

# **CS516 - CLOUD COMPUTING**

**Professor: Unubold Tumenbayar**

**Student: Ba Luan Tran**

## **What is Amazon Web Services (AWS)?**

Amazon Web Services (AWS) is the world's most comprehensive and broadly adopted cloud platform, offering over 200 fully featured services from data centers globally.

Millions of customers—including the fastest-growing startups, largest enterprises, and leading government agencies—are using AWS to lower costs, become more agile, and innovate faster.

## **What is Amazon Cognito?**

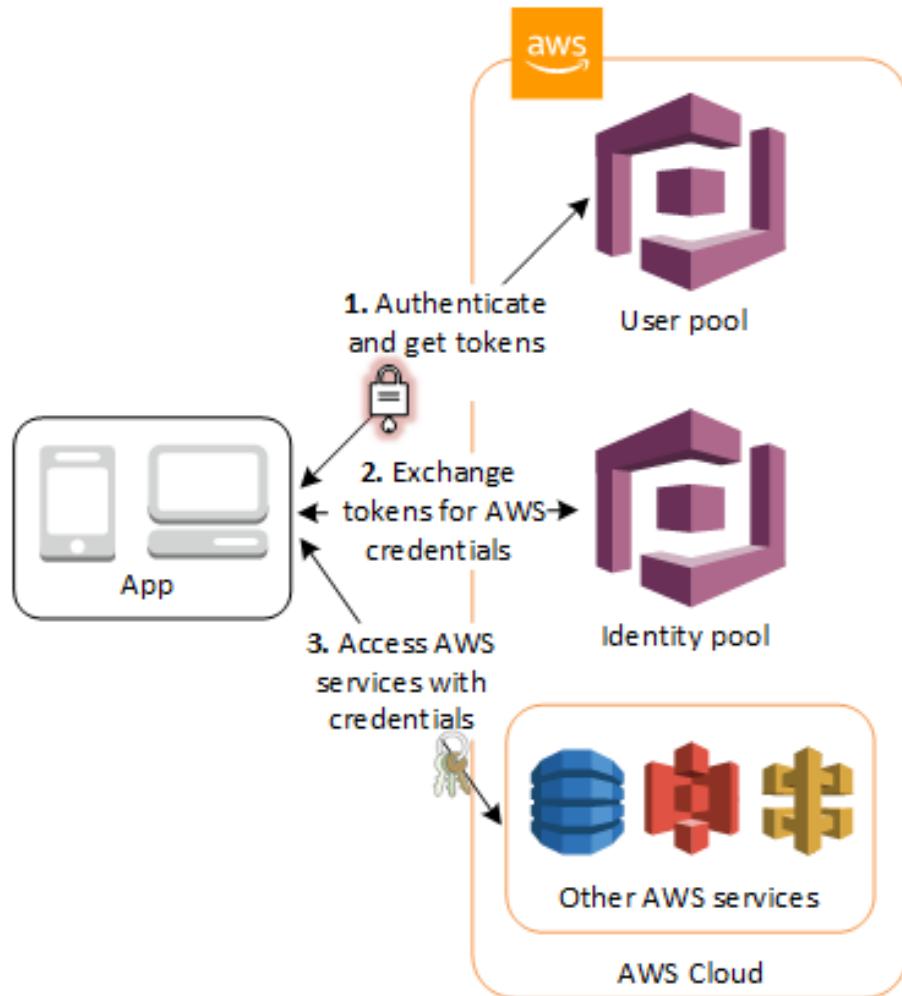
Amazon Cognito provides authentication, authorization, and user management for your web and mobile apps. Your users can sign in directly with a user name and password, or through a third party such as Facebook, Amazon, Google or Apple.

The two main components of Amazon Cognito are user pools and identity pools. User pools are user directories that provide sign-up and sign-in options for your app users. Identity pools enable you to grant your users access to other AWS services. You can use identity pools and user pools separately or together.

An Amazon Cognito user pool and identity pool used together

See the diagram for a common Amazon Cognito scenario. Here the goal is to authenticate your user, and then grant your user access to another AWS service.

1. In the first step your app user signs in through a user pool and receives user pool tokens after a successful authentication.
2. Next, your app exchanges the user pool tokens for AWS credentials through an identity pool.
3. Finally, your app user can then use those AWS credentials to access other AWS services such as Amazon S3 or DynamoDB.



For more examples using identity pools and user pools, see [Common Amazon Cognito scenarios](#).

Amazon Cognito is compliant with SOC 1-3, PCI DSS, ISO 27001, and is HIPAA-BAA eligible. For more information, see [AWS services in scope](#). See also [Regional data considerations](#).

## Topics

- [Features of Amazon Cognito](#)
- [Getting started with Amazon Cognito](#)
- [Regional availability](#)
- [Pricing for Amazon Cognito](#)
- [Using the Amazon Cognito console](#)
- [Using this service with an AWS SDK](#)

---

# Features of Amazon Cognito

## User pools

A user pool is a user directory in Amazon Cognito. With a user pool, your users can sign in to your web or mobile app through Amazon Cognito, or federate through a third-party identity provider (IdP). Whether your users sign in directly or through a third party, all members of the user pool have a directory profile that you can access through an SDK.

User pools provide:

- Sign-up and sign-in services.
- A built-in, customizable web UI to sign in users.
- Social sign-in with Facebook, Google, Login with Amazon, and Sign in with Apple, and through SAML and OIDC identity providers from your user pool.
- User directory management and user profiles.
- Security features such as multi-factor authentication (MFA), checks for compromised credentials, account takeover protection, and phone and email verification.
- Customized workflows and user migration through AWS Lambda triggers.

For more information about user pools, see [Getting started with user pools](#) and the [Amazon Cognito user pools API reference](#).

## Identity pools

With an identity pool, your users can obtain temporary AWS credentials to access AWS services, such as Amazon S3 and DynamoDB. Identity pools support anonymous guest users, as well as the following identity providers that you can use to authenticate users for identity pools:

- Amazon Cognito user pools
- Social sign-in with Facebook, Google, Login with Amazon, and Sign in with Apple
- OpenID Connect (OIDC) providers
- SAML identity providers
- Developer authenticated identities

To save user profile information, your identity pool needs to be integrated with a user pool.

For more information about identity pools, see [Getting started with Amazon Cognito identity pools \(federated identities\)](#) and the [Amazon Cognito identity pools API reference](#).

## Implementation Example

1. Create a user pool in AWS Cognito:

The screenshot shows the AWS Cognito User Pools console. At the top, there's a navigation bar with links to RDS, IAM, S3, CloudFront, Certificate Manager, Route 53, DynamoDB, Simple Notification Service, and API Gateway. The user is signed in as 'Admin @ 6235-4951-0599' in the N. Virginia region.

The main page displays a 'User pool overview' card with the following details:

User pool name	ARN	Created time
Users	<a href="#">arn:aws:cognito-idp:us-east-1:23549510599:userpool/us-east-1_XzOGPLFVy</a>	February 5, 2023 at 21:34 CST
User pool ID	<a href="#">us-east-1_XzOGPLFVy</a>	Last updated time
		February 5, 2023 at 21:38 CST
Estimated number of users		
0		

Below the overview, there's a 'Getting started' section with a link to 'Create user'.

The navigation bar below the overview card includes tabs for 'Users' (which is selected), 'Groups', 'Sign-in experience', 'Sign-up experience', 'Messaging', and 'App integration'.

The 'Users (0)' section contains a table header with columns for 'User name', 'Email address', 'Email verified', and 'Confirmation status'. There are also buttons for 'Delete user' and 'Create user'.

At the bottom of the page, there are links for 'Privacy', 'Terms', and 'Cookie preferences', along with a copyright notice: '© 2023, Amazon Web Services, Inc. or its affiliates.'

[Option+S] N. Virginia ▾ Admin @ 6235-4951-0599 ▾

RDS IAM S3 CloudFront Certificate Manager Route 53 DynamoDB Simple Notification Service API Gateway >

Amazon Cognito > User pools > Users > App client: MyAppClient

## App client: MyAppClient [Info](#)

[Delete](#) [Edit](#)

### App client information

App client name MyAppClient	Authentication flow session duration 3 minutes	Created time February 5, 2023 at 21:34 CST
Client ID 3km8gpm8c0g4g950miklfspae v	Refresh token expiration 30 day(s)	Last updated time February 5, 2023 at 21:34 CST
Client secret -	Access token expiration 60 minutes	
Authentication flows ALLOW_REFRESH_TOKEN_AUTH ALLOW_USER_PASSWORD_AUTH ALLOW_USER_SRP_AUTH	ID token expiration 60 minutes	Advanced authentication settings Enable token revocation Enable prevent user existence errors

▶ **Attribute read and write permissions**  
Choose the standard and custom attributes this app can read and write. Required attributes are locked as writable. We recommend that you set immutable custom attributes as writable to allow the app client to set initial values during sign-up.

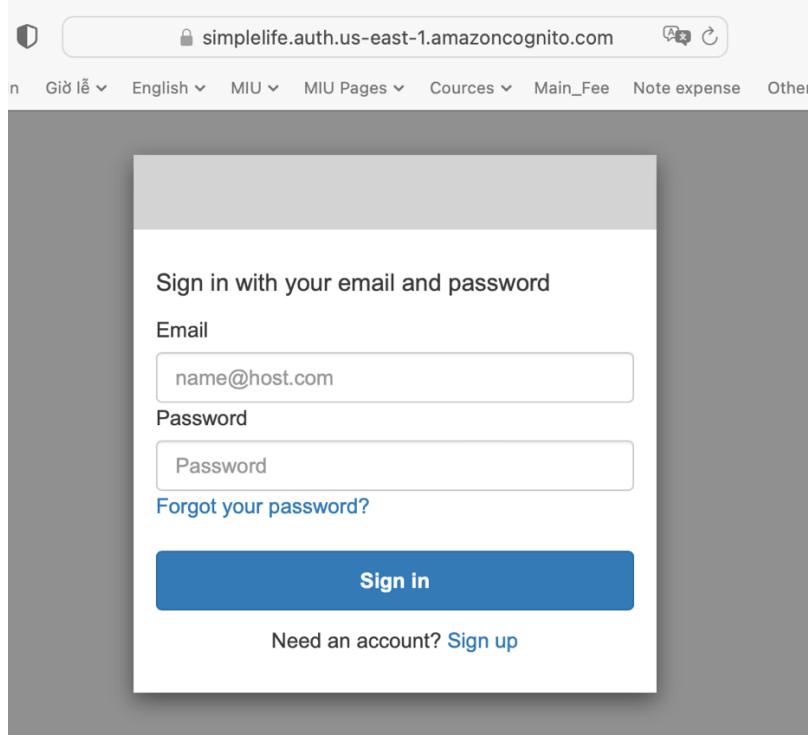
[Edit](#)

### Pinpoint analytics [Info](#)

[Enable](#)

© 2023, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

- After created the AWS Cognito, it will provides authentication, authorization, and user management for your web and mobile apps



<https://dispostable.com/inbox/testlab7BaLuan/>

**Disposable**

Inbox for testlab7BaLuan@dispostable.com

Continuously check for new messages

From	Subject	Date
0100018624cf05ed-793ca5af-5cc4-4baa-a073-1d680e5ad80d-000000@amazonses.com	Welcome to simplelife.studio	6 Feb 2023, 03:40

**Disposable**

testlab7BaLuan's inbox > "Welcome to simplelife.studio"

**Message "Welcome to simplelife.studio"**

**From:** no-reply@verificationemail.com

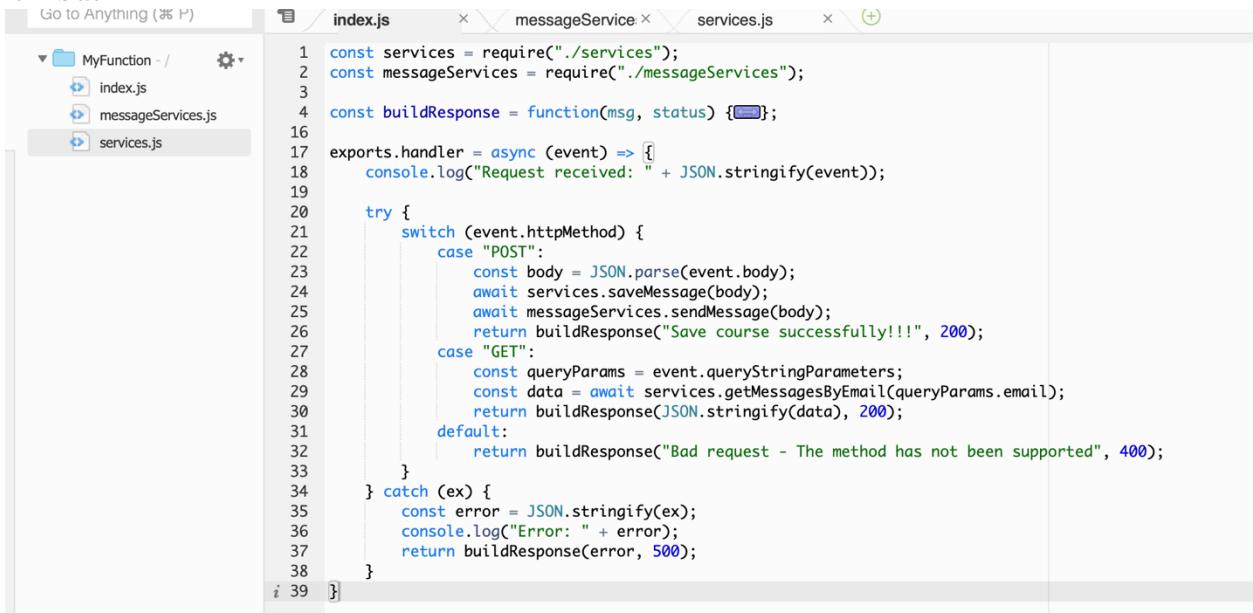
[Back to inbox](#) | [Download EML](#) | [Delete message](#)

Click [here](#) if you trust the sender and want to see original message.

Dear friend, Welcome to visit my site. Your verification code is 532749. Have a good day!!! Best regards, Ba Luan Tran

[Contact](#) | Unread messages older than 2 days, and read older than 2 months are automatically deleted.

2. Create a lambda to get/save messages from the website and create API Getway to the lambda

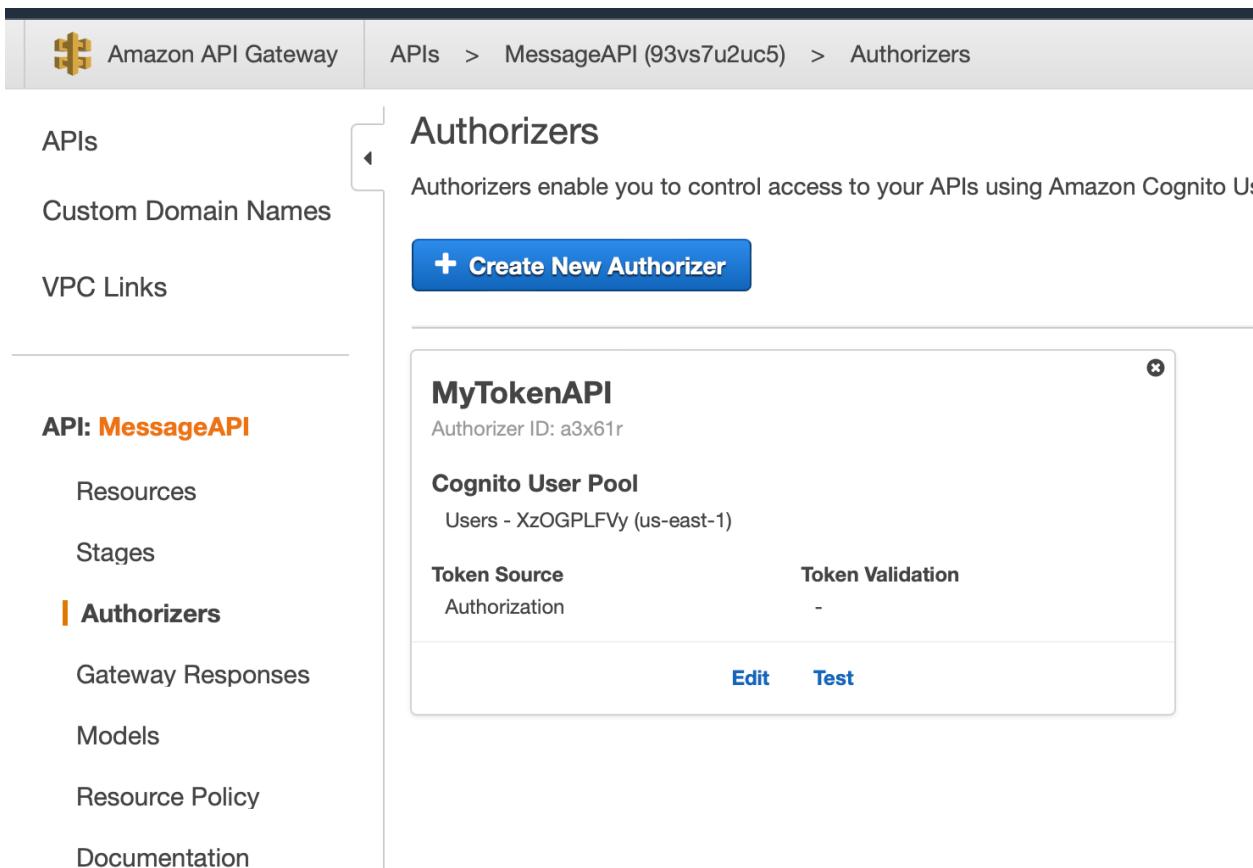


```

1 const services = require("./services");
2 const messageServices = require("./messageServices");
3
4 const buildResponse = function(msg, status) {
5   return {
6     statusCode: status,
7     body: msg,
8     headers: {
9       "Content-Type": "application/json"
10    }
11  };
12}
13
14 exports.handler = async (event) => {
15   console.log("Request received: " + JSON.stringify(event));
16
17   try {
18     switch (event.httpMethod) {
19       case "POST":
20         const body = JSON.parse(event.body);
21         await services.saveMessage(body);
22         await messageServices.sendMessage(body);
23         return buildResponse("Save course successfully!!!", 200);
24       case "GET":
25         const queryParams = event.queryStringParameters;
26         const data = await services.getMessagesByEmail(queryParams.email);
27         return buildResponse(JSON.stringify(data), 200);
28       default:
29         return buildResponse("Bad request - The method has not been supported", 400);
30     }
31   } catch (ex) {
32     const error = JSON.stringify(ex);
33     console.log("Error: " + error);
34     return buildResponse(error, 500);
35   }
36 }
37
38
39

```

3. Create an Authorizers of the Cognito User Pool and assign to the API Gateway



**APIs > MessageAPI (93vs7u2uc5) > Authorizers**

**Authorizers**

Authorizers enable you to control access to your APIs using Amazon Cognito User Pools.

**+ Create New Authorizer**

MyTokenAPI	
Authorizer ID: a3x61r	
<b>Cognito User Pool</b>	
Users - XzOGPLFVY (us-east-1)	
<b>Token Source</b>	<b>Token Validation</b>
Authorization	-
<b>Edit    Test</b>	

**API: MessageAPI**

- Resources
- Stages
- Authorizers**
- Gateway Responses
- Models
- Resource Policy
- Documentation

Resources

Actions ▾

## /message Methods

▼ /

▼ /message

GET

OPTIONS

POST

### GET

arn:aws:lambda:us-east-1:623549510599:function...

Authorization COGNITO\_USER\_POOLS

API Key Required

### OPTIONS

Mock Endpoint

Authorization None

API Key Not required

### POST

Authorization None

API Key Required

# Integrating Amazon Cognito into a website

1. Send some messages in the site (<https://www.simplelife.studio/>)

The screenshot shows a website layout with a navigation bar at the top. The navigation bar includes a blue cat icon logo, the text 'JAVA DEVELOPER', and links for HOME, ABOUT, WORK, EDUCATION, CONTACT (which is underlined), and SIGN IN.

**Send A Message**

Name: Thomas Tran

Email: thomas.tran@dispostable.com

Phone Number (optional): 2065937990

Subject: Can I get an appointment with you?

Your Message:

I'm Thomas.  
I see you have good experiences about Java and Reactjs.  
I want to see you in the next week to discuss technologies,  
structure to apply a new project  
Have a good day!

**Get in Touch**

Whether you want to get in touch, talk about a project collaboration, or just say hi, I'd love to hear from you.  
Simply fill the form and send me an email.

Fairfield, IA 52557  
View larger map

Map showing locations in Fairfield, IA, including Golden Dome Market and Cafe, Maharishi International University, Evergreen Cemetery, Everybody's Whole Foods, and Utopia Park. It also shows Fishback Lake and E Kirkwood Ave.

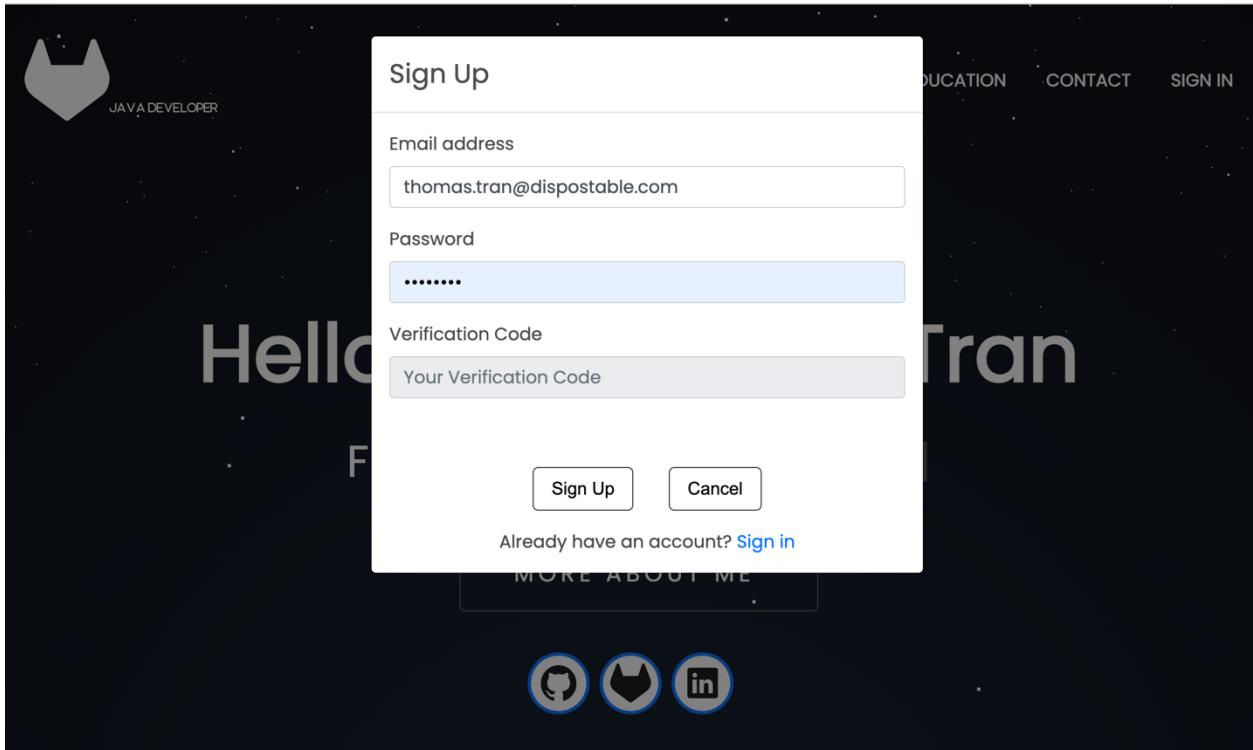
Keyboard shortcuts | Map data ©2023 Google | Terms of Use | Report a map error

📍 1000 N 4th St, Fairfield, Iowa 52557  
📞 (206) 593-7990  
✉️ ba.tran@miu.edu

Social media icons for GitHub, LinkedIn, and another blue cat icon.

Send Message

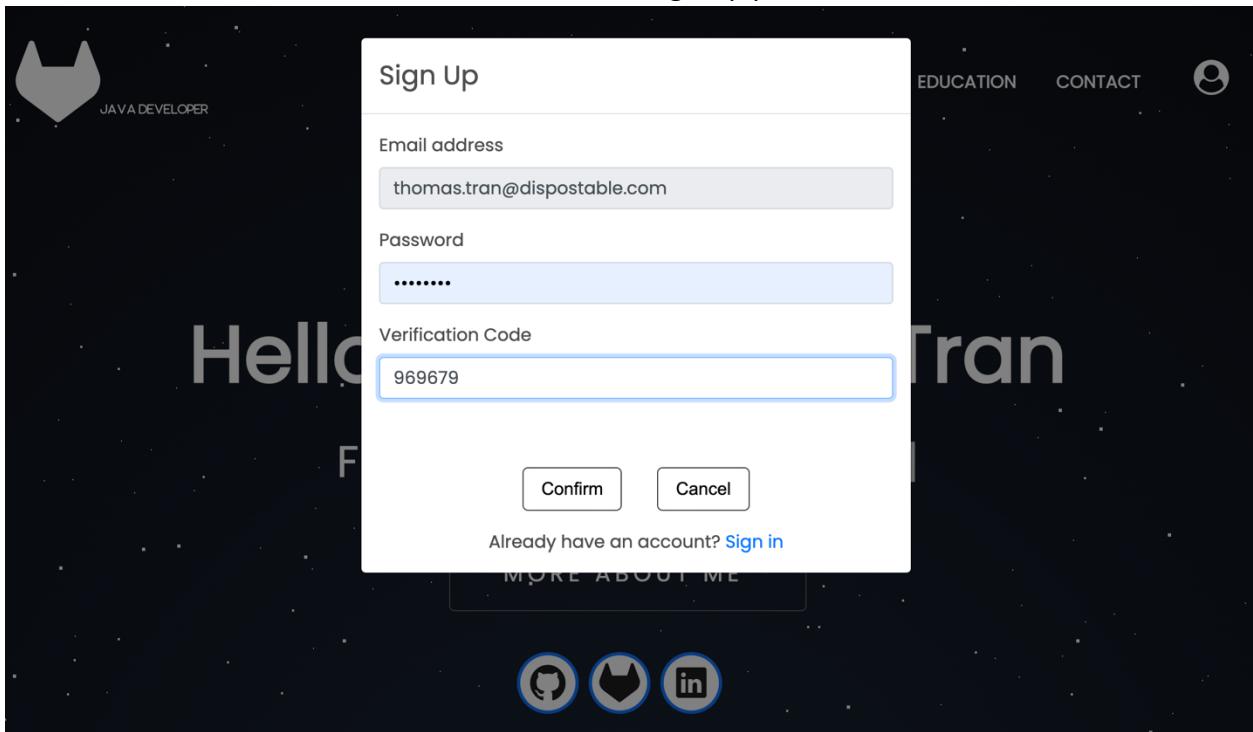
2. Sign up to the site with an email which used to send the messages



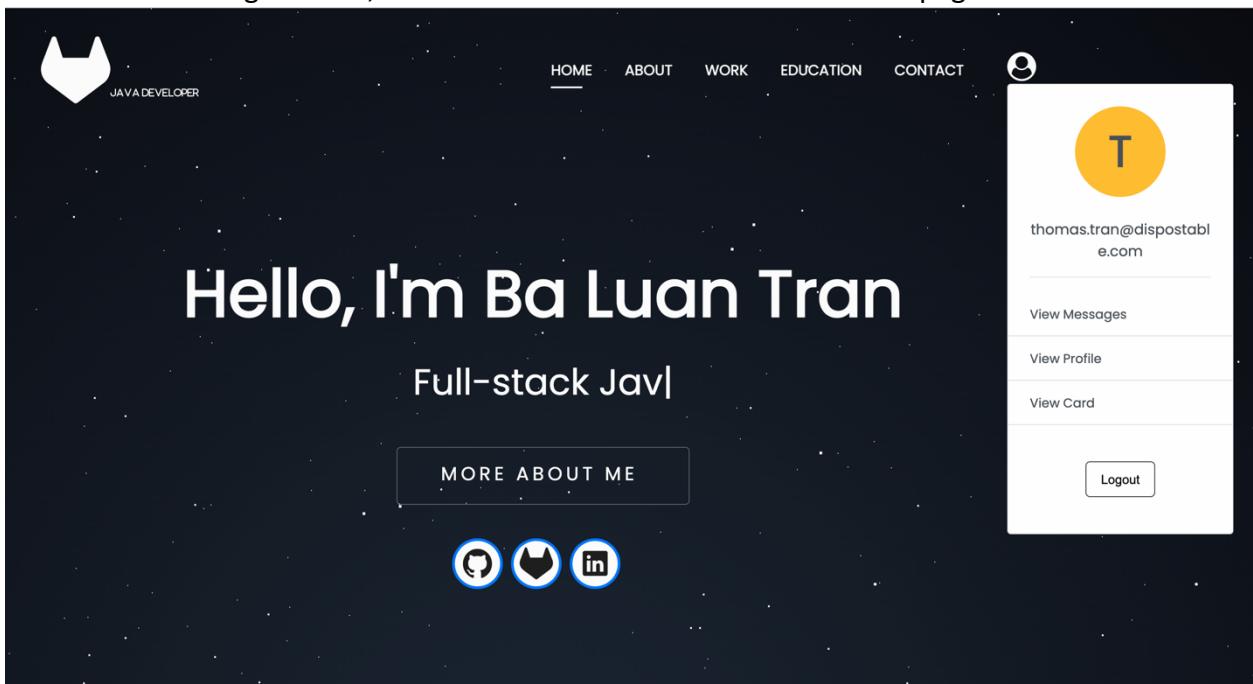
- An verification code will be sent to your email

The screenshot shows an email inbox from "Disposable". The inbox contains a single message titled "Message "Welcome to simplelife.studio"" from "no-reply@verificationemail.com". The message body says: "Dear friend, Welcome to visit my site. Your verification code is 969679. Have a good day!!! Best regards, Ba Luan Tran". Below the message, there is a note: "Click [here](#) if you trust the sender and want to see original message." At the bottom of the inbox, there is a link: "Contact | Unread messages older than 2 days, and read older than 2 months are automatically deleted."

- Enter the code and click Confirm to finish the Sign Up process



3. After successful registration, we will see the Profile icon in the header page



4. If we click the icon, the Profile dropdown will be displayed and we can see all sent messages by clicking the Message menu

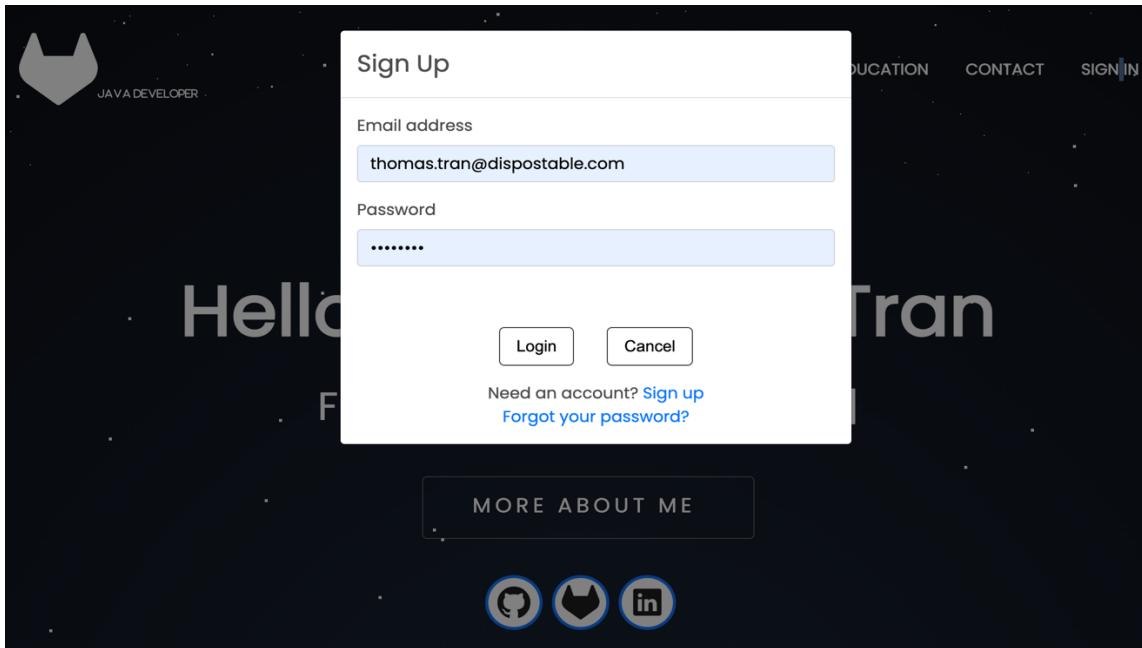
The screenshot shows a 'Messages' section with a table. The table has columns for Name, Subject, Content, and Datetime. There are two rows of data:

Name	Subject	Content	Datetime
Thomas Tran	Can I get an appointment with you?	I'm Thomas. I see you have good experiences about Java and Reactjs. I want to see you in the next week to discuss technologies, structure to apply a new project Have a good day!	2/6/2023, 8:59:42 PM
Thomas Tran	Welcome to my team	Welcome to my team, Ba Luan Tran. If you need any help, please call or message to me. Hope good things come to you!	2/6/2023, 9:02:22 PM

At the bottom right of the modal is a 'Close' button.

5. Beside that the website also supports Sign In, Forgot Password with the AWS Cognito:

The screenshot shows a 'Forgot your password?' form. It includes fields for 'Email address' (thomas.tran@dispostable.com) and buttons for 'Reset my password' and 'Cancel'. Below the form is a link 'Need an account? [Sign up](#)'. At the bottom of the page, there is a 'MORE ABOUT ME' button and social media icons for GitHub, LinkedIn, and another platform.



---

# CI/CD: GitHub Action to Sync S3 Bucket

## 1. Create a workflow in GitHub and create AWS secrets variables

main · cs516 / .github / workflows / prod.yml

luan-tran-89 · Update prod.yml ✓

1 contributor

38 lines (36 sloc) | 915 Bytes

```
1 name: Production Build
2 on:
3   pull_request:
4     push:
5       branches:
6         - main
7   env:
8     CI: "false"
9   jobs:
10    build:
11      runs-on: ubuntu-latest
12
13    strategy:
14      matrix:
15        node-version: [ 16.x ]
16
17    steps:
18      - uses: actions/checkout@v3
19      - name: Use Node.js ${{ matrix.node-version }}
20        uses: actions/setup-node@v3
21        with:
22          node-version: ${{ matrix.node-version }}
23      - name: Yarn Install
24        run: |
25          yarn install
26      - name: Production Build
27        run: |
28          yarn build
29      - name: Deploy to S3
30        uses: jakejarvis/s3-sync-action@master
31        with:
32          args: --acl public-read --delete
33          env:
34            AWS_S3_BUCKET: ${{ secrets.AWS_S3_BUCKET }}
35            AWS_ACCESS_KEY_ID: ${{ secrets.AWS_ACCESS_KEY_ID }}
36            AWS_SECRET_ACCESS_KEY: ${{ secrets.AWS_SECRET_ACCESS_KEY }}
37            AWS_REGION: ${{ secrets.AWS_REGION }}
38            SOURCE_DIR: "build"
```

2. After running the build action, the build folder will be updated to the AWS bucket as below:

```
build (16.x)
succeeded 1 hour ago in 1m 40s
Search logs
```

> Set up job 3s

> Build jakejarvis/s3-sync-action@master 23s

> Run actions/checkout@v3 1s

> Use Node.js 16.x 1s

> Yarn Install 28s

> Production Build 39s

> Deploy to S3 4s

1 ▶ Run jakejarvis/s3-sync-action@master

11 /usr/bin/docker run --name c5cfbbf60417343a3bc91d18530fa9f07\_468901 --label 49859c --workdir /github/workspace --rm -e "CI" -e "AWS\_S3\_BUCKET" -e "AWS\_ACCESS\_KEY\_ID" -e "AWS\_SECRET\_ACCESS\_KEY" -e "AWS\_REGION" -e "INPUT\_ARGS" -e "HOME" -e "GITHUB\_JOB" -e "GITHUB\_REF" -e "GITHUB\_SHA" -e "GITHUB\_REPOSITORY" -e "GITHUB\_REPOSITORY\_OWNER" -e "GITHUB\_REPOSITORY\_OWNER\_ID" -e "GITHUB\_RUN\_ID" -e "GITHUB\_RUN\_NUMBER" -e "GITHUB\_RETENTION\_DAYS" -e "GITHUB\_RUN\_ATTEMPT" -e "GITHUB\_ACTOR\_ID" -e "GITHUB\_ACTOR" -e "GITHUB\_TRIGGERING\_ACTOR" -e "GITHUB\_WORKFLOW" -e "GITHUB\_HEAD\_REF" -e "GITHUB\_BASE\_REF" -e "GITHUB\_EVENT\_NAME" -e "GITHUB\_SERVER\_URL" -e "GITHUB\_API\_URL" -e "GITHUB\_GRAPHQL\_URL" -e "GITHUB\_REF\_NAME" -e "GITHUB\_REF\_PROTECTED" -e "GITHUB\_REF\_TYPE" -e "GITHUB\_WORKFLOW\_REF" -e "GITHUB\_WORKFLOW\_SHA" -e "GITHUB\_WORKSPACE" -e "GITHUB\_ACTION" -e "GITHUB\_EVENT\_PATH" -e "GITHUB\_ACTION\_REPOSITORY" -e "GITHUB\_ACTION\_REF" -e "GITHUB\_PATH" -e "GITHUB\_ENV" -e "GITHUB\_STEP\_SUMMARY" -e "GITHUB\_STATE" -e "GITHUB\_OUTPUT" -e "RUNNER\_OS" -e "RUNNER\_ARCH" -e "RUNNER\_NAME" -e "RUNNER\_TOOL\_CACHE" -e "RUNNER\_TEMP" -e "RUNNER\_WORKSPACE" -e "ACTIONS\_RUNTIME\_URL" -e "ACTIONS\_RUNTIME\_TOKEN" -e "ACTIONS\_CACHE\_URL" -e GITHUB\_ACTIONS=true -v "/var/run/docker.sock":"/var/run/docker.sock" -v "/home/runner/work/\_temp/\_github\_home":"/github/home" -v "/home/runner/work/\_temp/\_github\_workflow":"/github/workflow" -v "/home/runner/work/\_temp/\_runner\_file\_commands":"/github/file\_commands" -v "/home/runner/work/c5fbbf60417343a3bc91d18530fa9f07 --acl public-read --delete

12 upload: build/asset-manifest.json to s3://\*\*\*\*/asset-manifest.json

13 upload: build/lib/animate/animate.min.css to s3://\*\*\*\*/lib/animate/animate.min.css

14 upload: build/lib/animate/animate.css to s3://\*\*\*\*/lib/animate/animate.css

15 upload: build/lib/bootstrap/css/bootstrap.css to s3://\*\*\*\*/lib/bootstrap/css/bootstrap.css

16 upload: build/lib/bootstrap/css/bootstrap.min.css to s3://\*\*\*\*/lib/bootstrap/css/bootstrap.min.css

17 upload: build/lib/bootstrap/css/bootstrap.min.css.map to s3://\*\*\*\*/lib/bootstrap/css/bootstrap.min.css.map

18 upload: build/index.html to s3://\*\*\*\*/index.html

19 upload: build/gitlab.svg to s3://\*\*\*\*/gitlab.svg

20 upload: build/css/style-orange.css to s3://\*\*\*\*/css/style-orange.css

21 upload: build/js/main.js to s3://\*\*\*\*/js/main.js

22 upload: build/css/style-sky-blue.css to s3://\*\*\*\*/css/style-sky-blue.css

23 upload: build/lib/bootstrap/js/bootstrap.js to s3://\*\*\*\*/lib/bootstrap/js/bootstrap.js

24 upload: build/css/style-green.css to s3://\*\*\*\*/css/style-green.css

25 upload: build/css/style-red.css to s3://\*\*\*\*/css/style-red.css

# Automatic Cloudfront invalidation with Amazon Lambda:

Create a lambda function and add a trigger to AWS S3.

If the AWS S3 is updated objects, the trigger will be called the lambda function and this will create a Cloudfront invalidation to clear cache.

The screenshot shows the AWS Lambda console interface. At the top, there is a navigation bar with tabs: Code, Test, Monitor, Configuration, Aliases, and Versions. The 'Code' tab is currently selected. Below the navigation bar, there is a main workspace divided into several sections. On the left, there is a sidebar labeled 'Environment' with a search bar 'Go to Anything (⌘ P)' and a tree view showing a folder 'CloudFrontFunction' containing a file 'lambda\_function.py'. In the center, there is a code editor window titled 'lambda\_function' with the following Python code:

```
from __future__ import print_function
import boto3
import time

def lambda_handler(event, context):
    path = []
    for items in event["Records"]:
        if items["s3"]["object"]["key"] == "index.html":
            path.append("/")
    if path:
        client = boto3.client('cloudfront')
        invalidation = client.create_invalidation(DistributionId = 'E3PZH5RQJ07NZH',
                                                InvalidationsBatch={
                                                    'Paths': {
                                                        'Quantity': 1,
                                                        'Items': path
                                                    },
                                                    'CallerReference': str(time.time())
                                                })
    return {"status": "success"}
```

On the right side of the workspace, there are two buttons: '+ Add destination' and '+ Add trigger'. The '+ Add trigger' button is highlighted with a red box. Below the workspace, there is a horizontal bar with buttons for File, Edit, Find, View, Go, Tools, Window, Test, Deploy, and a magnifying glass icon.

---

## Resources:

**S3:** <https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html>

**Lambda** [https://aws.amazon.com/lambda/getting-started/?trk=dca4b539-ba5f-4c78-bd55-e8e5f7a26221&sc\\_icampaign=lambda\\_ict\\_gs\\_functions&sc\\_icontent=awssm-11768\\_engage&sc\\_iplace=aws-console-lambda](https://aws.amazon.com/lambda/getting-started/?trk=dca4b539-ba5f-4c78-bd55-e8e5f7a26221&sc_icampaign=lambda_ict_gs_functions&sc_icontent=awssm-11768_engage&sc_iplace=aws-console-lambda)

**SNS:** <https://docs.aws.amazon.com/sns/latest/dg/welcome.html>

**API Gateway:** <https://docs.aws.amazon.com/apigateway/latest/developerguide/welcome.html>

**AWS Cognito:** [https://docs.amazonaws.cn/en\\_us/cognito/latest/developerguide/what-is-amazon-cognito.html](https://docs.amazonaws.cn/en_us/cognito/latest/developerguide/what-is-amazon-cognito.html)

**Route53:** <https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/Welcome.html>

**DynamoDB:**

<https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Introduction.html>

**Integration between ReactJS app and AWS Cognito:**

<https://www.npmjs.com/package/amazon-cognito-identity-js>

**Github:** <https://github.com/luan-tran-89/cs516/tree/main/final-project>

---

## Reference Supported Links:

**GitHub WorkFlow:** <https://docs.github.com/en/actions/quickstart>

**GitHub Action to Sync S3 Bucket:** <https://github.com/jakejarvis/s3-sync-action>

**Automatic Cloudfront invalidation with Amazon Lambda:**

[https://blog.miguelangelnieto.net/posts/Automatic\\_Cloudfront\\_invalidation\\_with\\_Amazon\\_Lambda.html](https://blog.miguelangelnieto.net/posts/Automatic_Cloudfront_invalidation_with_Amazon_Lambda.html)