CLOUD COMPUTING

AWS – CAPSTONE 3

Build a serverless web application using Amplify, Cognito, API Gateway, Lambda & DynamoDB

Rajeswari Subramanian | December,2022

GREAT LAKES EXECUTIVE LEARNING | Great Learning

Contents

Project Overview:

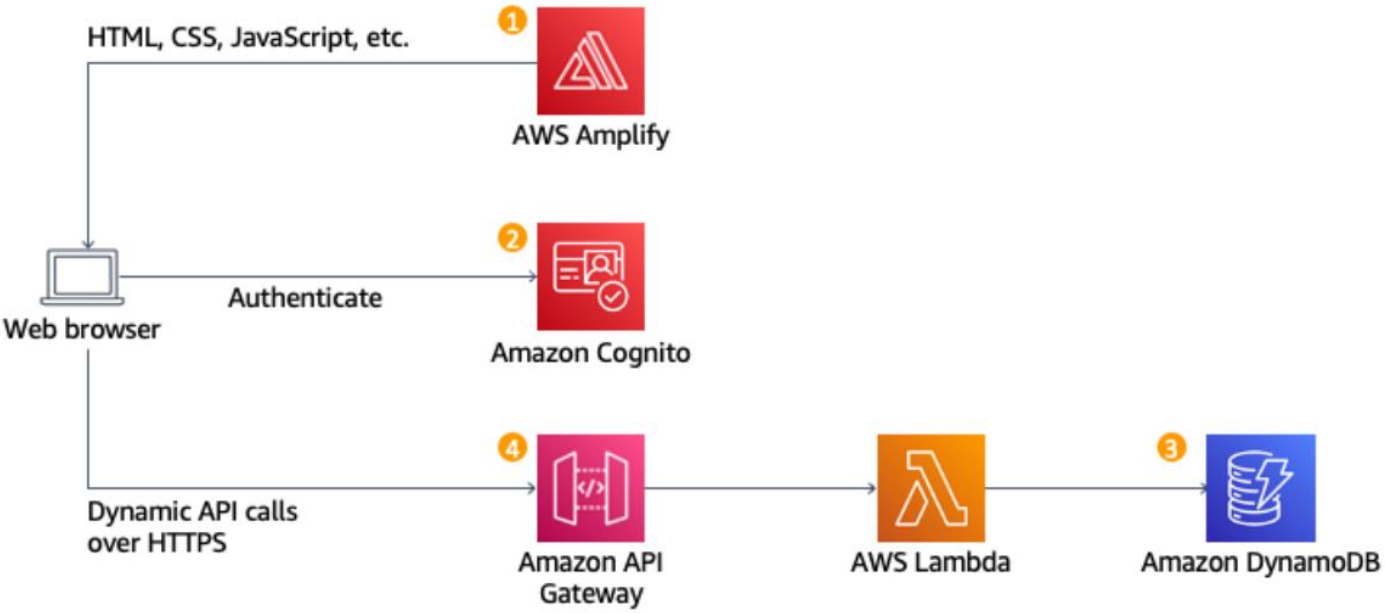
In this project. We are creating simple serverless web application that enables users to request unicorn rides from the Wild Rydes fleet. The application will present users with an HTML based user interface for indicating the location where they would like to be picked up and will interface on the backend with a RESTful web service to submit the request and dispatch a nearby unicorn. The application will also provide facilities for users to register with the service and log in before requesting rides.

Modules

This capstone project is divided into four modules. Each module describes a scenario of what we're going to build and step-by-step directions to help to implement the architecture.

1. Host a Static Website: Configure AWS Amplify to host the static resources for the web application with continuous deployment built in
2. Manage Users: Create an Amazon Cognito user pool to manage users' accounts
3. Build a Serverless Backend: Build a backend process for handling requests our web application
4. Deploy a RESTful API: Use Amazon API Gateway to expose the Lambda function, that built in the previous module as a RESTful API

Architectural Diagram



1. **Static Web Hosting:**

AWS Amplify hosts static web resources including HTML, CSS, JavaScript, and image files which are loaded in the user's browser.

2. **User Management:**

Amazon Cognito provides user management and authentication functions to secure the backend API.

3. **Serverless Backend:**

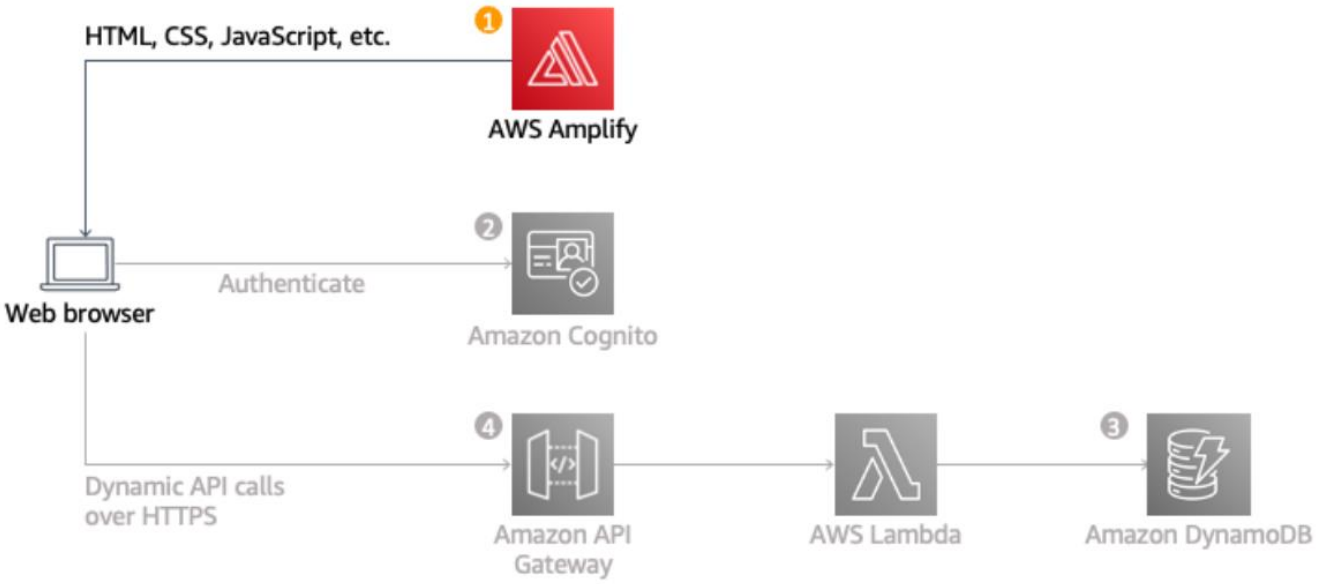
Amazon DynamoDB provides a persistence layer where data can be stored by the API's Lambda function.

4. **RESTful API:**

JavaScript executed in the browser sends and receives data from a public backend API built using Lambda and API Gateway.

Module 1: Static Web Hosting

Architectural Overview

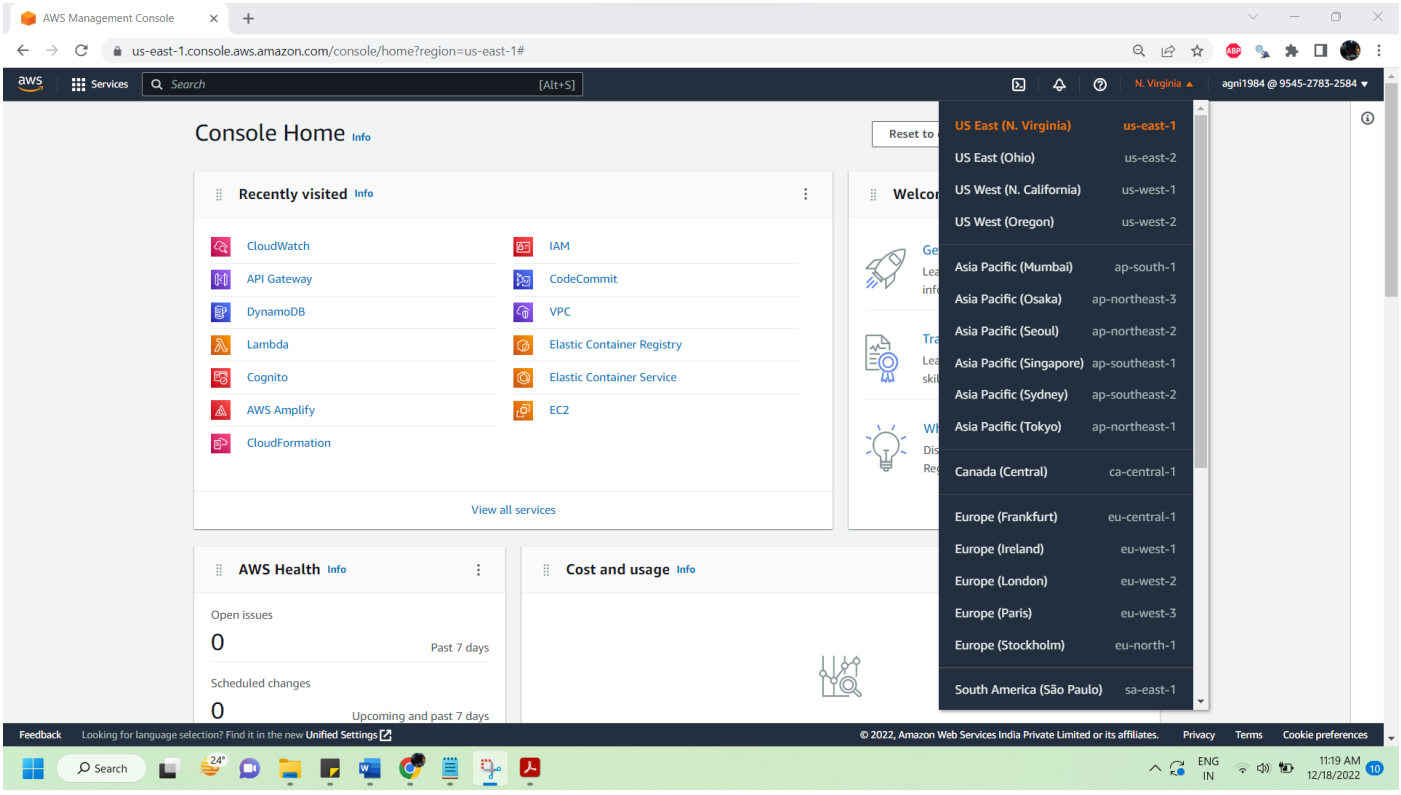


The architecture for this module is straightforward. All of our static web content including HTML, CSS, JavaScript, images, and other files will be managed by AWS Amplify Console. End users will then access the site using the public website URL exposed by AWS Amplify Console. We don't need to run any web servers or use other services to make our site available.

Step 1: Select Region

This web application can be deployed in any AWS Region that supports all the services used in this application, which include AWS Amplify, AWS CodeCommit, Amazon Cognito, AWS Lambda, Amazon API Gateway, and Amazon DynamoDB.

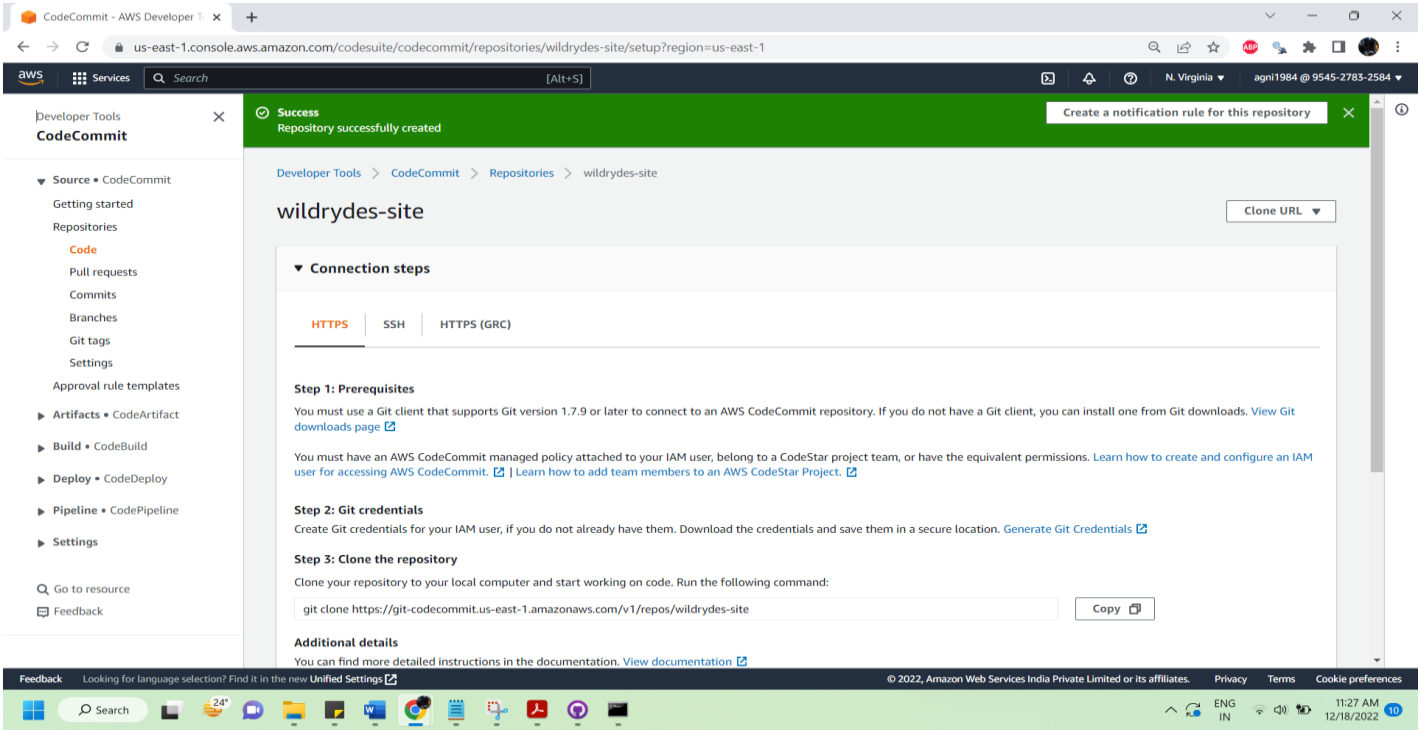
Select ***"US East (N. Virginia)"*** Region from the dropdown in the upper right corner of the AWS Management Console.



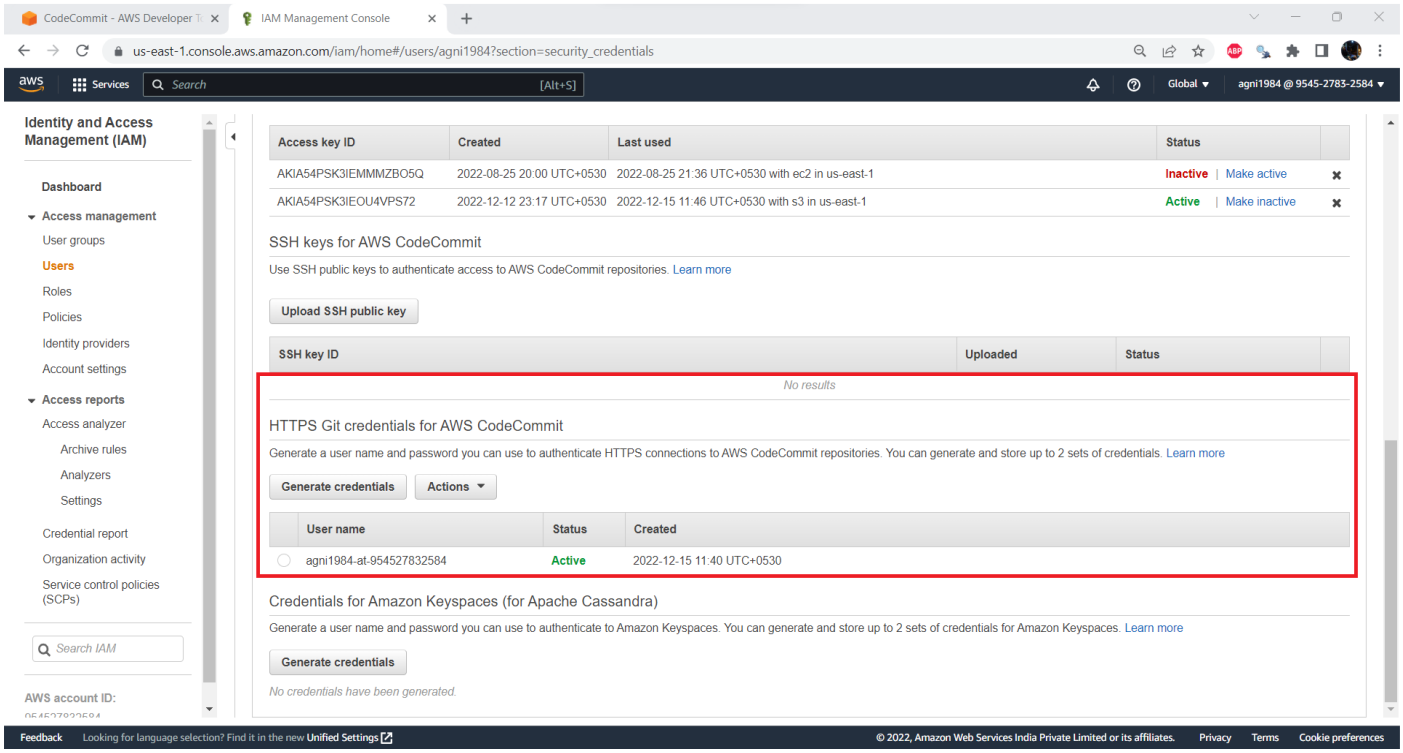
Step 2: Create a Git Repository

'CodeCommit' used to store & manage our application code. Follow the below steps to create 'Codecommit' repository.

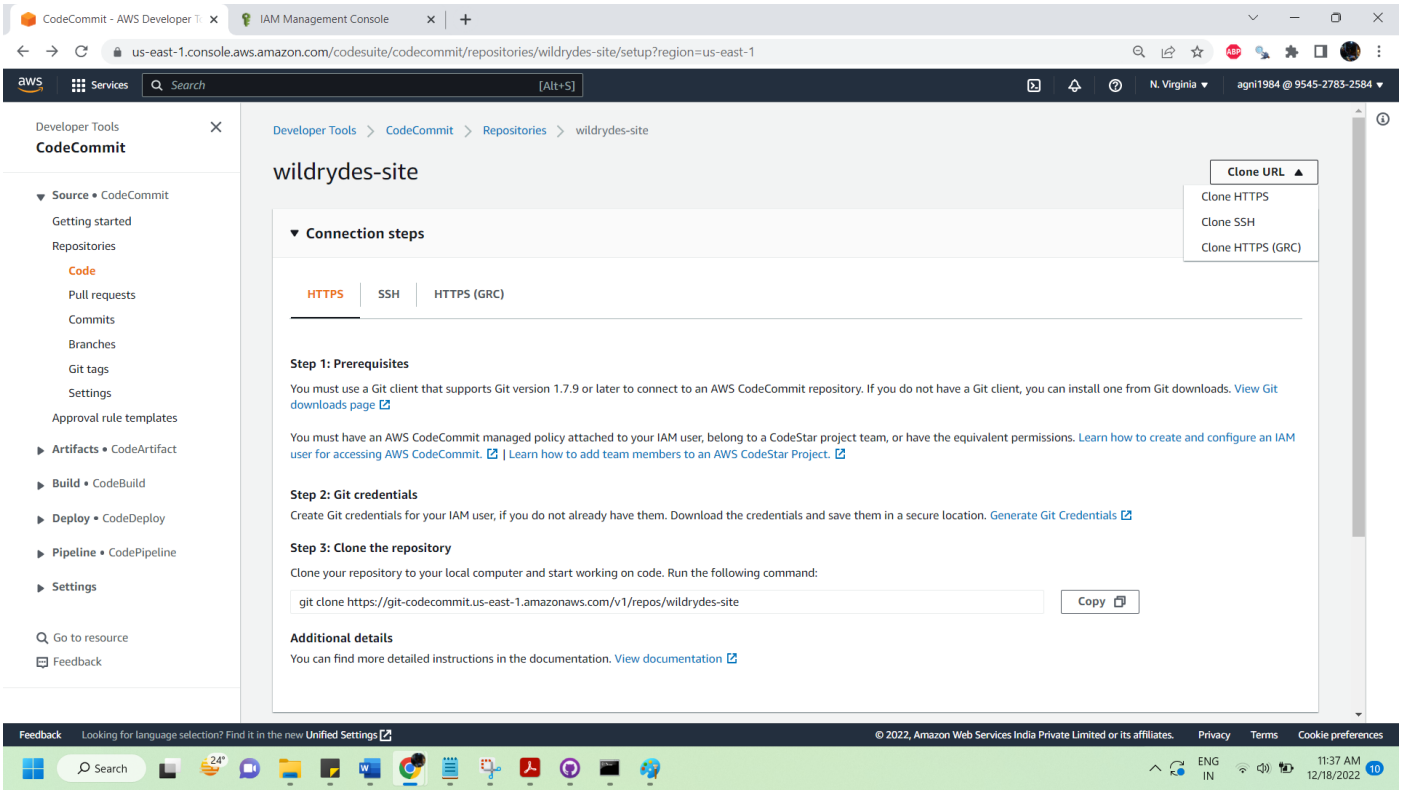
- Open the AWS CodeCommit console
- Select Create Repository
- Set the Repository name* to **"wildrydes-site"**
- Select Create
- Now that the repository is created.

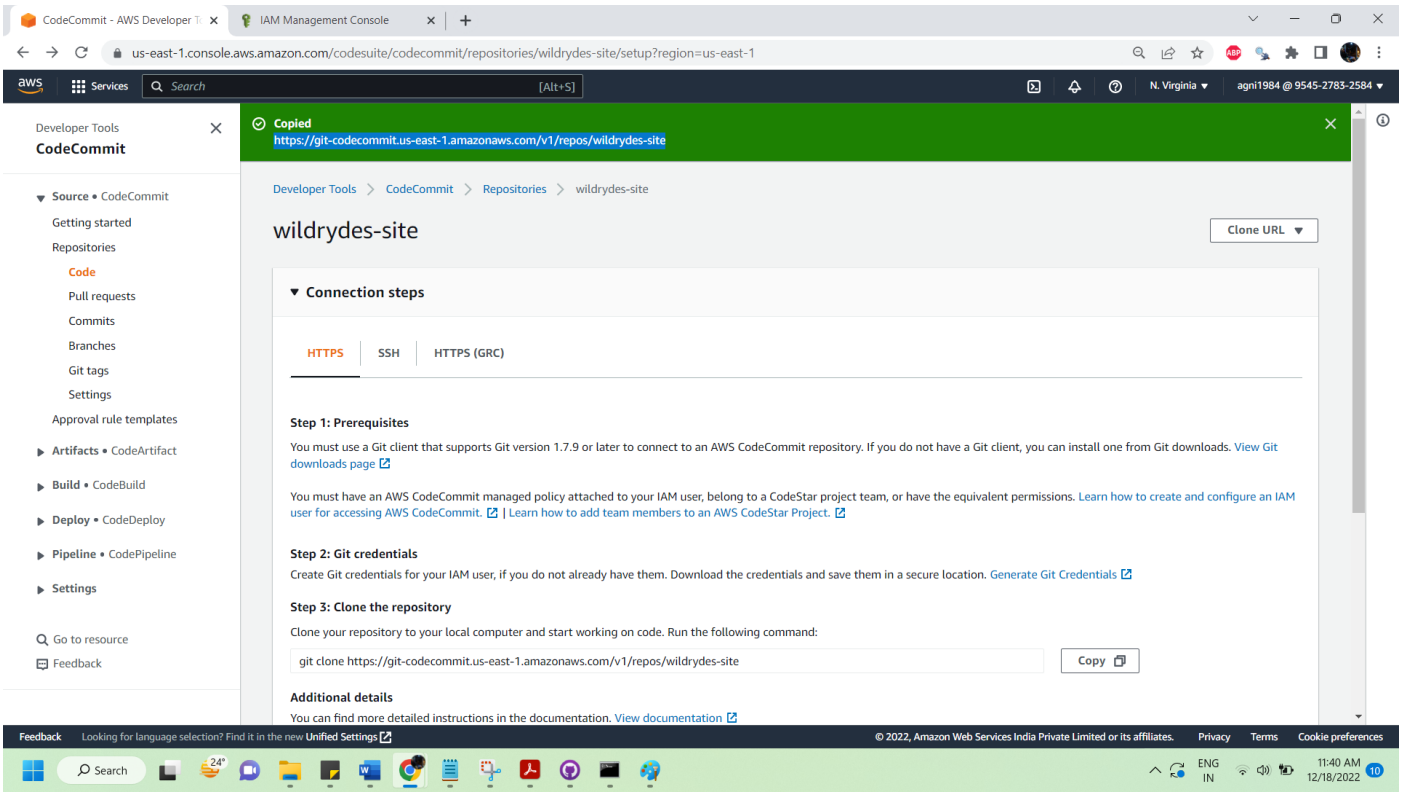


- Generate Git credentials in the IAM user, in IAM console following these instructions.
    - In the IAM console, in the navigation pane, choose **Users**, and from the list of users, choose IAM user.
    - On the user details page, choose the **Security Credentials** tab, and in **HTTPS Git credentials for AWS CodeCommit**, choose **Generate**.
    - Copy the user name and password that IAM generated, either by showing, copying, and then pasting this information into a secure file on the local computer, or by choosing **Download credentials** to download this information as a .CSV file. We need this information to connect to CodeCommit.



➢ In the CodeCommit console, From the Clone URL drop down, select Clone HTTPS





➢ From a terminal window run git clone and the HTTPS URL of the repository:

```
E:\GreatLakes_Cloud\CapstoneProject\aws-capstone3>git clone https://git-codecommit.us-east-1.amazonaws.com/v1/repos/wildrydes-site
Cloning into 'wildrydes-site'...
warning: You appear to have cloned an empty repository.
```

Step 3: Populate the Git Repository

Copy the web site content from an existing publicly accessible S3 bucket's content to the repository.

a. Change directory into our repository and copy the static files from S3:

```
E:\GreatLakes_Cloud\CapstoneProject\aws-capstone3>cd wildrydes-site

E:\GreatLakes_Cloud\CapstoneProject\aws-capstone3\wildrydes-site>aws s3 cp
s3://wildrydes-us-east-1/WebApplication/1_StaticWebHosting/website ./ --recursive
```

```
E:\GreatLakes_Cloud\CapstoneProject\aws-capstone3>cd wildrydes-site

E:\GreatLakes_Cloud\CapstoneProject\aws-capstone3\wildrydes-site>aws s3 cp s3://wildrydes-us-east-1/WebApplication/1_StaticWebHosting/website ./ --recursive
```

b. Commit the files to our Git service

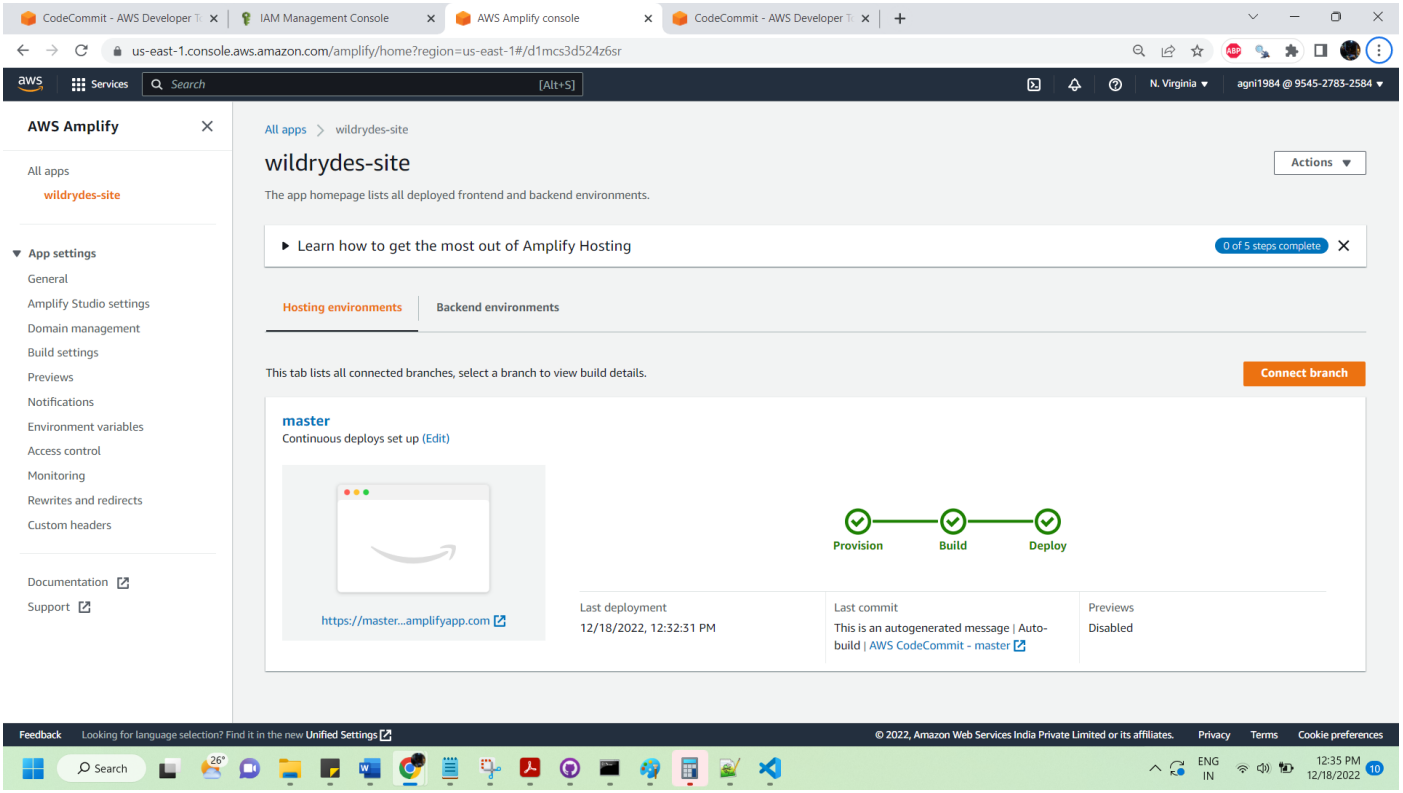
```
$ git add .
$ git commit -m 'new'
$ git push
```

```
E:\GreatLakes_Cloud\CapstoneProject\aws-capstone3\wildrydes-site>git push
Enumerating objects: 95, done.
Counting objects: 100% (95/95), done.
Delta compression using up to 8 threads
Compressing objects: 100% (94/94), done.
Writing objects: 100% (95/95), 9.44 MiB | 1.93 MiB/s, done.
Total 95 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Validating objects: 100%
To https://git-codecommit.us-east-1.amazonaws.com/v1/repos/wildrydes-site
 * [new branch]      master -> master
```

Step 4: Enable Web Hosting with the Amplify Console

AWS Amplify takes care of the work of setting up a place to store static web application code and provides a number of helpful capabilities to simplify both the lifecycle of that application as well as enable best practices. Follow the below steps to deploy code using AWS Amplify,

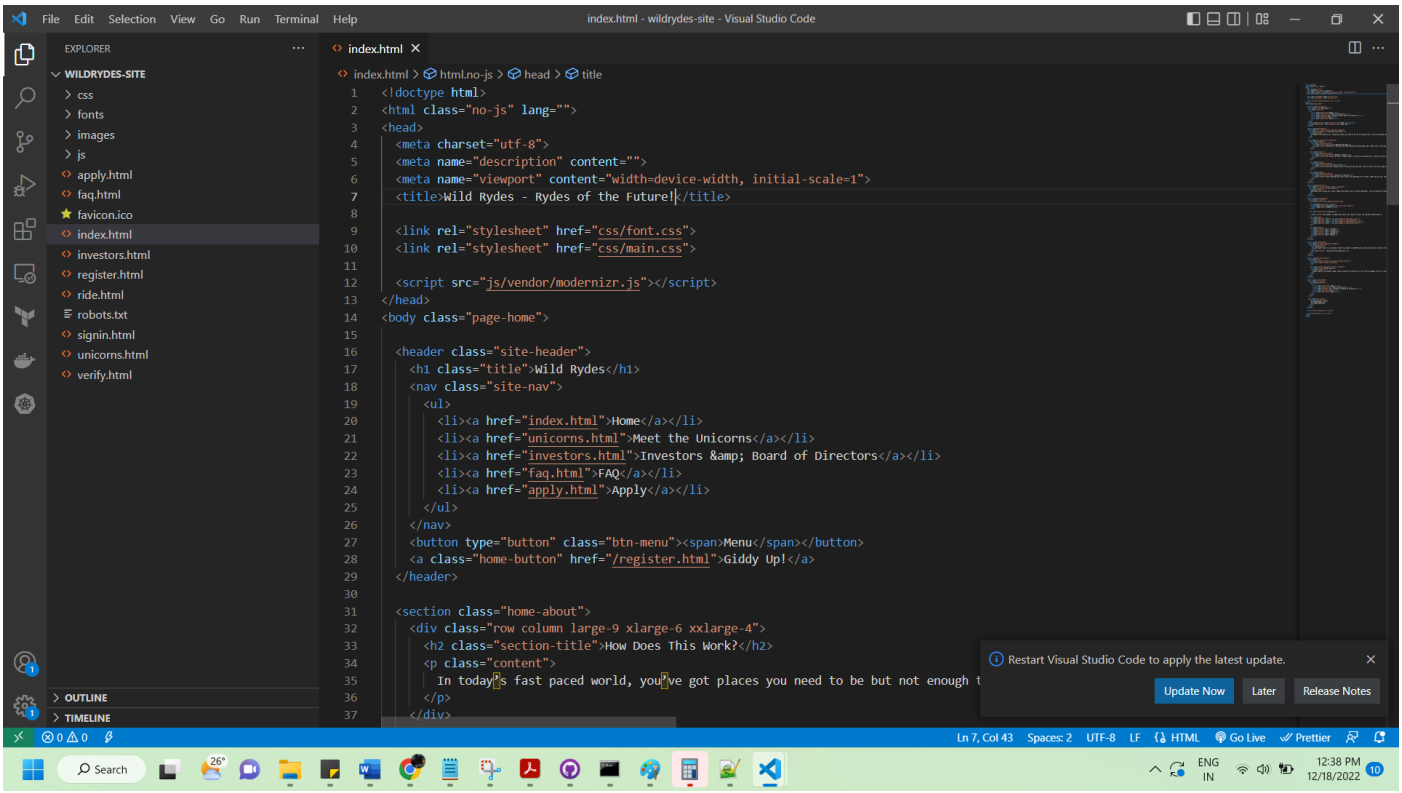
- Launch the Amplify Console page
- Click Get Started under Deploy with Amplify Console
- Go to New App on the top right and choose Host Web App
- Select CodeCommit under Get started with Amplify Hosting
- Select the Repository service provider used today and select Next
- From the dropdown select the Repository and Branch just create
- On the "Configure build settings" page leave all the defaults and select Next.
- On the "Review" page select Save and deploy
- Amplify Console to create the necessary resources and to deploy our code.
- Once completed, click on the site image to launch Wild Rydes site.



Step 5: Modify site

The AWS Amplify Console will rebuild and redeploy the app when it detects changes to the connected repository. Make a change to the main page to test out this process.

- From local machine, open `wildryde-site/index.html` in visual studio and modify the title line so that it says: ***<title>Wild Rydes - Rydes of the Future!</title>***

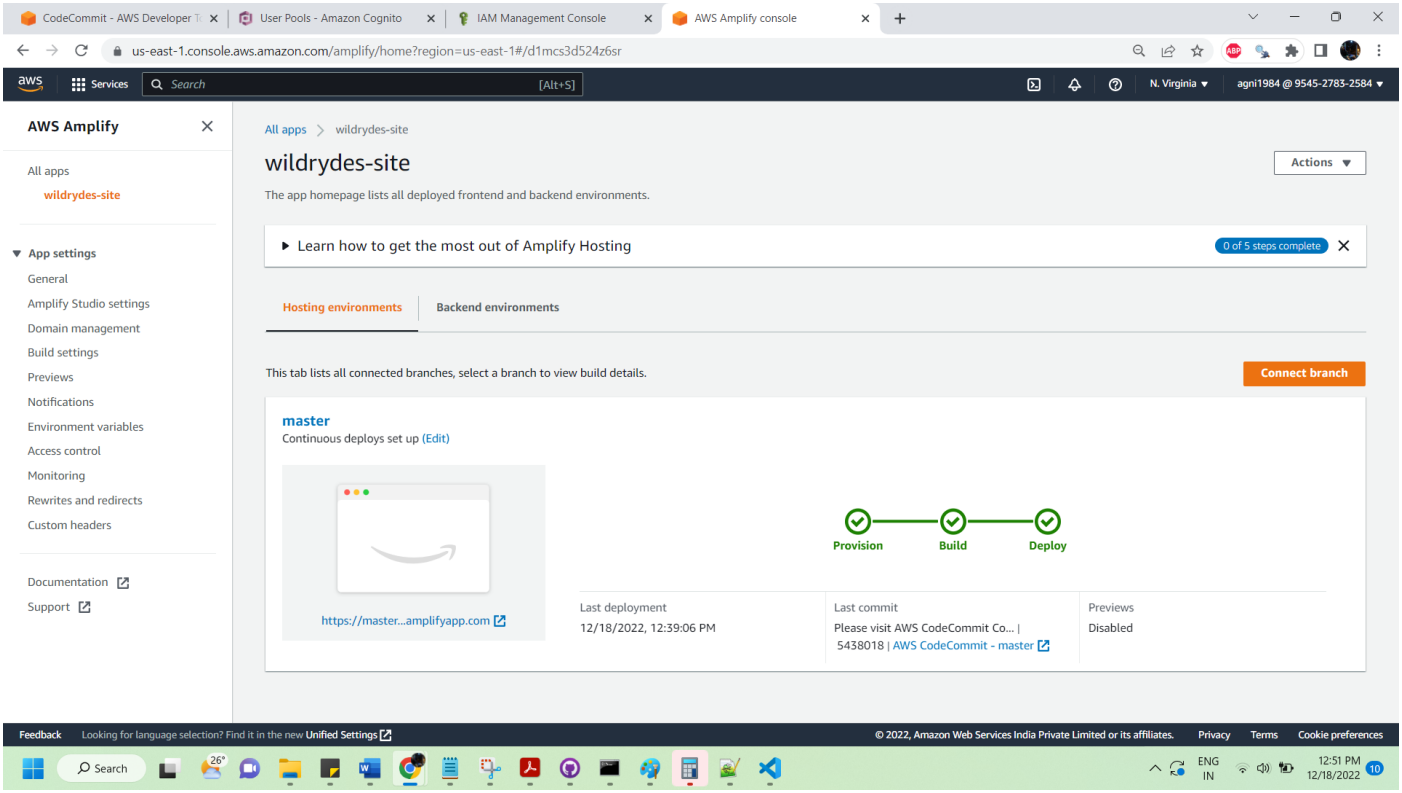


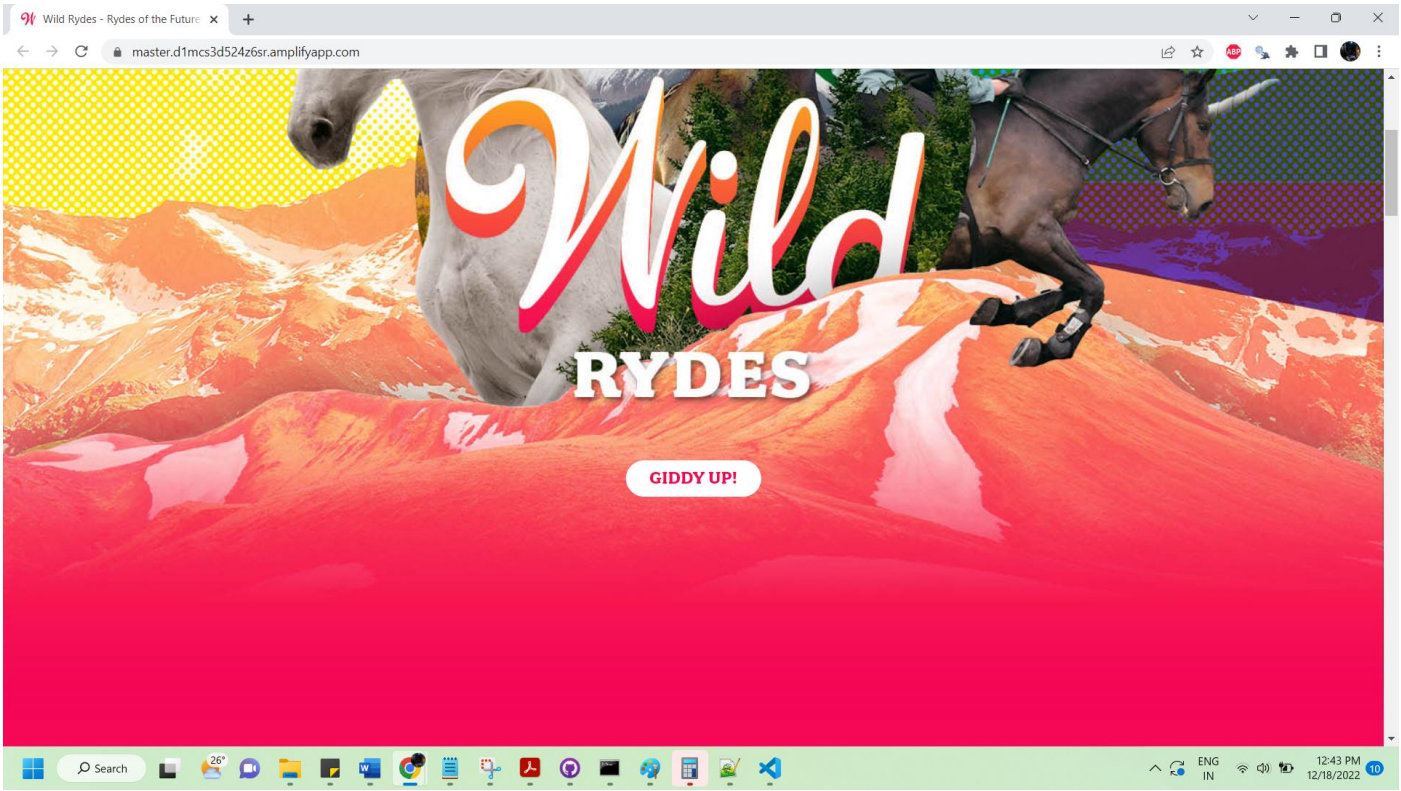
- Save the file and commit to git repository again. Amplify Console will begin to build the site again soon after it notices the update to the repository.

```
$ git add index.html
$ git commit -m "updated title"
```

```
E:\GreatLakes_Cloud\CapstoneProject\aws-capstone3\wildrydes-site>git add index.html
warning: in the working copy of 'index.html', LF will be replaced by CRLF the next time Git touches it

E:\GreatLakes_Cloud\CapstoneProject\aws-capstone3\wildrydes-site>git commit -m "updated title"
[master 5438018] updated title
 1 file changed, 1 insertion(+), 1 deletion(-)

E:\GreatLakes_Cloud\CapstoneProject\aws-capstone3\wildrydes-site>git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 308 bytes | 308.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Validating objects: 100%
To https://git-codecommit.us-east-1.amazonaws.com/v1/repos/wildrydes-site
   053dc44..5438018  master -> master
```

b. Once completed, re-open the Wild Rydes site and notice the title change.





Module 2: User Management

In this module we will create an Amazon Cognito user pool to manage users' accounts. We'll deploy pages that enable customers to register as a new user, verify their email address, and sign into the site.

Architectural Overview

When users visit website, they will first register a new user account by providing an email address and password to register.

After users submit their registration, Amazon Cognito will send a confirmation email with a verification code to the address they provided. To confirm their account, users will return to our site and enter their email address and the verification code they received. We can also confirm user accounts using the Amazon Cognito console with a fake email address for testing.

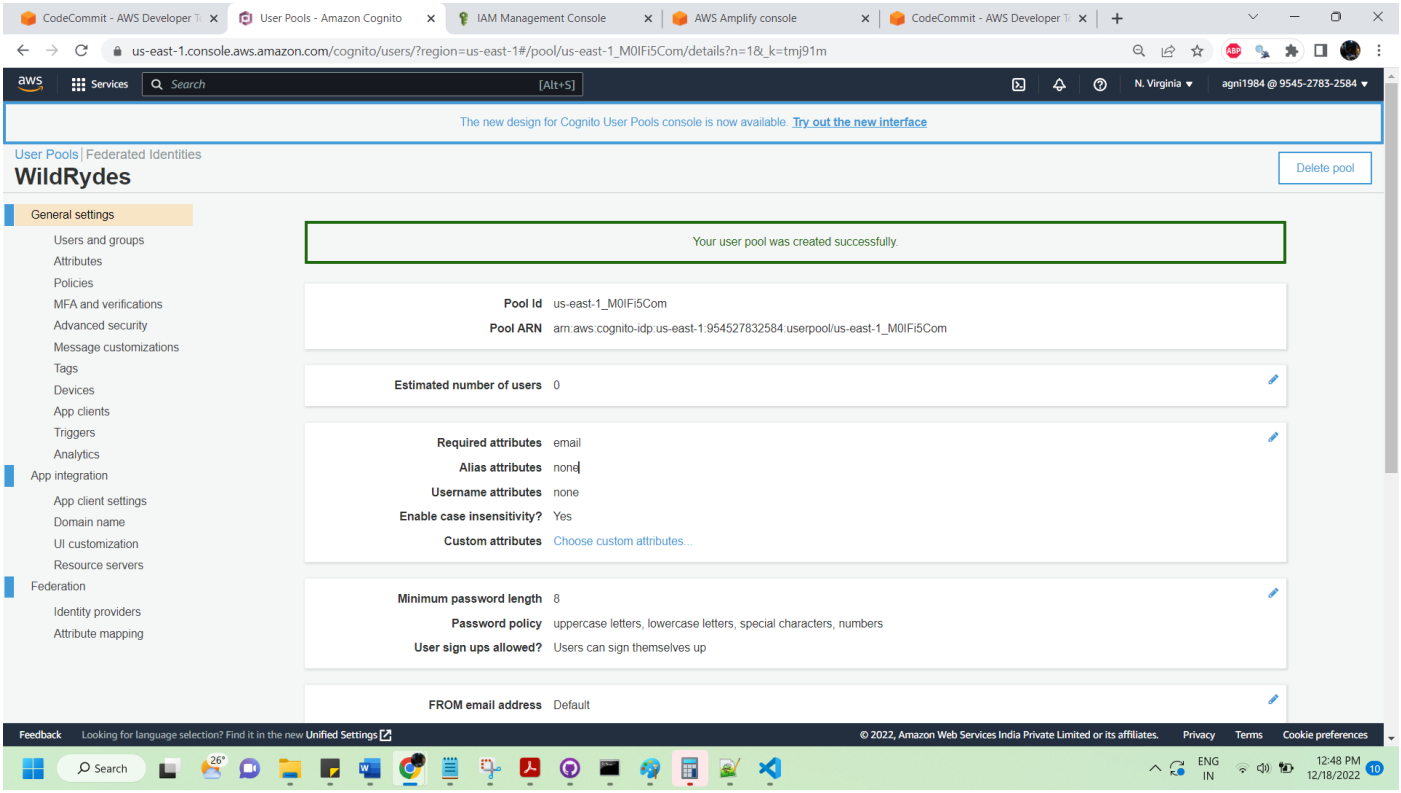
After users have a confirmed account (either using the email verification process or a manual confirmation through the console), they will be able to sign in. When users sign in, they enter their username (or email) and password. A JavaScript function then communicates with Amazon Cognito, authenticates using the Secure Remote Password protocol (SRP), and receives back a set of JSON Web Tokens (JWT). The JWTs contain claims about the identity of the user and will be used in the next module to authenticate against the RESTful API we build with Amazon API Gateway.

Step 1: Create an Amazon Cognito User Pool

Amazon Cognito provides two different mechanisms for authenticating users. We can use Cognito User Pools to add sign-up and sign-in functionality to our application or use Cognito Identity Pools to authenticate

users through social identity providers such as Facebook, Twitter, or Amazon, with SAML identity solutions, or by using our own identity system. In this project, we are using, user pool as the backend for the provided registration and sign-in pages.

- From the AWS Console click Services then select Cognito under Mobile Services
- Choose Manage our User Pools
- Choose Create a User Pool
- Provide a name for the user pool s **WildRydes**, then select Review Defaults
- On the review page, click Create pool
- Note the Pool Id on the Pool details page of our newly created user pool.
  Pool Id us-east-1_M0IFi5Com
  Pool ARN arn:aws:cognito-idp:us-east-1:954527832584:userpool/us-east-1_M0IFi5Com



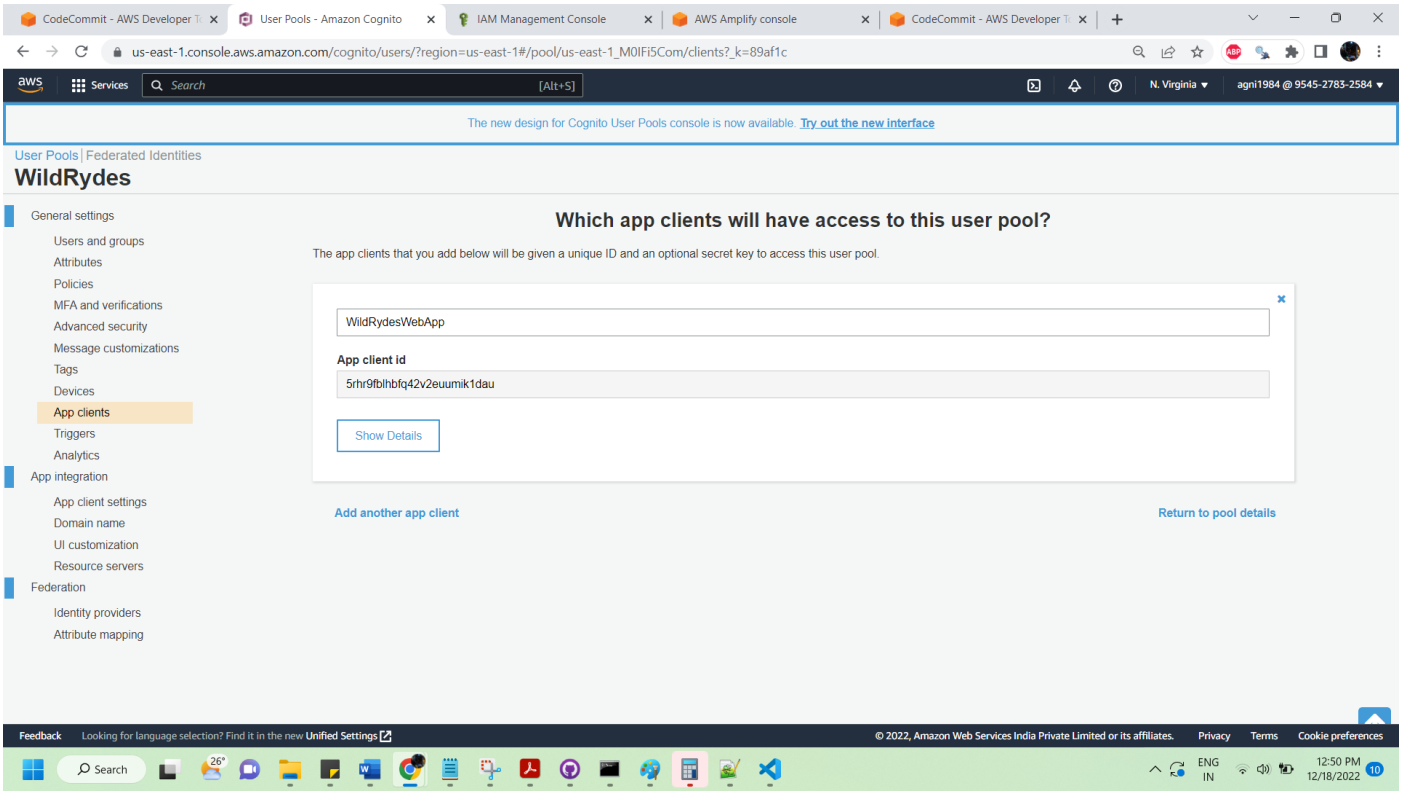
Step 2: Add an App to User Pool

Follow the below step to add 'App client' to 'user pool',

- From the Pool Details page for the user pool, select App clients from the left General Settings section in the navigation bar.
- Choose Add an app client.
- Give the app client a name such as **WildRydesWebApp**.

- Uncheck the Generate client secret option. Client secrets aren't currently supported for use with browser-based applications. Keep the default settings for the other options.
- Choose Create app client.
- Note the App client id for the newly created application.

app client id: 5rhr9fblhbfq42v2euumik1dau



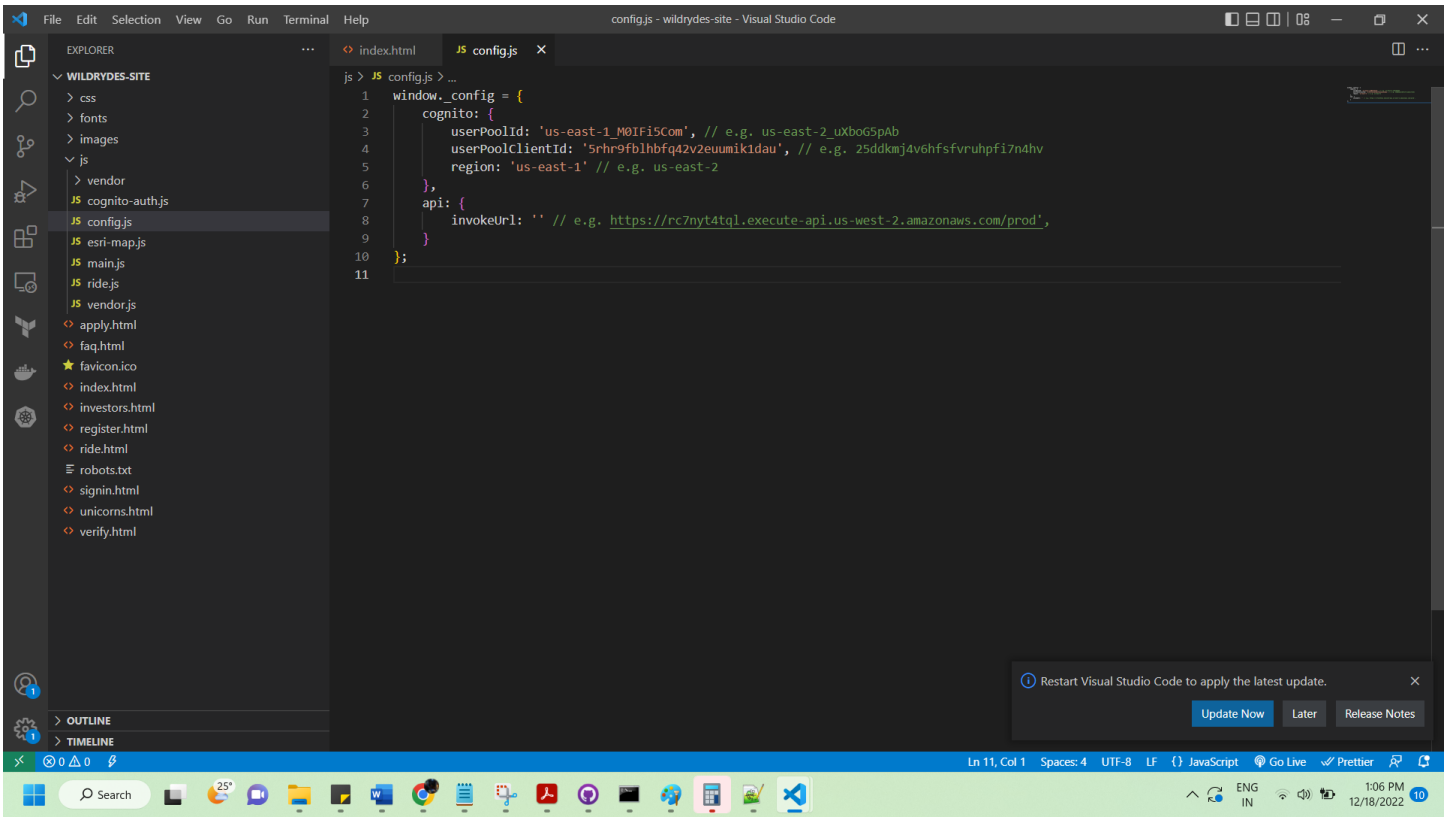
Step 3: Update the website config

The /js/config.js file contains settings for the user pool ID, app client ID and Region.

Pool Id us-east-1_M0IFi5Com

app client id: 5rhr9fblhbfq42v2euumik1dau

Region: us-east-1



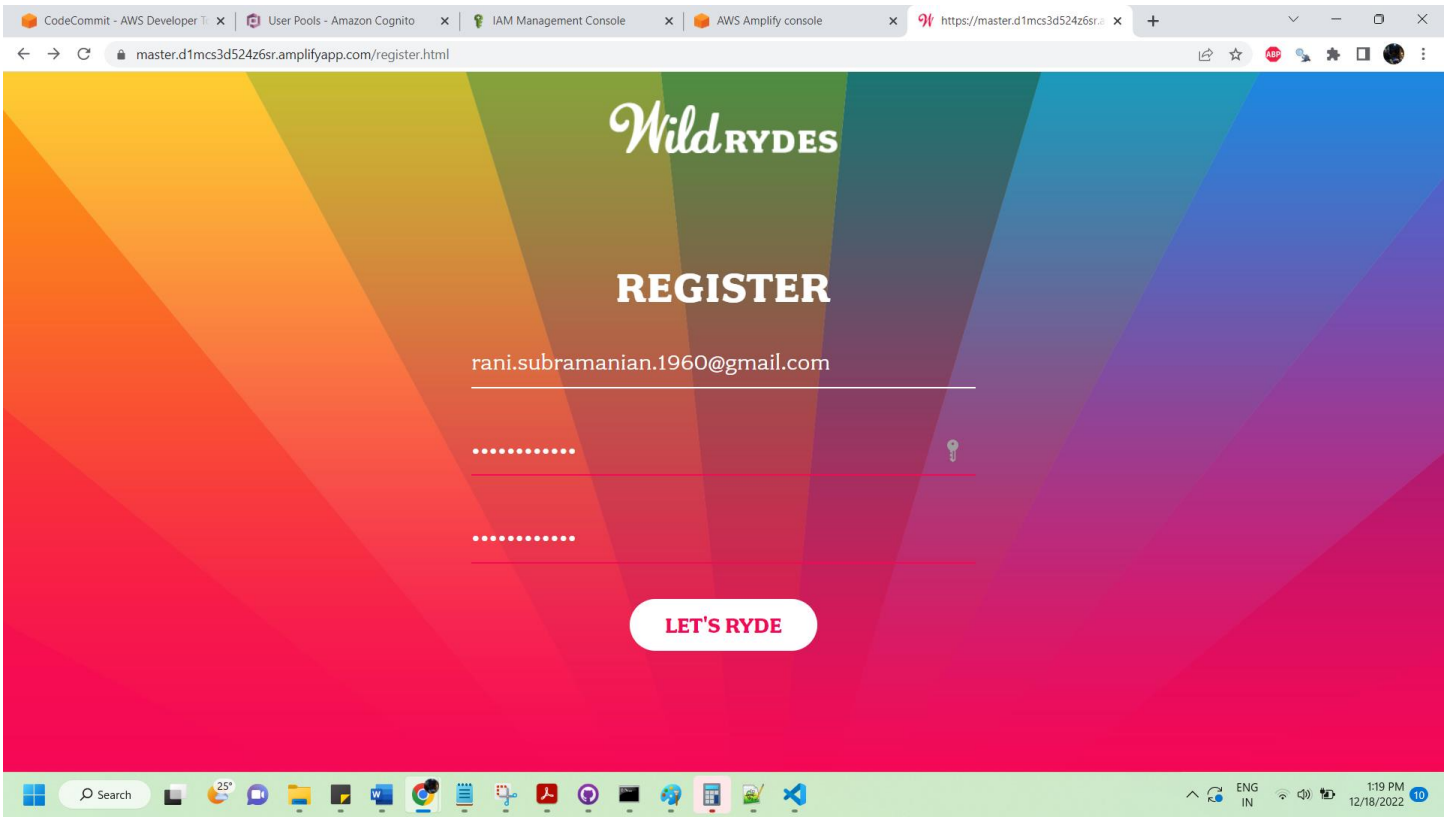
Save the modified file and push it to our Git repository to have it automatically deploy to Amplify Console.

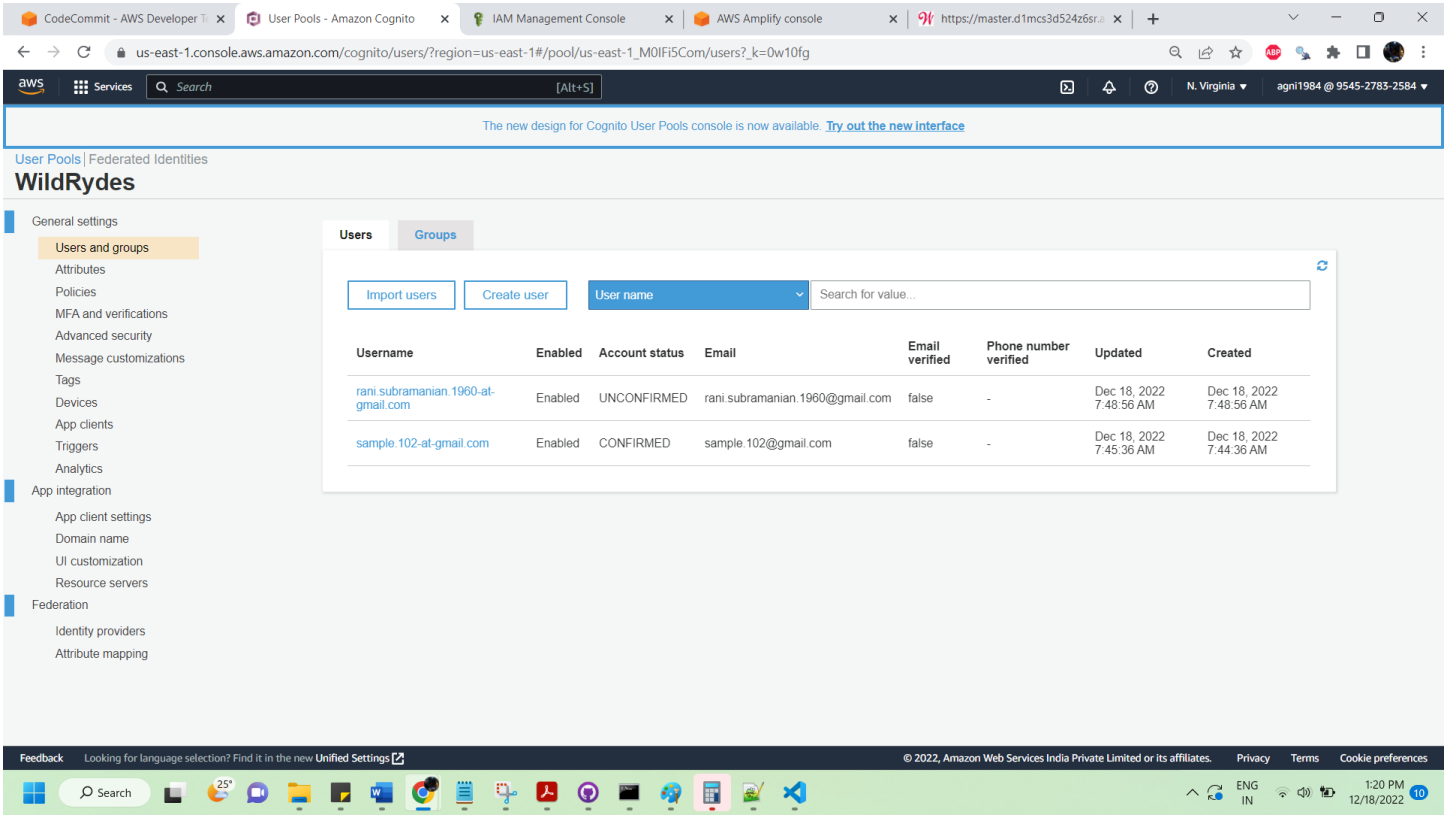
```
$ git add .
$ git commit -m "new_config"
$ git push
```

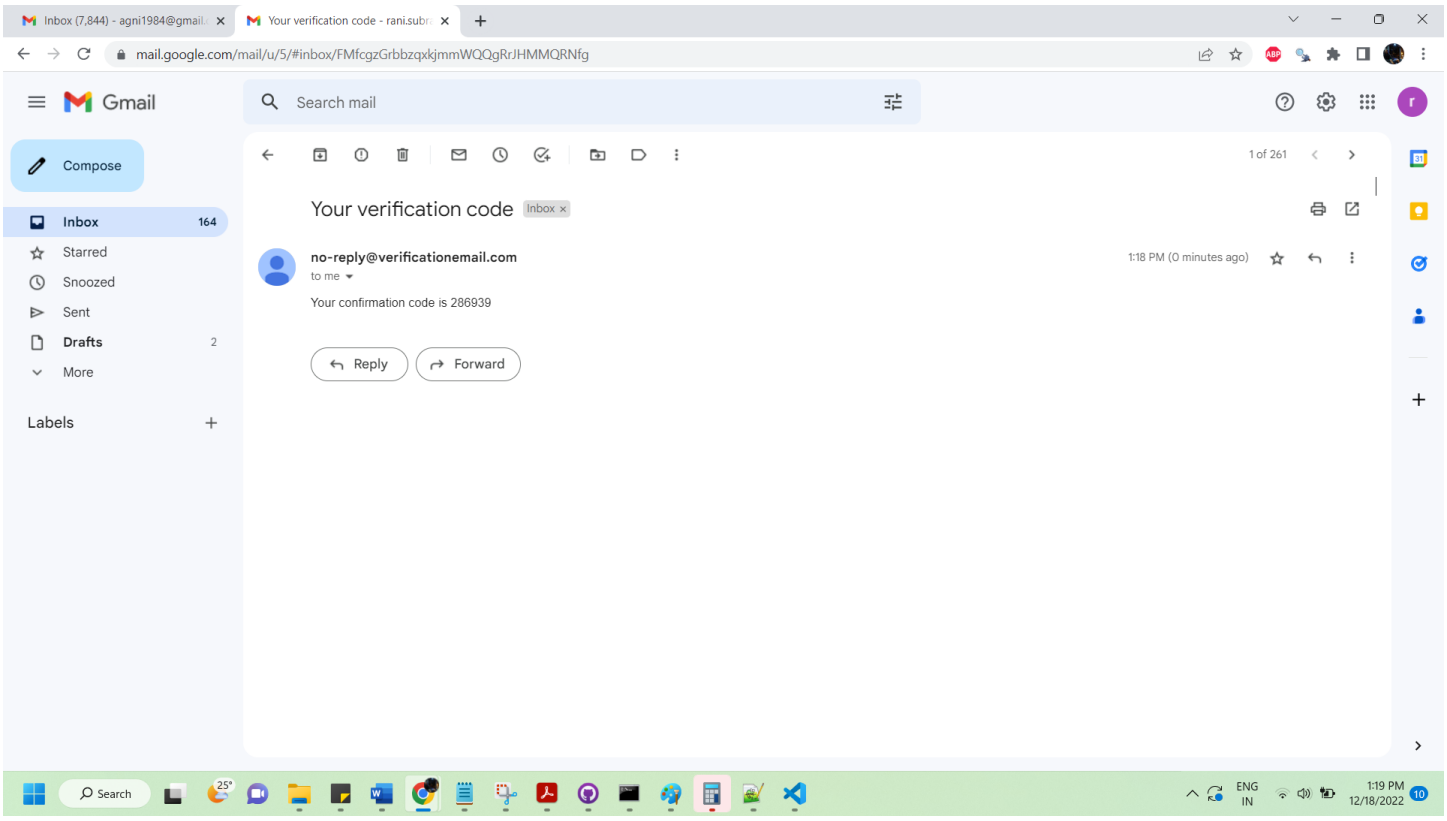
Step 4: Validate the Implementation

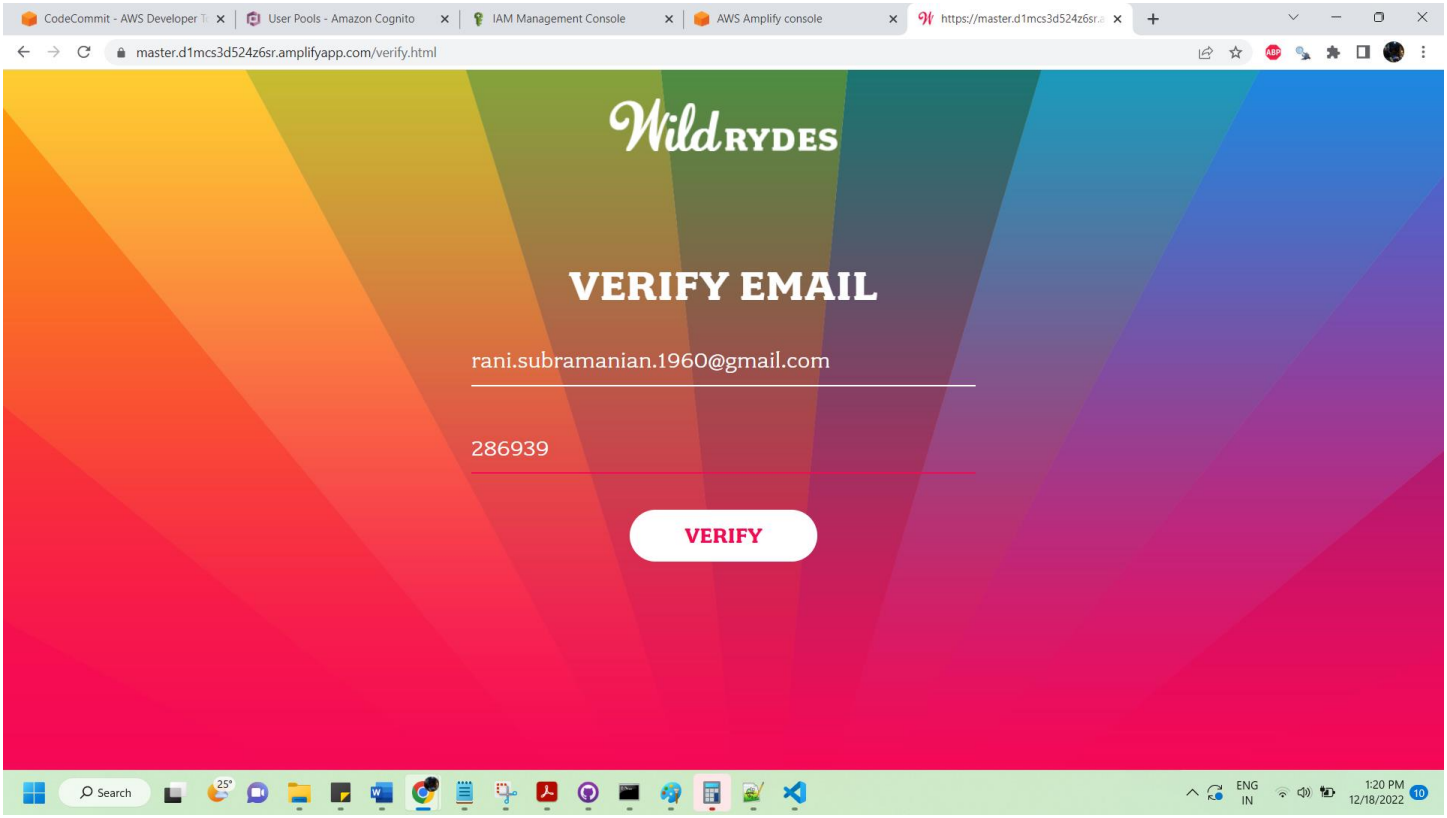
- Visit /register.html under the website domain, or choose the Giddy Up! button on the homepage of our site.

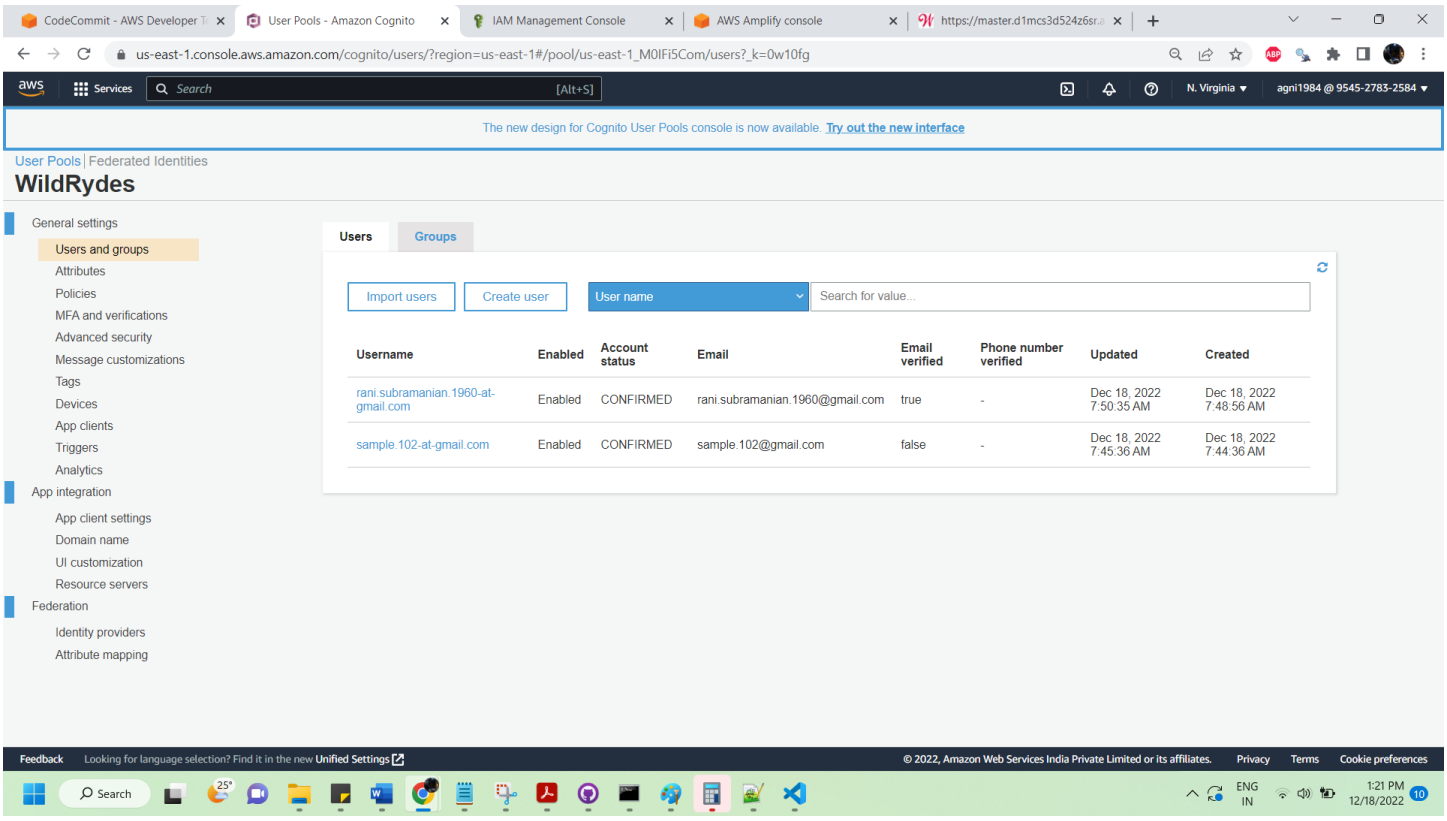
- Complete the registration form and choose Let's Ryde. We can use our own email or enter a fake email. Make sure to choose a password that contains at least one upper-case letter, a number, and a special character. Don't forget the password we entered for later. We should see an alert that confirms that our user has been created.
- Confirm our new user using one of the two following methods.
- For valid email address,
    - If we used an email address, we can complete the account verification process by visiting /verify.html under our website domain and entering the verification code that is emailed to us.

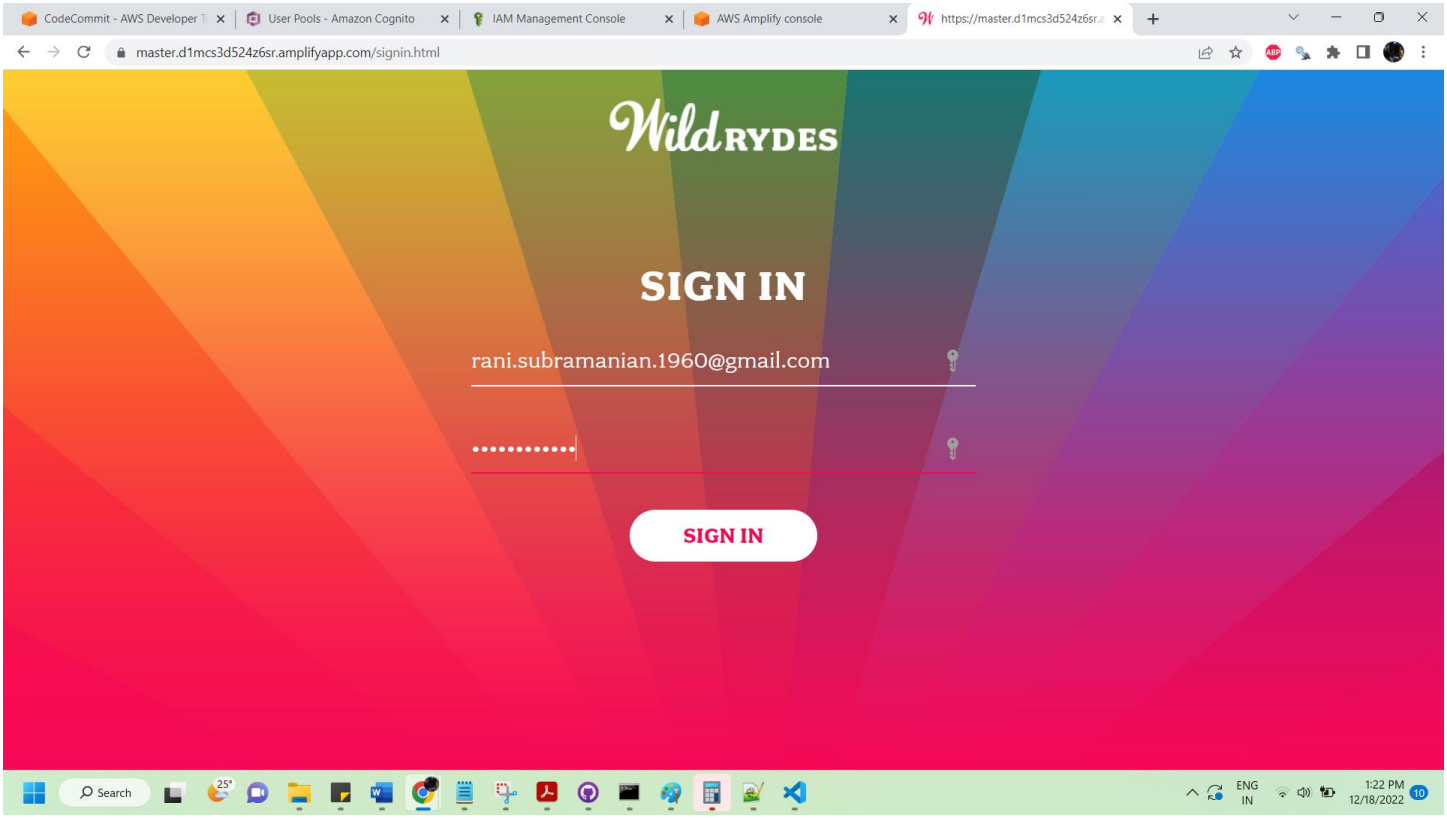


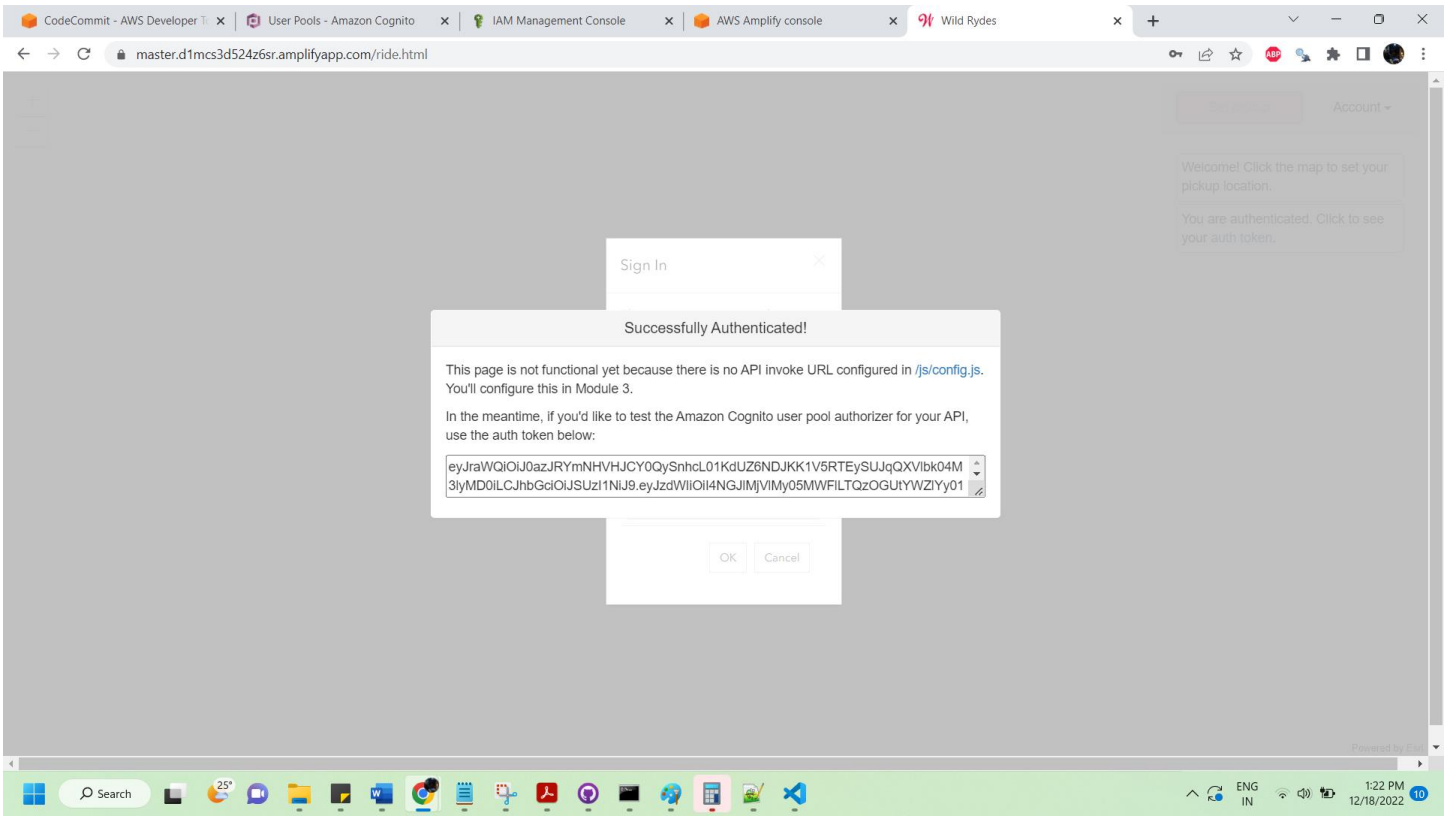

CodeCommit - AWS Developer
User Pools - Amazon Cognito
IAM Management Console
AWS Amplify console
us-east-1.console.aws.amazon.com/cognito/users/?region=us-east-1#/pool/us-east-1_M0IFi5Com/users?_k=0w10fg
Services
Search
[Alt+S]
N. Virginia
agni1984 @ 9545-2783-2584
The new design for Cognito User Pools console is now available. Try out the new interface
User Pools | Federated Identities
WildRydes
General settings
Users and groups
Attributes
Policies
MFA and verifications
Advanced security
Message customizations
Tags
Devices
App clients
Triggers
Analytics
App integration
App client settings
Domain name
UI customization
Resource servers
Federation
Identity providers
Attribute mapping
Users
Groups
Import users
Create user
User name
Search for value...
Username
Enabled
Account status
Email
Email verified
Phone number verified
Updated
Created
rani.subramanian.1960-at-gmail.com
Enabled
UNCONFIRMED
rani.subramanian.1960@gmail.com
false
-
Dec 18, 2022 7:48:56 AM
Dec 18, 2022 7:48:56 AM
sample.102-at-gmail.com
Enabled
CONFIRMED
sample.102@gmail.com
false
-
Dec 18, 2022 7:45:36 AM
Dec 18, 2022 7:44:36 AM
Feedback
Looking for language selection? Find it in the new Unified Settings
© 2022, Amazon Web Services India Private Limited or its affiliates.
Privacy
Terms
Cookie preferences
Search
ENG IN
1:20 PM 12/18/2022


Inbox (7,844) - agni1984@gmail.
Your verification code - rani.subr
mail.google.com/mail/u/5/#inbox/FMfcgzGrbbzqxkjmmWQQgRrJHMMQRNfg
Gmail
Search mail
Compose
Inbox 164
Starred
Snoozed
Sent
Drafts 2
More
Labels
1 of 261
Your verification code
Inbox
no-reply@verificationemail.com
to me
1:18 PM (0 minutes ago)
Your confirmation code is 286939
Reply
Forward
Search
ENG IN
1:19 PM 12/18/2022


CodeCommit - AWS Developer
User Pools - Amazon Cognito
IAM Management Console
AWS Amplify console
https://master.d1mcs3d524z6sr.a
master.d1mcs3d524z6sr.amplifyapp.com/verify.html
WildRydes
VERIFY EMAIL
rani.subramanian.1960@gmail.com
286939
VERIFY
Search
ENG
IN
1:20 PM
12/18/2022

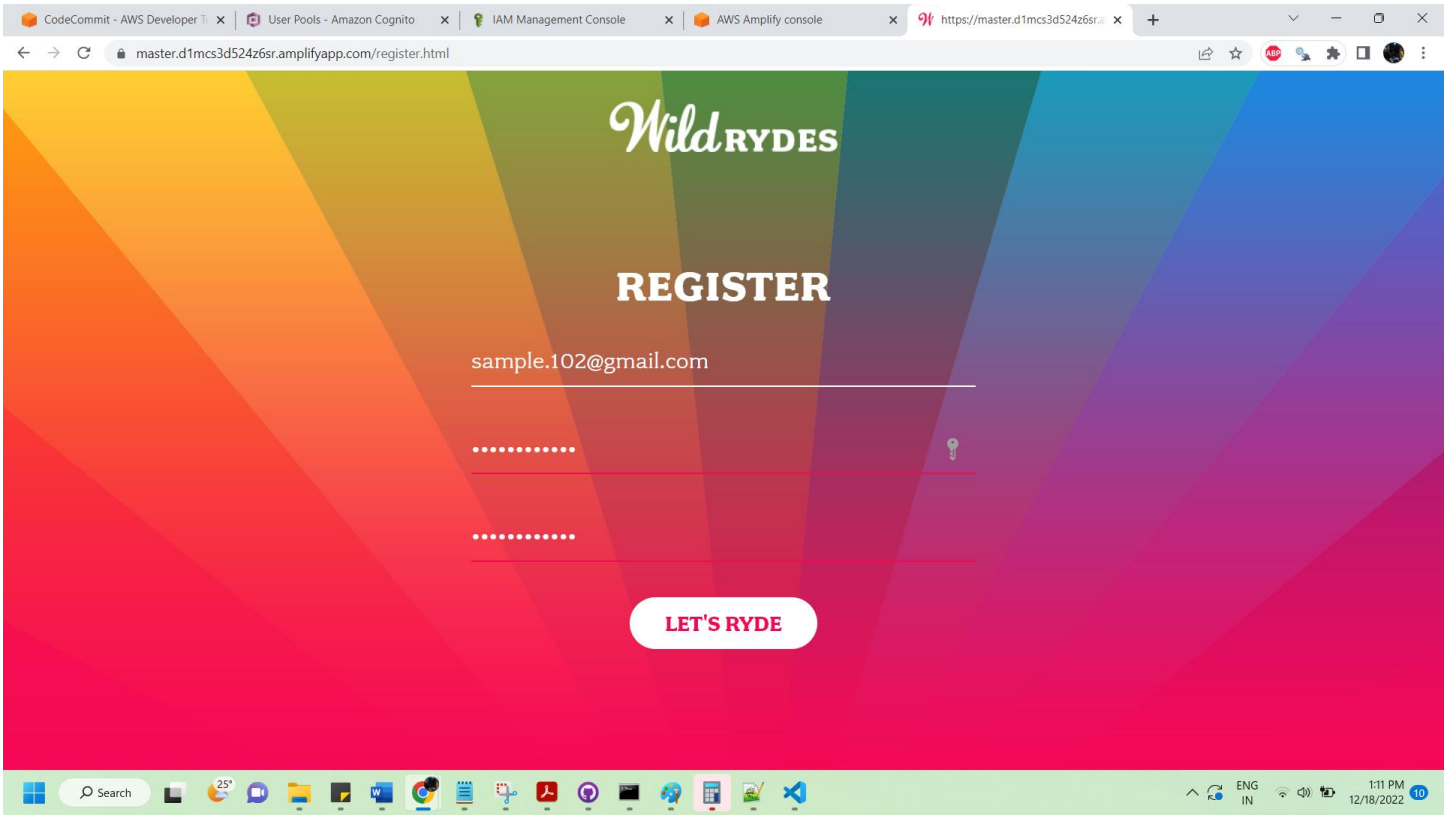

CodeCommit - AWS Developer
User Pools - Amazon Cognito
IAM Management Console
AWS Amplify console
https://master.d1mcs3d524z6sr.a
us-east-1.console.aws.amazon.com/cognito/users/?region=us-east-1#/pool/us-east-1_M0IFi5Com/users?_k=0w10fg
aws
Services
Search
[Alt+S]
N. Virginia
agni1984 @ 9545-2783-2584
The new design for Cognito User Pools console is now available. Try out the new interface
User Pools | Federated Identities
WildRydes
General settings
Users and groups
Attributes
Policies
MFA and verifications
Advanced security
Message customizations
Tags
Devices
App clients
Triggers
Analytics
App integration
App client settings
Domain name
UI customization
Resource servers
Federation
Identity providers
Attribute mapping
Users
Groups
Import users
Create user
User name
Search for value...
Username
Enabled
Account status
Email
Email verified
Phone number verified
Updated
Created
rani.subramanian.1960-at-gmail.com
Enabled
CONFIRMED
rani.subramanian.1960@gmail.com
true
-
Dec 18, 2022 7:50:35 AM
Dec 18, 2022 7:48:56 AM
sample.102-at-gmail.com
Enabled
CONFIRMED
sample.102@gmail.com
false
-
Dec 18, 2022 7:45:36 AM
Dec 18, 2022 7:44:36 AM
Feedback
Looking for language selection? Find it in the new Unified Settings
© 2022, Amazon Web Services India Private Limited or its affiliates.
Privacy
Terms
Cookie preferences
Search
ENG
IN
1:21 PM
12/18/2022

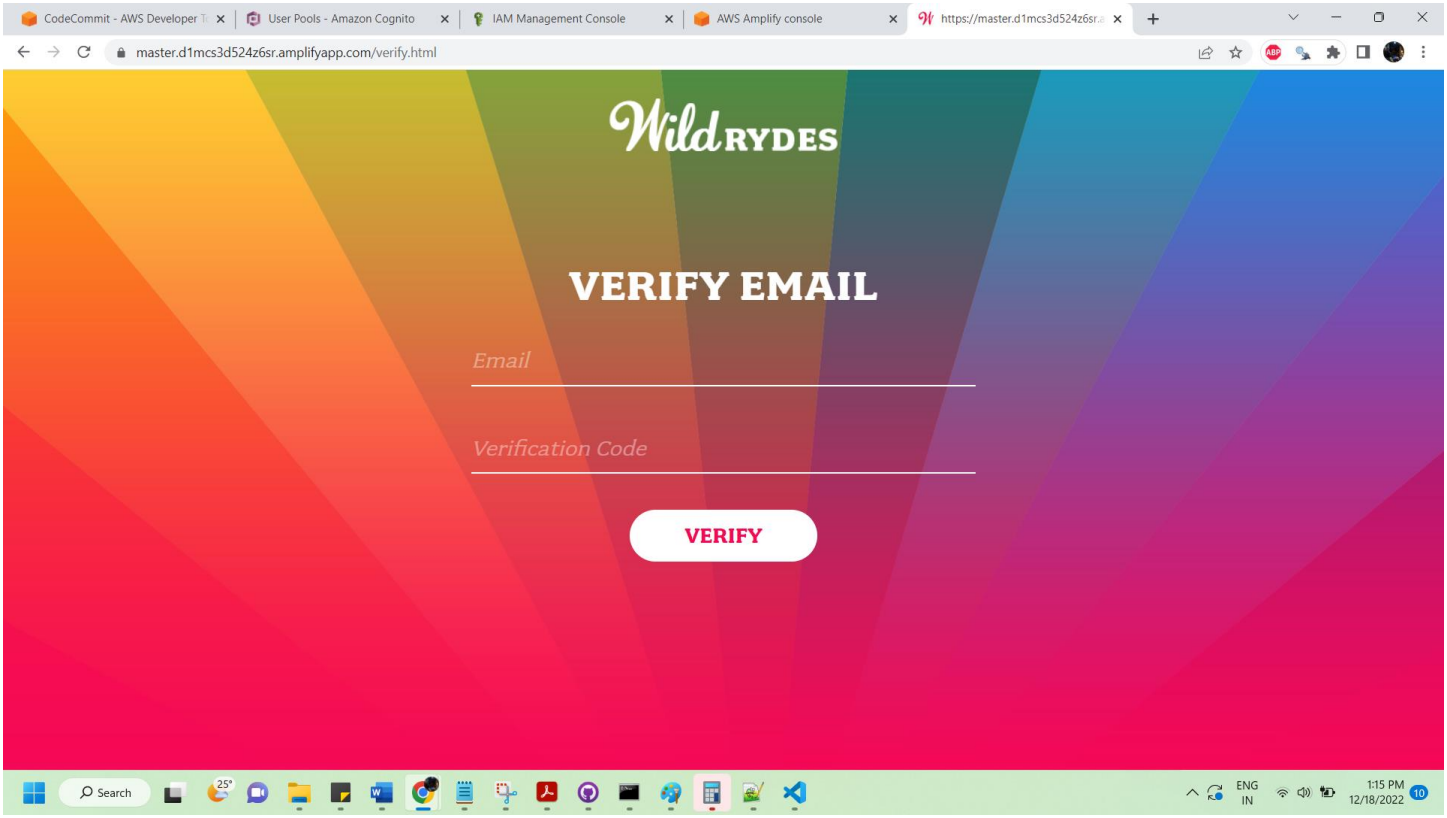


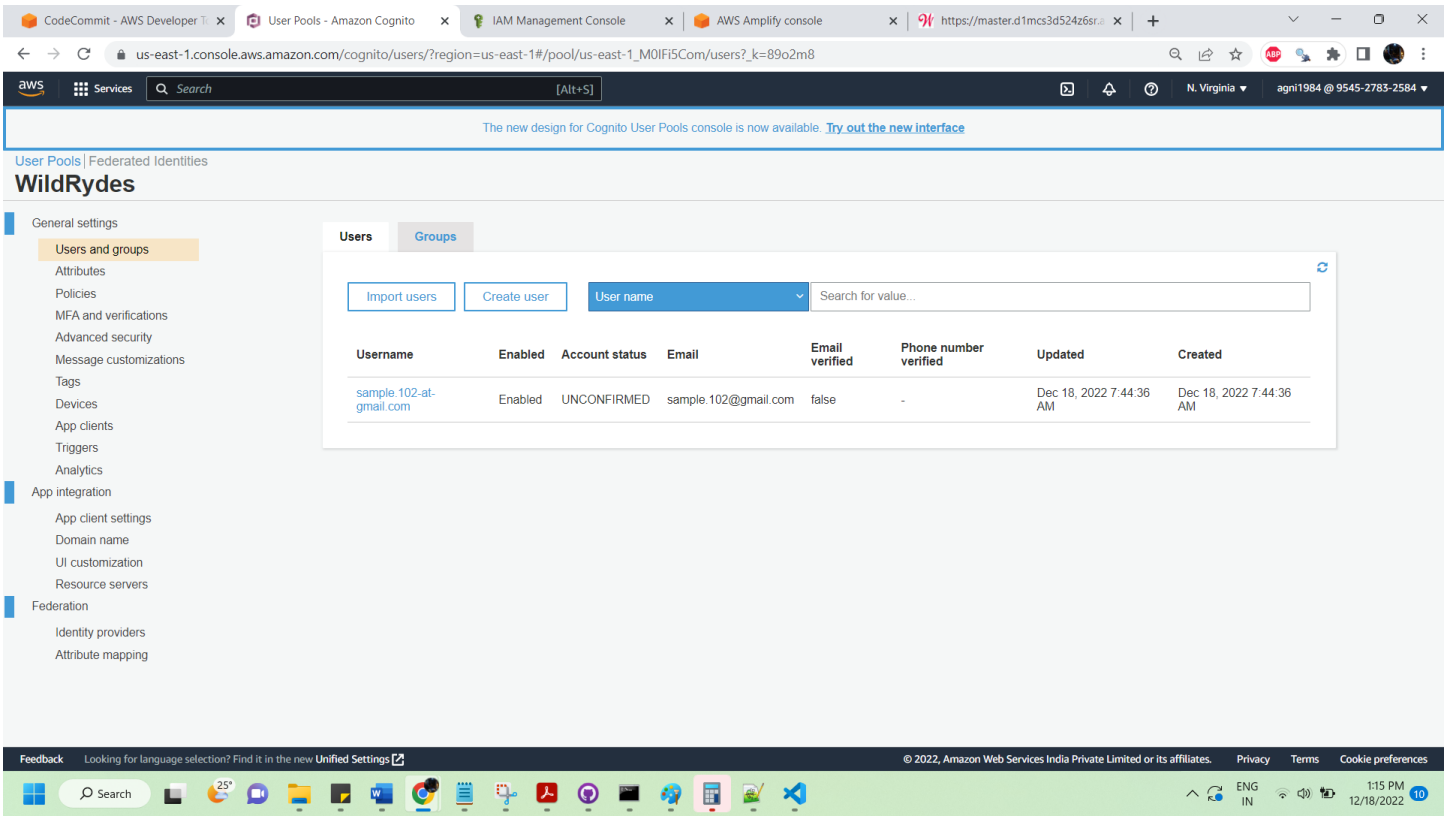


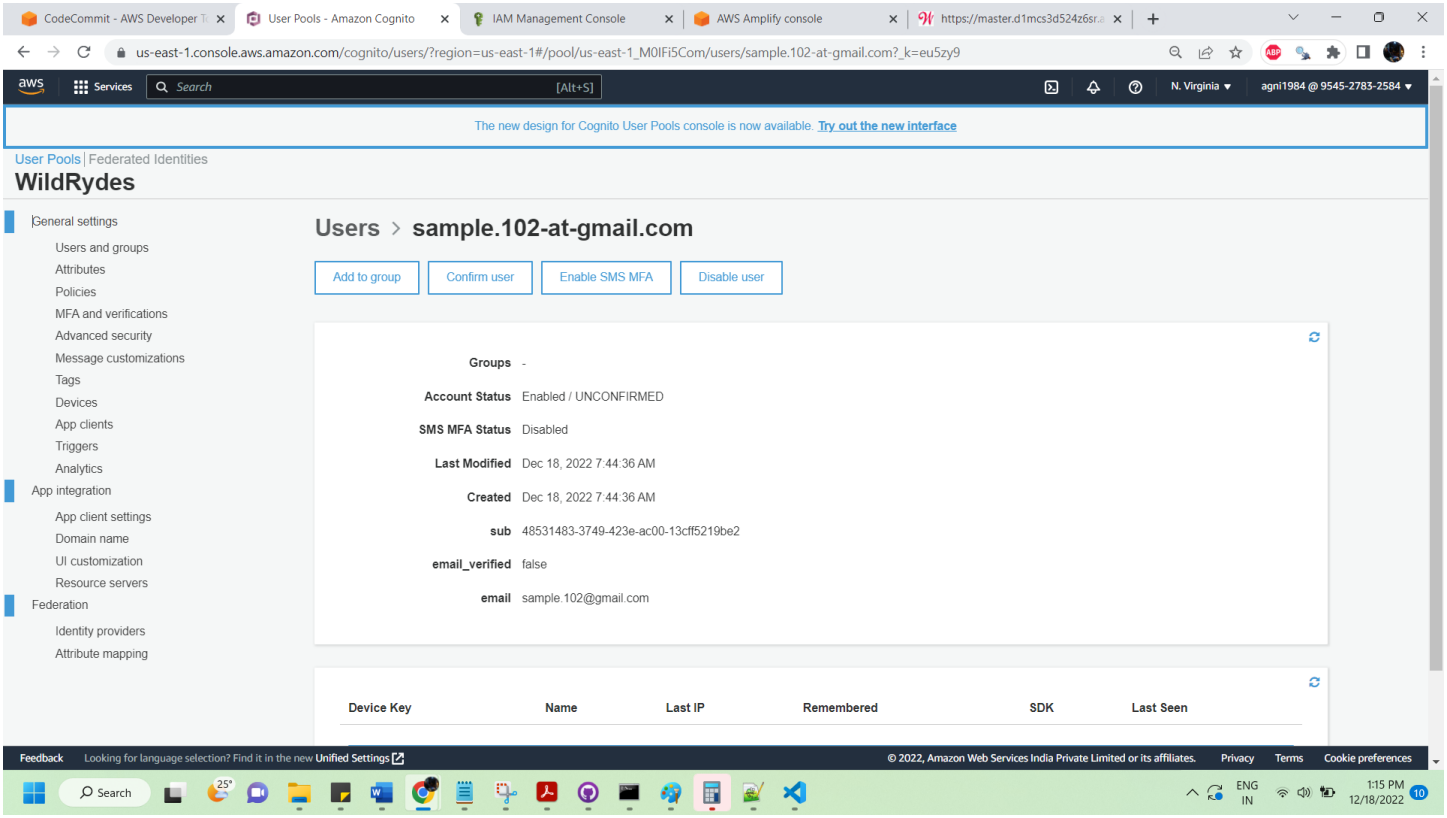
- For Dummy email address,
  - If dummy email address is used, confirm the user manually through the Cognito console.

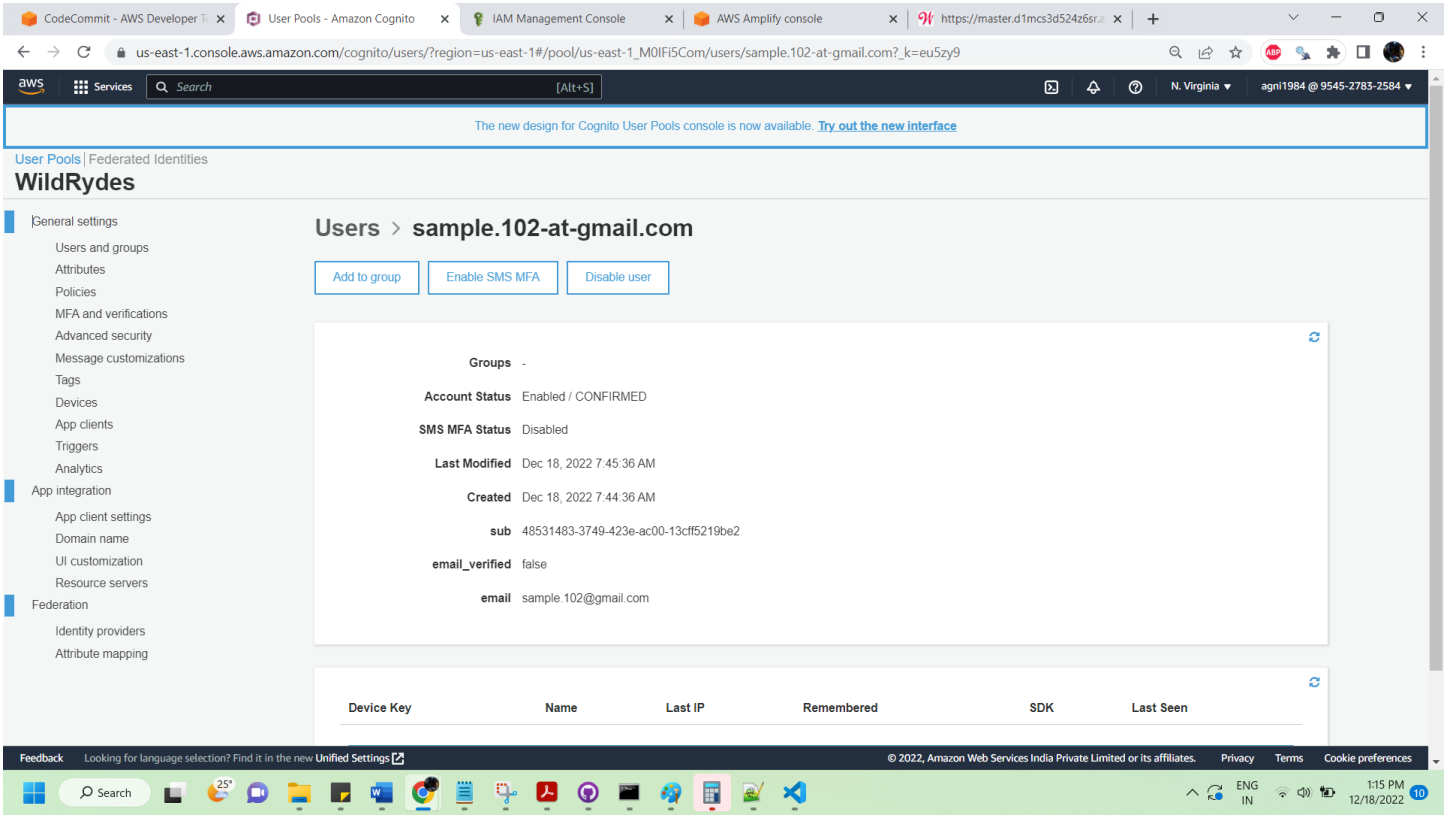
- From the AWS console, click Services then select Cognito under Security, Identity & Compliance.
- Choose Manage our User Pools
- Select the WildRydes user pool and click Users and groups in the left navigation bar.
- User corresponding to the email address that we submitted through the registration page should be seen. Choose that username to view the user detail page.
- Choose Confirm user to finalize the account creation process.
- After confirming the new user using either the /verify.html page or the Cognito console, visit /signin.html and log in using the email address and password entered during the registration step.
- If successful , we should be redirected to /ride.html. we should see a notification that the API is not configured.

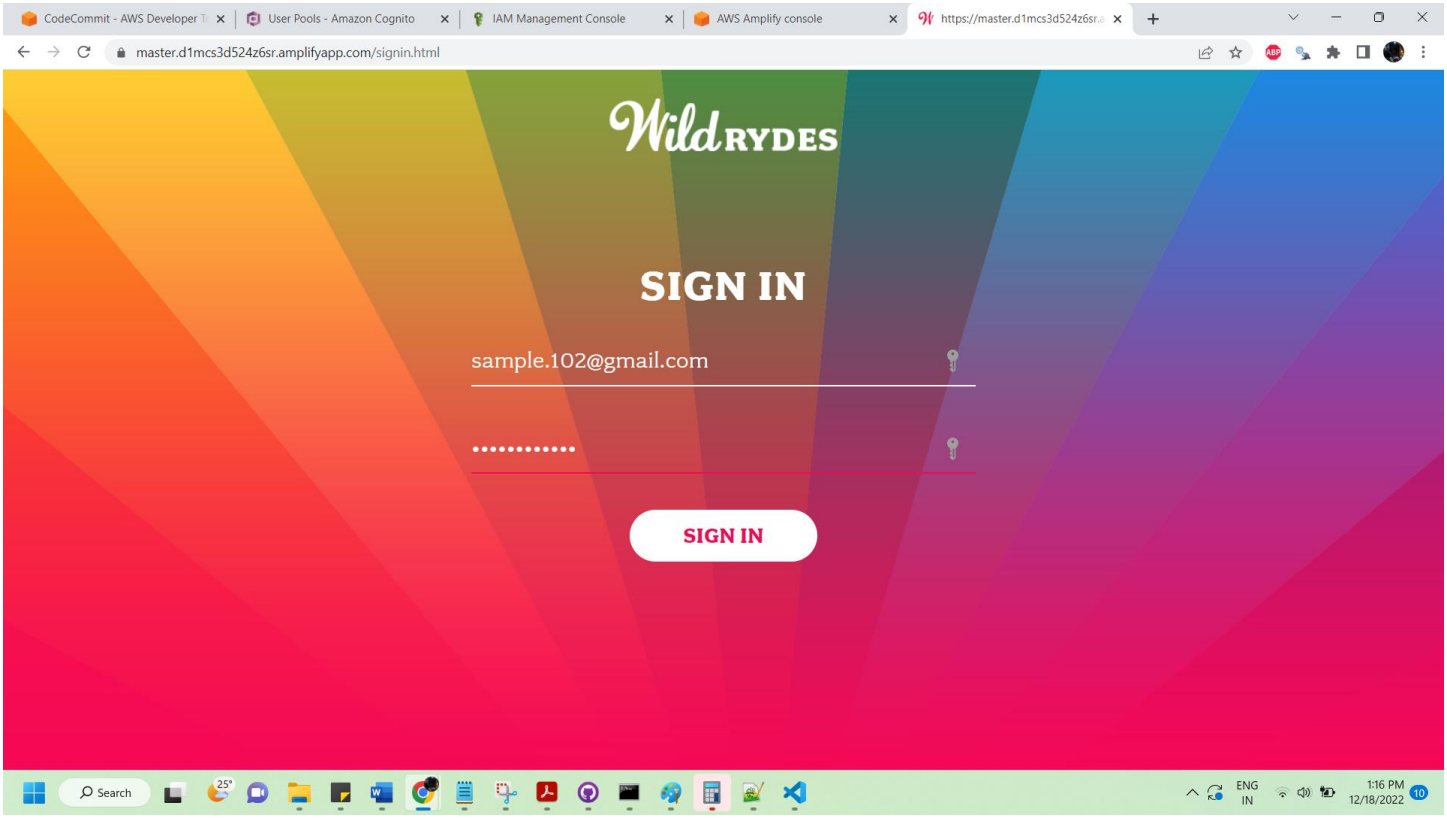


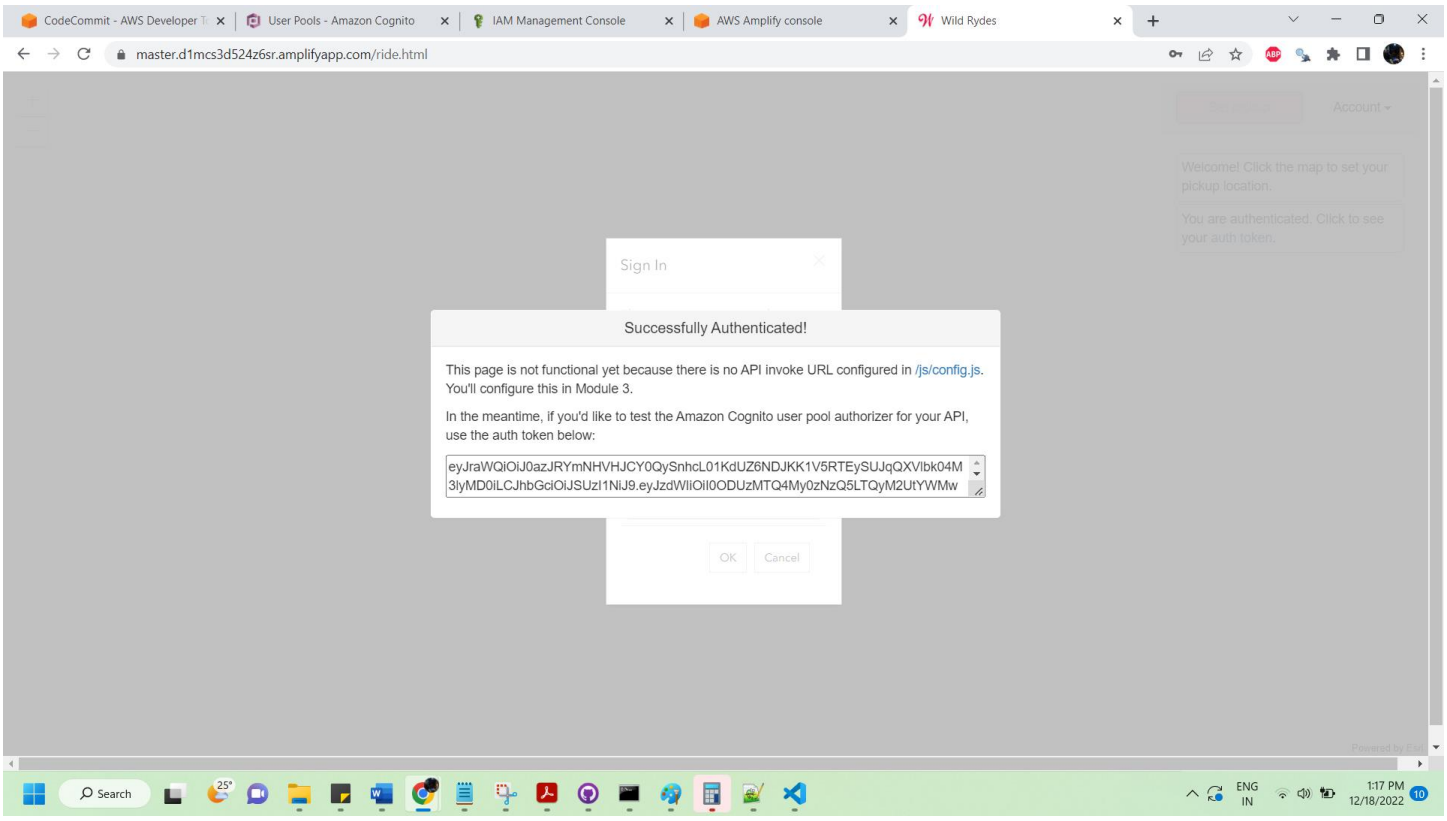

CodeCommit - AWS Developer
User Pools - Amazon Cognito
IAM Management Console
AWS Amplify console
https://master.d1mcs3d524z6sr.a
master.d1mcs3d524z6sr.amplifyapp.com/verify.html
WildRydes
VERIFY EMAIL
Email
Verification Code
VERIFY
Search
ENG
IN
1:15 PM
12/18/2022


CodeCommit - AWS Developer
User Pools - Amazon Cognito
IAM Management Console
AWS Amplify console
https://master.d1mcs3d524z6sr.a
us-east-1.console.aws.amazon.com/cognito/users/?region=us-east-1#/pool/us-east-1_M0IFi5Com/users?_k=89o2m8
aws
Services
Search
[Alt+S]
N. Virginia
agni1984 @ 9545-2783-2584
The new design for Cognito User Pools console is now available. Try out the new interface
User Pools | Federated Identities
WildRydes
General settings
Users and groups
Attributes
Policies
MFA and verifications
Advanced security
Message customizations
Tags
Devices
App clients
Triggers
Analytics
App integration
App client settings
Domain name
UI customization
Resource servers
Federation
Identity providers
Attribute mapping
Users
Groups
Import users
Create user
User name
Search for value...
Username
Enabled
Account status
Email
Email verified
Phone number verified
Updated
Created
sample.102-at-gmail.com
Enabled
UNCONFIRMED
sample.102@gmail.com
false
-
Dec 18, 2022 7:44:36 AM
Dec 18, 2022 7:44:36 AM
Feedback
Looking for language selection? Find it in the new Unified Settings
© 2022, Amazon Web Services India Private Limited or its affiliates.
Privacy
Terms
Cookie preferences
Search
ENG
IN
1:15 PM
12/18/2022


CodeCommit - AWS Developer
User Pools - Amazon Cognito
IAM Management Console
AWS Amplify console
us-east-1.console.aws.amazon.com/cognito/users/?region=us-east-1#/pool/us-east-1_M0IFi5Com/users/sample.102-at-gmail.com?_k=eu5zy9
Services
Search
[Alt+S]
N. Virginia
agni1984 @ 9545-2783-2584
The new design for Cognito User Pools console is now available. Try out the new interface
User Pools | Federated Identities
WildRydes
General settings
Users and groups
Attributes
Policies
MFA and verifications
Advanced security
Message customizations
Tags
Devices
App clients
Triggers
Analytics
App integration
App client settings
Domain name
UI customization
Resource servers
Federation
Identity providers
Attribute mapping
Users > sample.102-at-gmail.com
Add to group
Confirm user
Enable SMS MFA
Disable user
Groups -
Account Status Enabled / UNCONFIRMED
SMS MFA Status Disabled
Last Modified Dec 18, 2022 7:44:36 AM
Created Dec 18, 2022 7:44:36 AM
sub 48531483-3749-423e-ac00-13cff5219be2
email_verified false
email sample.102@gmail.com
Device Key
Name
Last IP
Remembered
SDK
Last Seen
Feedback
Looking for language selection? Find it in the new Unified Settings
© 2022, Amazon Web Services India Private Limited or its affiliates.
Privacy
Terms
Cookie preferences
Search
ENG IN
1:15 PM 12/18/2022


CodeCommit - AWS Developer
User Pools - Amazon Cognito
IAM Management Console
AWS Amplify console
us-east-1.console.aws.amazon.com/cognito/users/?region=us-east-1#/pool/us-east-1_M0IFi5Com/users/sample.102-at-gmail.com?_k=eu5zy9
Services
Search
[Alt+S]
N. Virginia
agni1984 @ 9545-2783-2584
The new design for Cognito User Pools console is now available. Try out the new interface
User Pools | Federated Identities
WildRydes
General settings
Users and groups
Attributes
Policies
MFA and verifications
Advanced security
Message customizations
Tags
Devices
App clients
Triggers
Analytics
App integration
App client settings
Domain name
UI customization
Resource servers
Federation
Identity providers
Attribute mapping
Users > sample.102-at-gmail.com
Add to group
Enable SMS MFA
Disable user
Groups -
Account Status Enabled / CONFIRMED
SMS MFA Status Disabled
Last Modified Dec 18, 2022 7:45:36 AM
Created Dec 18, 2022 7:44:36 AM
sub 48531483-3749-423e-ac00-13cff5219be2
email_verified false
email sample.102@gmail.com
Device Key
Name
Last IP
Remembered
SDK
Last Seen
Feedback
Looking for language selection? Find it in the new Unified Settings
© 2022, Amazon Web Services India Private Limited or its affiliates.
Privacy
Terms
Cookie preferences
Search
ENG IN
1:15 PM 12/18/2022

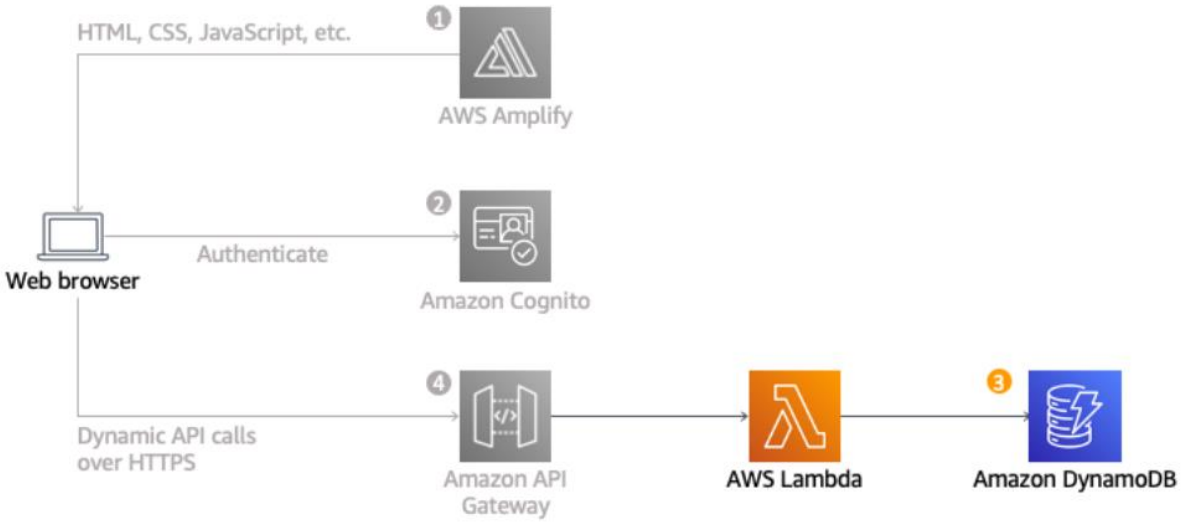

CodeCommit - AWS Developer
User Pools - Amazon Cognito
IAM Management Console
AWS Amplify console
https://master.d1mcs3d524z6sr.a
master.d1mcs3d524z6sr.amplifyapp.com/signin.html
WildRYDES
SIGN IN
sample.102@gmail.com
SIGN IN
Search
ENG
IN
1:16 PM
12/18/2022


CodeCommit - AWS Developer
User Pools - Amazon Cognito
IAM Management Console
AWS Amplify console
Wild Rydes
master.d1mcs3d524z6sr.amplifyapp.com/ride.html
Account
Welcome! Click the map to set your pickup location.
You are authenticated. Click to see your auth token.
Sign In
Successfully Authenticated!
This page is not functional yet because there is no API invoke URL configured in /js/config.js. You'll configure this in Module 3.
In the meantime, if you'd like to test the Amazon Cognito user pool authorizer for your API, use the auth token below:
eyJraWQiOiJ0azJRYmNHVHJCY0QySnhcL01KdUZ6NDJKK1V5RTEySUJqQXVlbk04M
3lyMD0iLCJhbGciOiJSUzI1NiJ9.eyJzdWIiOiI0ODUzMTQ4My0zNzQ5LTQyM2UtYWMw
OK
Cancel
Search
ENG
IN
1:17 PM
12/18/2022

Module 3: Serverless Backend

In this module, AWS Lambda and Amazon DynamoDB is used to build a backend process for handling requests for the web application. The browser application that we deployed in the first module allows users to request that a unicorn be sent to a location of their choice. To fulfill those requests, the JavaScript running in the browser will need to invoke a service running in the cloud.

Architectural Overview



We are implementing a Lambda function that will be invoked each time a user requests a unicorn. The function will select a unicorn from the fleet, record the request in a DynamoDB table, and then respond to the frontend application with details about the unicorn being dispatched.

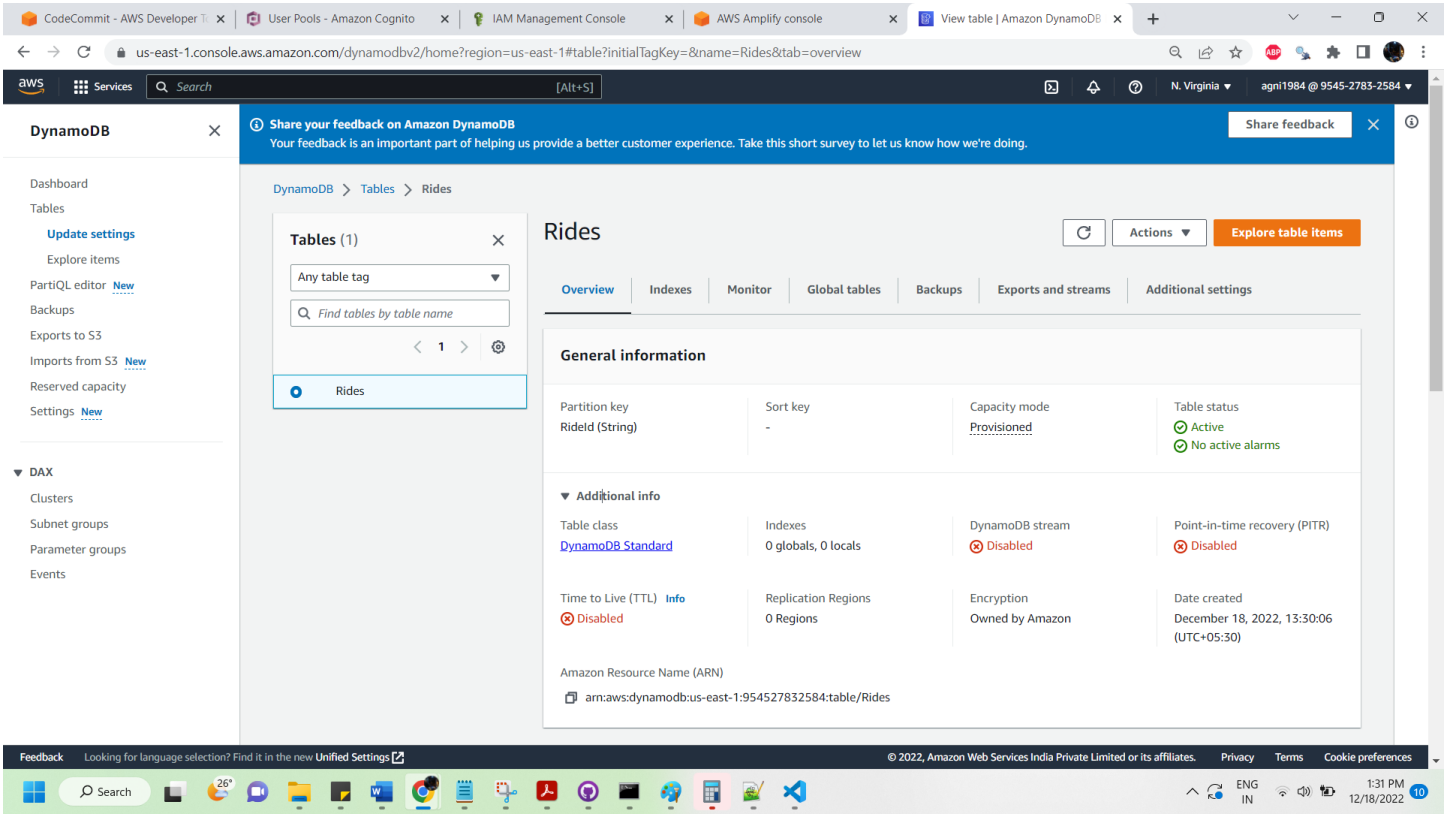
The function is invoked from the browser using Amazon API Gateway. API Gateway will be implemented in the next module.

Step 1: Create an Amazon DynamoDB Table

Create Dynamo DB table called ‘Rides’ and give it a partition key called ‘RideId’ with type String. Follow the below steps to create DynamoDB table,

- From the AWS Management Console, choose Services then select DynamoDB under Databases.
- Choose Create table.
- Enter ‘Rides’ for the Table name. This field is case sensitive.
- Enter ‘RideId’ for the Partition key and select String for the key type. This field is case sensitive.
- Check the Use default settings box and choose Create.
- Scroll to the bottom of the Overview section of the new table and note the ARN.

ARN: arn:aws:dynamodb:us-east-1:954527832584:table/Rides

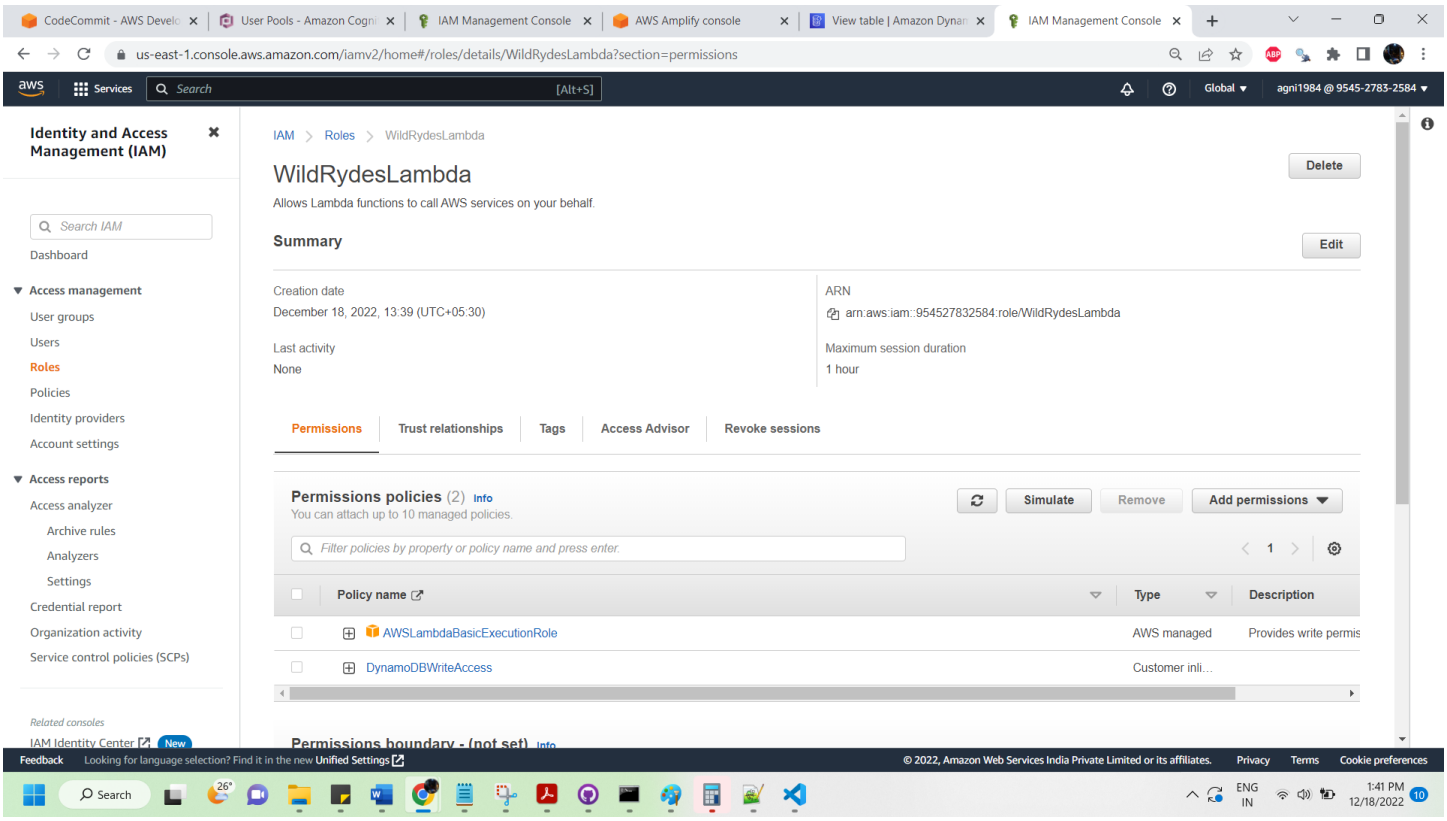


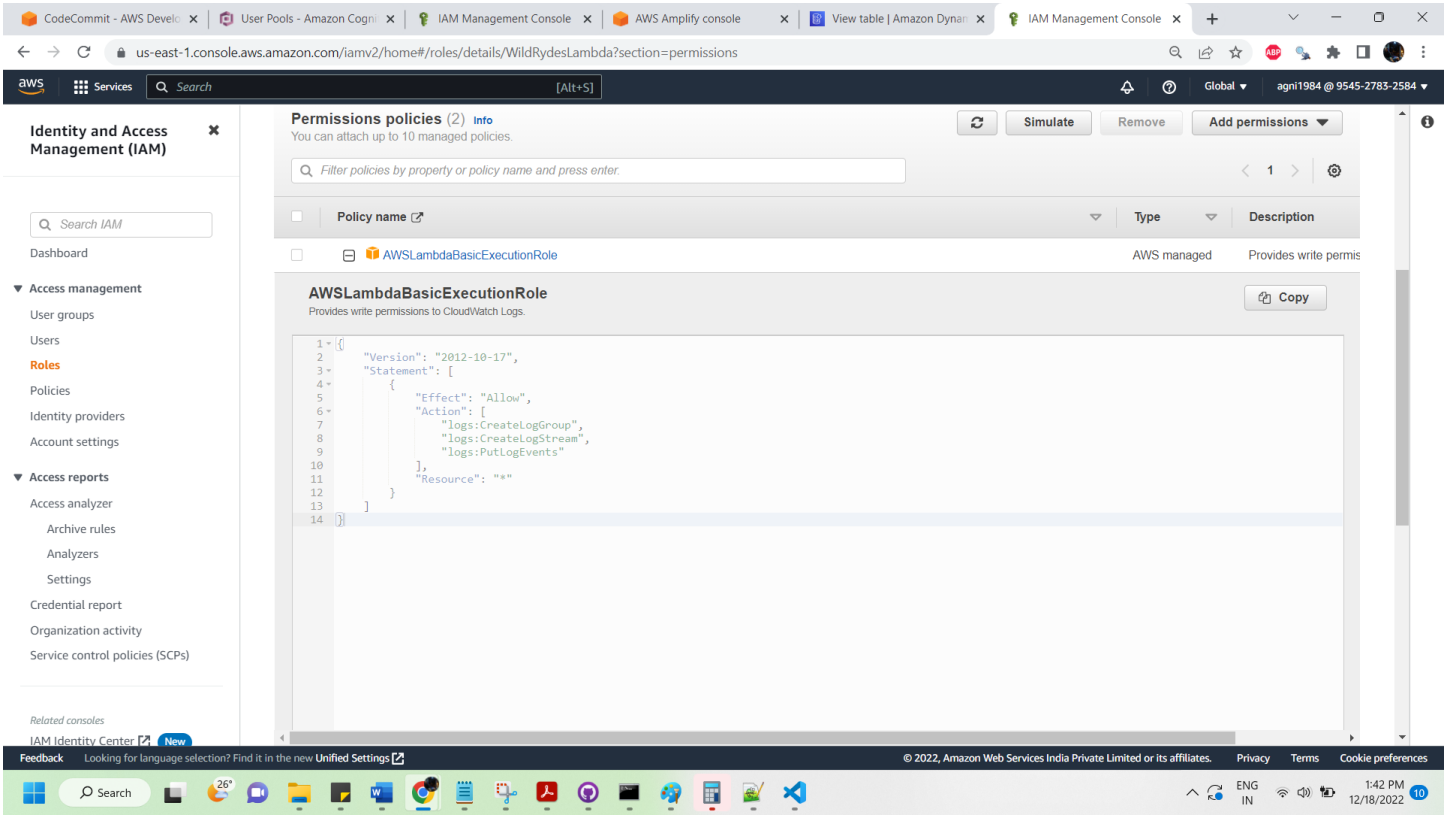
Step 2: Create an IAM Role for the Lambda function

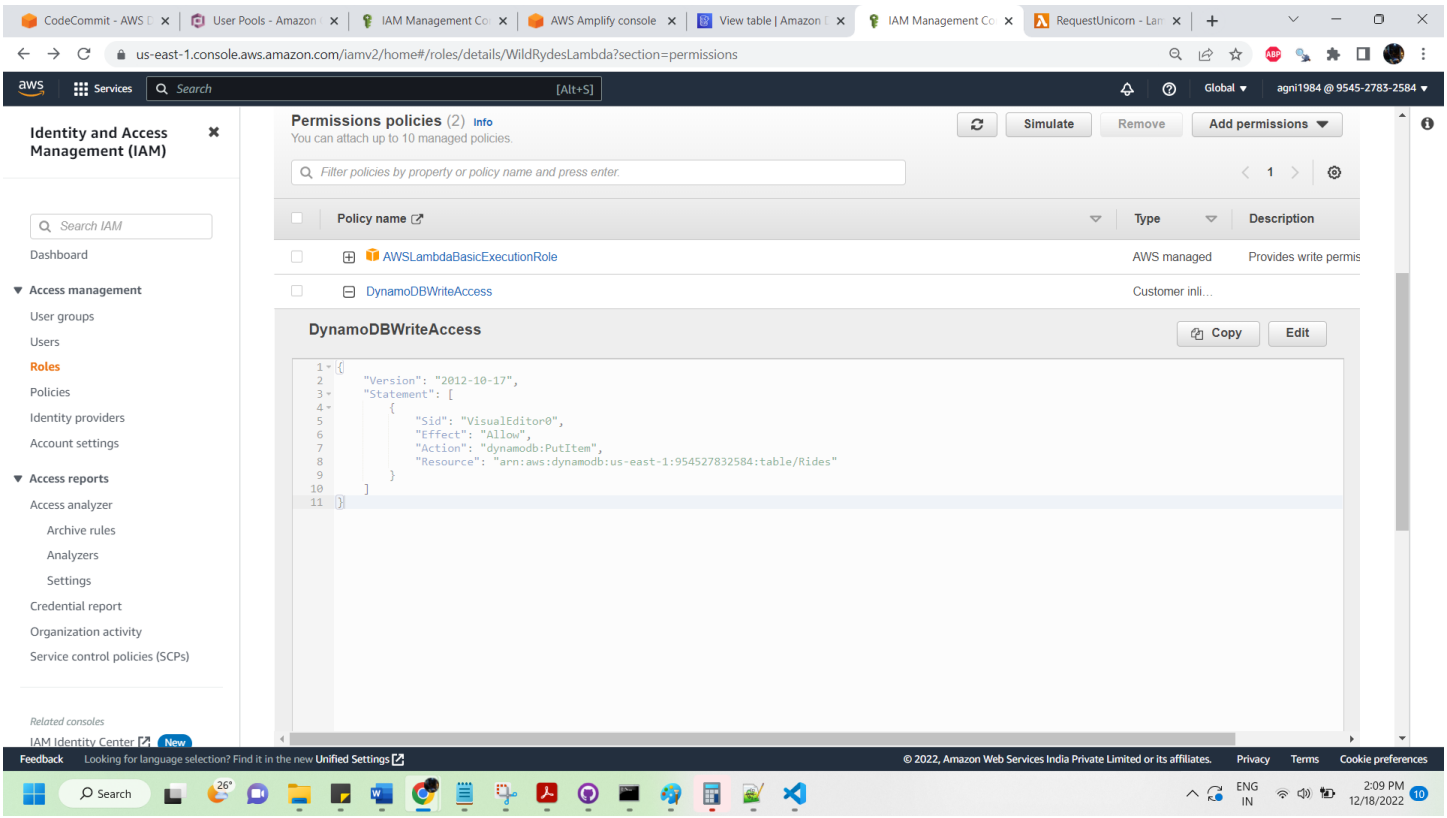
Every Lambda function has an IAM role associated with it. This role defines what other AWS services the function is allowed to interact with. Create an IAM role that grants our Lambda function permission to write logs to Amazon CloudWatch Logs and access to write items to our DynamoDB table. Follow below steps to create IAM Role,

- From the AWS Management Console, click on Services and then select IAM in the Security, Identity & Compliance section.
- Select Roles in the left navigation pane and then choose Create Role.
- Select Lambda for the role type from the AWS service group, then click Next: Permissions.
- Begin typing **AWSLambdaBasicExecutionRole** in the Filter text box and check the box next to that role.
- Choose Next Step.
- Enter **WildRydesLambda** for the Role Name. Keep other parameters as default.
- Choose Create Role.
- Type **WildRydesLambda** into the filter box on the Roles page and choose the role we just created.
- On the Permissions tab, on the left under Add permissions, choose Create Inline Policy.
- Select Choose a service.
- Begin typing DynamoDB into the search box labeled Find a service and select DynamoDB when it appears.

- Choose Select actions.
- Begin typing PutItem into the search box labeled Filter actions and check the box next to PutItem when it appears.
- Select the Resources section.
- With the Specific option selected, choose the Add ARN link in the table section.
- Paste the ARN of the table we created in the previous section in the Specify ARN for table field, and choose Add.
- Choose Review Policy.
- Enter DynamoDBWriteAccess for the policy name and choose Create policy.



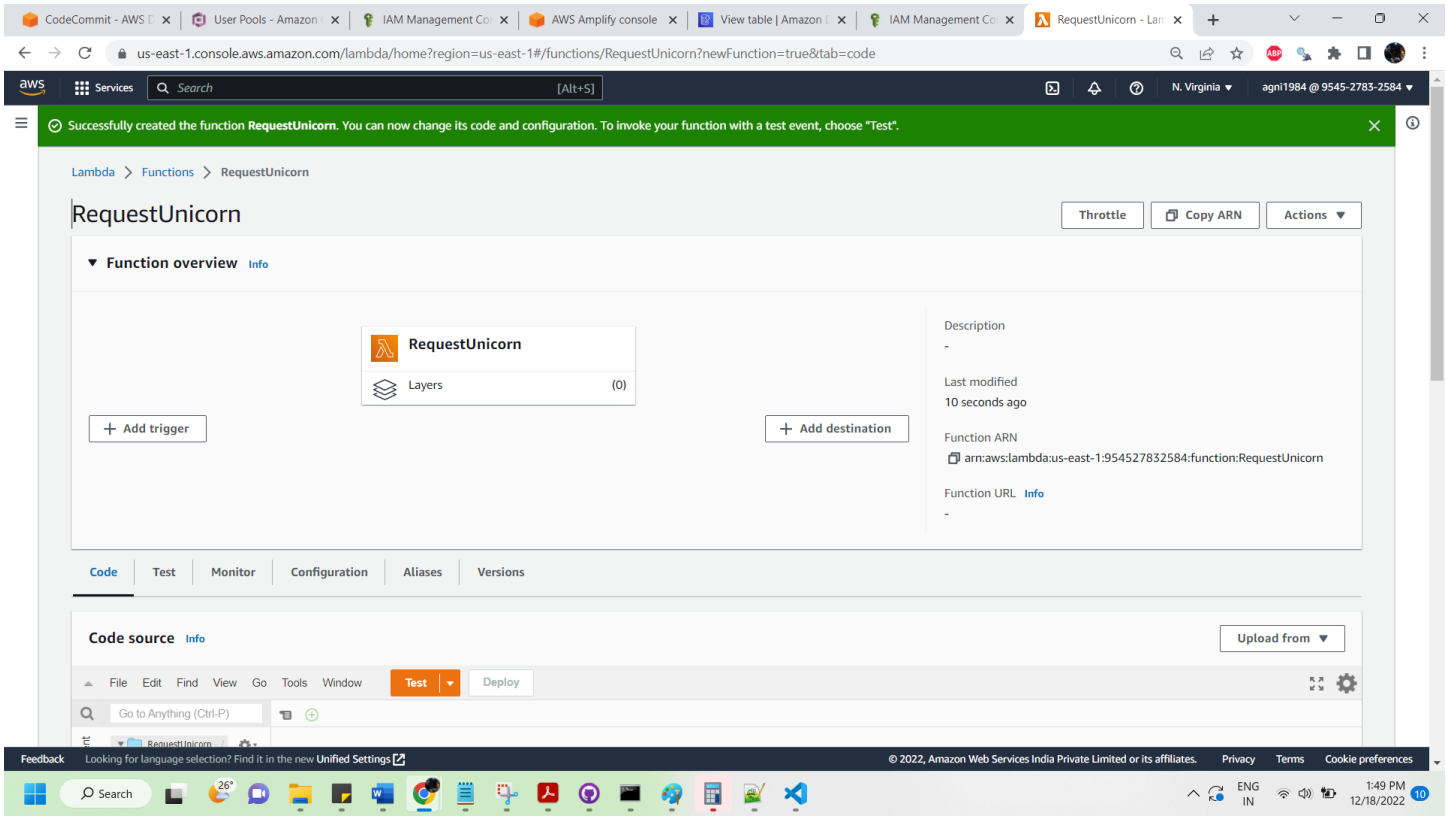

us-east-1.console.aws.amazon.com/iamv2/home#/roles/details/WildRydesLambda?section=permissions
Identity and Access Management (IAM)
Permissions policies (2) Info
You can attach up to 10 managed policies.
Simulate
Remove
Add permissions
Filter policies by property or policy name and press enter.
Policy name
Type
Description
AWSLambdaBasicExecutionRole
AWS managed
Provides write permis
AWSLambdaBasicExecutionRole
Provides write permissions to CloudWatch Logs.
Copy
{
"Version": "2012-10-17",
"Statement": [
{
"Effect": "Allow",
"Action": [
"logs:CreateLogGroup",
"logs:CreateLogStream",
"logs:PutLogEvents"
],
"Resource": "*"
}
]
}
Dashboard
Access management
User groups
Users
Roles
Policies
Identity providers
Account settings
Access reports
Access analyzer
Archive rules
Analyzers
Settings
Credential report
Organization activity
Service control policies (SCPs)
Related consoles
IAM Identity Center
New
Feedback
Looking for language selection? Find it in the new Unified Settings
© 2022, Amazon Web Services India Private Limited or its affiliates.
Privacy
Terms
Cookie preferences


us-east-1.console.aws.amazon.com/iamv2/home#/roles/details/WildRydesLambda?section=permissions
Identity and Access Management (IAM)
Permissions policies (2) Info
You can attach up to 10 managed policies.
Simulate
Remove
Add permissions
Filter policies by property or policy name and press enter.
Policy name
Type
Description
AWSLambdaBasicExecutionRole
AWS managed
Provides write permis
DynamoDBWriteAccess
Customer inli...
DynamoDBWriteAccess
Copy
Edit
{
"Version": "2012-10-17",
"Statement": [
{
"Sid": "VisualEditor0",
"Effect": "Allow",
"Action": "dynamodb:PutItem",
"Resource": "arn:aws:dynamodb:us-east-1:954527832584:table/Rides"
}
]
}
Dashboard
Access management
User groups
Users
Roles
Policies
Identity providers
Account settings
Access reports
Access analyzer
Archive rules
Analyzers
Settings
Credential report
Organization activity
Service control policies (SCPs)
Related consoles
IAM Identity Center
New
Feedback
Looking for language selection? Find it in the new Unified Settings
© 2022, Amazon Web Services India Private Limited or its affiliates.
Privacy
Terms
Cookie preferences

Step 3: Create a Lambda Function for Handling Requests

AWS Lambda will run our code in response to events such as an HTTP request. Build the AWS Lambda function that will process API requests from the web application to dispatch a unicorn. Follow the below steps, to create Lambda function,

- Choose Services then select Lambda in the Compute section.
- Click Create function.
- Keep the default Author from scratch card selected.
- Enter RequestUnicorn in the Name field.
- Select Node.js 16.x for the Runtime.
- Ensure Choose an existing role is selected from the Role dropdown.
- Select WildRydesLambda from the Existing Role dropdown.
- Click on Create function.



- Scroll down to the Function code section and replace the existing code in the index.js code editor with the contents of requestUnicorn.js.

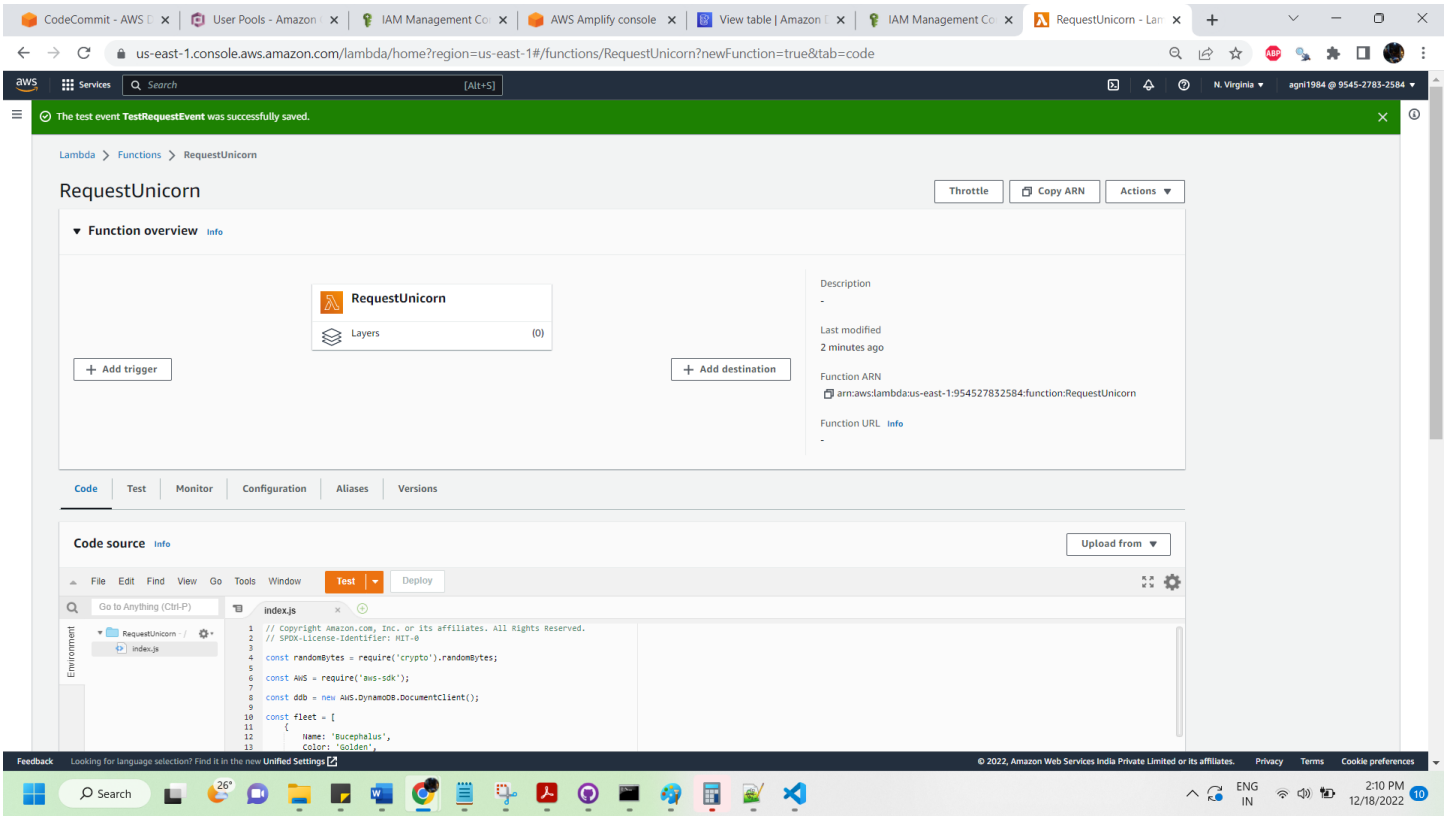
```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: MIT-0

const randomBytes = require('crypto').randomBytes;
```

```
const AWS = require('aws-sdk');

const ddb = new AWS.DynamoDB.DocumentClient();

const fleet = [
    {
        Name: 'Bucephalus',
        Color: 'Golden',
        Gender: 'Male',
    },
    {
        Name: 'Shadowfax',
        Color: 'White',
        Gender: 'Male',
    },
    {
        Name: 'Rocinante',
        Color: 'Yellow',
        Gender: 'Female',
    },
];

exports.handler = (event, context, callback) => {
    if (!event.requestContext.authorizer) {
      errorResponse('Authorization not configured', context.awsRequestId, callback);
      return;
    }

    const rideId = toUrlString(randomBytes(16));
    console.log('Received event (', rideId, '): ', event);

    // Because we're using a Cognito User Pools authorizer, all of the claims
    // included in the authentication token are provided in the request context.
    // This includes the username as well as other attributes.
    const username = event.requestContext.authorizer.claims['cognito:username'];

    // The body field of the event in a proxy integration is a raw string.
    // In order to extract meaningful values, we need to first parse this string
    // into an object. A more robust implementation might inspect the Content-Type
    // header first and use a different parsing strategy based on that value.
    const requestBody = JSON.parse(event.body);

    const pickupLocation = requestBody.PickupLocation;

    const unicorn = findUnicorn(pickupLocation);

    recordRide(rideId, username, unicorn).then(() => {
        // You can use the callback function to provide a return value from your Node.js
        // Lambda functions. The first parameter is used for failed invocations. The
        // second parameter specifies the result data of the invocation.

        // Because this Lambda function is called by an API Gateway proxy integration
        // the result object must use the following structure.
        callback(null, {
            statusCode: 201,
            body: JSON.stringify({
                RideId: rideId,
                Unicorn: unicorn,
```

```
                UnicornName: unicorn.Name,
                Eta: '30 seconds',
                Rider: username,
            }),
            headers: {
                'Access-Control-Allow-Origin': '*',
            },
        });
    }).catch((err) => {
        console.error(err);

        // If there is an error during processing, catch it and return
        // from the Lambda function successfully. Specify a 500 HTTP status
        // code and provide an error message in the body. This will provide a
        // more meaningful error response to the end client.
        errorResponse(err.message, context.awsRequestId, callback)
    });
};

// This is where you would implement logic to find the optimal unicorn for
// this ride (possibly invoking another Lambda function as a microservice.)
// For simplicity, we'll just pick a unicorn at random.
function findUnicorn(pickupLocation) {
    console.log('Finding unicorn for ', pickupLocation.Latitude, ', ',
pickupLocation.Longitude);
    return fleet[Math.floor(Math.random() * fleet.length)];
}

function recordRide(rideId, username, unicorn) {
    return ddb.put({
        TableName: 'Rides',
        Item: {
            RideId: rideId,
            User: username,
            Unicorn: unicorn,
            UnicornName: unicorn.Name,
            RequestTime: new Date().toISOString(),
        },
    }).promise();
}

function toUrlString(buffer) {
    return buffer.toString('base64')
        .replace(/\+/g, '-')
        .replace(/\//g, '_')
        .replace(/=/g, '');
}

function errorResponse(errorMessage, awsRequestId, callback) {
  callback(null, {
    statusCode: 500,
    body: JSON.stringify({
      Error: errorMessage,
      Reference: awsRequestId,
    }),
    headers: {
      'Access-Control-Allow-Origin': '*',
    },
  });
```

```
}
```

- Choose Deploy.



Step 4: Validate the Implementation

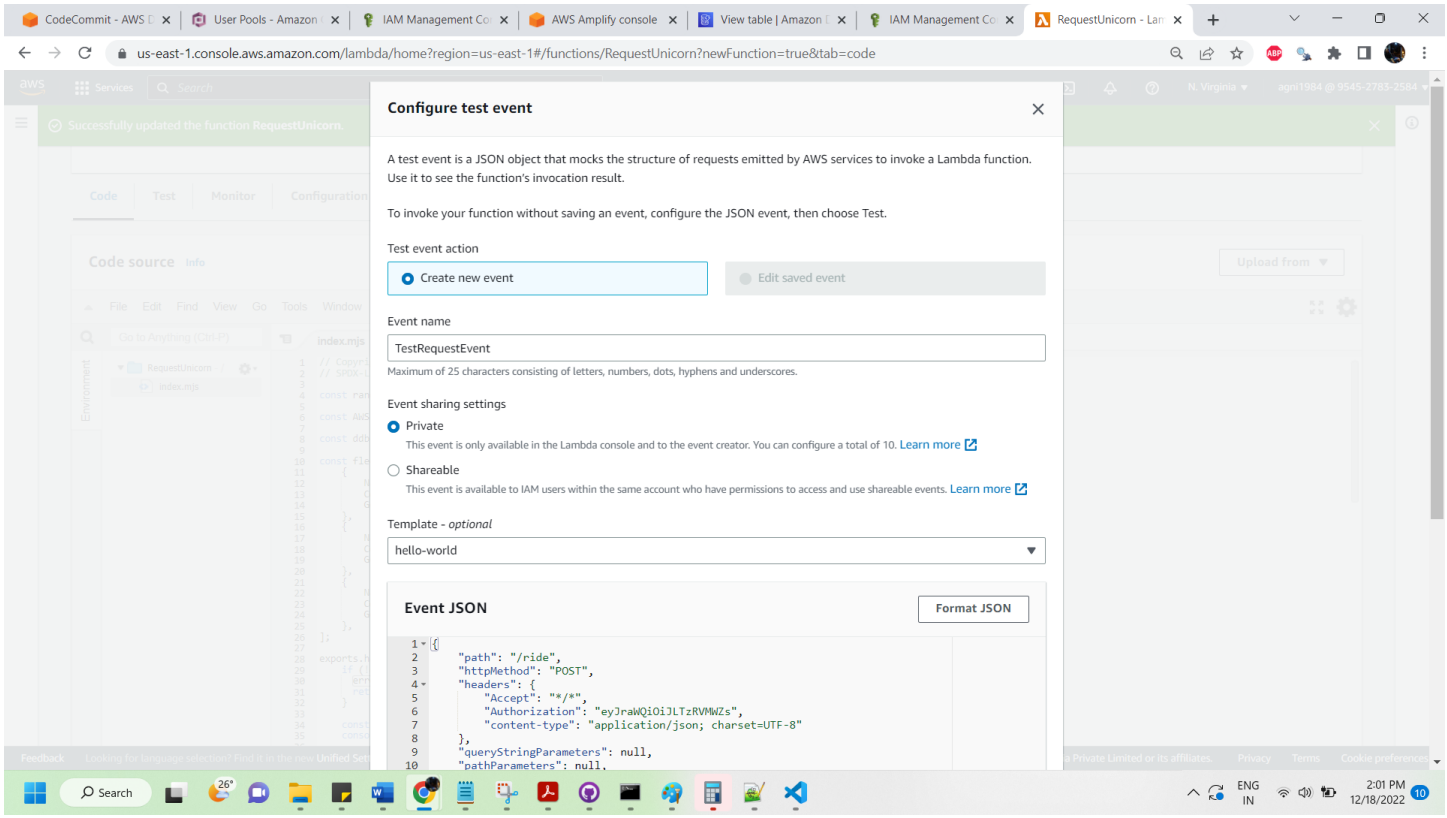
Test the created AWS lambda function, by following below steps,

- From the main edit screen for our function, select Test and choose Configure test event from the dropdown.
- Keep Create new event selected.
- Enter TestRequestEvent in the Event name field
- Copy and paste the following test event into the editor:

```
{
    "path": "/ride",
    "httpMethod": "POST",
    "headers": {
        "Accept": "*/*",
        "Authorization": "eyJraWQiOiJLTzRVMWZs",
        "content-type": "application/json; charset=UTF-8"
    },
    "queryStringParameters": null,
    "pathParameters": null,
    "requestContext": {
        "authorizer": {
            "claims": {
                "cognito:username": "the_username"
```

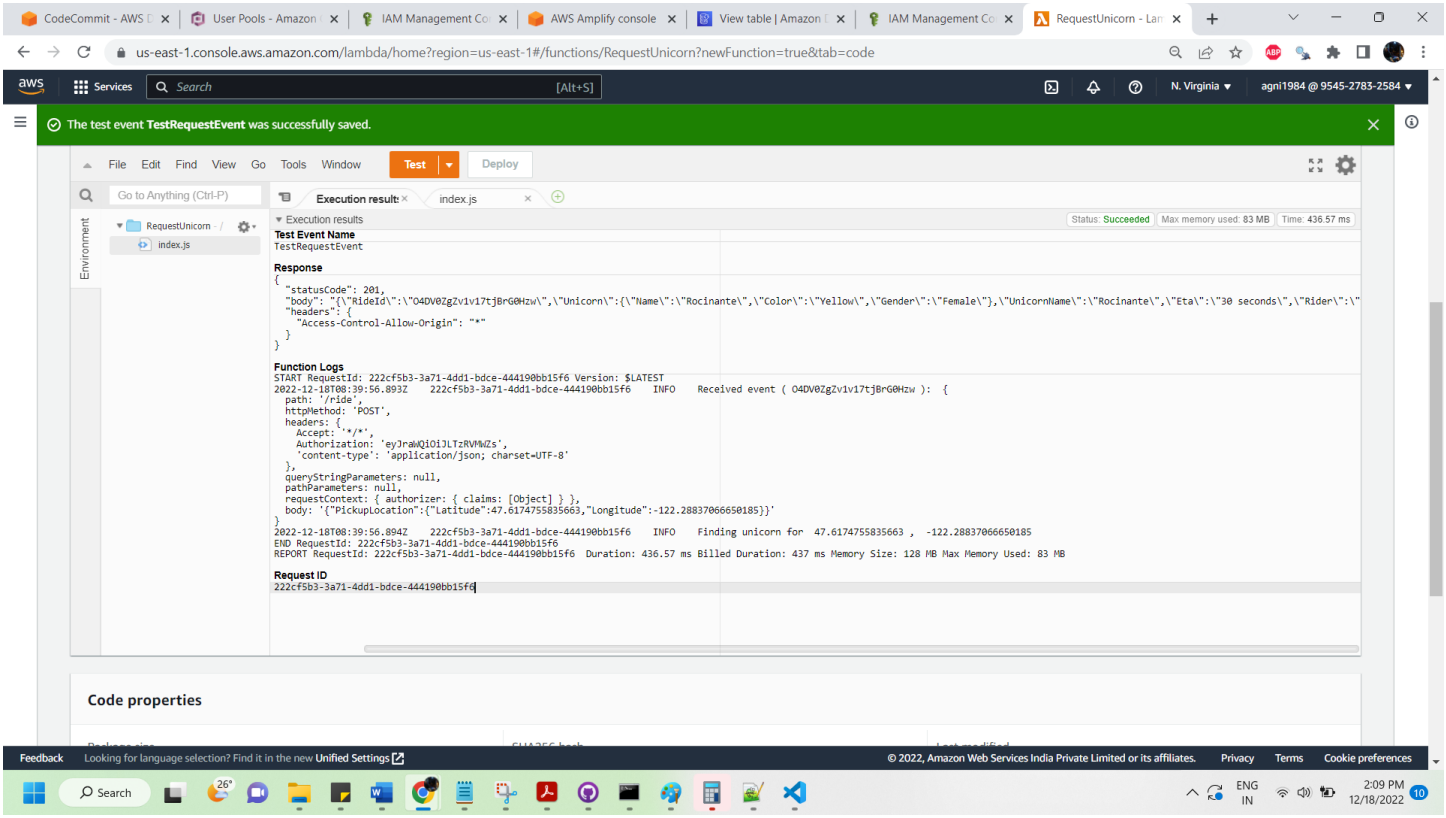
```
            }
        }
    },
    "body": "{\"PickupLocation\":{\"Latitude\":47.6174755835663,\"Longitude\":-
122.28837066650185}}"
}
```

- Choose Save.



- On the main function edit screen click Test with TestRequestEvent selected in the dropdown.
- Scroll to the top of the page and expand the Details section of the Execution result section.
- Verify that the execution succeeded and that the function result looks like the following:

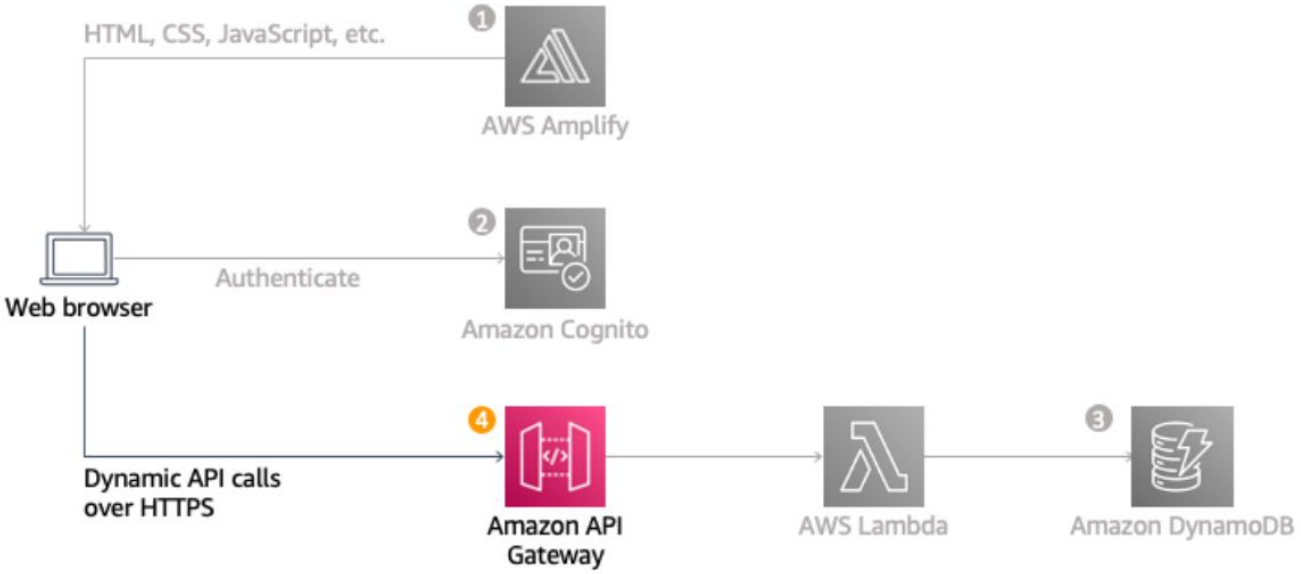
```
{
    "statusCode": 201,
    "body":
"{\"RideId\":\"SvLnijIAtg6inAFUBRT+Fg==\",\"Unicorn\":{\"Name\":\"Rocinante\",\"Color\":\
"Yellow\",\"Gender\":\"Female\"},\"Eta\":\"30 seconds\"}",
    "headers": {
        "Access-Control-Allow-Origin": "*"
    }
}
```



Module 4: RESTful API

In this module, Amazon API Gateway is used to expose the Lambda function that, we built in the previous module as a RESTful API. This API will be accessible on the public Internet. It will be secured using the Amazon Cognito user pool we created in the previous module. Using this configuration, we will then turn our statically hosted website into a dynamic web application by adding client-side JavaScript that makes AJAX calls to the exposed APIs.

Architectural Overview



The static website we deployed in the first module already has a page configured to interact with the API we will build in this module. The page at /ride.html has a simple map-based interface for requesting a unicorn ride. After authenticating using the /signin.html page, users will be able to select their pickup location by clicking a point on the map and then requesting a ride by choosing the "Request Unicorn" button in the upper right corner.

This module will focus on the steps required to build the cloud components of the API.

Step 1: Create a New REST API

- In the AWS Management Console, click Services then select API Gateway under Application Services.
- Choose Create API.
- Select Build under REST API and enter WildRydes for the API Name.

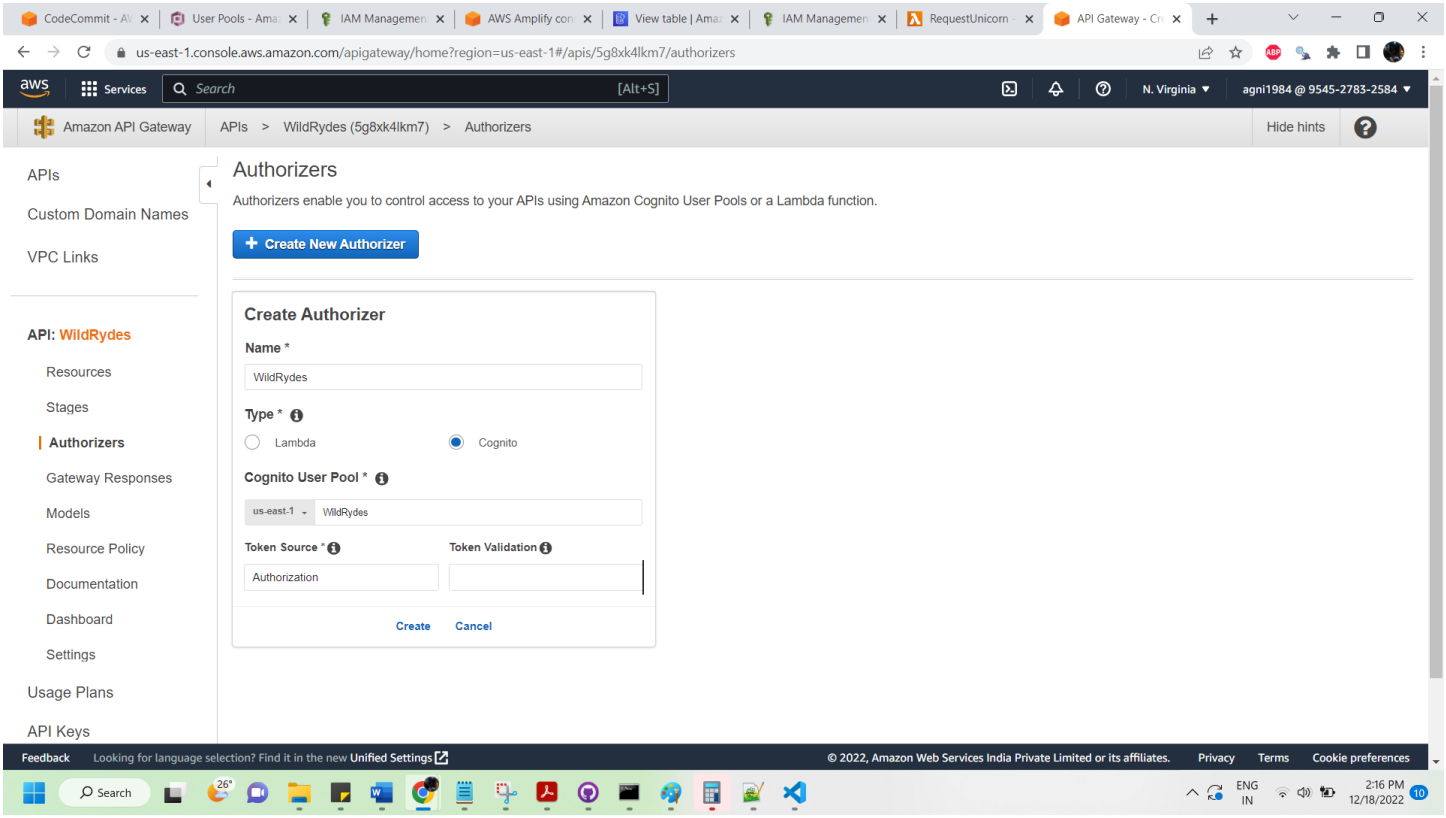
- Choose Edge optimized in the Endpoint Type dropdown. Note: Edge optimized are best for public services being accessed from the Internet. Regional endpoints are typically used for APIs that are accessed primarily from within the same AWS Region.
- Choose Create API

Step 2: Create a Cognito User Pool Authorizer

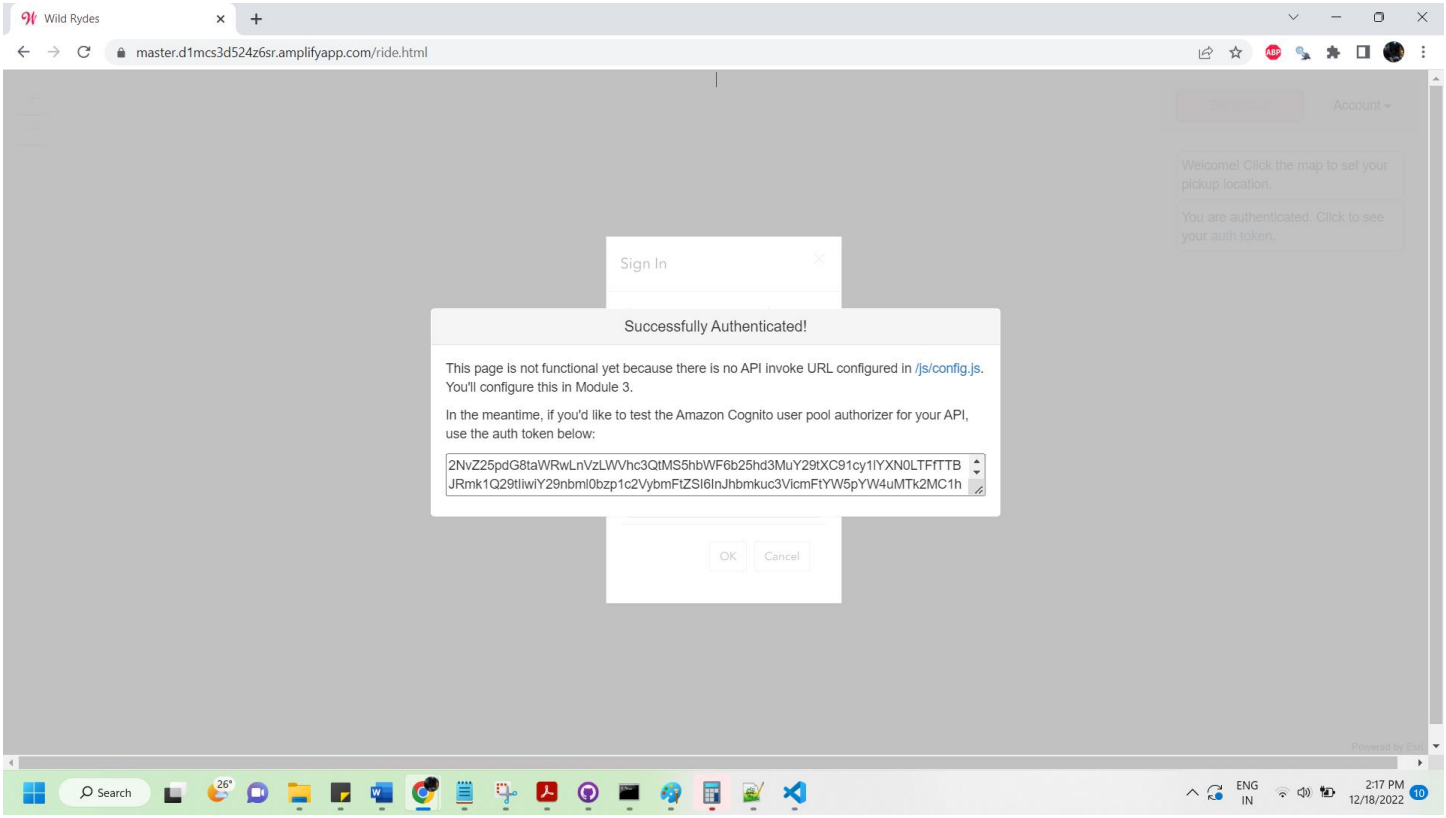
Amazon API Gateway can use the JWT tokens returned by Cognito User Pools to authenticate API calls. In this step we'll configure an authorizer for our API to use the user pool we created in Module 2.

In the Amazon API Gateway console, create a new Cognito user pool authorizer for our API. Configure it with the details of the user pool that we created in the previous module. We can test the configuration in the console by copying and pasting the auth token presented to us, after we log in via the /signin.html page of our current website.

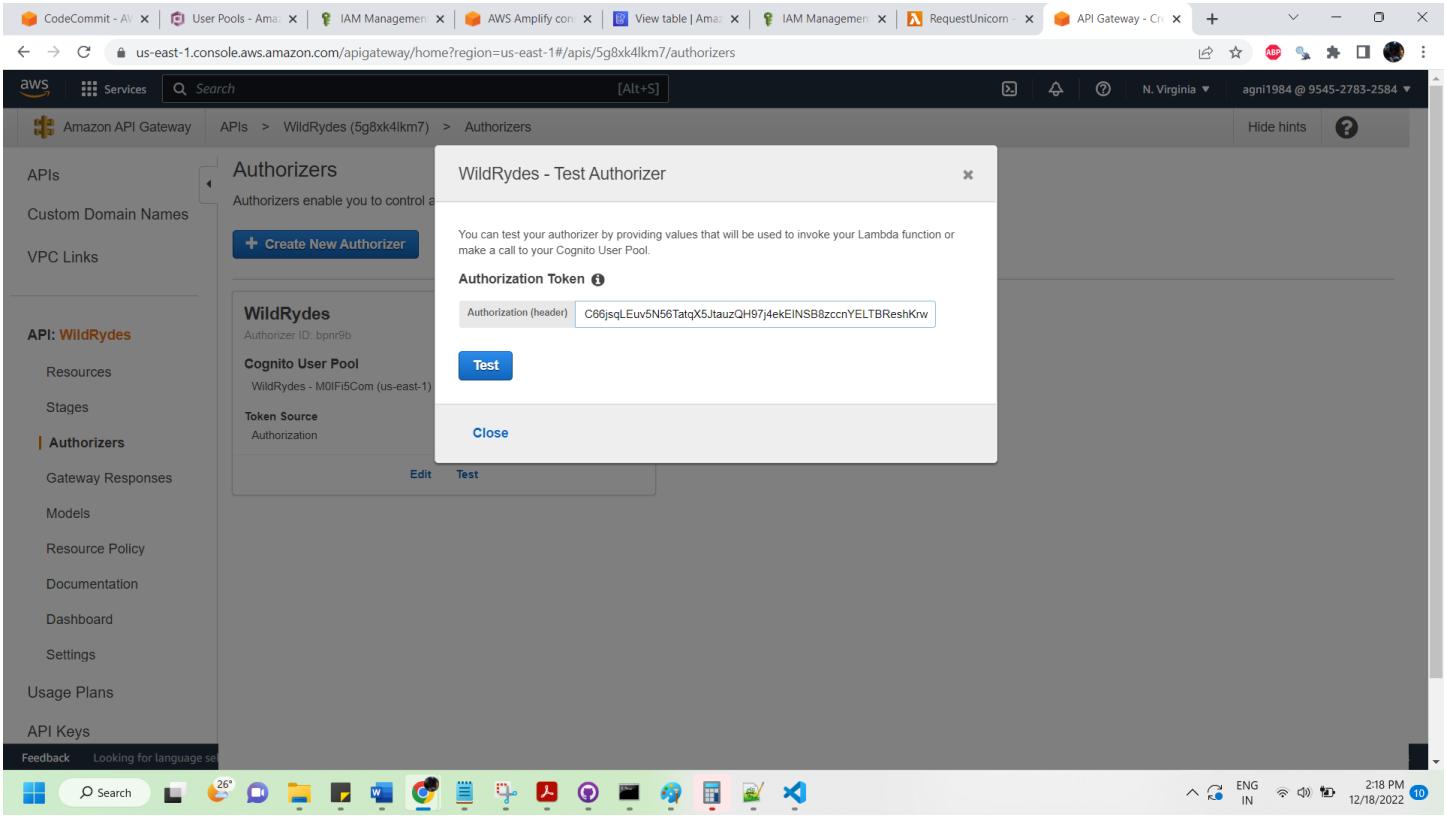
- Under our newly created API, choose Authorizers.
- Choose Create New Authorizer.
- Enter WildRydes for the Authorizer name.
- Select Cognito for the type.
- Enter WildRydes (or the name we gave our user pool) in the Cognito User Pool input.
- Enter Authorization for the Token Source.
- Choose Create.

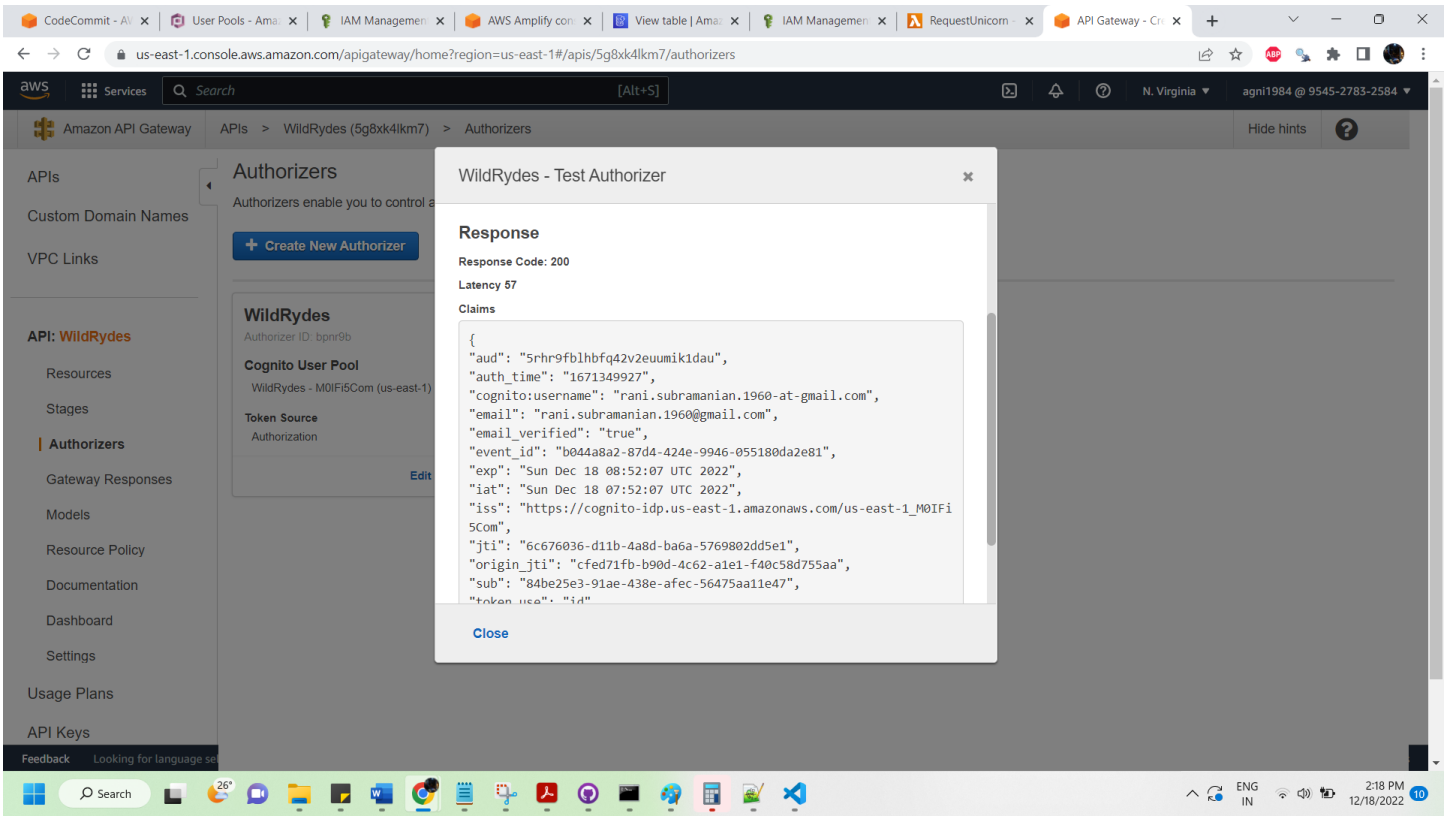


- Verify our authorizer configuration
- Open a new browser tab and visit /ride.html under our website's domain.
- If we are redirected to the sign-in page, sign in with the user we created in the last module. We will be redirected back to /ride.html.
- Copy the auth token from the notification on the /ride.html,



- Go back to previous tab where we have just finished creating the Authorizer
- Click Test at the bottom of the card for the authorizer.
- Paste the auth token into the Authorization Token field in the popup dialog.
- Click Test button and verify that the response code is 200 and that we see the claims for our user displayed.

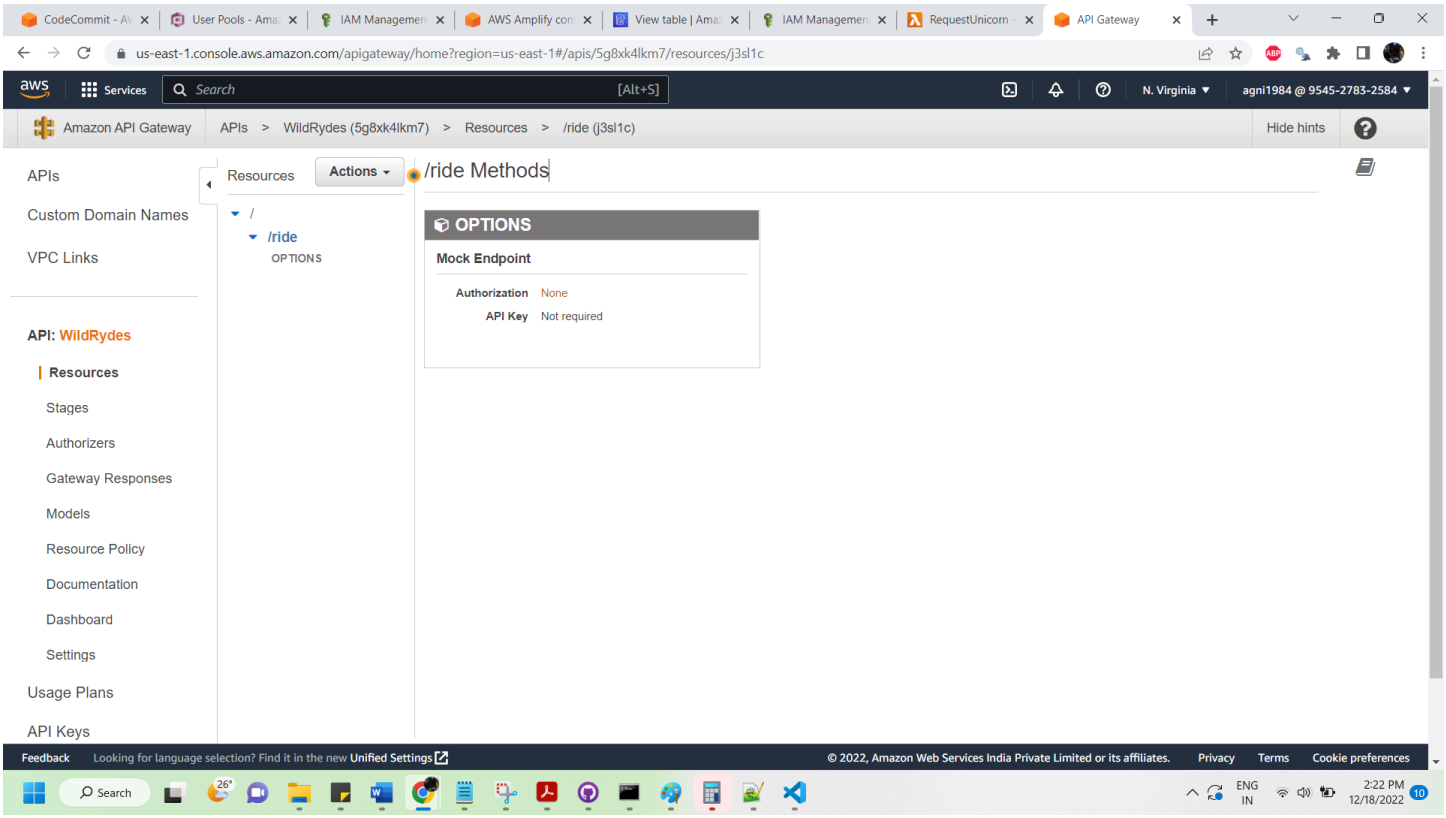

us-east-1.console.aws.amazon.com/apigateway/home?region=us-east-1#/apis/5g8xk4lkm7/authorizers
Amazon API Gateway
APIs > WildRydes (5g8xk4lkm7) > Authorizers
Authorizers
Create New Authorizer
WildRydes
Cognito User Pool
Token Source
Authorization
Edit
Test
WildRydes - Test Authorizer
You can test your authorizer by providing values that will be used to invoke your Lambda function or make a call to your Cognito User Pool.
Authorization Token
Authorization (header)
C66jsqLEuv5N56TatqX5JtauzQH97j4ekEINSB8zccnYELTBReshKrw
Test
Close


us-east-1.console.aws.amazon.com/apigateway/home?region=us-east-1#/apis/5g8xk4lkm7/authorizers
Amazon API Gateway
APIs > WildRydes (5g8xk4lkm7) > Authorizers
Authorizers
Create New Authorizer
WildRydes
Cognito User Pool
Token Source
Authorization
WildRydes - Test Authorizer
Response
Response Code: 200
Latency 57
Claims
{
"aud": "5rhr9fblhbfq42v2euumik1dau",
"auth_time": "1671349927",
"cognito:username": "rani.subramanian.1960-at-gmail.com",
"email": "rani.subramanian.1960@gmail.com",
"email_verified": "true",
"event_id": "b044a8a2-87d4-424e-9946-055180da2e81",
"exp": "Sun Dec 18 08:52:07 UTC 2022",
"iat": "Sun Dec 18 07:52:07 UTC 2022",
"iss": "https://cognito-idp.us-east-1.amazonaws.com/us-east-1_M0IFi5Com",
"jti": "6c676036-d11b-4a8d-ba6a-5769802dd5e1",
"origin_jti": "cfed71fb-b90d-4c62-a1e1-f40c58d755aa",
"sub": "84be25e3-91ae-438e-afec-56475aa11e47",
Close

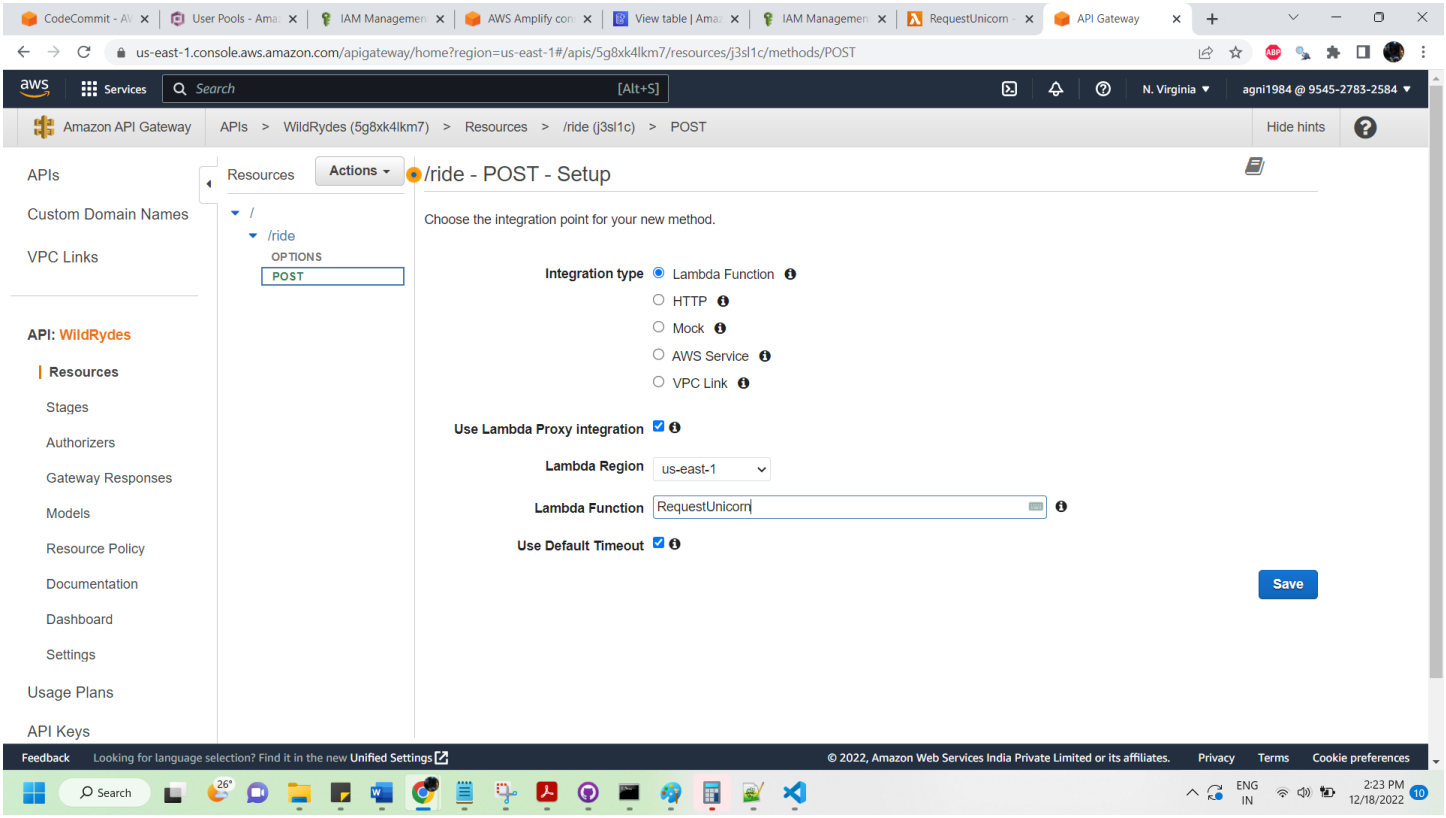
Step 3: Create a new resource and method

Create a new resource called /ride within our API. Then create a POST method for that resource and configure it to use a Lambda proxy integration backed by the RequestUnicorn function we created in the first step of this module.

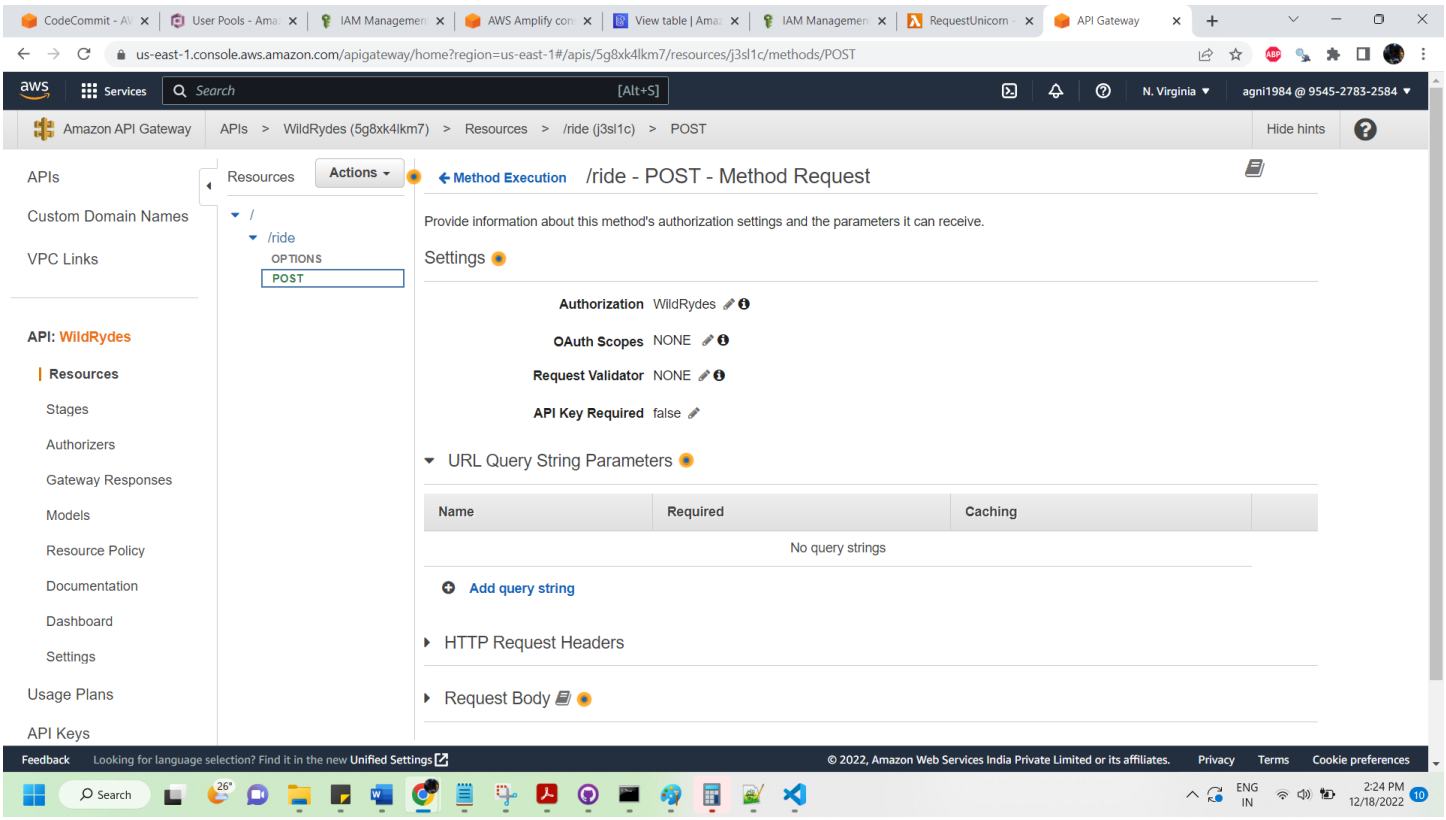
- In the left nav, click on Resources under our WildRydes API.
- From the Actions dropdown select Create Resource.
- Enter ride as the Resource Name.
- Ensure the Resource Path is set to ride.
- Select Enable API Gateway CORS for the resource.
- Click Create Resource.



- With the newly created /ride resource selected, from the Action dropdown select Create Method.
- Select POST from the new dropdown that appears, then click the checkmark.
- Select Lambda Function for the integration type.
- Check the box for Use Lambda Proxy integration.
- Select the Region we are using for Lambda Region.
- Enter the name of the function we created in the previous module, RequestUnicorn, for Lambda Function.
- Choose Save. Please note, if we get an error that our function does not exist, check that the region we selected matches the one we used in the previous module.



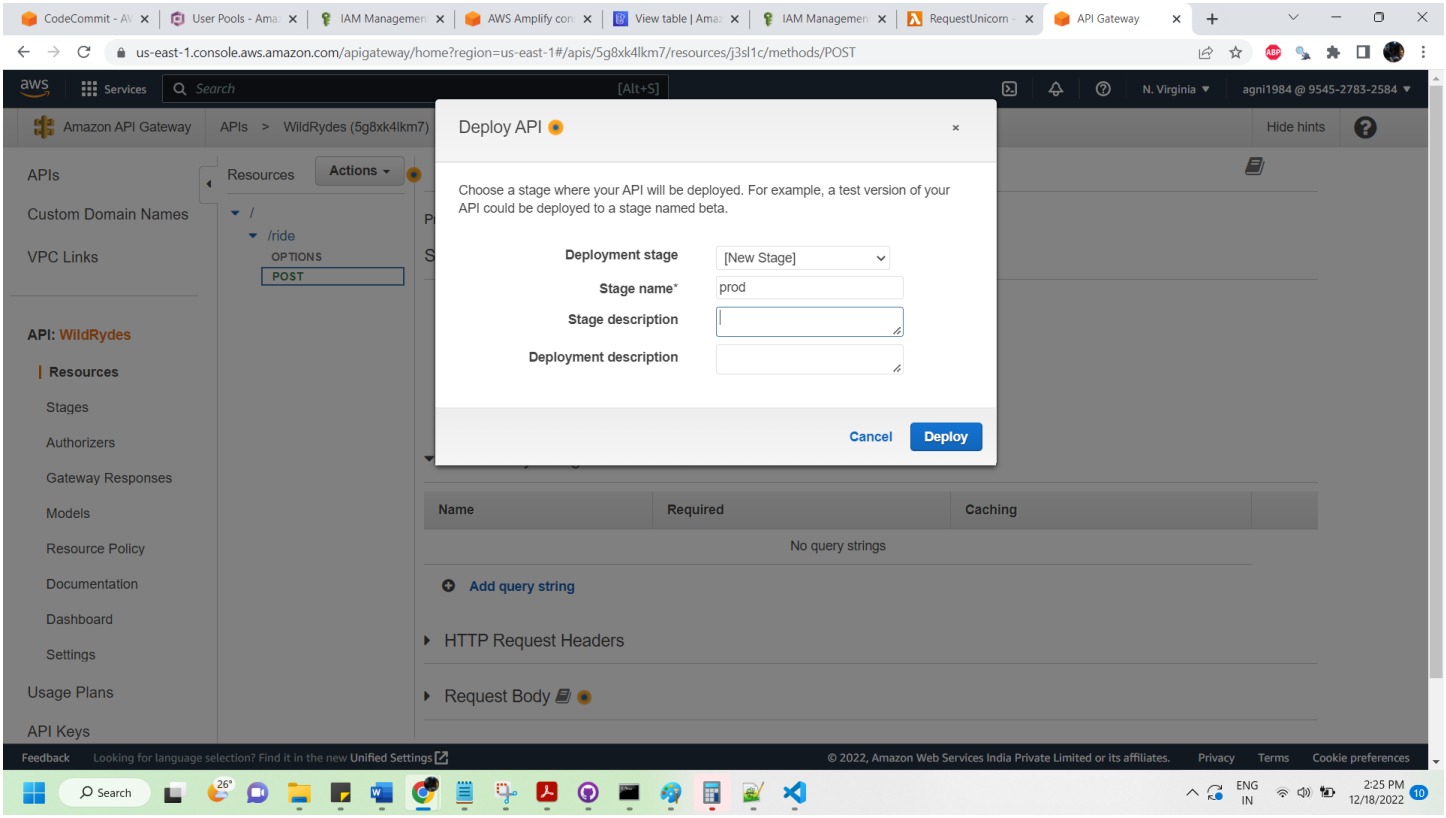
- When prompted to give Amazon API Gateway permission to invoke our function, choose OK.
- Choose on the Method Request card.
- Choose the pencil icon next to Authorization.
- Select the WildRydes Cognito user pool authorizer from the drop-down list, and click the checkmark icon.

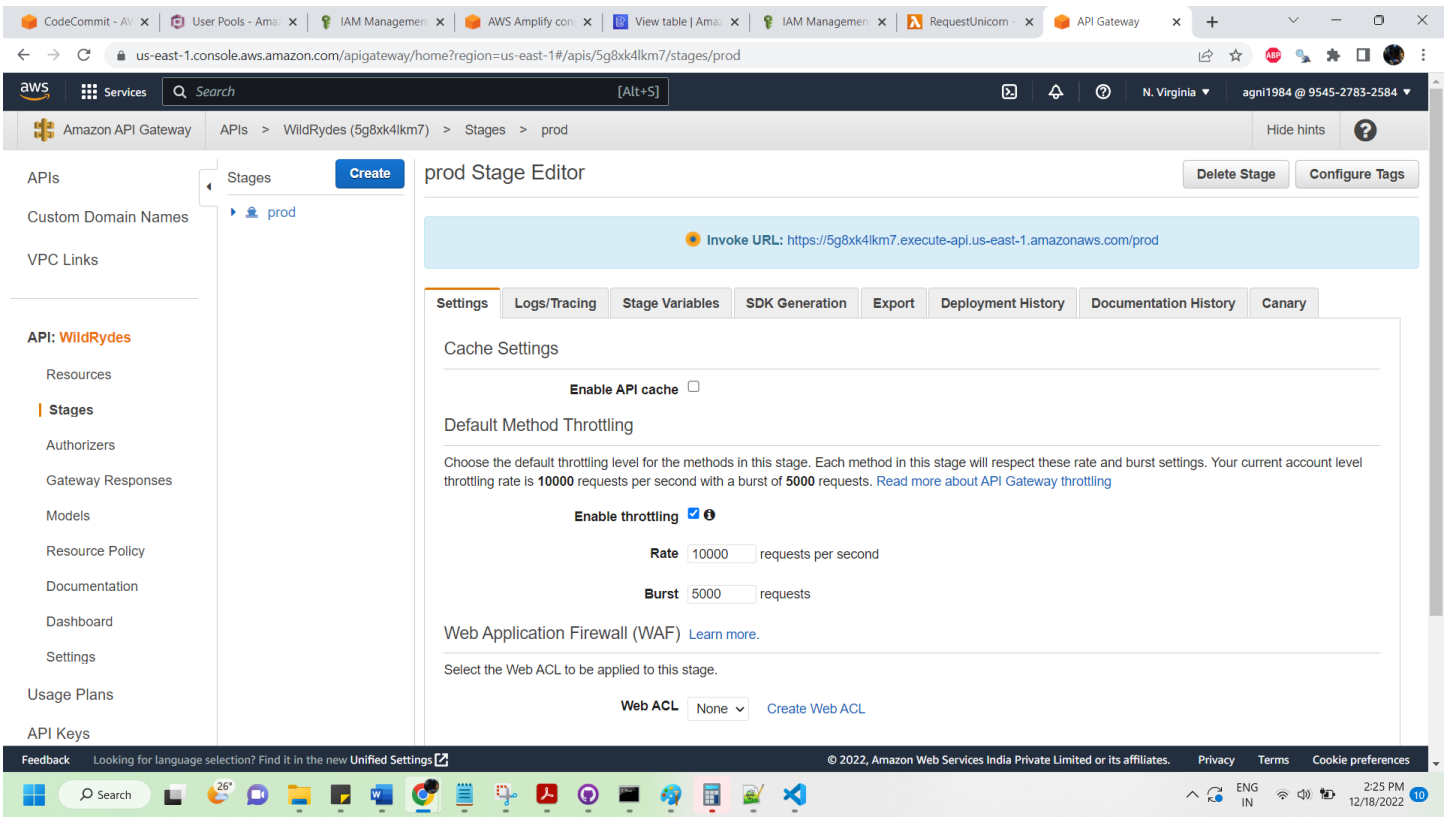


Step 4: Deploy our API

From the Amazon API Gateway console, choose Actions, Deploy API. We'll be prompted to create a new stage. Use prod for the stage name.

- In the Actions drop-down list select Deploy API.
- Select [New Stage] in the Deployment stage drop-down list.
- Enter prod for the Stage Name.
- Choose Deploy.
- Note the Invoke URL. We will use it in the next section.



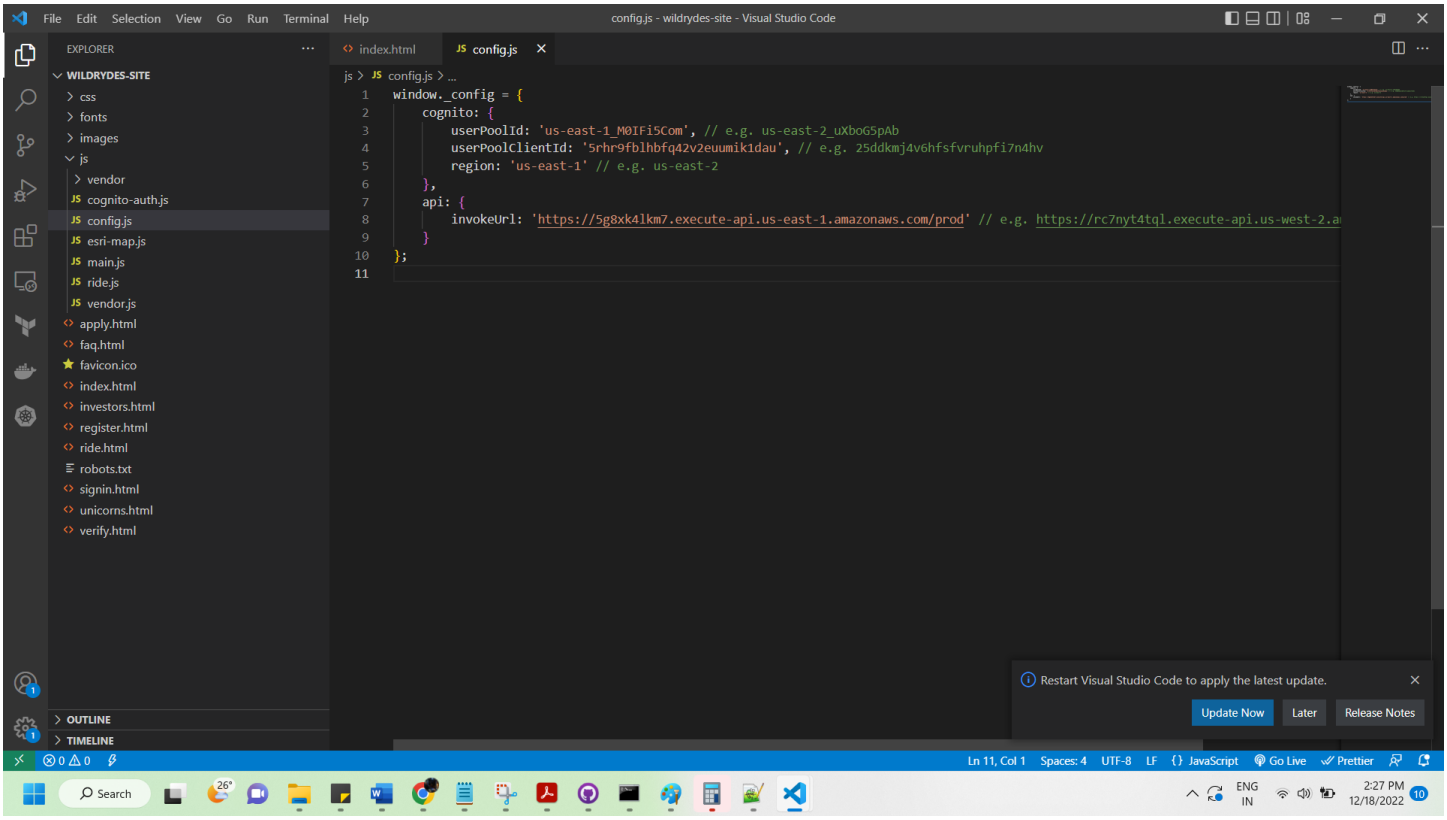


Invoke URL: https://5g8xk4lkm7.execute-api.us-east-1.amazonaws.com/prod

Step 5: Update the Website Config

Update the /js/config.js file in our website deployment to include the invoke URL of the stage we just created. We should copy the invoke URL directly from the top of the stage editor page on the Amazon API Gateway console and paste it into the _config.api.invokeUrl key of our sites /js/config.js file.

- Open the config.js file in a text editor.
- Update the invokeUrl setting under the api key in the config.js file. Set the value to the Invoke URL for the deployment stage, which we created in the previous section.



- Save the modified file and push it to Git repository to have it automatically deploy to Amplify Console.

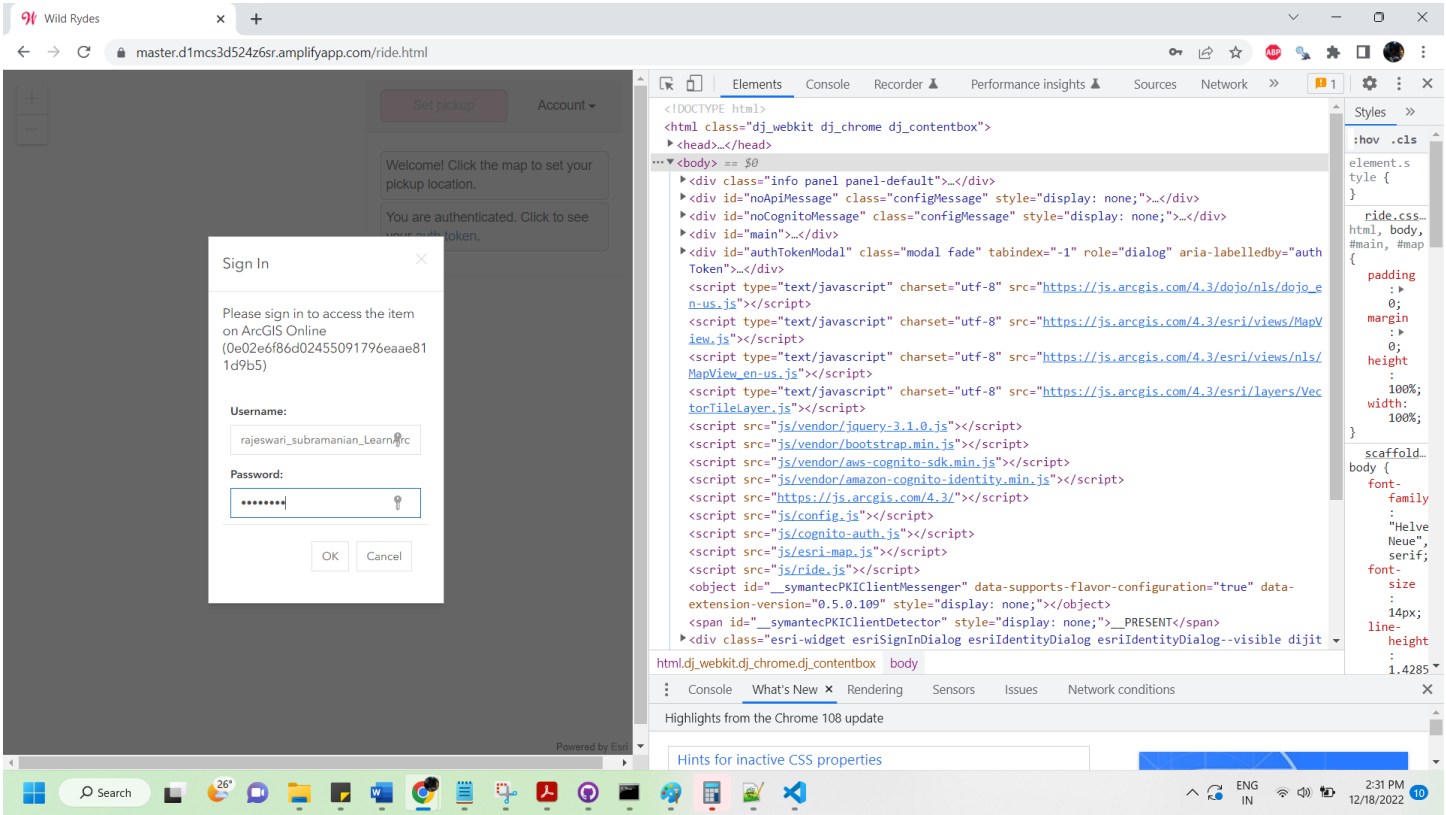
```
$ git add .
$ git commit -m "new_configuration"
$ git push
```

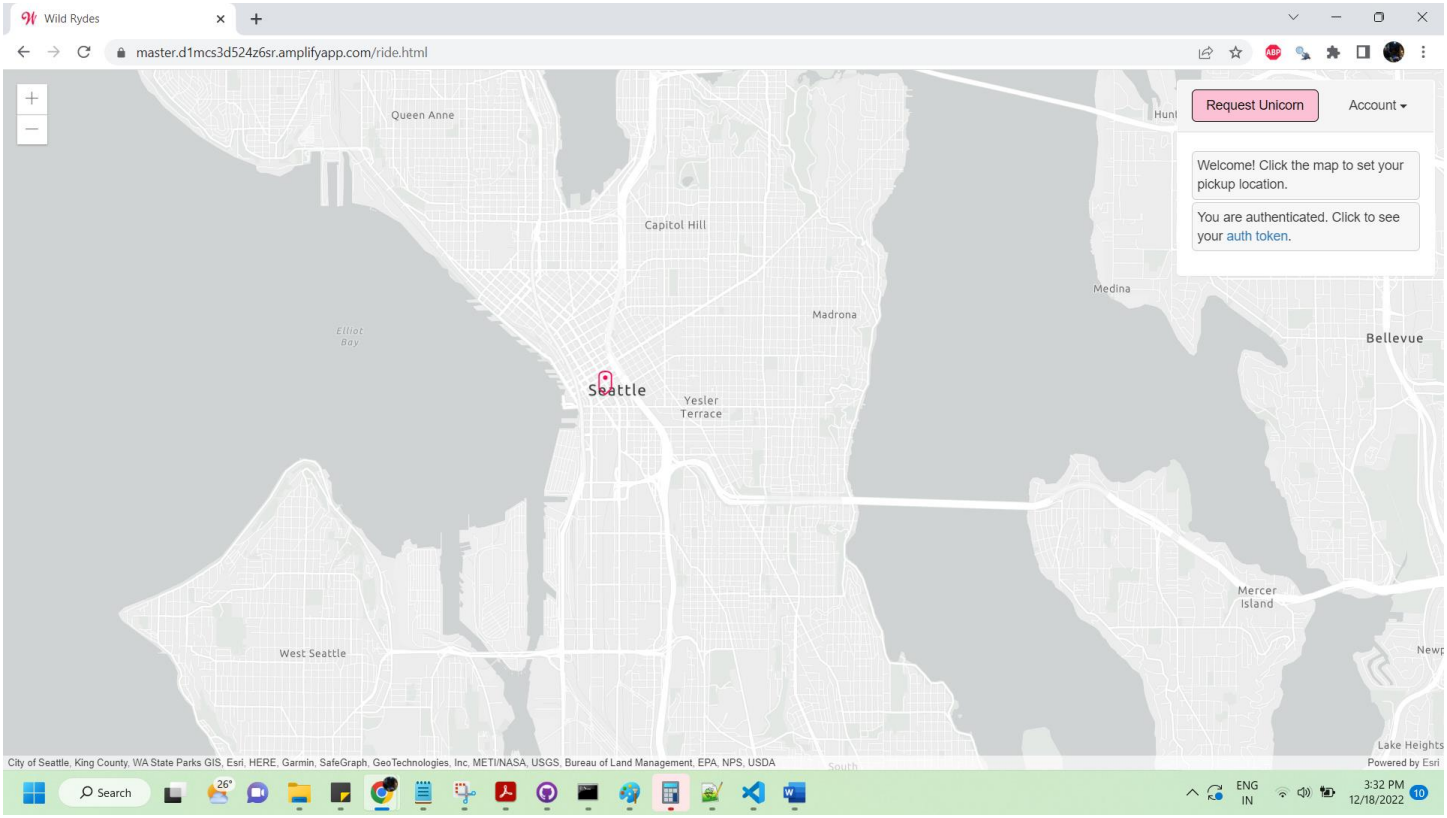
```
E:\GreatLakes_Cloud\CapstoneProject\aws-capstone3\wildrydes-site>git add .
warning: in the working copy of 'js/config.js', LF will be replaced by CRLF the next time Git touches it

E:\GreatLakes_Cloud\CapstoneProject\aws-capstone3\wildrydes-site>git commit -m "new_configuration"
[master 416cbbb] new_configuration
 1 file changed, 1 insertion(+), 1 deletion(-)

E:\GreatLakes_Cloud\CapstoneProject\aws-capstone3\wildrydes-site>git push
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 410 bytes | 410.00 KiB/s, done.
Total 4 (delta 3), reused 0 (delta 0), pack-reused 0
remote: Validating objects: 100%
To https://git-codecommit.us-east-1.amazonaws.com/v1/repos/wildrydes-site
   18fb109..416cbbb  master -> master

E:\GreatLakes_Cloud\CapstoneProject\aws-capstone3\wildrydes-site>
```

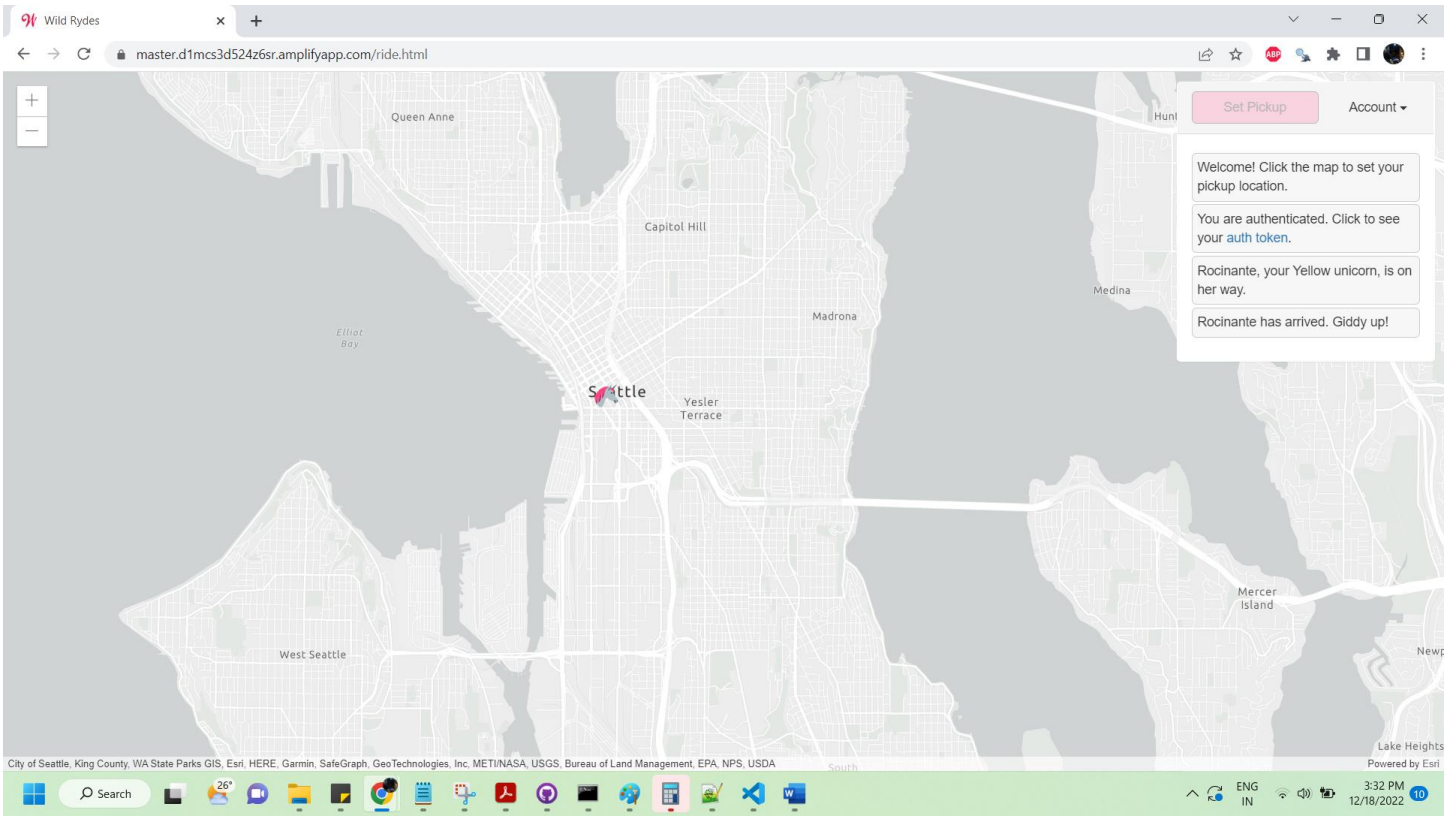
Step 6: Validate the implementation

Follow below steps, to validate that, our implementation is working fine.

- Visit /ride.html under website domain.
- If we are redirected to the ArcGIS sign-in page, sign in with the user credentials of ArcGIS.
- After the map has loaded, click anywhere on the map to set a pickup location.
- Choose Request Unicorn. We should see a notification in the right sidebar that a unicorn is on its way and then see a unicorn icon fly to our pickup location.







Note: To view the map, change arcgis version from 4.3 to 4.6 in ride.html and push the code again. Open the app after redeployment.

Cost Analysis

Problem: Cost Analysis of the implemented Solution. Assume your solution is used by 1000 users for a month, and give monthly billing estimates.

Solution: based on AWS pricing calculator, average cost per month for this solution is **$256.**423 including upfront cost ($180) for first month, and **$76.423** from next month onwards

Estimate summary

Upfront cost	Monthly cost	Total 12 months cost	Currency
180	76.423	1097.08	USD

* Includes upfront cost

Detailed Estimate

Region	Service	Upfront	Monthly	First 12 mo	Currency	Configuration summary
US East (N. Virginia)	DynamoDB provisioned capacity	180	26.39	496.68	USD	Table class (Standard), Average item size (all attributes) (1 KB), Write reserved capacity term (1 year), Read reserved capacity term (1 year), Data storage size (1 GB)
US East (N. Virginia)	Amazon API Gateway	0	0	0	USD	HTTP API requests units (millions), Average size of each request (34 KB), REST API request units (exact number), Cache memory size (GB) (None), WebSocket message units (thousands), Average message size (32 KB), Requests (0 per month), Requests (1000 per month)
US East (N. Virginia)	Amazon Cognito	0	50	600	USD	Advanced security features (Enabled), Number of monthly active users (MAU) (1000)
US East (N. Virginia)	AWS Lambda	0	0	0	USD	Architecture (x86), Architecture (x86), Amount of ephemeral storage allocated (512 MB)
US East (N. Virginia)	AWS Amplify	0	0.033	0.4	USD	Number of build minutes (1 per month), Data stored per month (1 GB)

Acknowledgement

* AWS Pricing Calculator provides only an estimate of your AWS fees and doesn't include any taxes that might apply. Your actual fees depend on a variety of factors, including your actual usage of AWS services.

Refer, AWS pricing estimation, for further details,

https://calculator.aws/#/estimate?id=15b7a4ca7667d8e1b17963f494a590fb7c22399e

Lessons & Observations

- Learnt to create serverless web app application.
- Learnt to host the web application on a front-end web server and connect it to a backend database.
- Learnt to set up user authentication using AWS Cognito and will be able to collect and analyse user behaviour.