Team Amazonian Prime

Installation Documentation

April. 4, 2023
Version 1.0

Document Information

Revision History

Date	Version	Status	Prepared by	Comments
April 4, 2023	1.0	Submitted	Amazonian Prime	

Document Control

Role	Name	E-mail	Telephone
Professor	Jerry Jim		
TA	Marie Salomon		
TA	Shijun Shen		
Project Sponsor	Peter Smith		
	Mahmoud Al Khatib	mahmoudalkhatib.ubc@gmail.com	
	Michael He	michaelhe17@gmail.com	
	Joshua Luong	joshualuong@hotmail.com	
	Tristan Martinuson	tmartinuson@gmail.com	
	Elaine Shi	elaineshi328@gmail.com	
	William Suryawidjaja	suryawidjajaw@gmail.com	

Approval

Role	Name	Signature	Sign-off Date
Amazonian Prime Member	Mahmoud Al Khatib	Mahmoud Al Khatib	April 4, 2023
Amazonian Prime Member	Michael He	Michael He	April 4, 2023
Amazonian Prime Member	Joshua Luong	Joshua Luong	April 4, 2023
Amazonian Prime Member	Tristan Martinuson	Tristan Martinusion	April 4, 2023
Amazonian Prime Member	Elaine Shi	Elaine Shi	April 4, 2023
Amazonian Prime Member	William Suryawidjaja	William Suryawidjaja	April 4, 2023

Table of Contents

Document Information **2**
Table of Contents **3**
Introduction **4**
Programming Environment **5**
Programming Tools and Languages 5
AWS Services 5
Getting Started **6**
Registering with AWS 6
Setup Development Environment **7**
Installing VSCode 7
Installing AWS CLI 7
Installing AWS SAM CLI 7
Configuring AWS Credentials 7
Github Actions Deployment **8**
Creating Pipeline AWS Resources 8
Setting up AWS Credential Secrets on Github 8
Verifying Build Deployments 10
Viewing a Deployment on AWS 11
Reconfiguring Google OAuth **12**
Registering for Cloud Platform Account 12
Integrating Client ID into the System 16
SES Email Verification **17**
Adding Receiver Accounts 17
Adding Sender Accounts 17
Configuration Document **18**
Invalidate Cache 18

Introduction

This document provides a comprehensive guide for installing and deploying the necessary services required to run Amazonian Prime, a web-based e-commerce platform that provides a range of services, including online shopping, digital storefront, and administration.

Amazonian Prime is a highly scalable and reliable platform that requires a set of robust and efficient services to support its functionality. These AWS services include RDS, S3, Lambda, Step Functions, CloudFront and CloudFormation. This document aims to provide a step-by-step guide to help system administrators and developers set up and deploy these services in a production environment.

The document assumes that the readers have a basic understanding of command line navigation, networking, and system administration. It provides detailed instructions on how to set up the necessary services using AWS SAM CLI and the AWS Web Console.

The installation and deployment process covered in this document involves several stages, including system configuration, service installation, and integration. Each stage is described in detail, with clear instructions on how to set up and configure the required services.

By following the instructions in this document, system administrators and developers can set up their own production environment of Amazonian Prime. This will ensure that the user can explore their very own environment of Amazonian Prime that can handle high traffic volumes and provide a reliable and seamless user experience.

This document serves as a valuable resource for those looking to set up and deploy Amazonian Prime. It provides a comprehensive guide to help ensure a smooth and successful installation and deployment process.

Programming Environment

Programming Tools and Languages

Operating System: Windows 10

Visual Studio Code for IDE (version 1.75)

- https://code.visualstudio.com/download
- Extensions: AWS Toolkit, ES7+ React/Redux/React-Native snippets

Yarn v.1.22.19

- https://classic.yarnpkg.com/lang/en/docs/install/

BackEnd
Language: Nodejs (version 18.14.0) for Lambda
AWS Services:

- AWS API Gateway
- AWS Step Functions
- AWS Aurora Database MySQL (MySQL engine version 8.0)
- AWS S3
- AWS CloudFormation
- AWS SES
- AWS SNS & SQS

Libraries:

- Google OAuth (google-auth-library version 8.7.0)

Tools/Platforms:

- Docker (version 4.16.3)
- Git (version 2.39.1)

FrontEnd
Language: Typescript (version 4.4.2)
Framework:

- React (version 18.2.0)
- React Dom (version 18.2.0)
- Redux (version 5.0.0)
- SASS (version 1.58.0)

Libraries:

- Material UI Design (version 5.11.8)
- Axios (version 1.3.2)

AWS Services

AWS CLI - https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html

AWS SAM CLI -
https://docs.aws.amazon.com/serverless-application-model/latest/developerguide/install-sam-cli.html

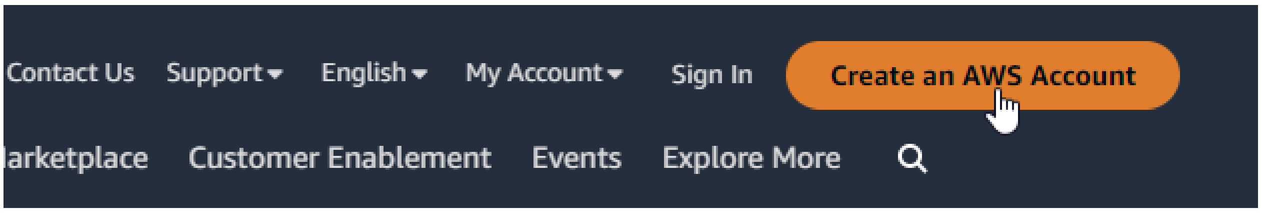
AWS Account - https://aws.amazon.com/console/

Lambda, S3, RDS, VPC, API Gateway, Step Function

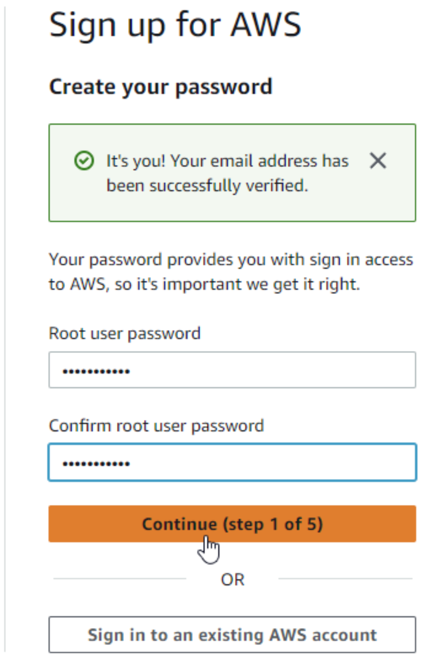
Getting Started

Registering with AWS

1. Go to the AWS website (https://aws.amazon.com/console/) and click on the "Create an AWS Account" button.



2. Enter your email address and name and choose to verify your email. Enter the verification code sent to your inbox. Then click on "Verify".



Sign up for AWS

Create your password

It's you! Your email address has been successfully verified.

Your password provides you with sign in access to AWS, so it's important we get it right.

Root user password: •••••••••••

Confirm root user password: •••••••••••

3. Enter your desired root user password as well as other personal information AWS asks you for, such as your name, address, and phone number. Select for "Personal" use when prompted.
4. Enter your payment information. You can use a credit card to pay for your AWS usage.
5. Review the AWS Customer Agreement and click through to finalize creating an account.
6. Once you've confirmed your account, you can sign in to the AWS Management Console. From here, you can access all of the AWS services and resources available to you.

Setup Development Environment

Installing VSCode

1. Go to the VSCode website (https://code.visualstudio.com/download) and click on the corresponding download button for your operating system (For the purposes of this document we will assume Windows).

2. Once the download is complete, open the downloaded file to begin the installation process.

3. Follow the installation wizard prompts to complete the installation. On a Windows machine, this involves clicking "Next" through the wizard, agreeing to the license terms, selecting the installation location (or accepting the default), and selecting any additional tasks (such as adding VSCode to your PATH environment variable).

4. Once the installation is complete, launch VSCode. On Windows, you can do this by double-clicking on the VSCode icon on your desktop or in your Start menu.

5. You can now start using VSCode to view, write and edit code.

Installing AWS CLI

1. Click on the following guide:
   https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html

2. Follow the instructions found under your current operating system

3. Verify AWS CLI is installed correctly with the following command:
   ```
   aws --version
   ```

Installing AWS SAM CLI

1. Click on the following guide:
   https://docs.aws.amazon.com/serverless-application-model/latest/developerguide/install-sam-cli.html

2. Follow the instructions found under your current operating system

3. Verify AWS SAM CLI is installed correctly with the following command:
   ```
   sam --version
   ```

Configuring AWS Credentials

1. Click on the following guide:
   https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-files.html

2. Follow the instructions found to add your AWS credentials to AWS SAM CLI and AWS CLI

GitHub Actions Deployment

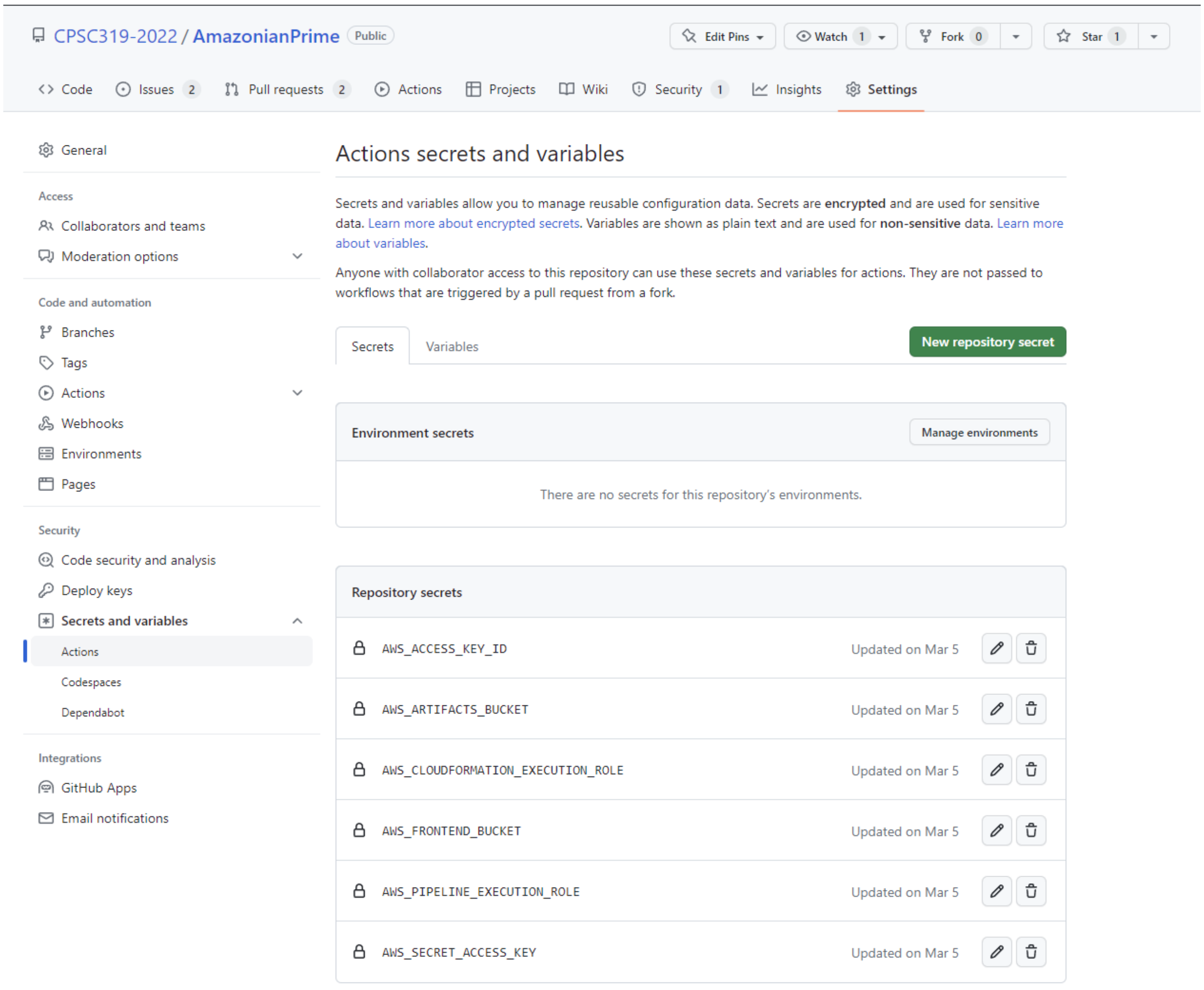
Creating Pipeline AWS Resources

1. In your preferred console instance with AWS SAM installed, navigate to ~\backend and run the following command:
    a. `sam pipeline bootstrap --no-interactive --stage dev --region us-west-2`
2. Let the command finish creating the following: an IAM user and role for the pipeline, IAM role for CloudFormation, and a S3 bucket for CloudFormation.
3. Record the console output containing AWS access key and secret access key ids, these will be needed later for the next step in setting up GitHub secrets.
4. If console output does not provide these keys, they can be found stored under the newly created AWS pipeline config found here:
    a. \backend\.aws-sam\pipeline\pipelineconfig.toml

Setting Up AWS Credential Secrets on GitHub

1. On the GitHub repository, create the following secrets and their corresponding values found under Settings -> Secrets and Variables -> Actions (use the ones created by running sam pipeline in the previous step):

Key	Value (Examples)
AWS_ACCESS_KEY_ID	AKIA****************
AWS_ARTIFACTS_BUCKET	aws-sam-cli-managed-dev-pipeline-artifactsbucket-*************
AWS_CLOUDFORMATION_EXECUTION_ROLE	arn:aws:iam::************:role/aws-sam-cli-managed-dev-pipe-CloudFormationExecutionR-************
AWS_FRONTEND_BUCKET	aws-sam-cli-managed-dev-pipeline-FrontendBucket-*************
AWS_PIPELINE_EXECUTION_ROLE	arn:aws:iam::************:role/aws-sam-cli-managed-dev-pipe-PipelineExecutionRole-************
AWS_SECRET_ACCESS_KEY	YR0T************************************


CPSC319-2022 / AmazonianPrime Public

Actions secrets and variables

Secrets and variables allow you to manage reusable configuration data. Secrets are **encrypted** and are used for sensitive data. Learn more about encrypted secrets. Variables are shown as plain text and are used for **non-sensitive** data. Learn more about variables.

Anyone with collaborator access to this repository can use these secrets and variables for actions. They are not passed to workflows that are triggered by a pull request from a fork.

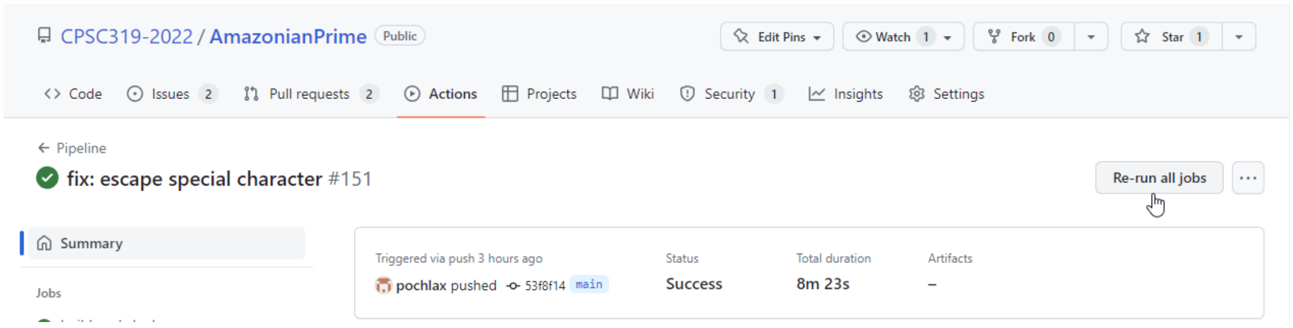
Environment secrets

There are no secrets for this repository's environments.

Repository secrets

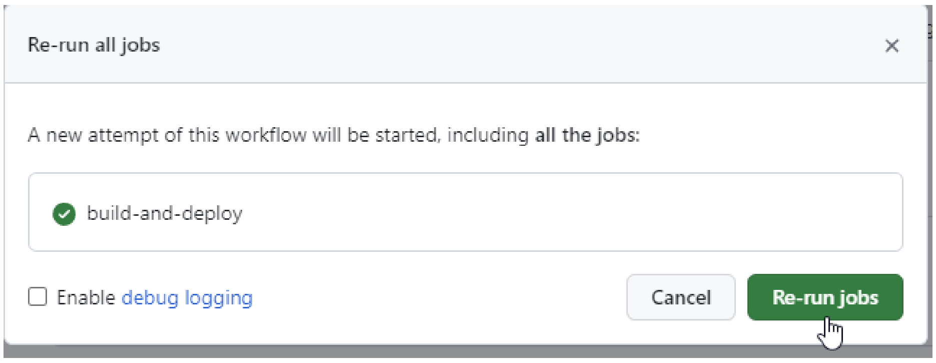
AWS_ACCESS_KEY_ID	Updated on Mar 5
AWS_ARTIFACTS_BUCKET	Updated on Mar 5
AWS_CLOUDFORMATION_EXECUTION_ROLE	Updated on Mar 5
AWS_FRONTEND_BUCKET	Updated on Mar 5
AWS_PIPELINE_EXECUTION_ROLE	Updated on Mar 5
AWS_SECRET_ACCESS_KEY	Updated on Mar 5

2. Once configured, deploy a change to the GitHub Repository to trigger a deployment pipeline and verify the GitHub action runs with the stored credentials (Can also be initiated manually via the Actions tab). The GitHub Action pipeline handles a majority of the work in deploying the backend and the frontend to their corresponding services in AWS.


CPSC319-2022 / AmazonianPrime Public

fix: escape special character #151

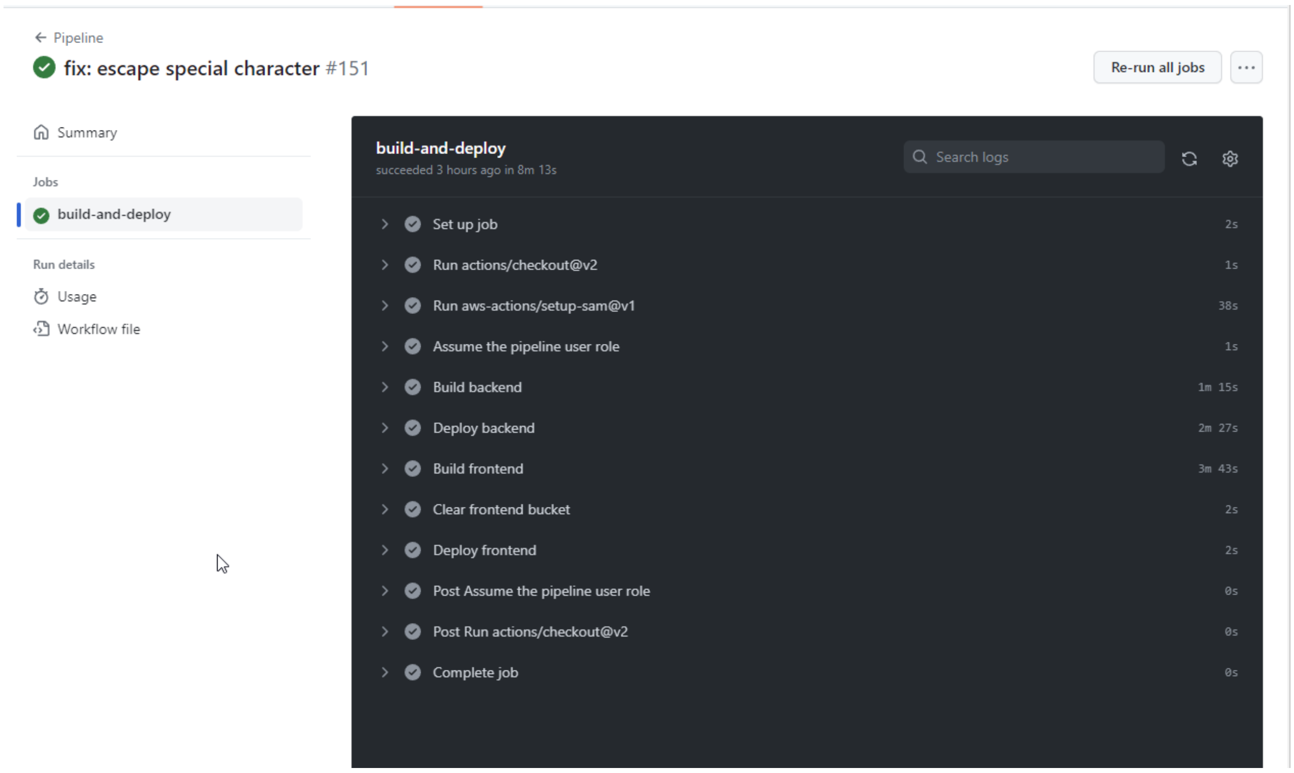
Triggered via push 3 hours ago	Status	Total duration	Artifacts
pochlax pushed 53f8f14 main	Success	8m 23s	–


Re-run all jobs

A new attempt of this workflow will be started, including **all the jobs**:

build-and-deploy

☐ Enable debug logging



fix: escape special character #151

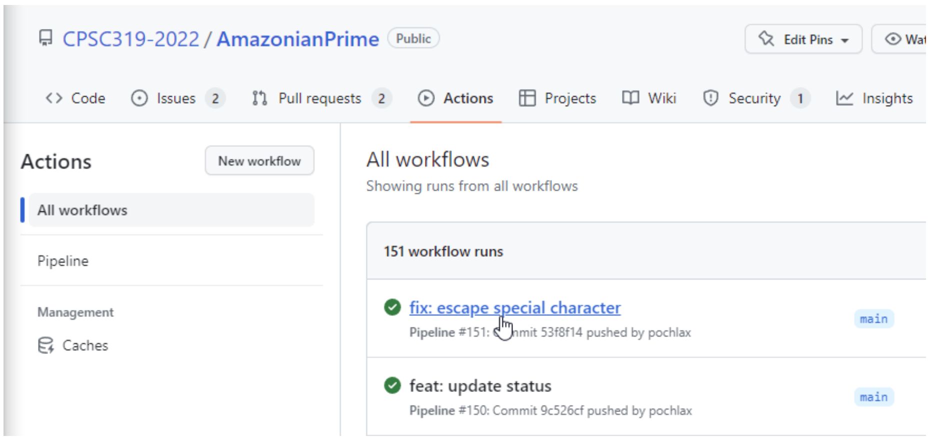
build-and-deploy

build-and-deploy
succeeded 3 hours ago in 8m 13s

```
Set up job                              2s
Run actions/checkout@v2                 1s
Run aws-actions/setup-sam@v1            38s
Assume the pipeline user role           1s
Build backend                           1m 15s
Deploy backend                          2m 27s
Build frontend                          3m 43s
Clear frontend bucket                   2s
Deploy frontend                         2s
Post Assume the pipeline user role      0s
Post Run actions/checkout@v2            0s
Complete job                            0s
```

Verifying Build Deployments

Build pipelines are initiated during Git Pushes to the main branch. Build status can be verified under the Actions tab, click on All workflows in the left sidebar to view the latest Git commits.



All workflows
Showing runs from all workflows

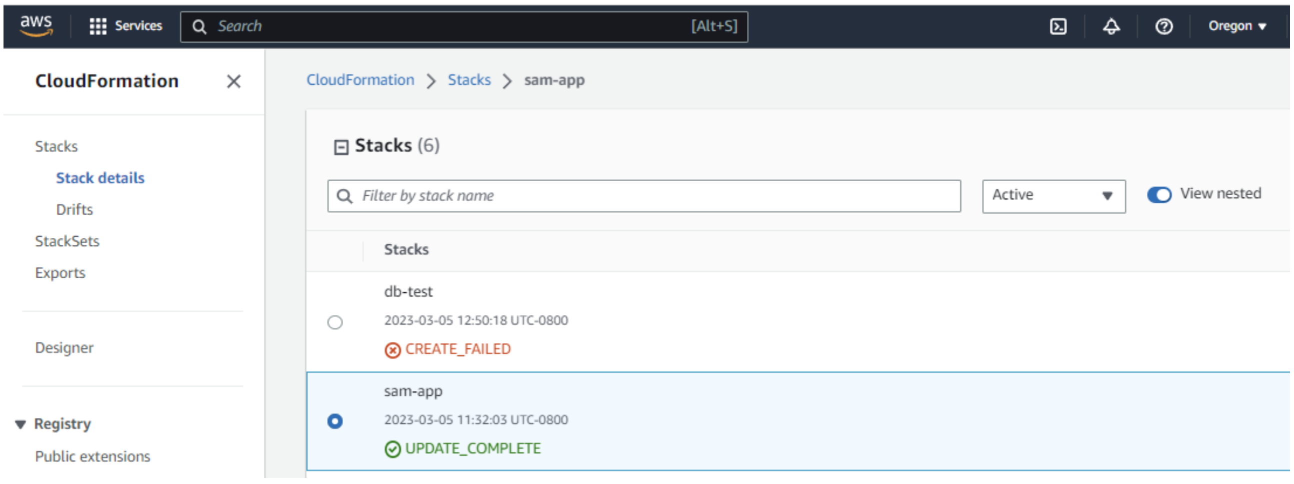
151 workflow runs

fix: escape special character
Pipeline #151: Commit 53f8f14 pushed by pochlax — main

feat: update status
Pipeline #150: Commit 9c526cf pushed by pochlax — main

Viewing a Deployment on AWS

1. Once a GitHub Action pipeline has successfully completed you can verify the stack by viewing it in CloudFormation on the AWS Web Console under sam-app.

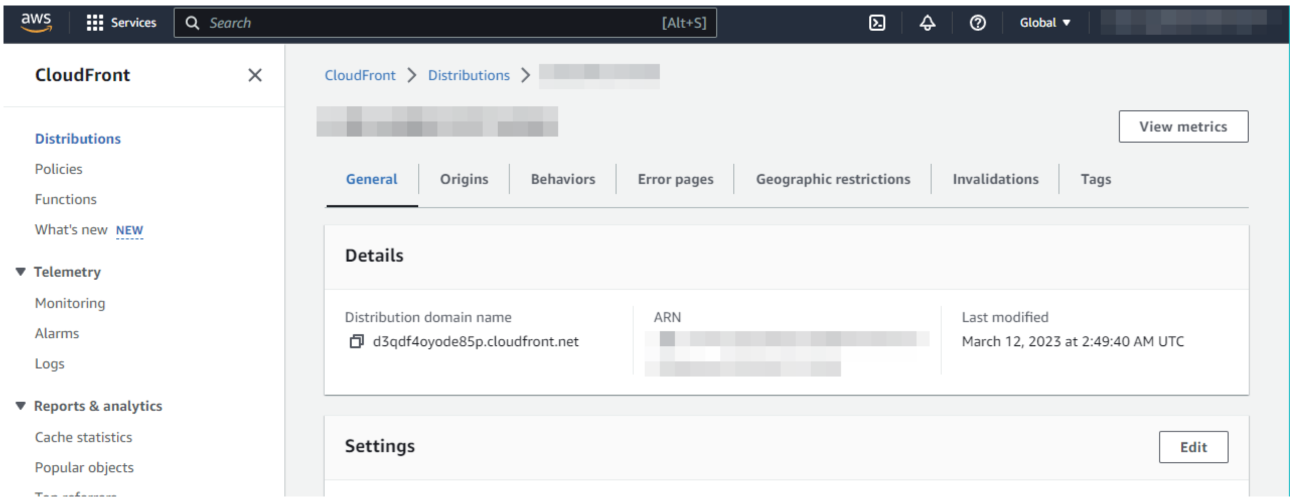


db-test
2023-03-05 12:50:18 UTC-0800
CREATE_FAILED

sam-app
2023-03-05 11:32:03 UTC-0800
UPDATE_COMPLETE

2. To access the deployment link for Amazonian Prime, navigate to CloudFront and click Distributions on the left hand sidebar to view the current Distributions available.

3. Click on the Distribution found and click on the URL found under Distribution domain name, under General -> Details.



Details

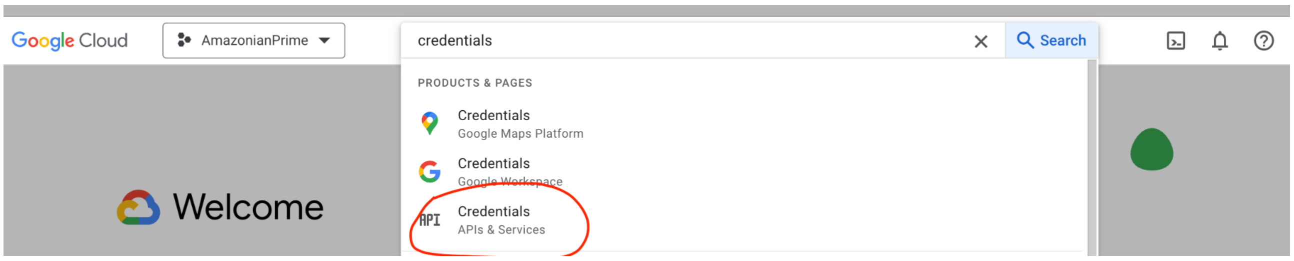
Distribution domain name	ARN	Last modified
d3qdf4oyode85p.cloudfront.net		March 12, 2023 at 2:49:40 AM UTC

Reconfiguring Google OAuth

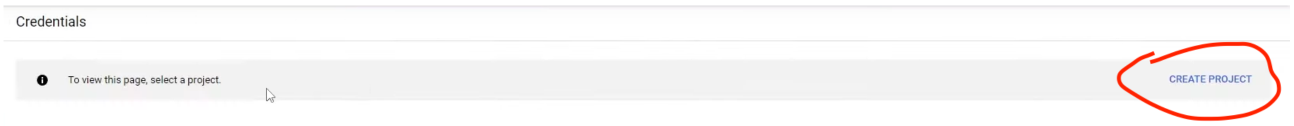
Google OAuth is currently hooked up and good to go. However, if you want to change the settings and scopes associated with the Google authentication flow, you'll need to create your own Google Cloud Platform account and connect it to the system.

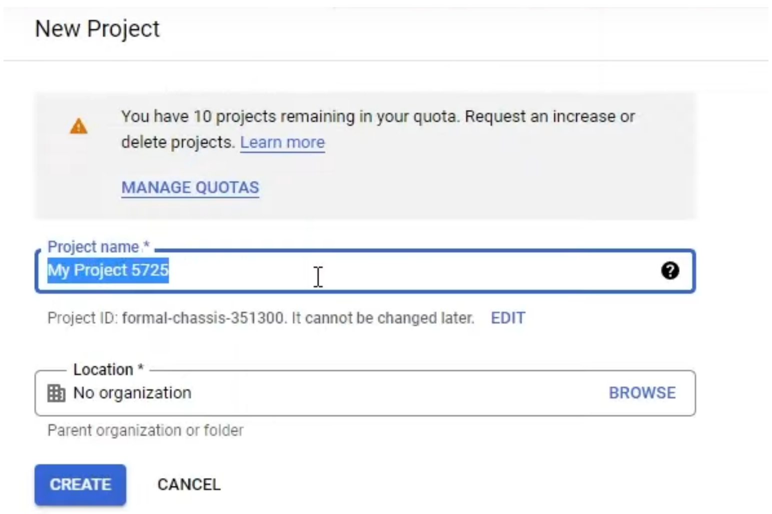
Registering for a Cloud Platform Account

Go to https://console.cloud.google.com/ and sign in with the google account you want to connect with the system. Search for credentials in the search bar and click on the one under APIs



From there you will be prompted to create a project. Click on the Create Project button, fill out the form, and hit Create.





You have 10 projects remaining in your quota. Request an increase or delete projects. Learn more

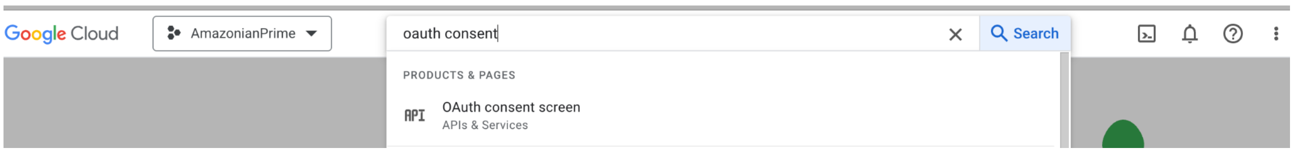
Project name *: My Project 5725

Project ID: formal-chassis-351300. It cannot be changed later. EDIT

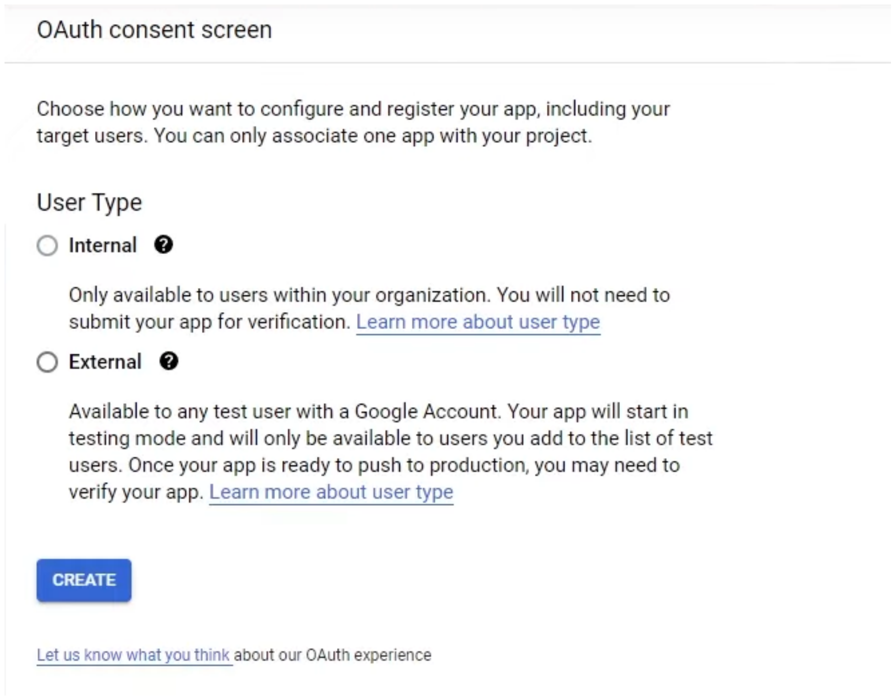
Location *: No organization

Parent organization or folder

After you have successfully created your project, search for "oath consent" and click on the one under APIs again.



Configure your user type as desired.


OAuth consent screen

Choose how you want to configure and register your app, including your target users. You can only associate one app with your project.

User Type

○ Internal

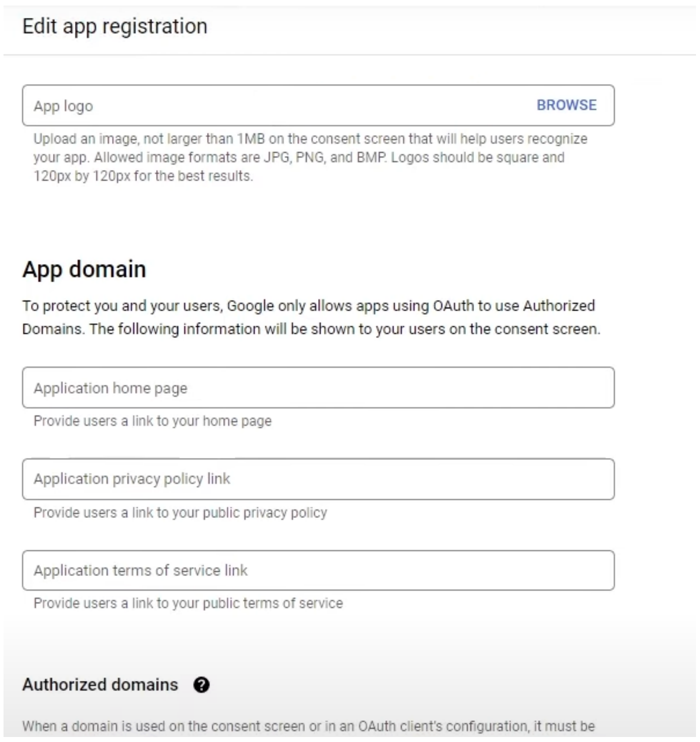
Only available to users within your organization. You will not need to submit your app for verification. Learn more about user type

○ External

Available to any test user with a Google Account. Your app will start in testing mode and will only be available to users you add to the list of test users. Once your app is ready to push to production, you may need to verify your app. Learn more about user type

Let us know what you think about our OAuth experience

Fill out the app registration form and hit Save and Continue:


Edit app registration

App logo

Upload an image, not larger than 1MB on the consent screen that will help users recognize your app. Allowed image formats are JPG, PNG, and BMP. Logos should be square and 120px by 120px for the best results.

App domain

To protect you and your users, Google only allows apps using OAuth to use Authorized Domains. The following information will be shown to your users on the consent screen.

Application home page

Provide users a link to your home page

Application privacy policy link

Provide users a link to your public privacy policy

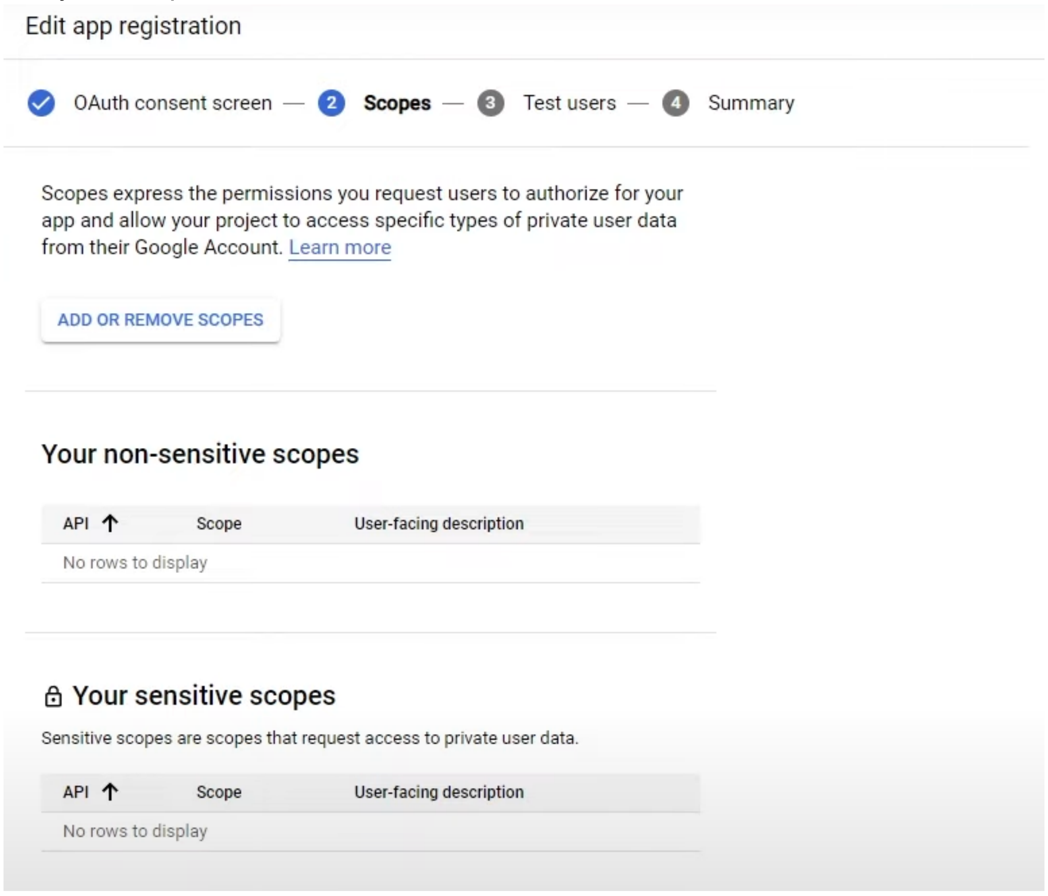
Application terms of service link

Provide users a link to your public terms of service

Authorized domains

When a domain is used on the consent screen or in an OAuth client's configuration, it must be

Edit your scopes as desired and hit Save and Continue:


Edit app registration

OAuth consent screen — 2 Scopes — 3 Test users — 4 Summary

Scopes express the permissions you request users to authorize for your app and allow your project to access specific types of private user data from their Google Account. Learn more

ADD OR REMOVE SCOPES

Your non-sensitive scopes

API ↑	Scope	User-facing description
No rows to display		

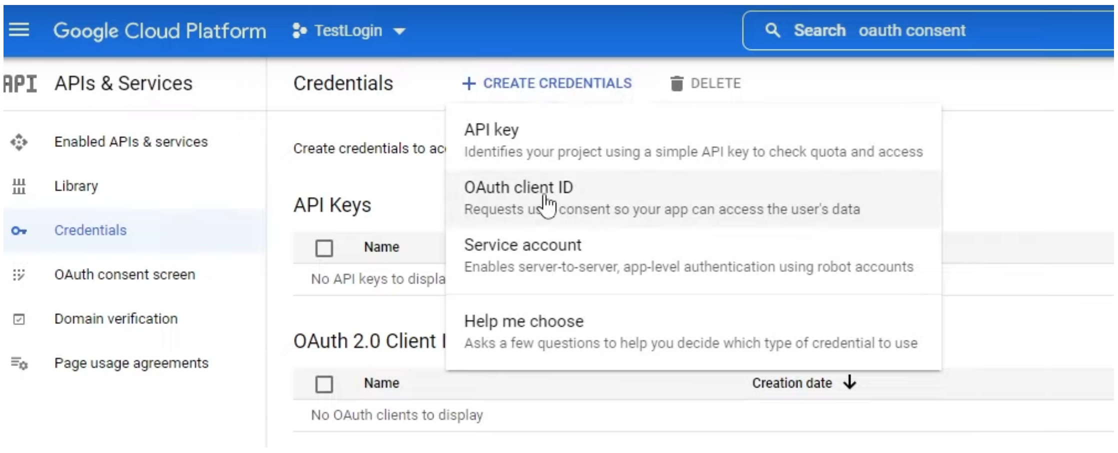
Your sensitive scopes

Sensitive scopes are scopes that request access to private user data.

API ↑	Scope	User-facing description
No rows to display		

Add test users and hit Save and Continue.

Go back to credentials under APIs. Create an OAuth client ID:



Create credentials to ac

API key
Identifies your project using a simple API key to check quota and access

OAuth client ID
Requests us consent so your app can access the user's data

Service account
Enables server-to-server, app-level authentication using robot accounts

Help me choose
Asks a few questions to help you decide which type of credential to use

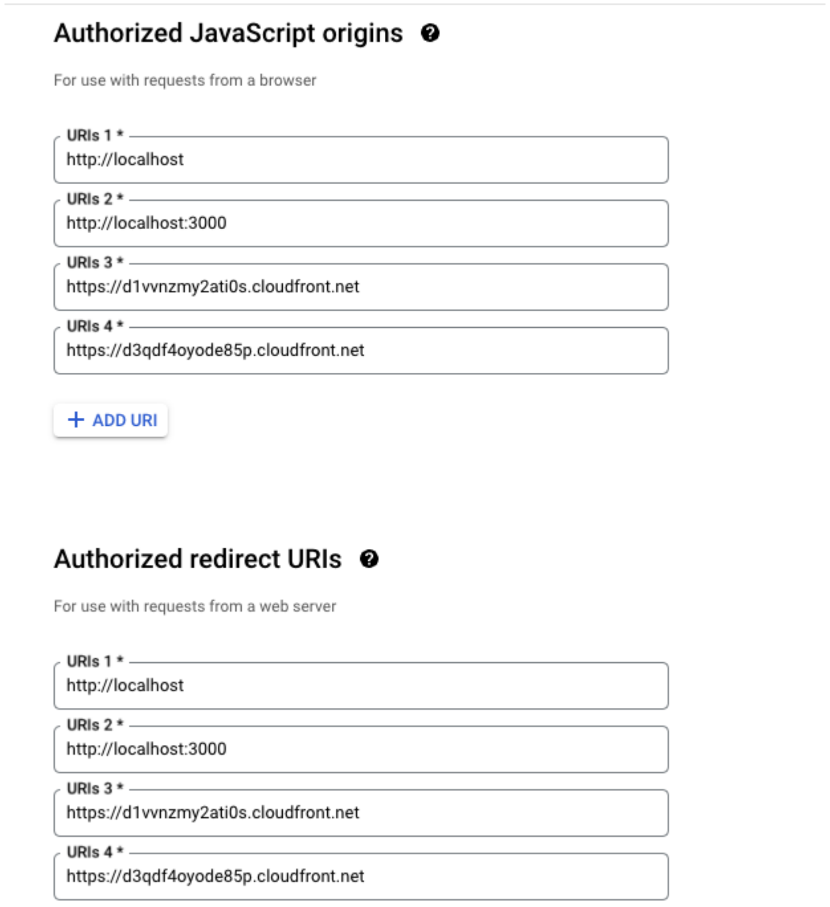
API Keys

Name
No API keys to displa

OAuth 2.0 Client I

Name	Creation date ↓
No OAuth clients to display	

Fill out the OAuth client ID registration form. Enter the website's URIs under Authorized Javascript origins and Authorized redirect URIs


Authorized JavaScript origins

For use with requests from a browser

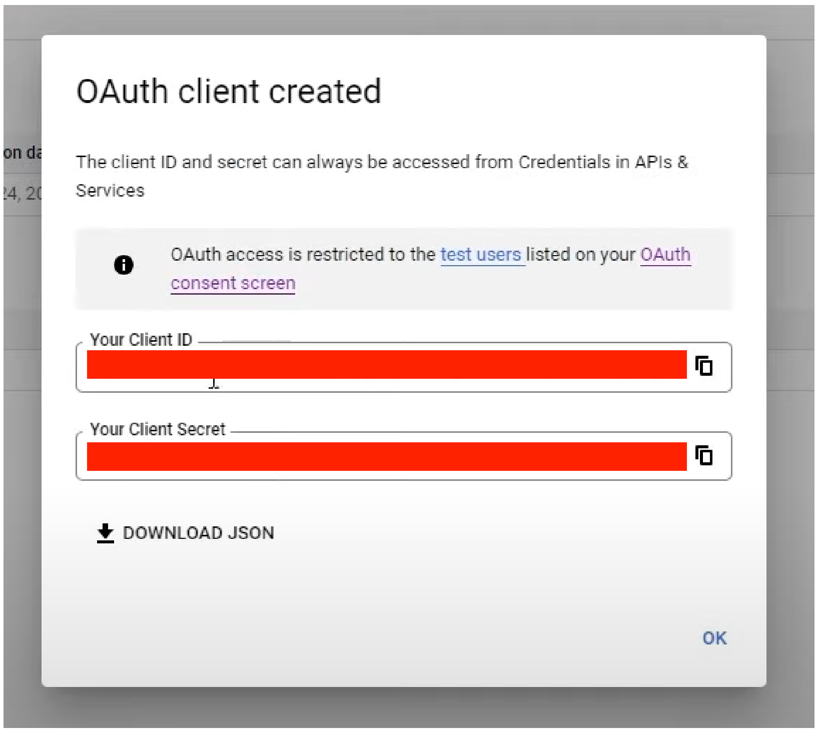
URIs 1 *	http://localhost
URIs 2 *	http://localhost:3000
URIs 3 *	https://d1vvnzmy2ati0s.cloudfront.net
URIs 4 *	https://d3qdf4oyode85p.cloudfront.net

Authorized redirect URIs

For use with requests from a web server

URIs 1 *	http://localhost
URIs 2 *	http://localhost:3000
URIs 3 *	https://d1vvnzmy2ati0s.cloudfront.net
URIs 4 *	https://d3qdf4oyode85p.cloudfront.net

Press Create and **save the Client ID and Client Secret:**


OAuth client created

The client ID and secret can always be accessed from Credentials in APIs & Services

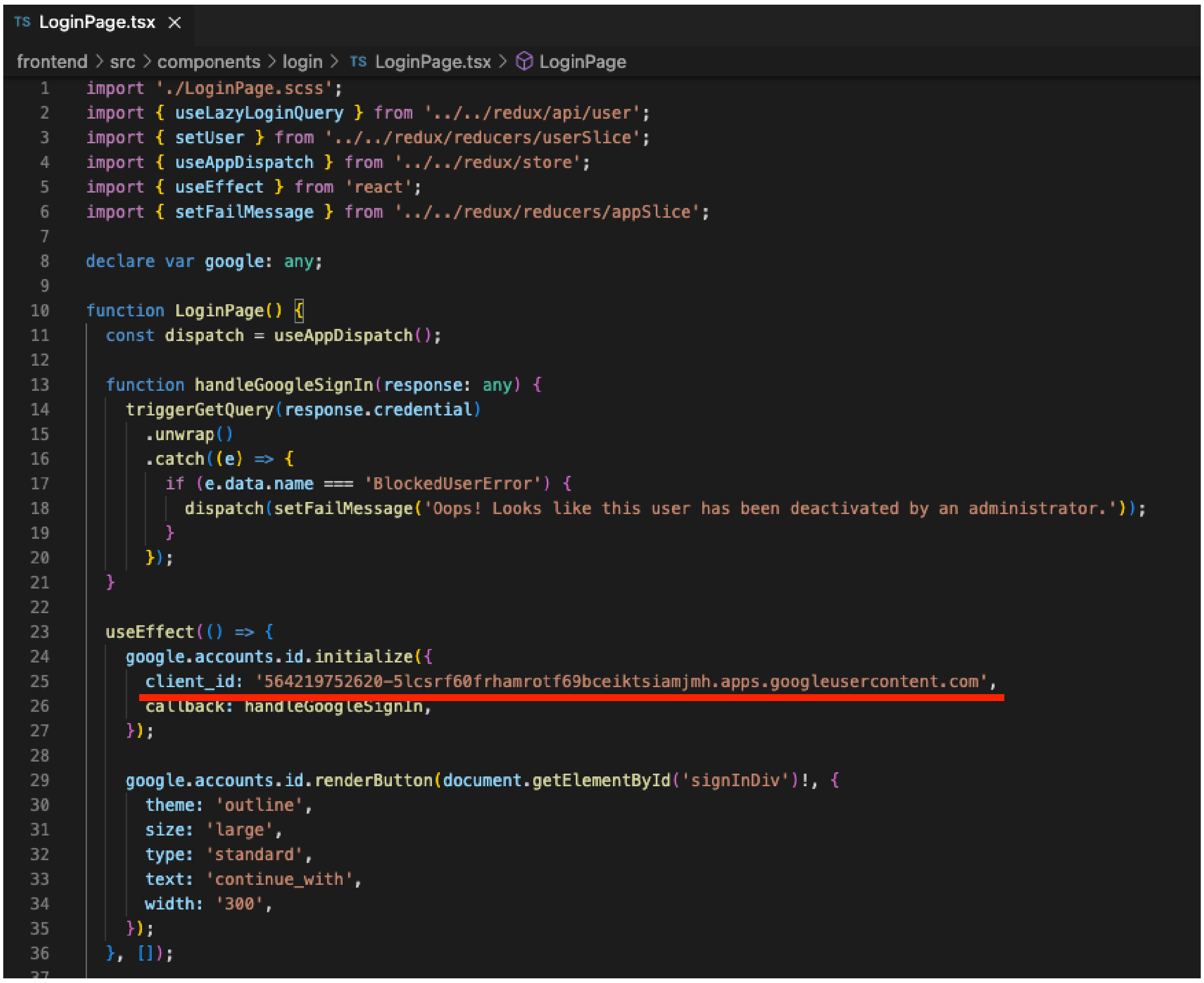
OAuth access is restricted to the test users listed on your OAuth consent screen

Your Client ID: [redacted]

Your Client Secret: [redacted]

Integrating Client ID Into the System

Under frontend/src/components/login/LoginPage.tsx in the codebase, paste the client id to the "client_id" field:


TS LoginPage.tsx ×
frontend > src > components > login > TS LoginPage.tsx > LoginPage

```tsx
import './LoginPage.scss';
import { useLazyLoginQuery } from '../../redux/api/user';
import { setUser } from '../../redux/reducers/userSlice';
import { useAppDispatch } from '../../redux/store';
import { useEffect } from 'react';
import { setFailMessage } from '../../redux/reducers/appSlice';

declare var google: any;

function LoginPage() {
  const dispatch = useAppDispatch();

  function handleGoogleSignIn(response: any) {
    triggerGetQuery(response.credential)
      .unwrap()
      .catch((e) => {
        if (e.data.name === 'BlockedUserError') {
          dispatch(setFailMessage('Oops! Looks like this user has been deactivated by an administrator.'));
        }
      });
  }

  useEffect(() => {
    google.accounts.id.initialize({
      client_id: '564219752620-5lcsrf60frhamrotf69bceiktsiamjmh.apps.googleusercontent.com',
      callback: handleGoogleSignIn,
    });

    google.accounts.id.renderButton(document.getElementById('signInDiv')!, {
      theme: 'outline',
      size: 'large',
      type: 'standard',
      text: 'continue_with',
      width: '300',
    });
  }, []);
```

You can now reconfigure the google authentication settings and scopes in the google console with your account, if desired.

SES Email Verification

Adding Receiver Accounts

1. Log in to the AWS Management Console and navigate to the AWS SES console (console.aws.amazon.com/ses).
    a. ex for this instance. https://us-west-2.console.aws.amazon.com/ses/home?region=us-west-2#/verified-identities

2. In the left-hand navigation menu, click on "Verified identities".

3. Click the "Create identity" button.

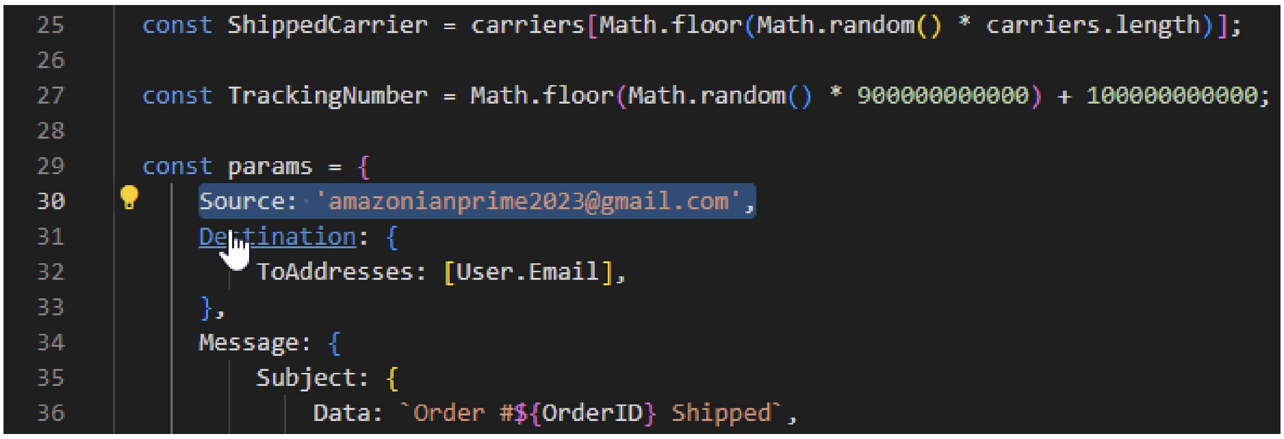
4. Click on the Email address radio button and enter the email address you want to add to SES and click the "Create identity" button. This will send an email to the specified email address with a verification link.

5. Open the email that was sent to the specified email address and click the verification link. Once the email address is verified, you can receive emails using SES from Amazonian Prime. If you want to send email from multiple email addresses, you will need to repeat this process for each email address.

Adding Sender Account

1. Follow the same steps as outlined in “Adding Receiver Accounts”

2. Several modifications will be needed to be made from the codebase to modify the address of the sender and can be found in the following files:
    a. \backend\functions\send-shipped:Line 30
    b. \backend\functions\send-order:Line 25
    c. \backend\functions\send-delivered:Line 25
    d. \backend\functions\email-confirmation:Line 10



```
25  const ShippedCarrier = carriers[Math.floor(Math.random() * carriers.length)];
26
27  const TrackingNumber = Math.floor(Math.random() * 900000000000) + 100000000000;
28
29  const params = {
30      Source: 'amazonianprime2023@gmail.com',
31      Destination: {
32          ToAddresses: [User.Email],
33      },
34      Message: {
35          Subject: {
36              Data: `Order #${OrderID} Shipped`,
```

3. Replace each instance of “Source: <email>” with a verified SES email account that will be used to send correspondence to other users.
4. Commit the changes using Git and Push to the main branch in order to deploy your changes.

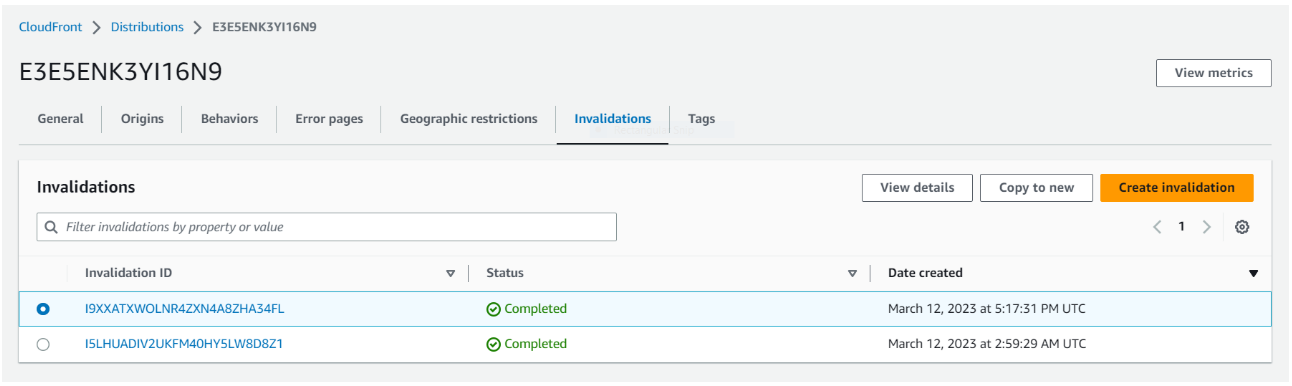
Configuration Document

If you follow the steps in the installation guide, then the deployment should be managed by the pipeline we created in Github actions. To trigger the pipeline, simply push to main or merge to main.

Sometimes you get an error while executing auto deployment, in which case, you need to invalidate the cache. This is outlined below:

Invalidate Cache

- Check if CloudFront is caching the frontend by looking for blank pages or syntax errors in the console on the production site.
- If you find an error in the console and the file it is referencing no longer exists in the S3 frontend bucket, then you need to clear the cache.
- Go to the CloudFront console and create an invalidation by copying one of the existing invalidations.
- In the new invalidation, specify the path or paths to the files you want to invalidate. You can use wildcards to specify multiple files.
- Click "Create Invalidation" to clear the cache.
- Inform the team that you plan to clear the cache, so they are aware of the change.



Invalidation ID	Status	Date created
I9XXATXWOLNR4ZXN4A8ZHA34FL	Completed	March 12, 2023 at 5:17:31 PM UTC
I5LHUADIV2UKFM40HY5LW8D8Z1	Completed	March 12, 2023 at 2:59:29 AM UTC