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.

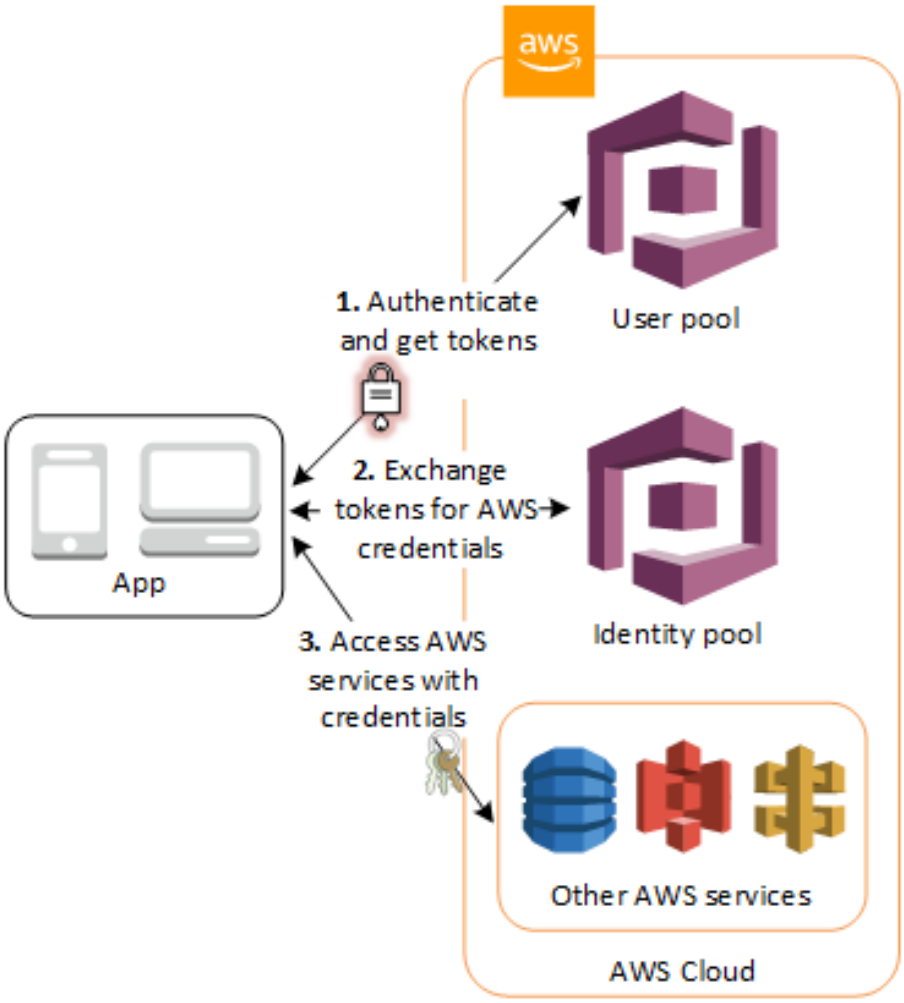
What is Amazon Cognito?

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:

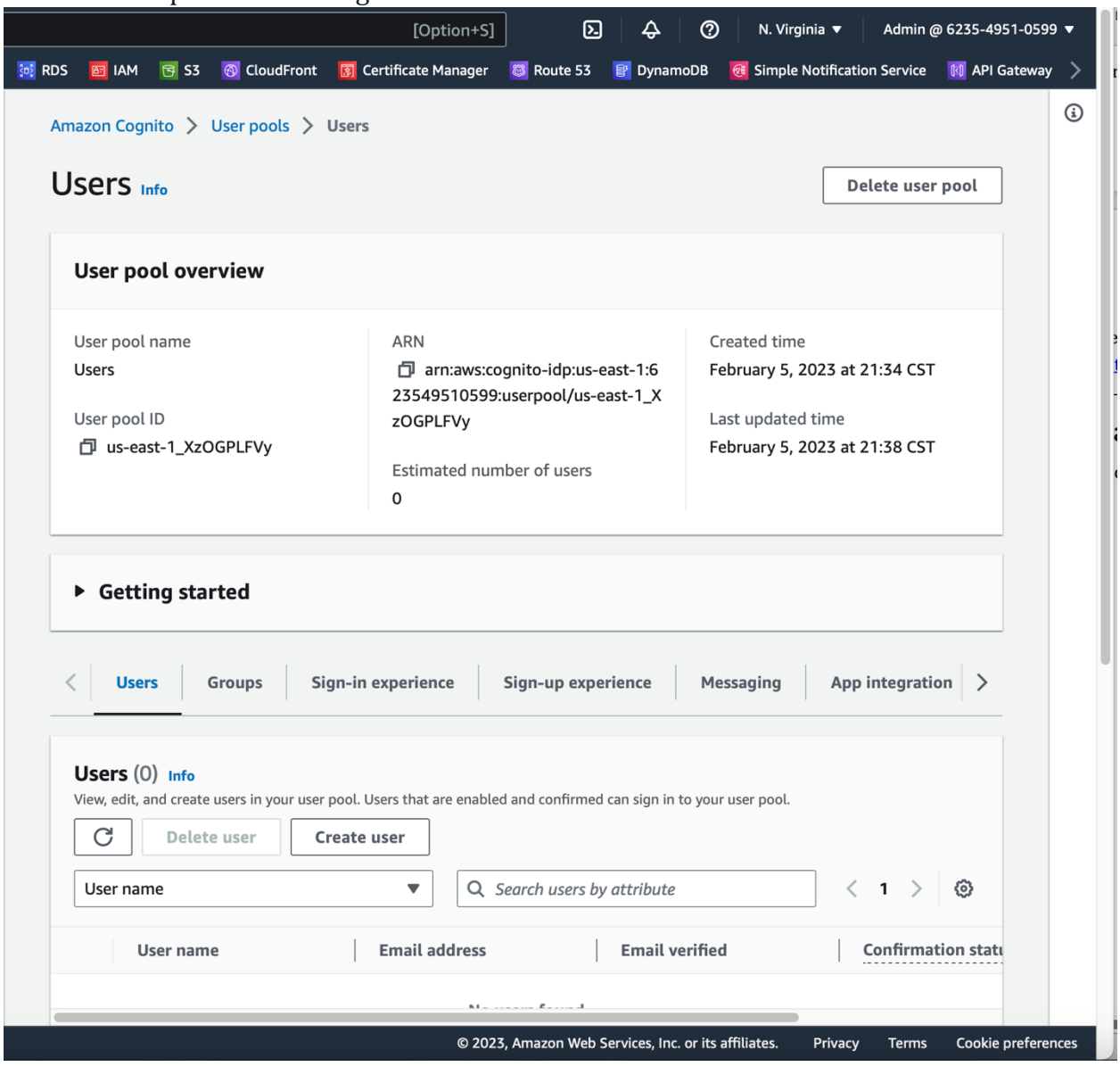
- Amazon Cognito user pools
- Social sign-in with Facebook, Google, Login with Amazon, and Sign in with Apple
- OpenID Connect (OIDC) providers
- SAML identity providers
- Developer authenticated identities

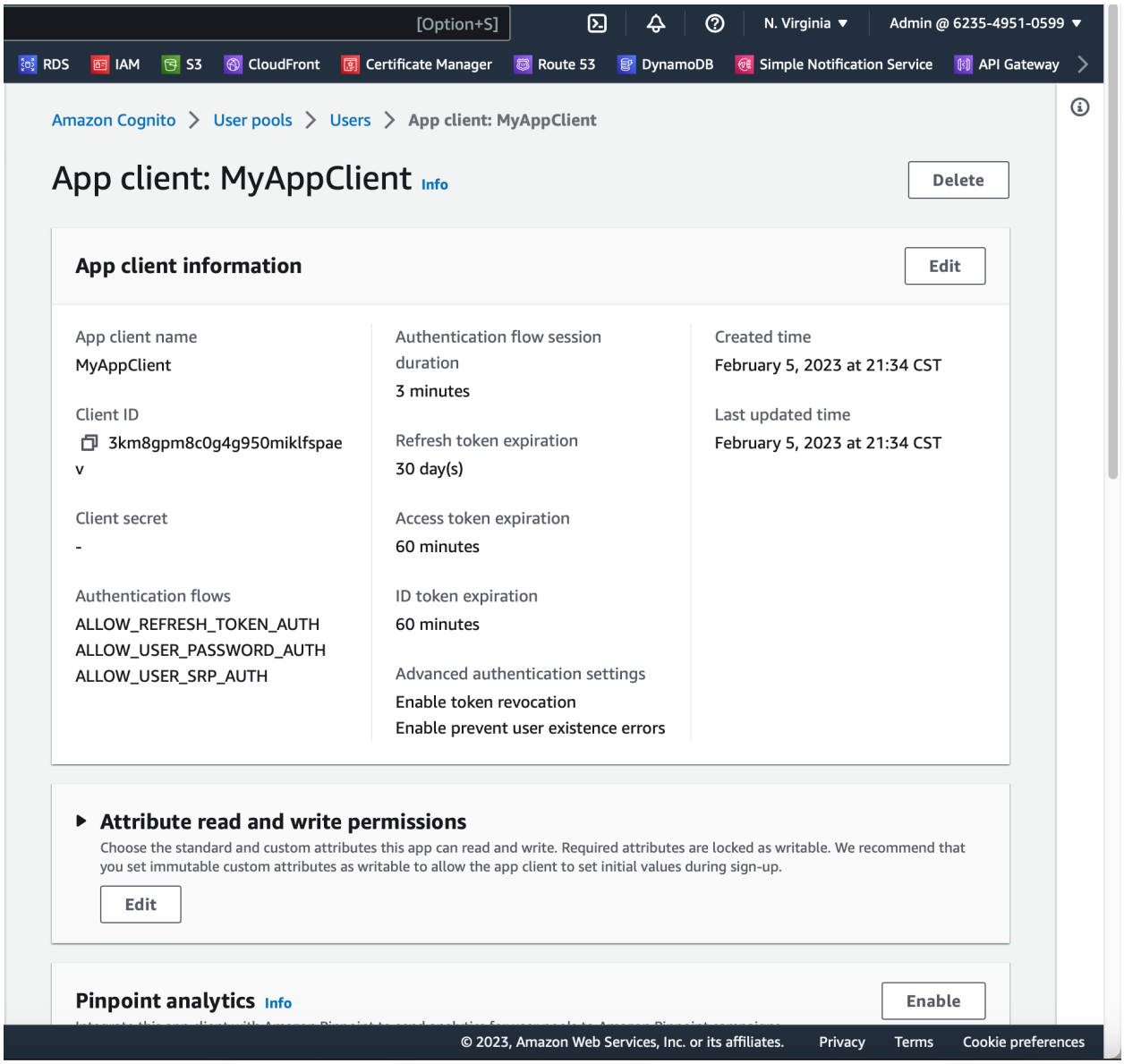
To save user profile information, your identity pool needs to be integrated with a user pool.

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

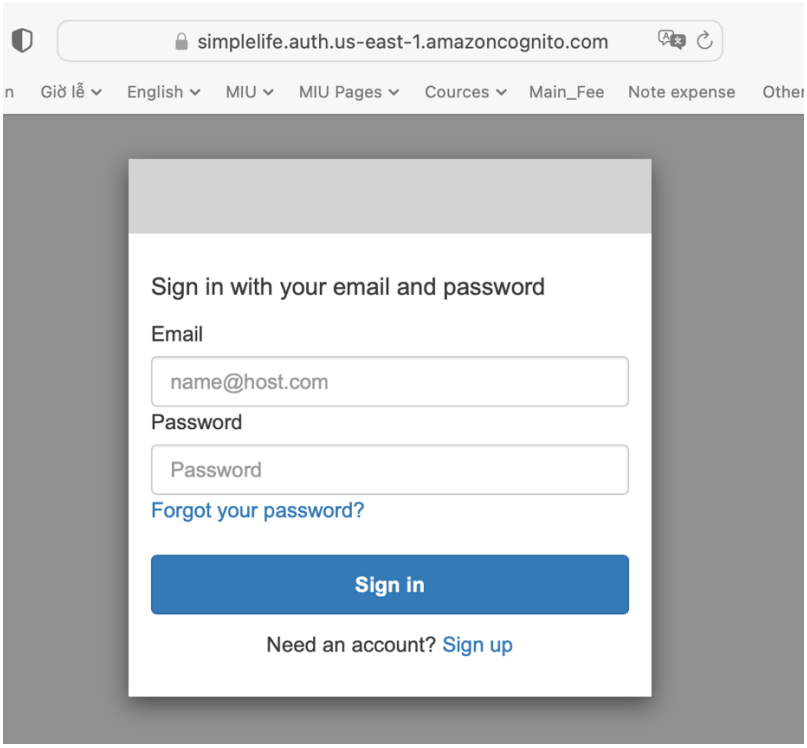
Implementation Example

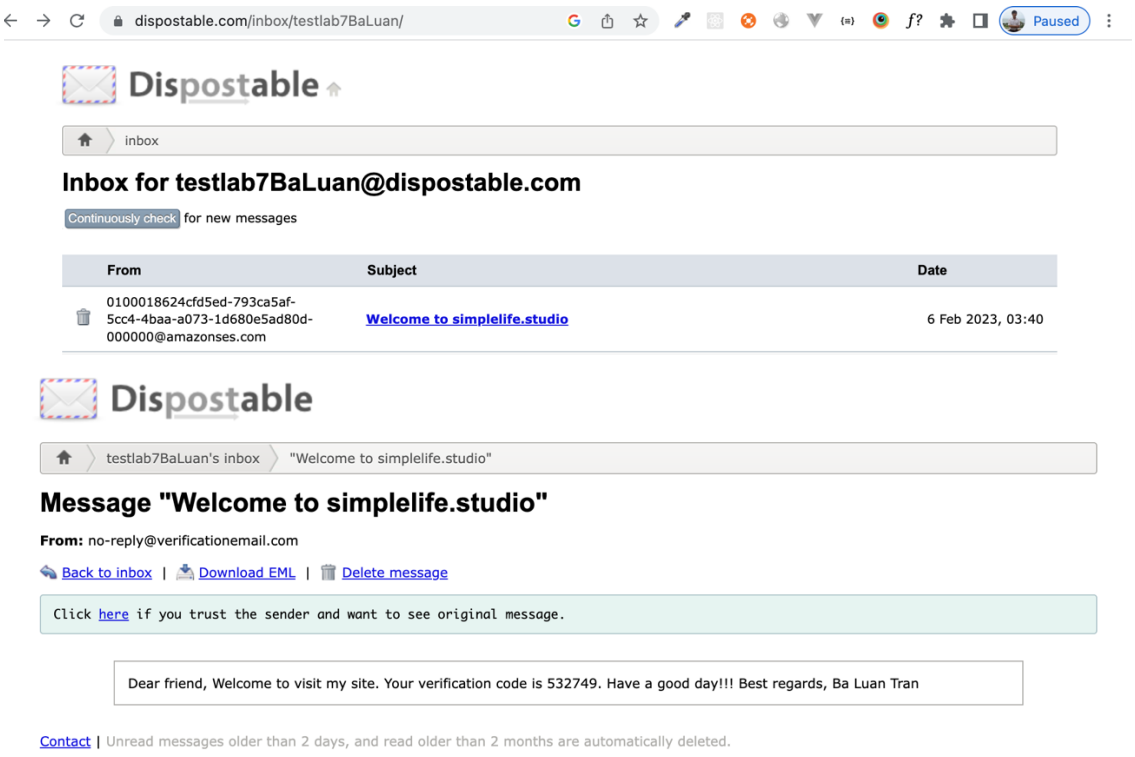
1. Create a user pool in AWS Cognito:



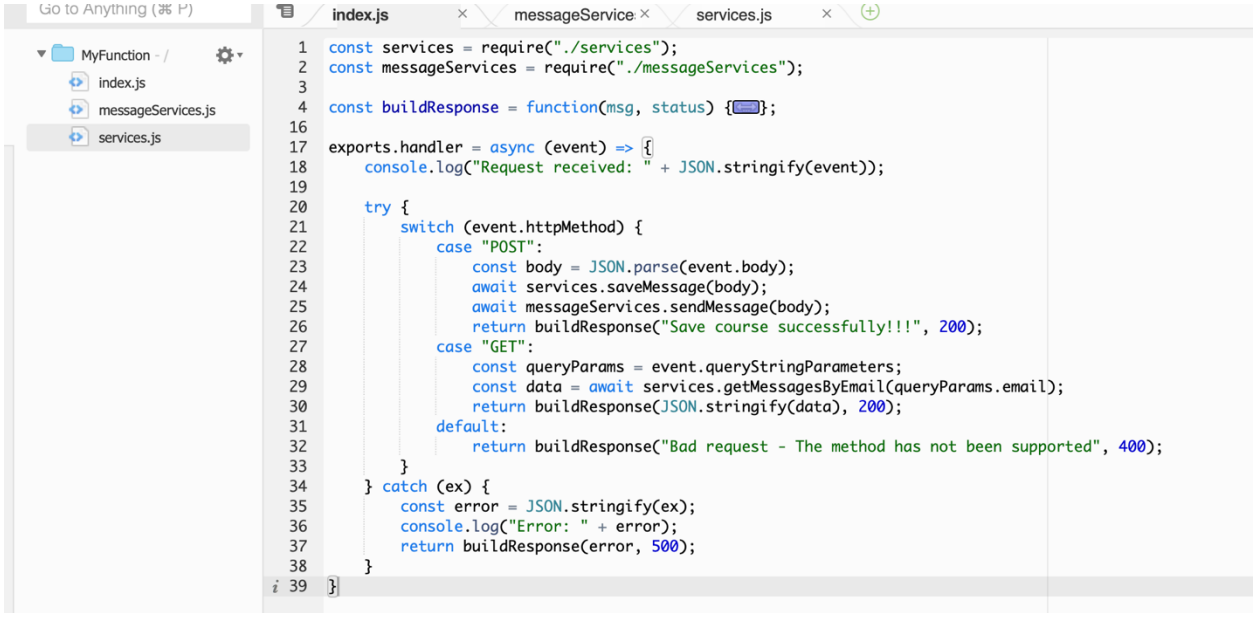

[Option+S]
N. Virginia
Admin @ 6235-4951-0599
RDS
IAM
S3
CloudFront
Certificate Manager
Route 53
DynamoDB
Simple Notification Service
API Gateway
Amazon Cognito > User pools > Users > App client: MyAppClient
App client: MyAppClient Info
Delete
App client information
Edit
App client name
MyAppClient
Client ID
3km8gpm8c0g4g950miklfspaev
Client secret
-
Authentication flows
ALLOW_REFRESH_TOKEN_AUTH
ALLOW_USER_PASSWORD_AUTH
ALLOW_USER_SRP_AUTH
Authentication flow session duration
3 minutes
Refresh token expiration
30 day(s)
Access token expiration
60 minutes
ID token expiration
60 minutes
Advanced authentication settings
Enable token revocation
Enable prevent user existence errors
Created time
February 5, 2023 at 21:34 CST
Last updated time
February 5, 2023 at 21:34 CST
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

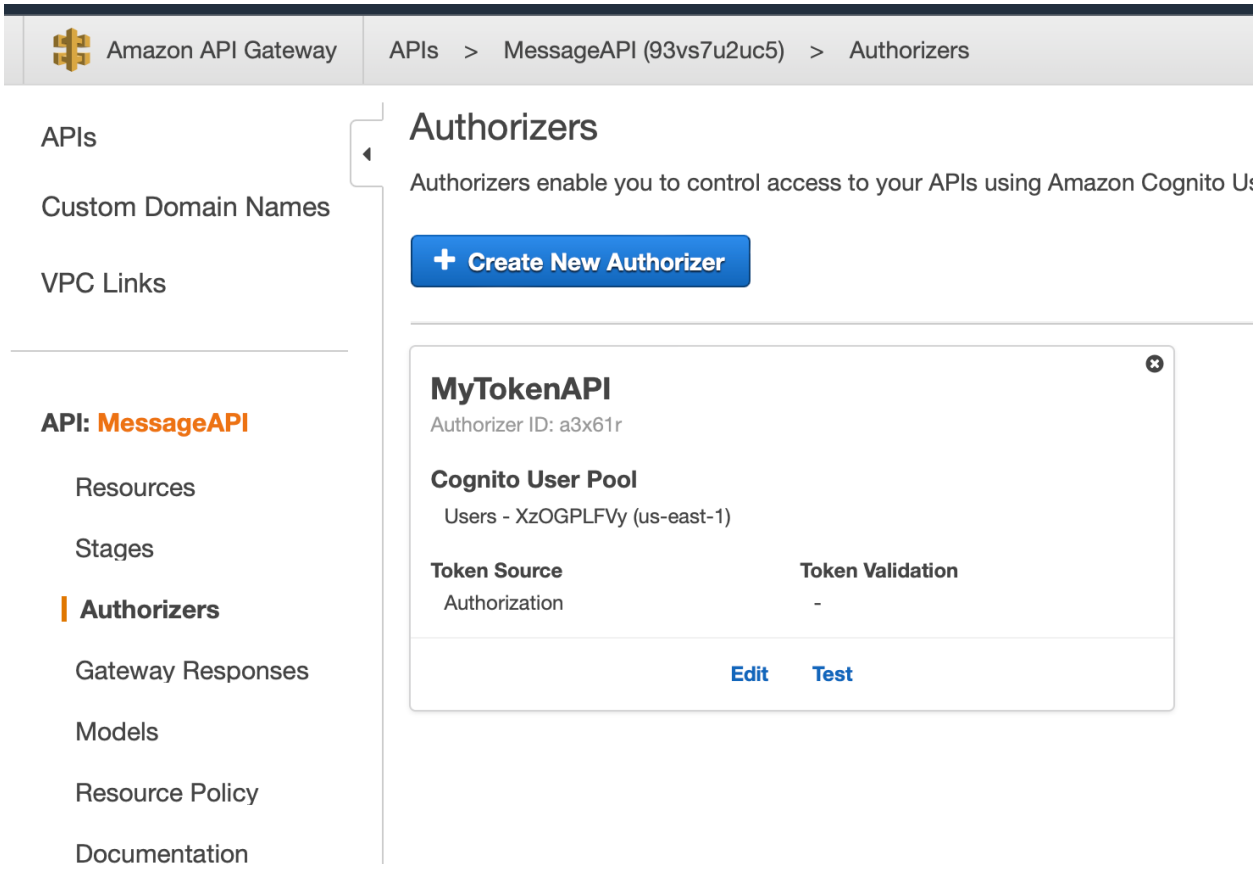


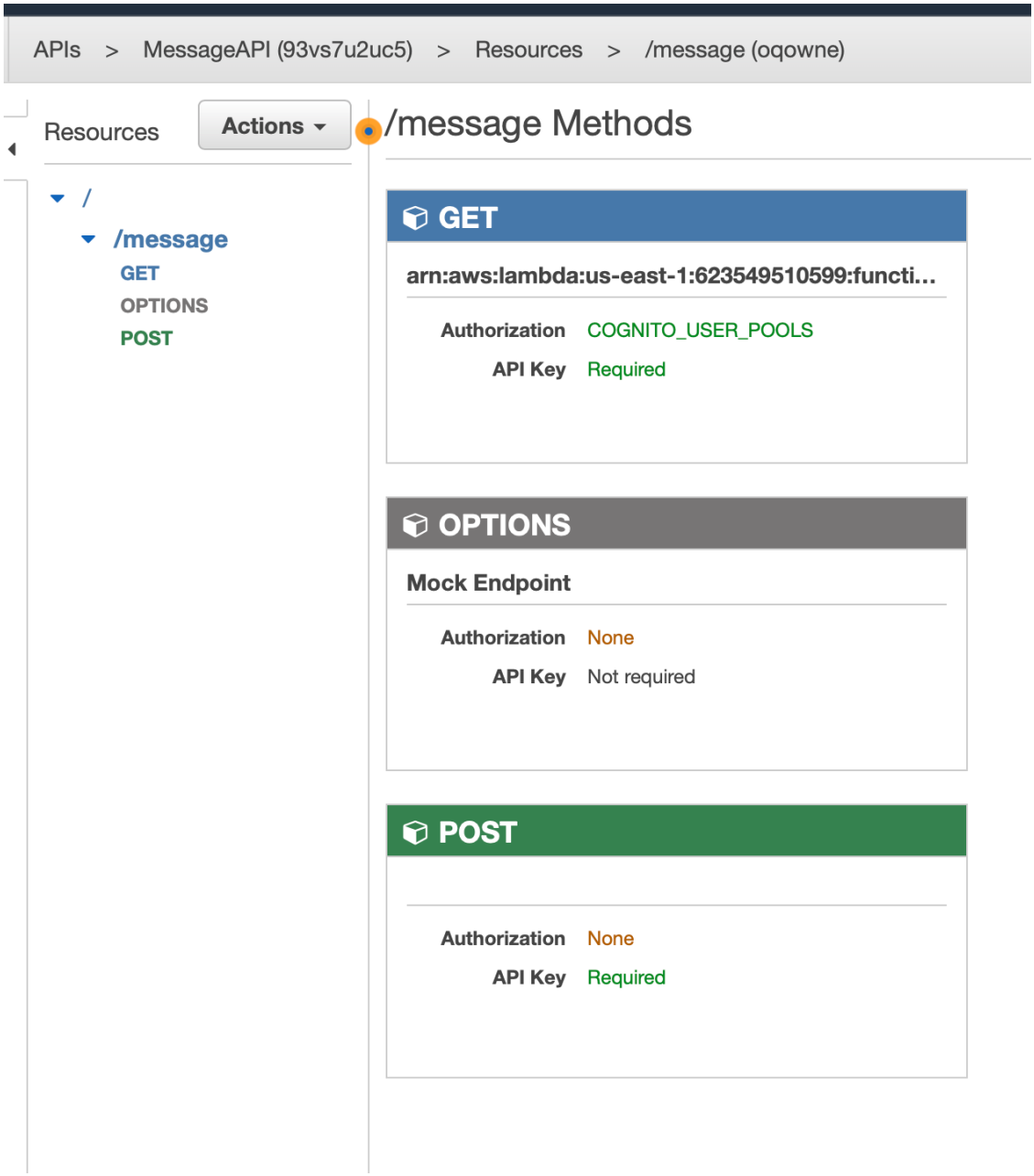


2. Create a lambda to get/save messages from the website and create API Getway to the lambda



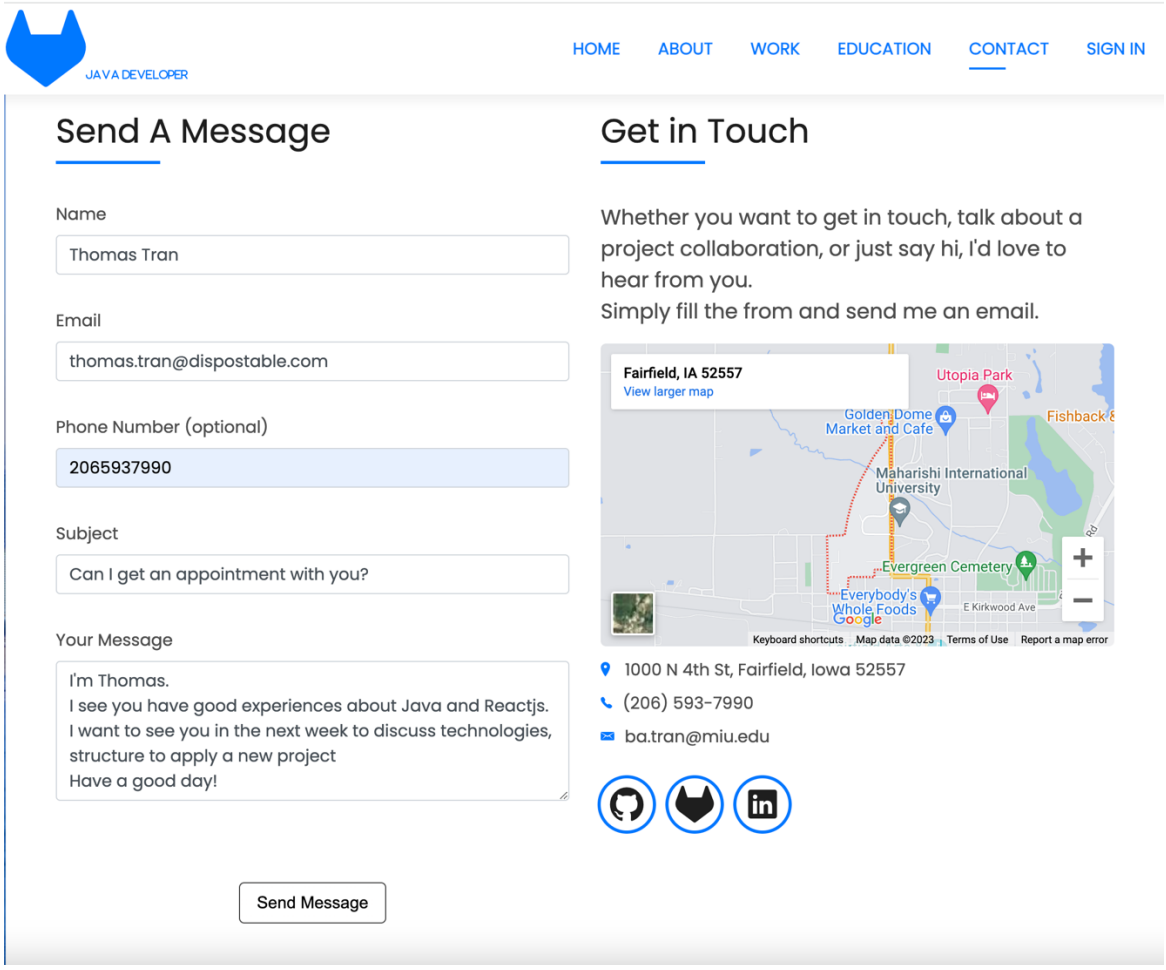
3. Create an Authorizers of the Cognito User Pool and assign to the API Gateway



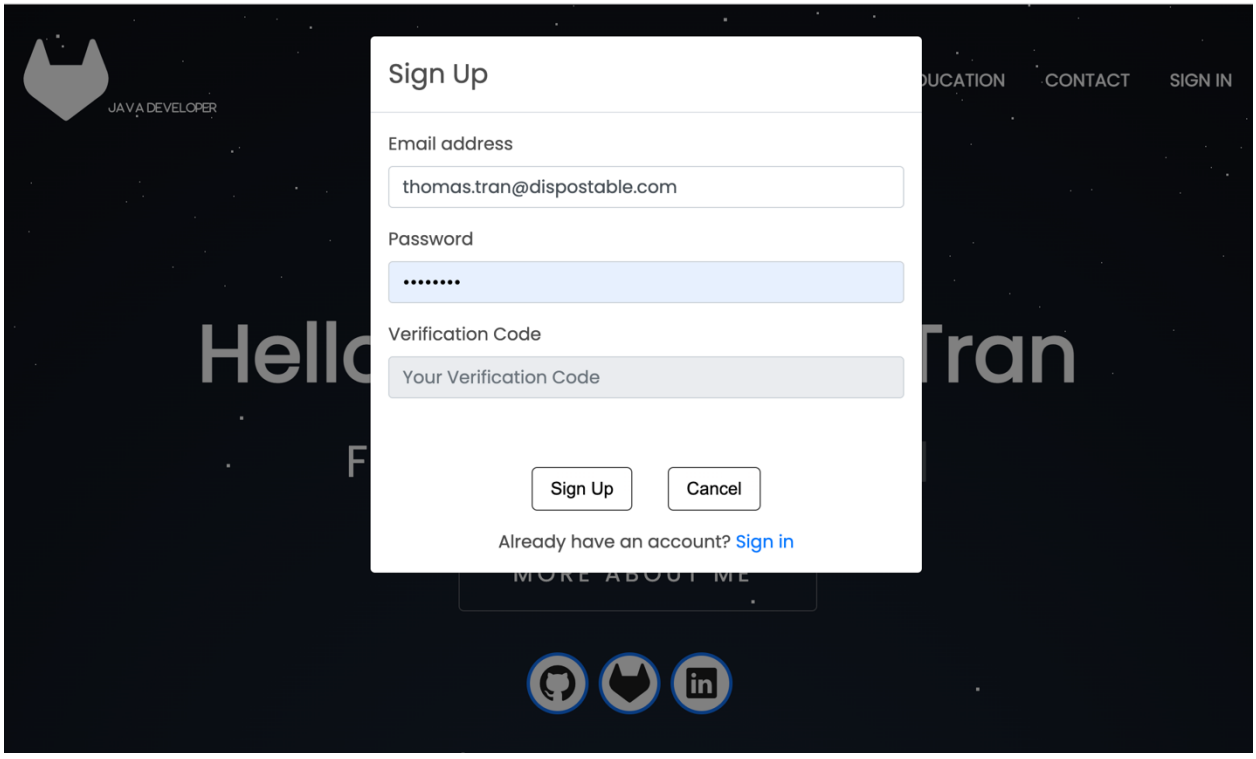

APIs > MessageAPI (93vs7u2uc5) > Resources > /message (oqowne)
Resources
Actions
/message Methods
/
/message
GET
OPTIONS
POST
GET
arn:aws:lambda:us-east-1:623549510599:functi...
Authorization COGNITO_USER_POOLS
API Key Required
OPTIONS
Mock Endpoint
Authorization None
API Key Not required
POST
Authorization None
API Key Required

Integrating Amazon Cognito into a website

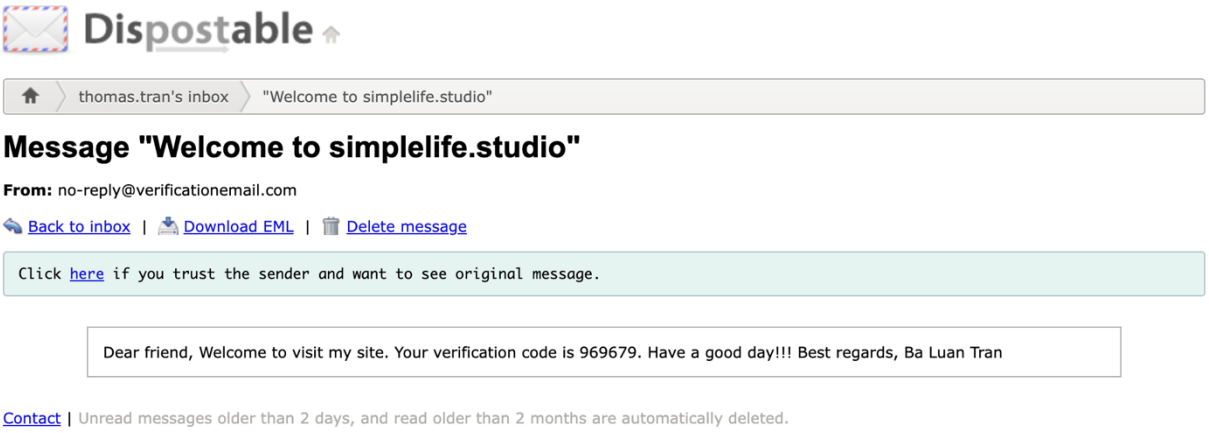
1. Send some messages in the site (https://www.simplelife.studio/)



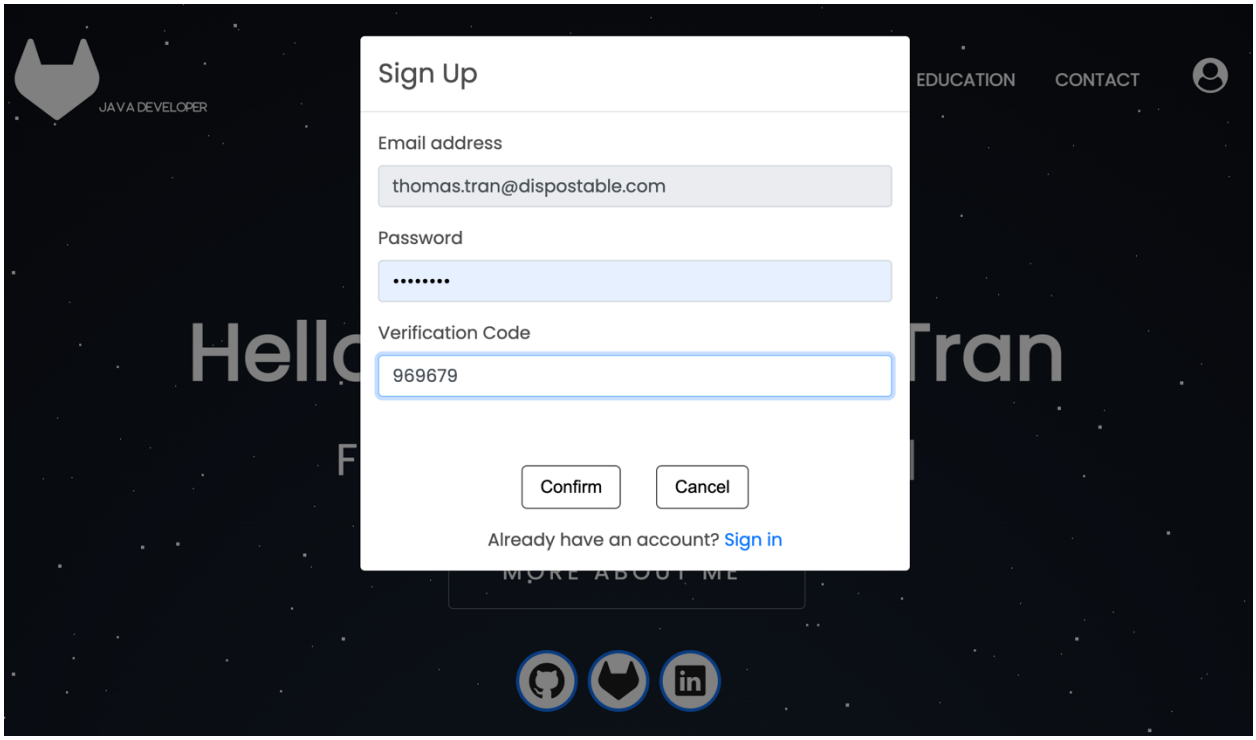
2. Sign up to the site with an email which used to send the messages



- An verification code will be sent to your email

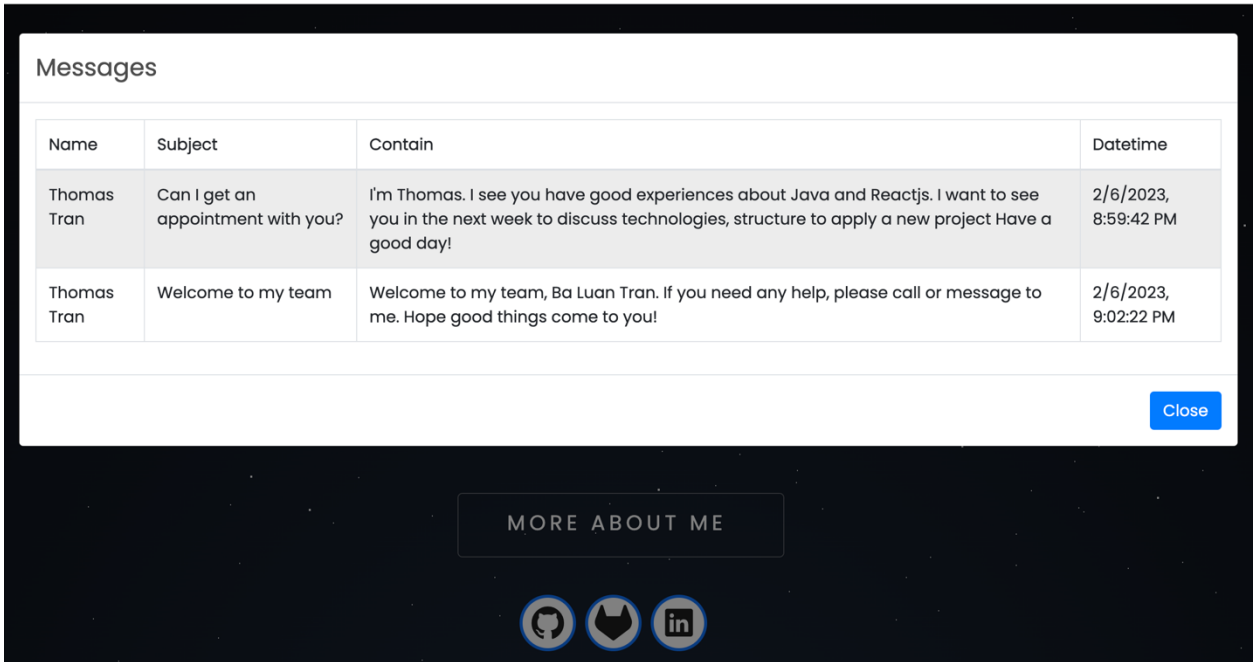


- Enter the code and click Confirm to finish the Sign Up process

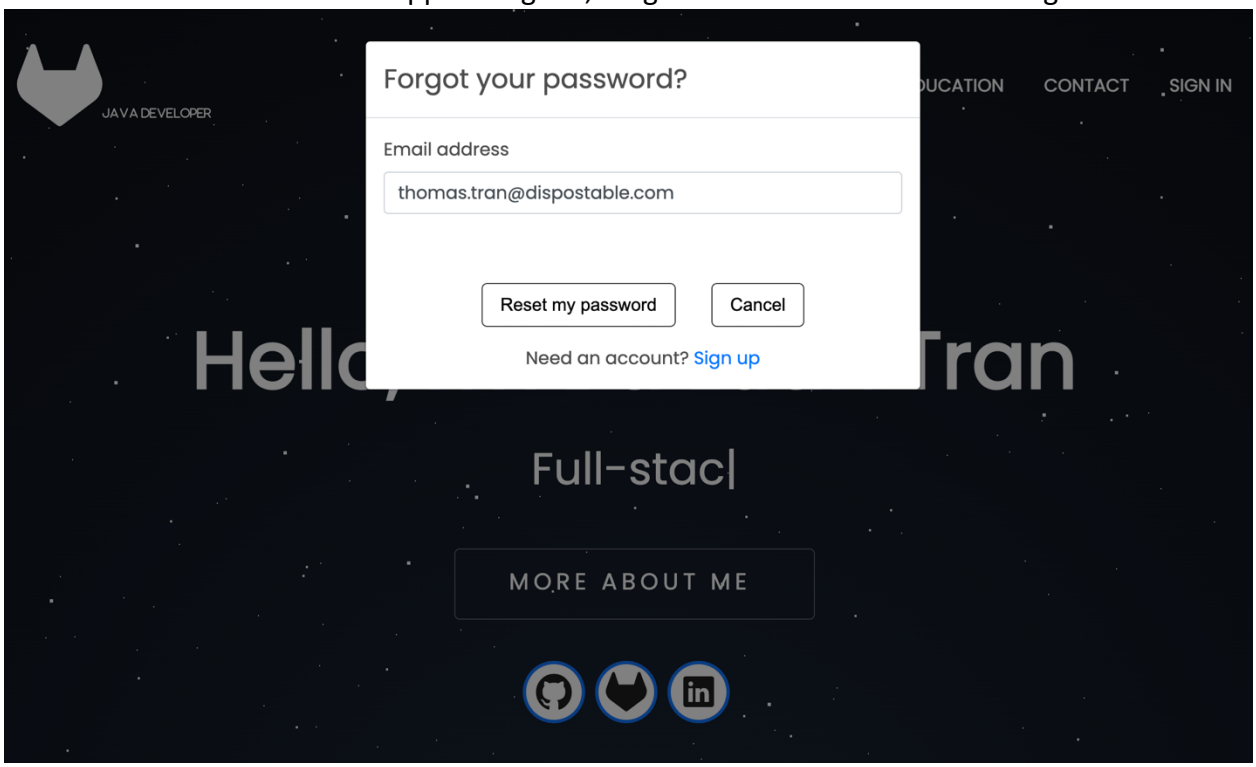


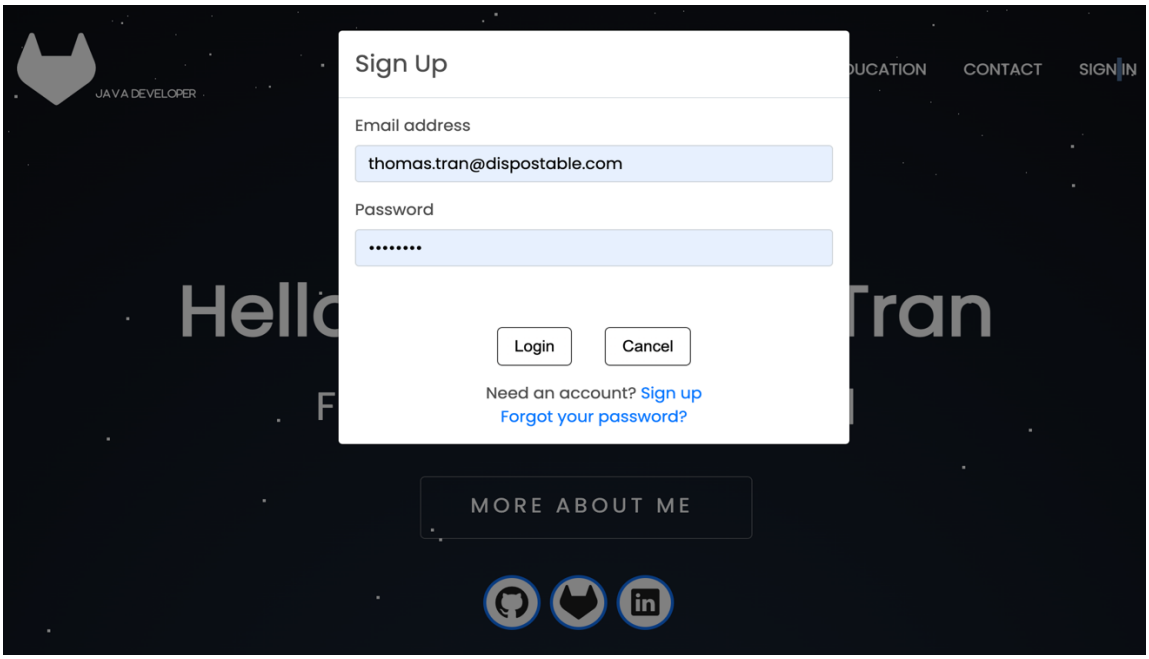
3. After successful registration, we will see the Profile icon in the header page

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



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




JAVA DEVELOPER
UCATION
CONTACT
SIGN IN
Sign Up
Email address
thomas.tran@dispostable.com
Password
Login
Cancel
Need an account? Sign up
Forgot your password?
Hello
Tran
MORE ABOUT ME

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

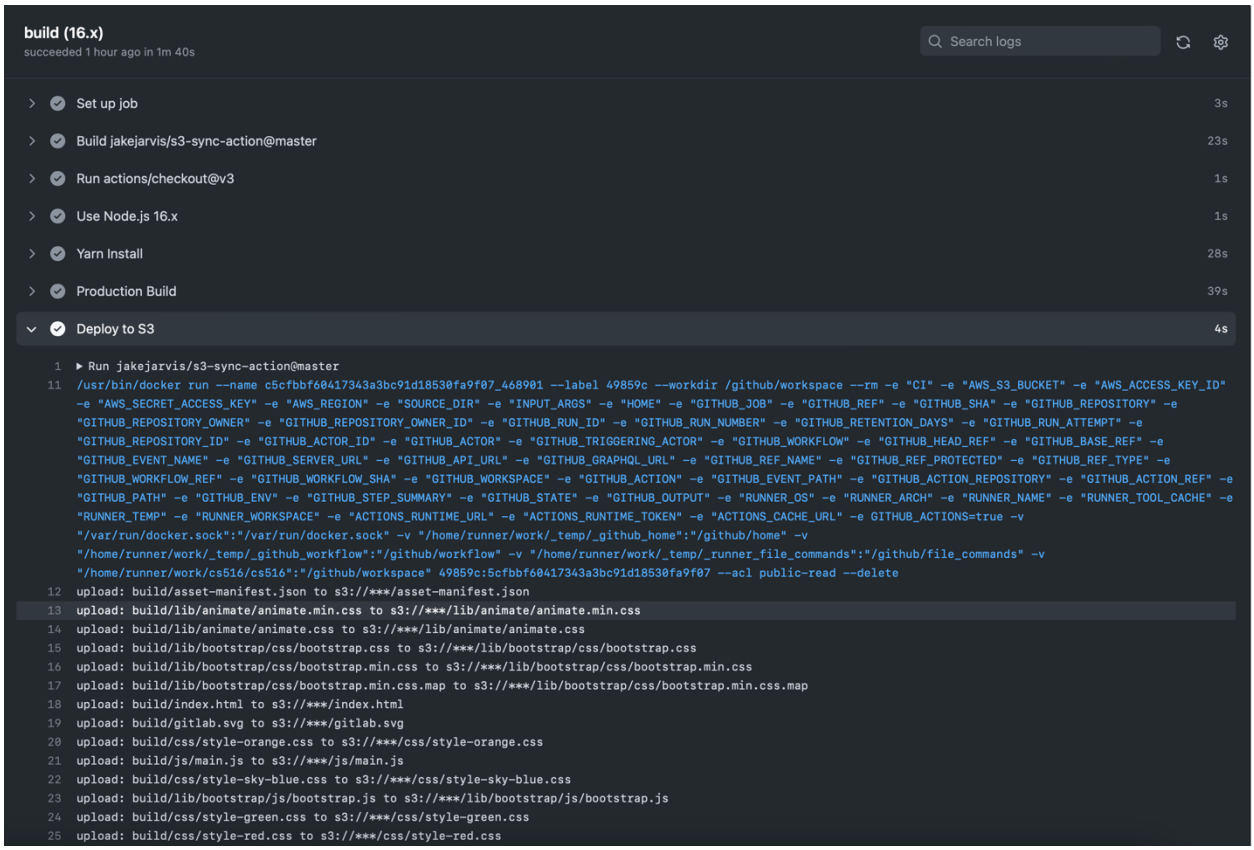
luan-tran-89 Update prod.yml ✓

1 contributor

38 lines (36 sloc) | 915 Bytes

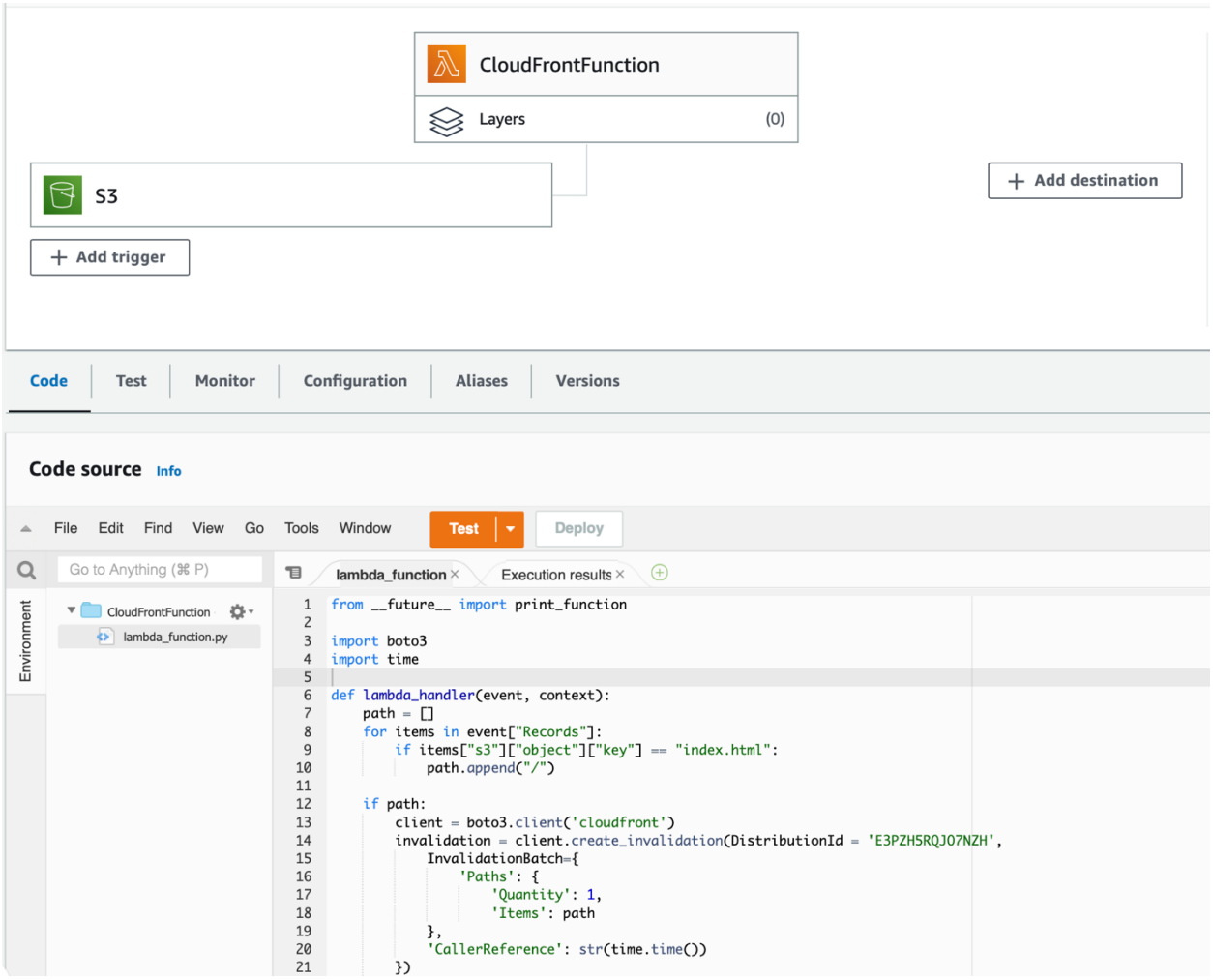
```yaml
name: Production Build
on:
  pull_request:
  push:
    branches:
      - main
env:
  CI: "false"
jobs:
  build:
    runs-on: ubuntu-latest

    strategy:
      matrix:
        node-version: [ 16.x ]

    steps:
    - uses: actions/checkout@v3
    - name: Use Node.js ${{ matrix.node-version }}
      uses: actions/setup-node@v3
      with:
        node-version: ${{ matrix.node-version }}
    - name: Yarn Install
      run: |
        yarn install
    - name: Production Build
      run: |
        yarn build
    - name: Deploy to S3
      uses: jakejarvis/s3-sync-action@master
      with:
        args: --acl public-read --delete
      env:
        AWS_S3_BUCKET: ${{ secrets.AWS_S3_BUCKET }}
        AWS_ACCESS_KEY_ID: ${{ secrets.AWS_ACCESS_KEY_ID }}
        AWS_SECRET_ACCESS_KEY: ${{ secrets.AWS_SECRET_ACCESS_KEY }}
        AWS_REGION: ${{ secrets.AWS_REGION }}
        SOURCE_DIR: "build"
```

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



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.



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

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

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