose of this exercise, we do not require additional configuration.

Click Next

Prerequisite [Contd.]

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

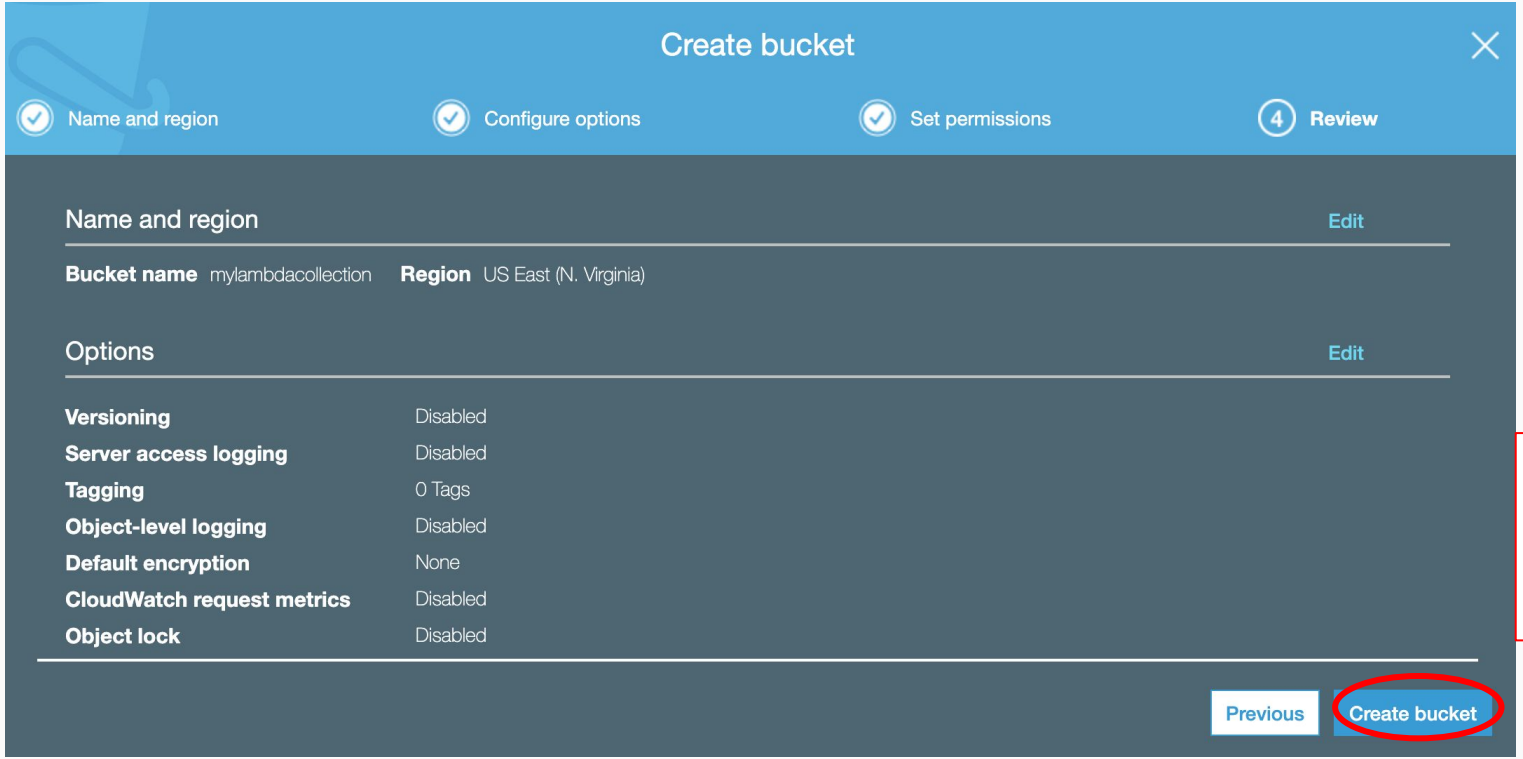


Set your bucket configuration as shown

Click Next

Prerequisite [Contd.]

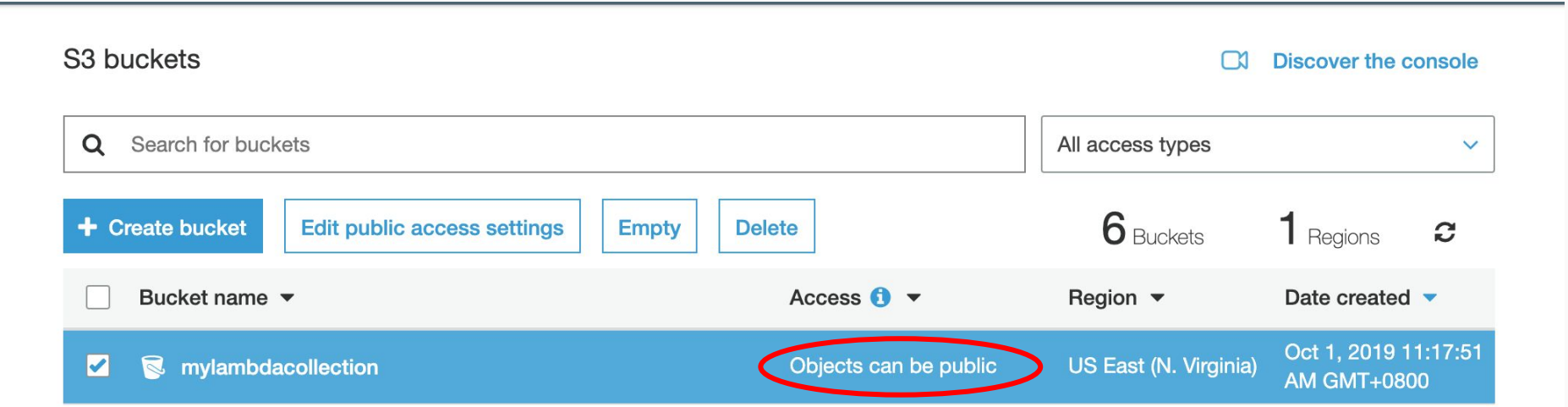
1. **Create an AWS S3 bucket with permissions to access objects publically.**



Review your changes and click on create bucket

Prerequisite [Contd.]

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



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

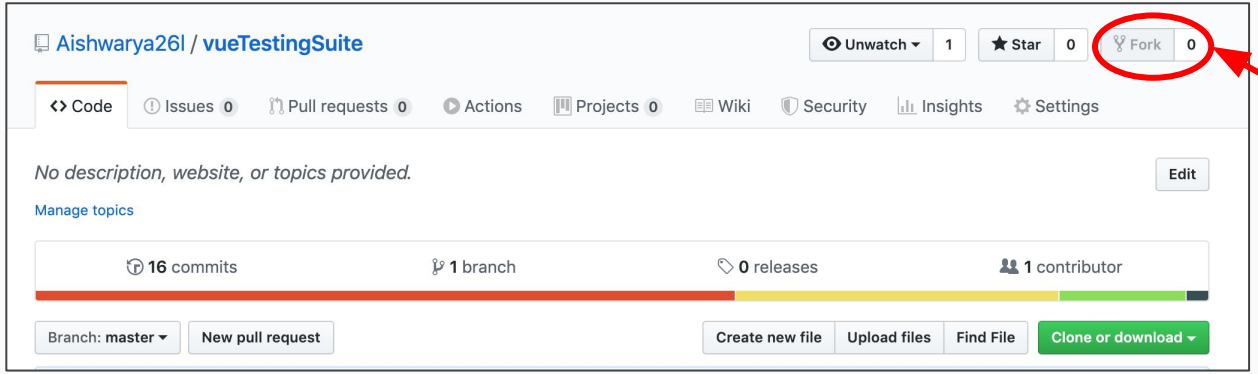
Repository link : https://github.com/Aishwarya26l/vueTestingSuite

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

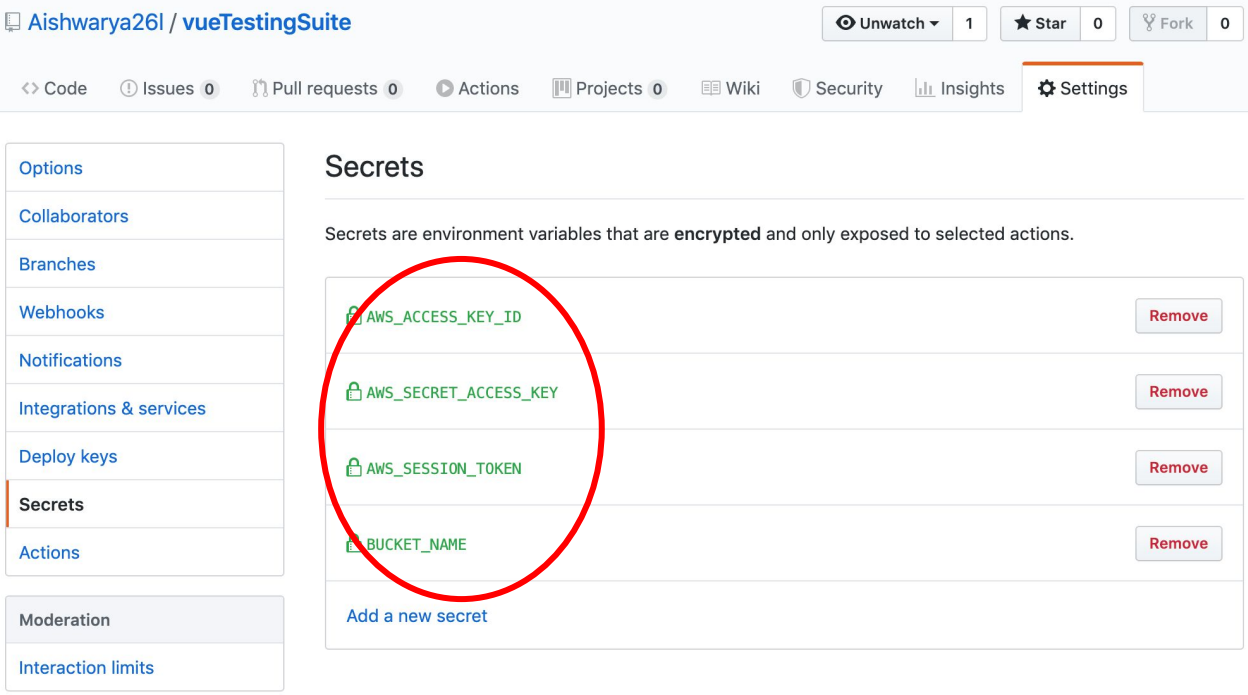


Click here to fork the repository

Steps to Deploy your Lambda Function

Step 2:

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



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

vocareum

My Classes Help aishwarya.l@nus.ed..

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 sigin 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 ()

$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:

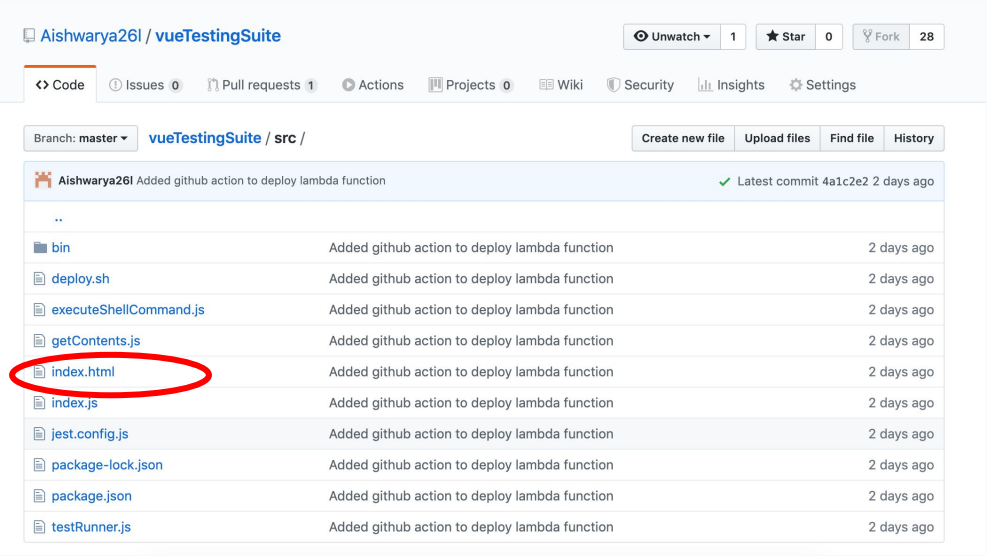
Copy the values corresponding to the keys required and add them in your Github console.

Steps to Deploy your Lambda Function

Step 3:

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

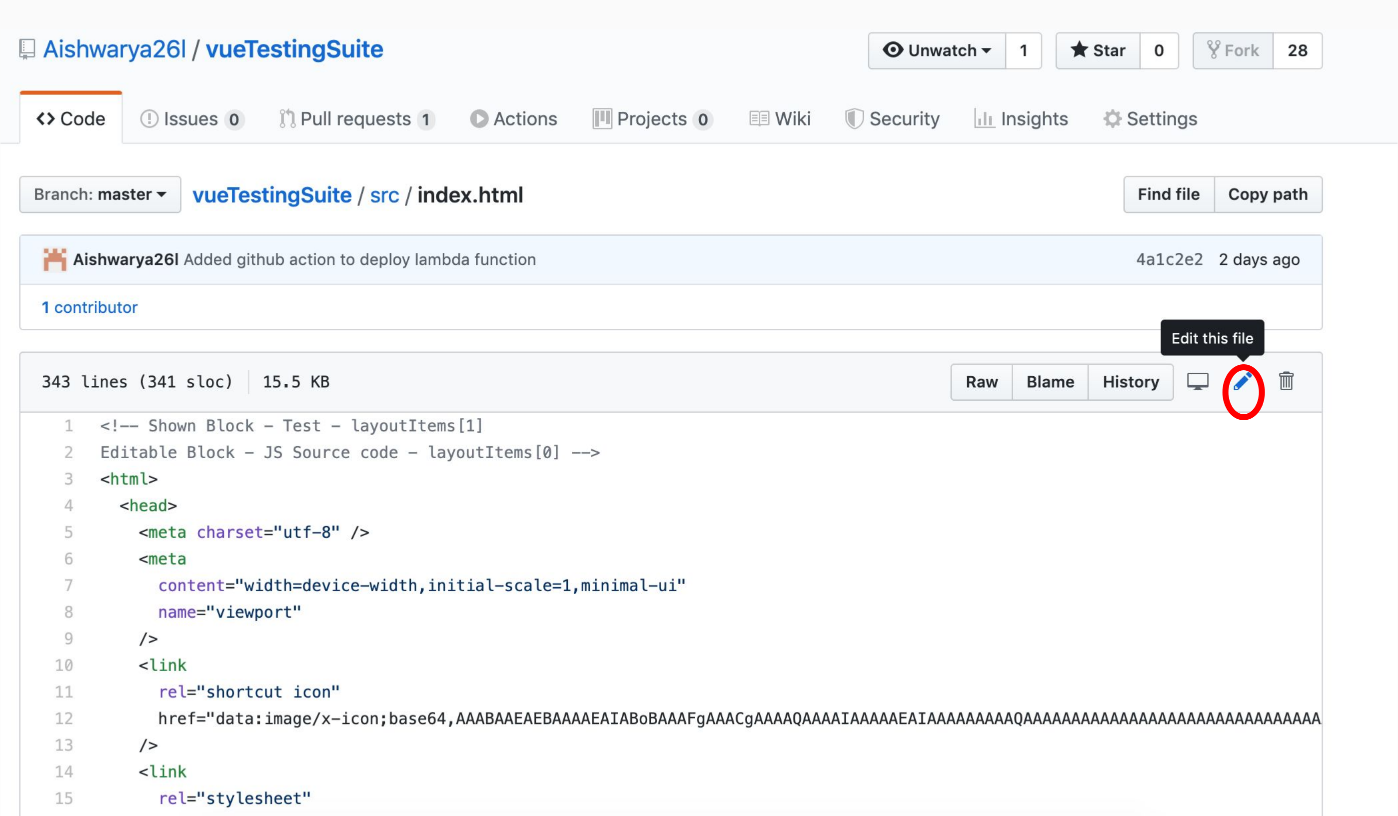
1. From the github console, click on the file you would like to modify. Example - src/index.html



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]:



vueTestingSuite / src / index.html

Aishwarya26l Added github action to deploy lambda function 4a1c2e2 2 days ago

1 contributor

343 lines (341 sloc) 15.5 KB

```
<!-- Shown Block - Test - layoutItems[1]
Editable Block - JS Source code - layoutItems[0] -->
<html>
  <head>
    <meta charset="utf-8" />
    <meta
      content="width=device-width,initial-scale=1,minimal-ui"
      name="viewport"
    />
    <link
      rel="shortcut icon"
      href="
    />
    <link
      rel="stylesheet"
```

Click on Edit this file

Steps to Deploy your Lambda Function

Step 3[Contd]:

```
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.

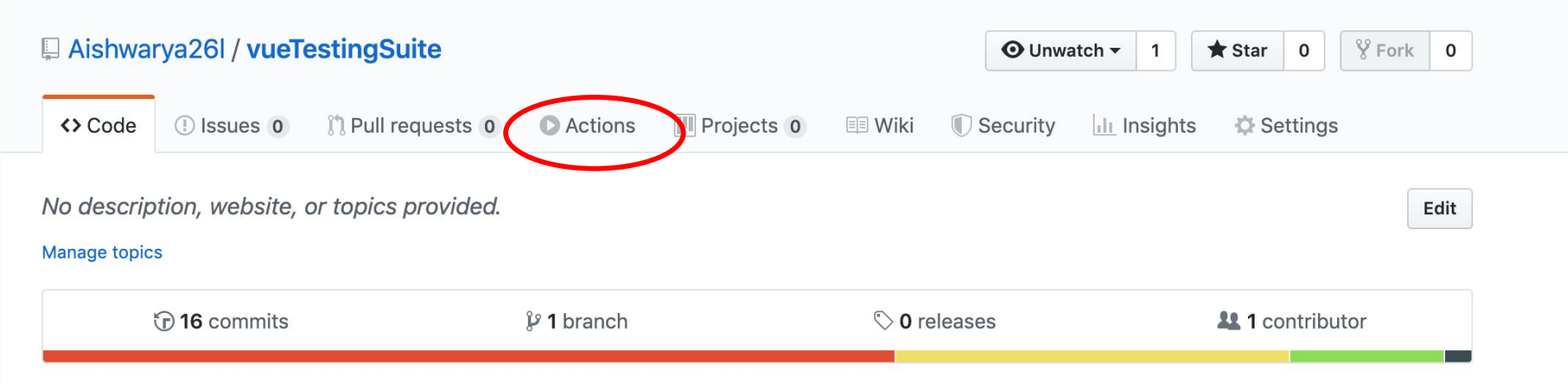
Click on commit changes.

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 -



Steps to Deploy your Lambda Function

Aishwarya26l / vueTestingSuite

Unwatch 1 | Star 0 | Fork 0

Code | Issues 0 | Pull requests 0 | Actions | Projects 0 | 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
   "/home/runner/work/vueTestingSuite/vueTestingSuite"
15 Sending build context to Docker daemon  2.655MB
16
17 Step 1/11 : FROM python:alpine
18 alpine: Pulling from library/python
19 9d48c3bd43c5: Already exists
```

•
•
•

```
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

aws Services Resource Groups vocstartsoft/user257930=aish... N. Virginia Support

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

aws Services Resource Groups vocstartsoft/user257930=aish... N. Virginia Support

CloudFormation

Stacks
Stack details
Drifts
StackSets
Exports

Designer

Previous console

Feedback

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

vueTestingSuite

Delete | Update | Stack actions | Create stack

Stack info | Events | Resources | Outputs | Parameters

Change sets

Overview

Stack ID
arn:aws:cloudformation:us-east-1:216730220204:stack/vueTestingSuite/fa58d230-e34e-11e9-b592-0eb40de15aba

Description
AWS API Gateway with a Lamb

Status
UPDATE_COMPLETE

Status reason
-

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

LABEL "com.github.actions.name"="lambda-github-actions"
LABEL "com.github.actions.description"="Deploy Lambda through GitHub A
LABEL "com.github.actions.icon"="upload-cloud"
LABEL "com.github.actions.color"="purple"

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"]
```

Docker linux image with python

Github Action name and other configurations

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.

entrypoint.sh

```
#!/bin/sh -l

export AWS_ACCESS_KEY_ID=$AWS_ACCESS_KEY_ID
export AWS_SECRET_ACCESS_KEY=$AWS_SECRET_ACCESS_KEY
export AWS_DEFAULT_REGION=$AWS_DEFAULT_REGION
export AWS_SESSION_TOKEN=$AWS_SESSION_TOKEN
export APINAME="$LAMBDA_FUNC_NAME-API"
export OVERLAY_S3URL="s3://${BUCKET_NAME}/${LAMBDA_FUNC_NAME}/lambda-

# Remove zipped contents if it exists
rm -f lambda-deploy.zip
rm -f lambda-deploy-overlay.tgz

# Move to src directory
cd src

echo
echo "***************************"
echo "*****NPM INSTALL***********"
echo "***************************"

# npm install node modules
if npm install
then
    echo "$(tput setaf 2)npm successfully installed$(tput sgr 0)"
else
    echo "$(tput setaf 2)**********ERROR***********$(tput sgr 0)"
fi

# zip project contents
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 to execute

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.yaml

.github > workflows > main.yaml > {} jobs > {} build > [ ] steps > {} 1 > {} env > abc BUCKET_NAME

```yaml
name: Deploy Lambda Function on push
on: push
jobs:
  build:
    name: Test action
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@master
      - uses: ./
        env:
          AWS_DEFAULT_REGION: "us-east-1"
          LAMBDA_FUNC_NAME: "vueTestingSuite"
          LAMBDA_RUNTIME: "nodejs10.x"
          LAMBDA_HANDLER: "index.handler"
          LAMBDA_MEMORY: 1024
          LAMBDA_TIMEOUT: 40
          BUCKET_NAME: ${{ secrets.BUCKET_NAME }}
          AWS_SESSION_TOKEN: ${{ secrets.AWS_SESSION_TOKEN }}
          AWS_SECRET_ACCESS_KEY: ${{ secrets.AWS_SECRET_ACCESS_KEY }}
          AWS_ACCESS_KEY_ID: ${{ secrets.AWS_ACCESS_KEY_ID }}
```

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

.github > workflows > main.yaml > {} jobs > {} build > [ ] steps > {} 1 > {} env > BUCKET_NAME

```yaml
name: Deploy Lambda Function on push
on: push
jobs:
  build:
    name: Test action
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@master
      - uses: ./
        env:
          AWS_DEFAULT_REGION: "us-east-1"
          LAMBDA_FUNC_NAME: "vueTestingSuite"
          LAMBDA_RUNTIME: "nodejs10.x"
          LAMBDA_HANDLER: "index.handler"
          LAMBDA_MEMORY: 1024
          LAMBDA_TIMEOUT: 40
          BUCKET_NAME: ${{ secrets.BUCKET_NAME }}
          AWS_SESSION_TOKEN: ${{ secrets.AWS_SESSION_TOKEN }}
          AWS_SECRET_ACCESS_KEY: ${{ secrets.AWS_SECRET_ACCESS_KEY }}
          AWS_ACCESS_KEY_ID: ${{ secrets.AWS_ACCESS_KEY_ID }}
```

- **`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

```yaml
AWSTemplateFormatVersion: "2010-09-09"
Transform: AWS::Serverless-2016-10-31
Description: AWS API Gateway with a Lambda Integration
Parameters:
  LambdaFuncName:
    Type: String
    Default: "pythonTest"
  LambdaRuntime:
    Type: String
    Default: "python3.7"
  LambdaHandler:
    Type: String
    Default: "index.lambda_handler"
  LambdaMemory:
    Type: Number
    Default: 128
  LambdaTimeout:
    Type: Number
    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.

template.yaml

template.yaml > {} Parameters > {} LambdaTimeout

```yaml
Resources:
  FunctionRole:
    Properties:
      AssumeRolePolicyDocument:
        Statement:
          - Action:
              - sts:AssumeRole
            Effect: Allow
            Principal:
              Service:
                - lambda.amazonaws.com
      ManagedPolicyArns:
        - arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole
        - arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess
      Path: /
    Type: AWS::IAM::Role
  LambdaFunc:
    Properties:
      CodeUri: ./lambda-deploy.zip
      Description: Vue Testing Suite
      FunctionName:
        Ref: "LambdaFuncName"
      Events:
        ExecuteFunc:
          Properties:
            Method: any
            Path: /
          Type: Api
      Handler:
        Ref: "LambdaHandler"
      MemorySize:
        Ref: "LambdaMemory"
      Role:
        Fn::GetAtt:
          - FunctionRole
          - Arn
      Runtime:
        Ref: "LambdaRuntime"
      Timeout:
        Ref: "LambdaTimeout"
    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.