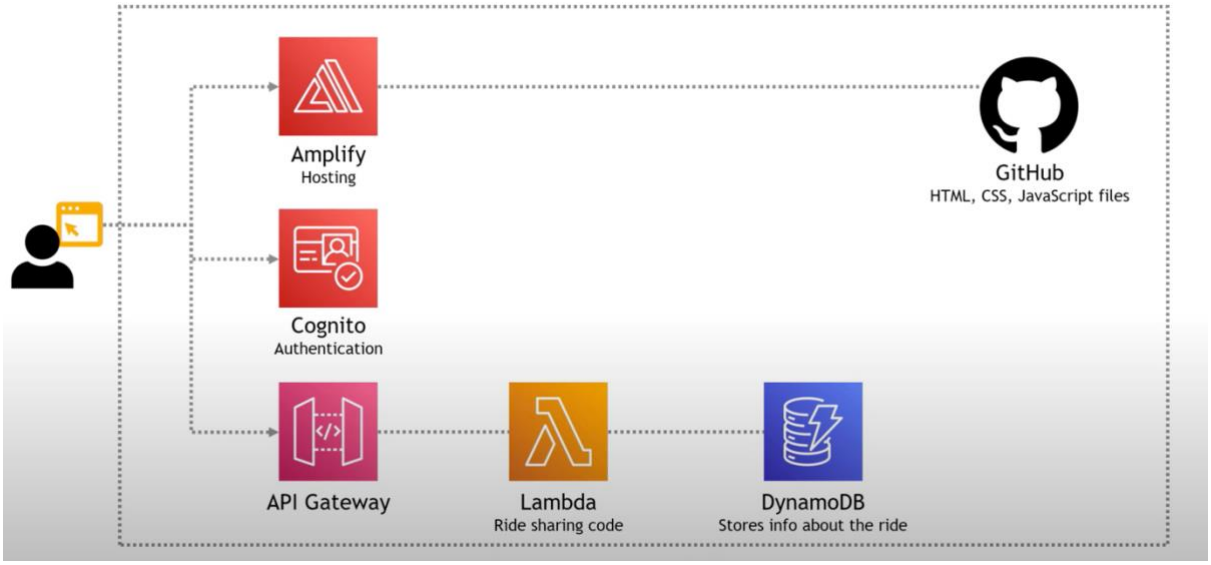
Wild Rydes Application (an AWS sample project)

In this project, we will be building **Wild Rydes**, a fun take on an Uber-like ride service, but with unicorns! The application will allow users to register, log in, and request a ride by simply clicking on a map. Along the way, you will gain hands-on experience with several key AWS services that power modern, serverless applications.

We will use the following seven AWS services to build this application: GitHub, Amplify, Cognito, Lambda, IAM, API Gateway and DynamoDB.

Throughout this project, you will not only interact with these services but also understand how they work together to provide a scalable, secure, and responsive application. By the end of this lab, you will have a working Wild Rydes application where users can interact with a unicorn ride-sharing system through a simple map interface.

Below is the architecture we are going to build.



AWS Amplify:
A development platform that simplifies building and deploying full-stack web and mobile applications. It helps set up and connect backend services like authentication, APIs, and storage while offering hosting for frontend applications.

Amazon DynamoDB:
A fully managed NoSQL database service that delivers high performance, scalability, and low latency. Ideal for applications needing fast and flexible key-value or document data storage.

AWS Lambda:
A serverless compute service that lets you run code without provisioning or managing servers. Code executes in response to triggers such as API calls, database changes, or events from other AWS services.

Amazon API Gateway:
A fully managed service for creating, publishing, and managing APIs. It acts as a bridge between your frontend application and backend services like Lambda, enabling secure and scalable API interactions.

Amazon Cognito:
A service for adding user authentication and authorization to your applications. It supports sign-up, sign-in, and access control with built-in integrations for identity providers like Google and Facebook.

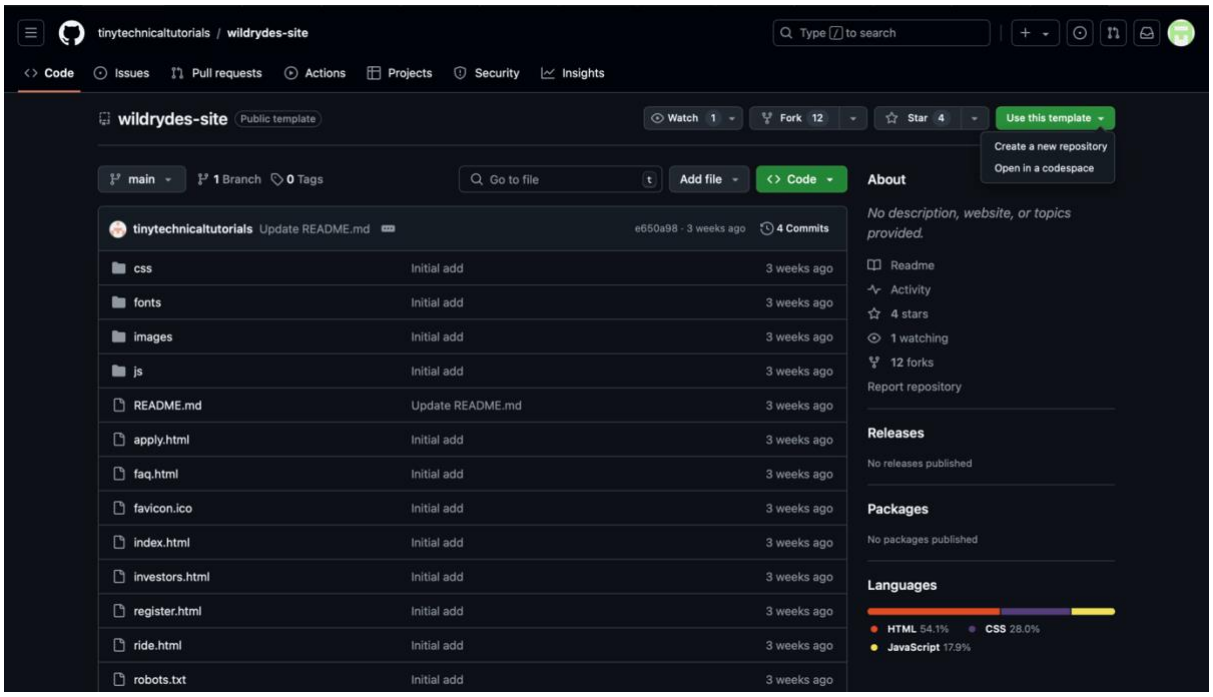
GitHub:
A version control and collaboration platform for managing code repositories. It allows teams to track changes, manage branches, and collaborate on projects efficiently.

Note: If you don't already have a GitHub account, go to https://github.com/ and create an account.

For this project, you will need the code files, and they are available in the repository linked below.

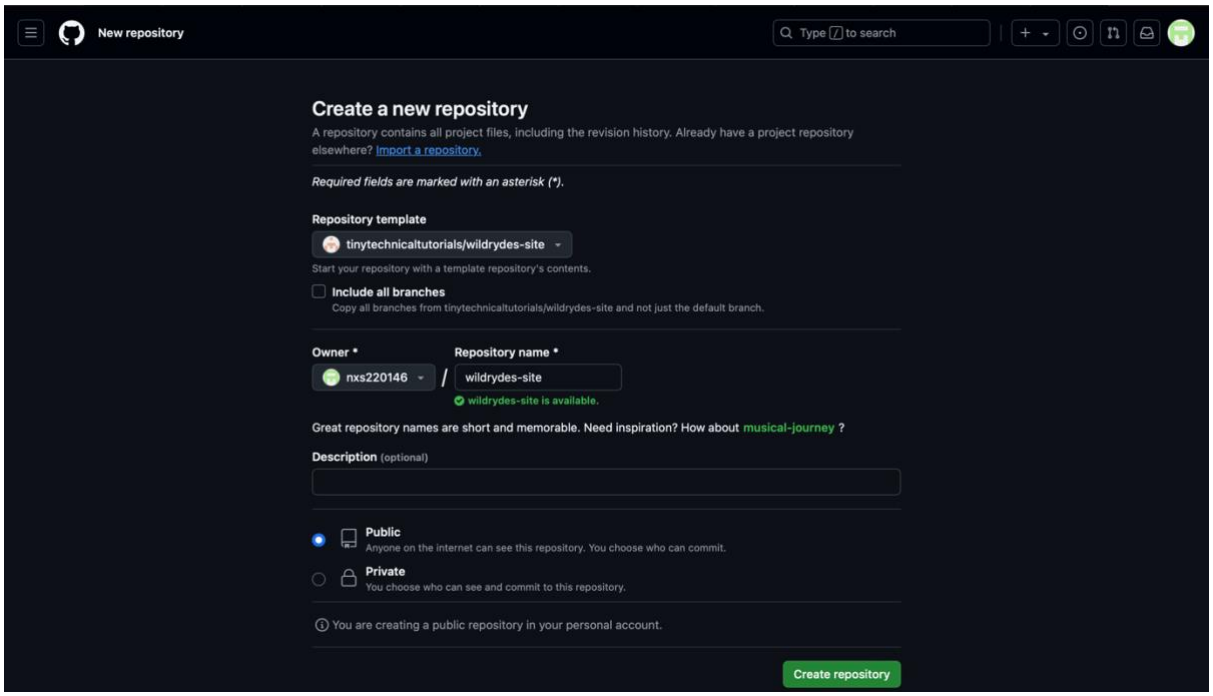
To bring the code to your GitHub account, follow these steps:

1. Go to the following link: https://github.com/tinytechnicaltutorials/wildrydes-site.
2. Click on the **Use this template** button at the top-right of the repository page.
3. Select **Create a new repository**.

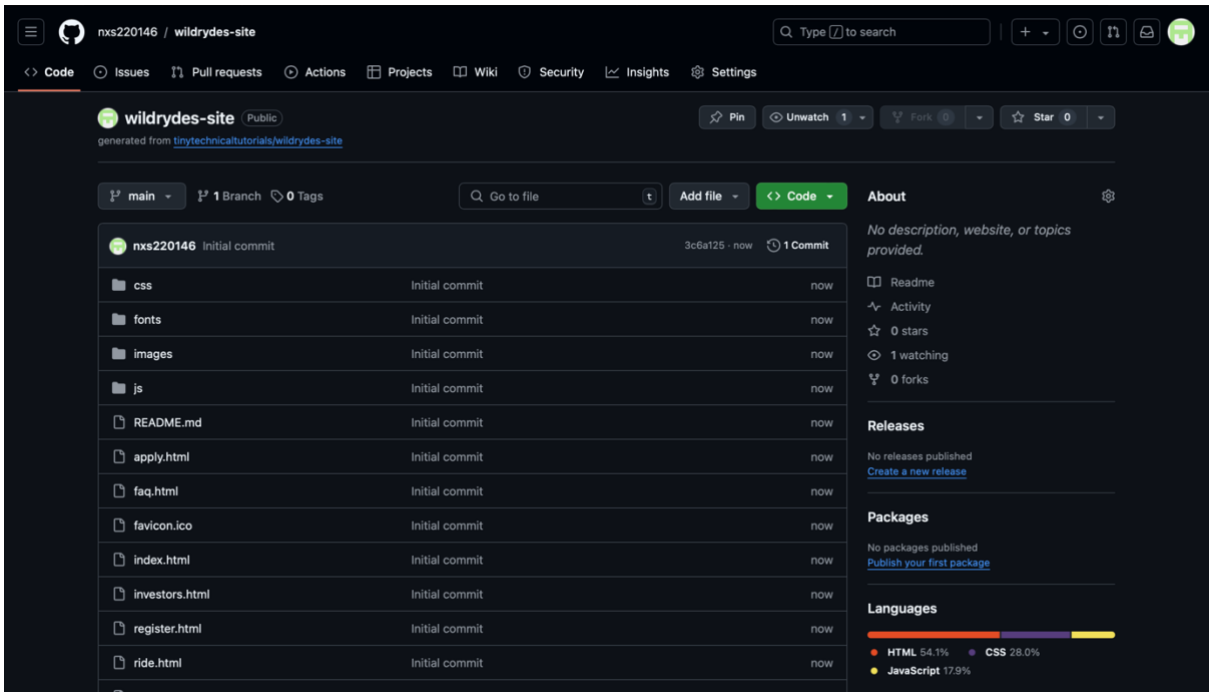


Fill in the **Repository name** as **wildrydes-site** .

Click **Create repository** to generate your own version of the repository.

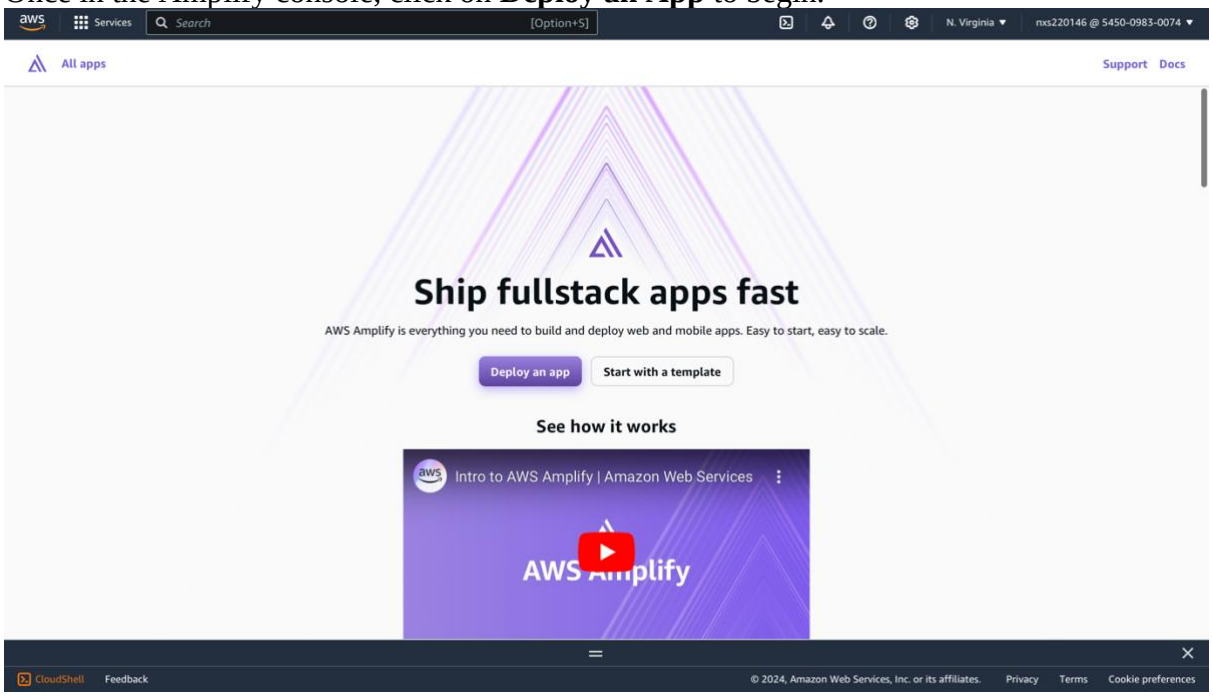


You should see something like the screenshot below. This screenshot shows the repository with all the code files now available in your GitHub account.

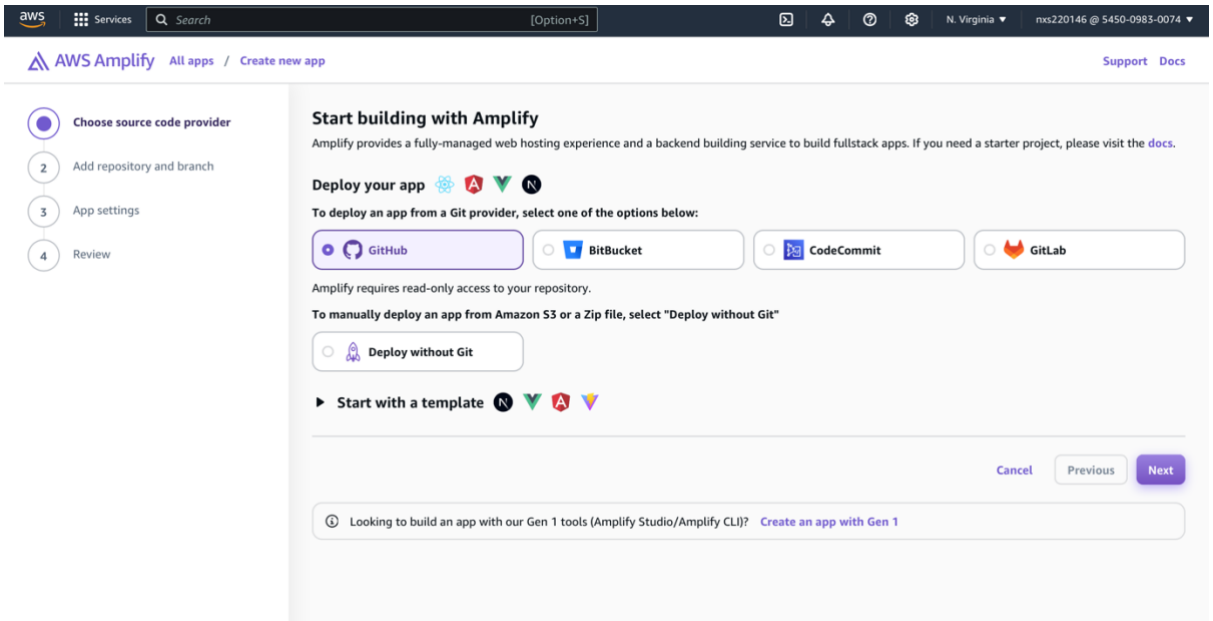


i. Creating a new app for hosting in AWS Amplify

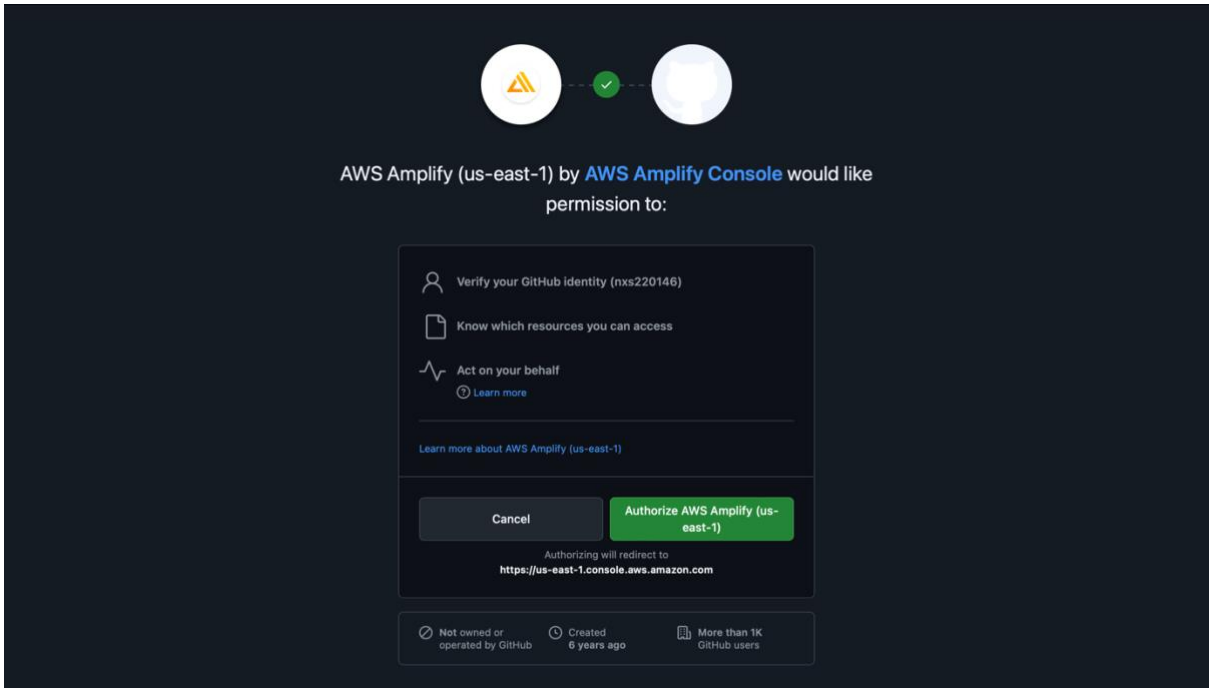
Click **Search** in the AWS Management Console, type **Amplify**, and select it from the results. Once in the Amplify console, click on **Deploy an App** to begin.



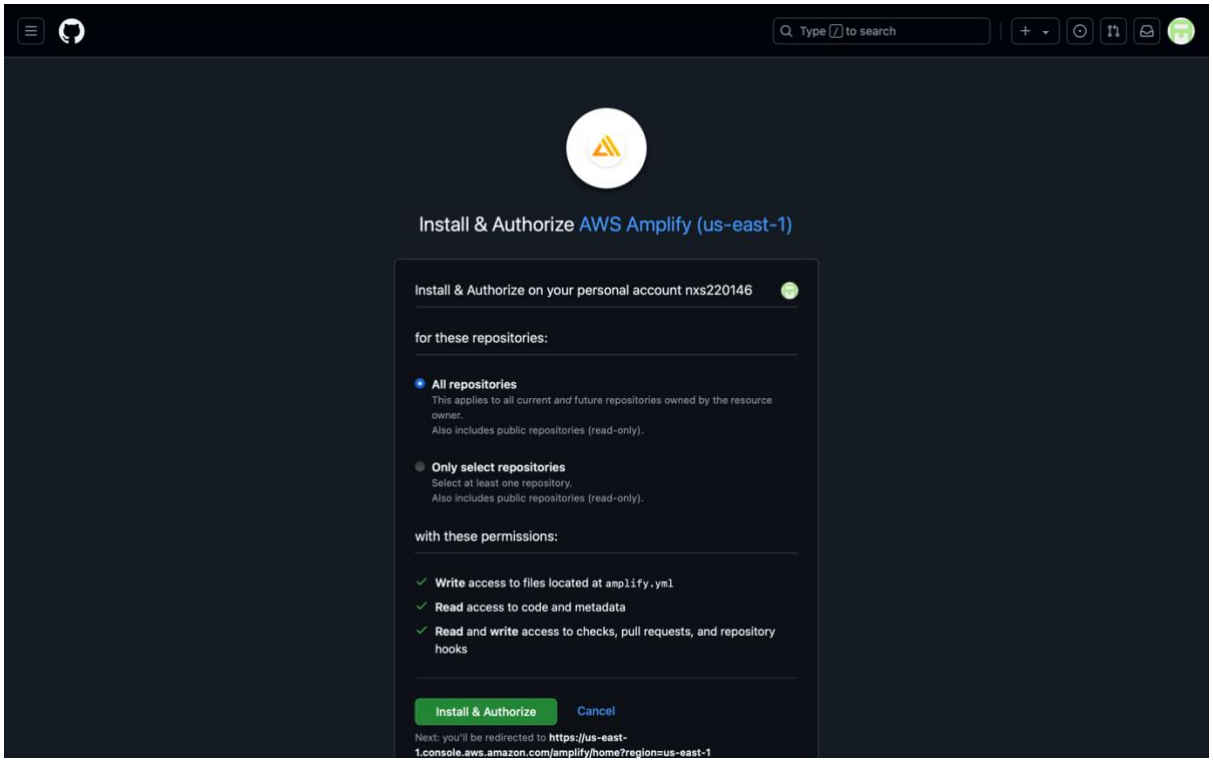
Choose **GitHub** as your repository source and click **Next** to proceed.



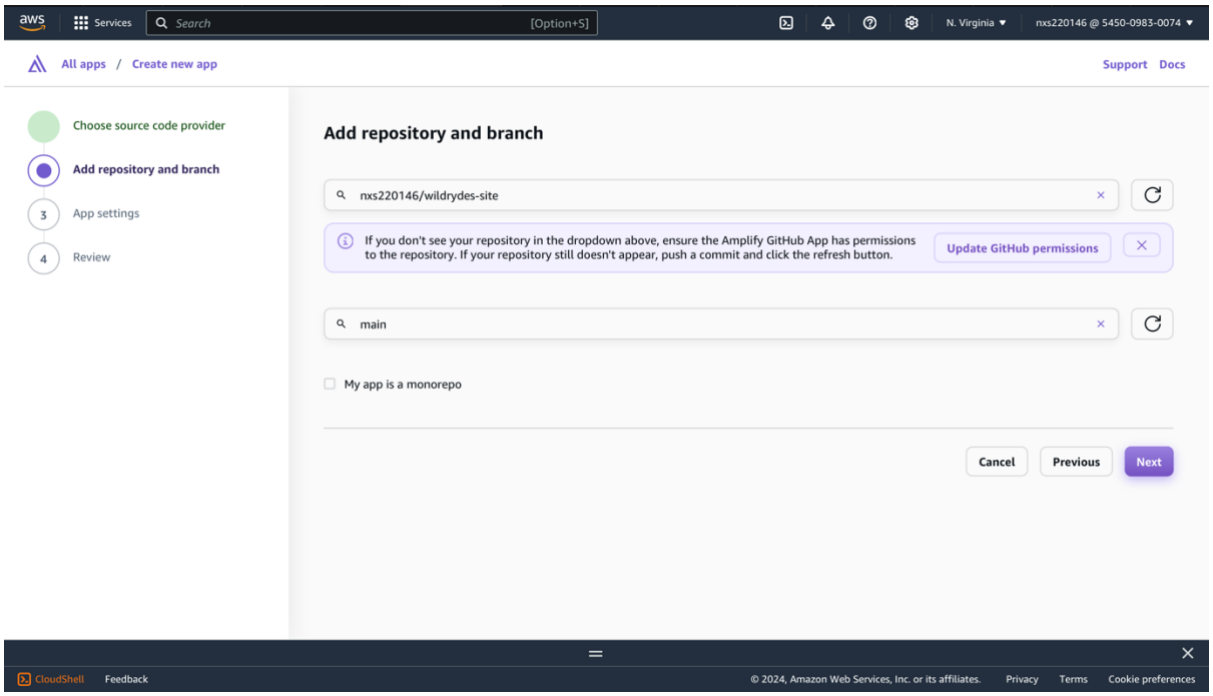
A prompt will appear asking you to authorize **AWS Amplify** to access your GitHub account. Follow the instructions to complete the authorization.



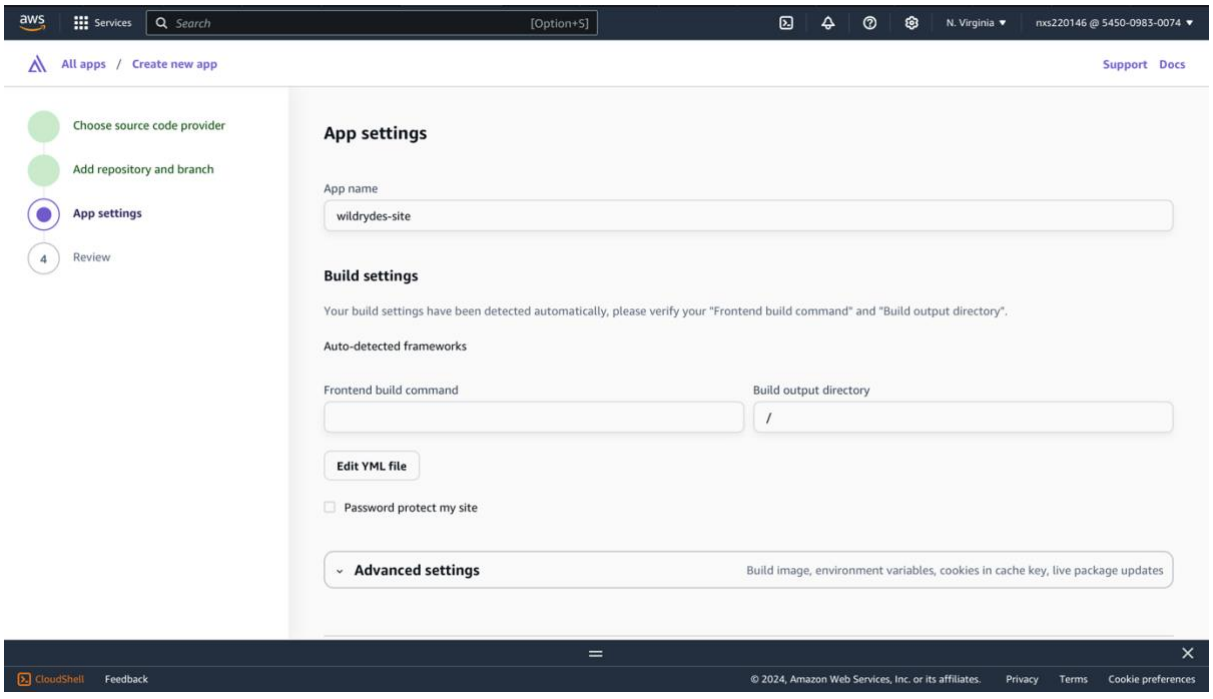
Click on the prompt. You can choose to authorize **AWS Amplify** for a specific repository or all repositories. In this case, I selected **All Repositories**, then click **Install and Authorize** to proceed.



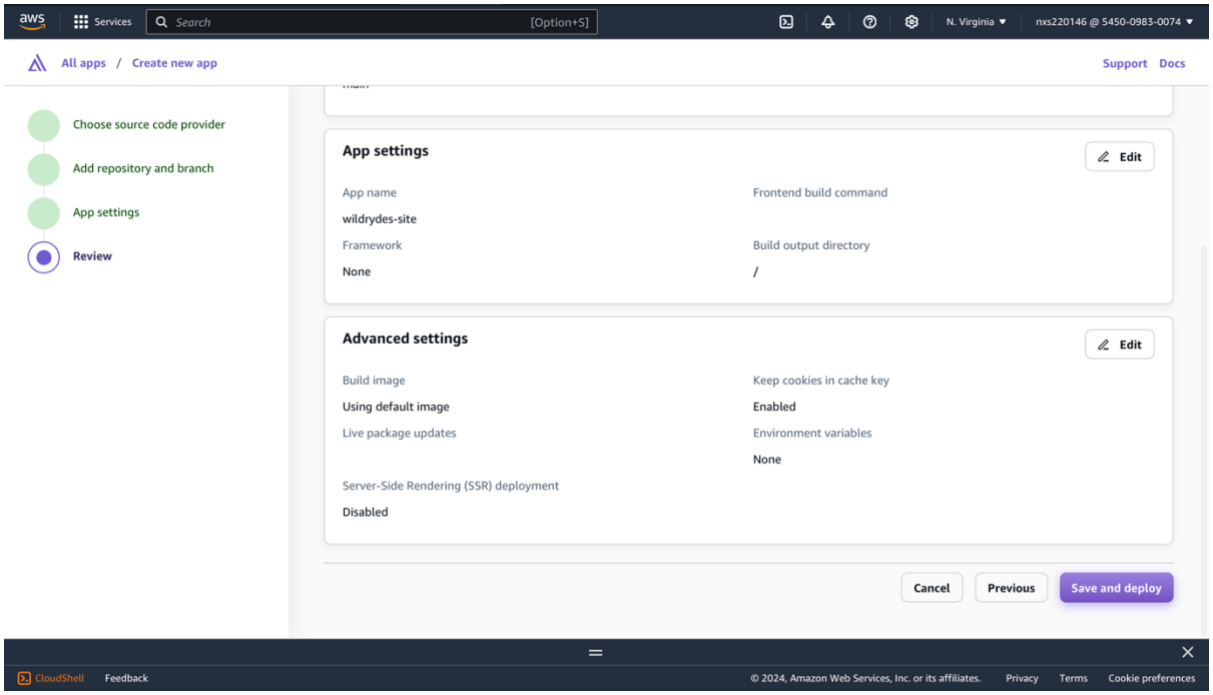
In the dropdown menu, select the desired repository. If the repository is not visible, click **Refresh** or **Update GitHub Permissions** to reload the list. Once selected, click **Next** to continue.



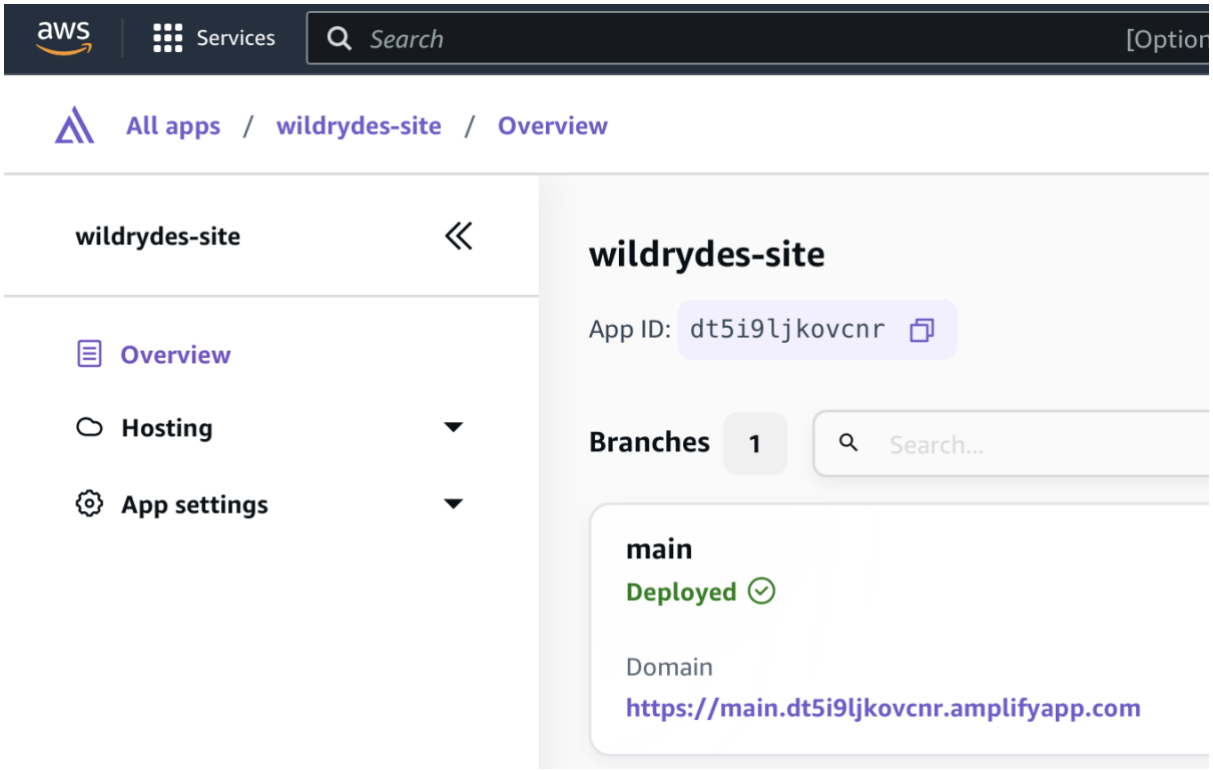
Enter **wildrydes-site** as the app name and click **Next** to proceed.



Click **Save and Deploy** to finalize the setup and start deploying your application.



After the deployment is complete, click on the **domain link** provided to view your application.



Note: The above screenshot is a deliverable. (It should capture the entire screen, with your **NETID** visible in the top-right corner.)

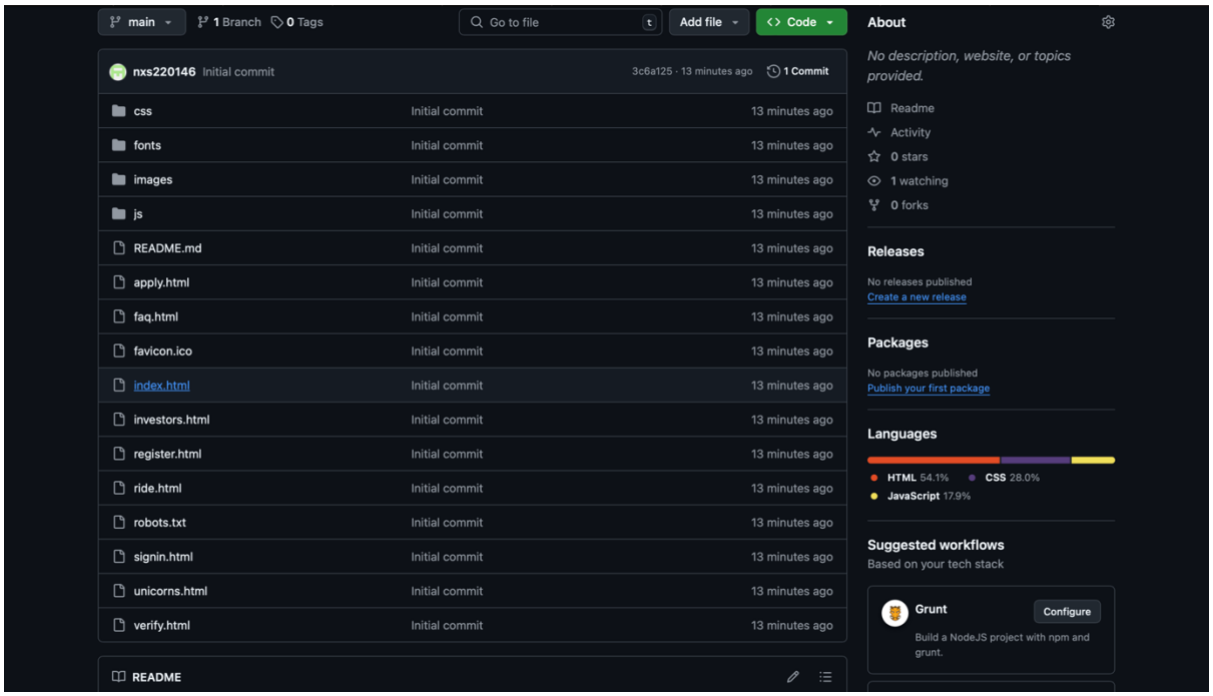
You will see the **WildRydes** website. Notice how we didn't need to manage any underlying infrastructure—AWS Amplify handled everything, allowing us to seamlessly deploy the application with minimal effort.

GIDDY UP!

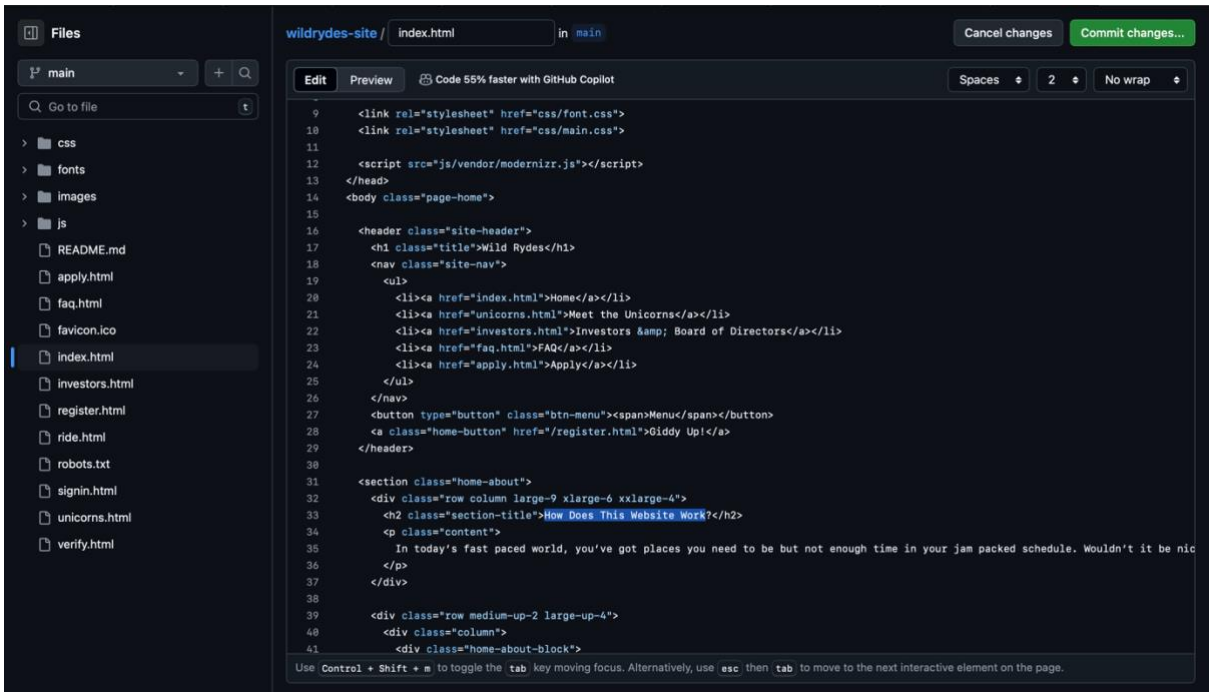
HOW DOES THIS WORK?

In today's fast paced world, you've got places you need to be but not enough time in your jam packed schedule. Wouldn't it be nice if there were a transportation service that changed the way you get around daily? Introducing Wild Rydes, an innovative transportation service that helps people get to their destination faster and hassle-free. Getting started is as easy as tapping a button in our app.

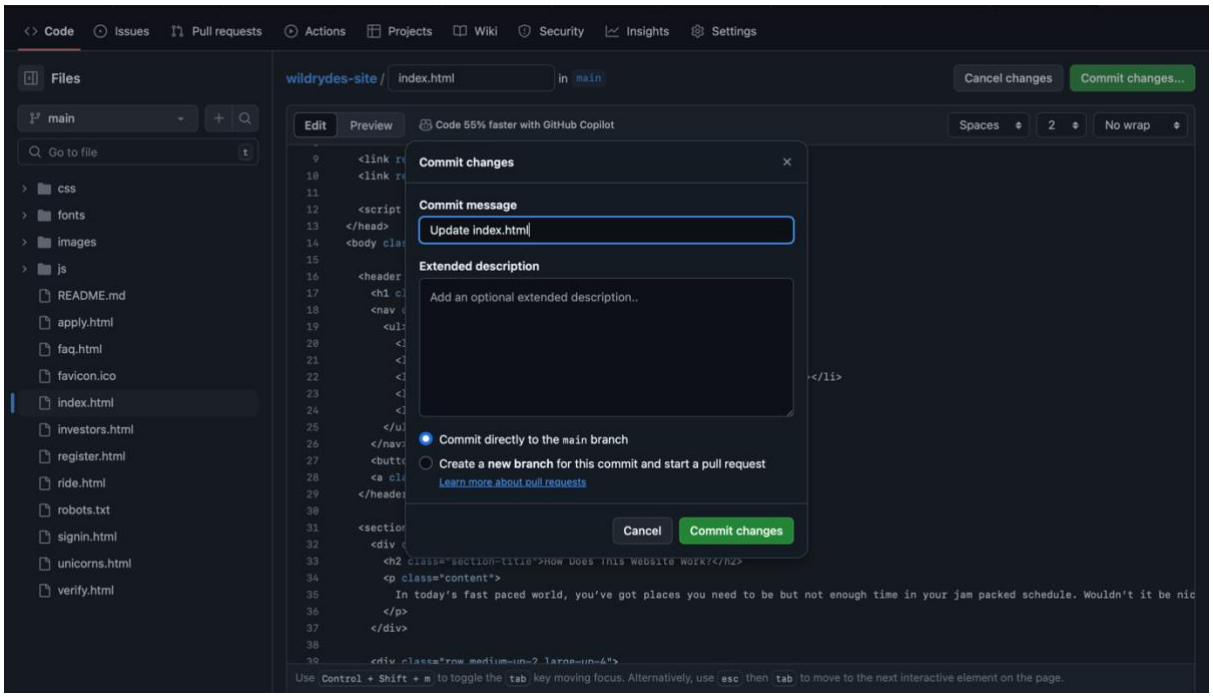
Go to your **GitHub repository** and click on **index.html**. Click on the **Edit this file** button to modify the file directly in GitHub.



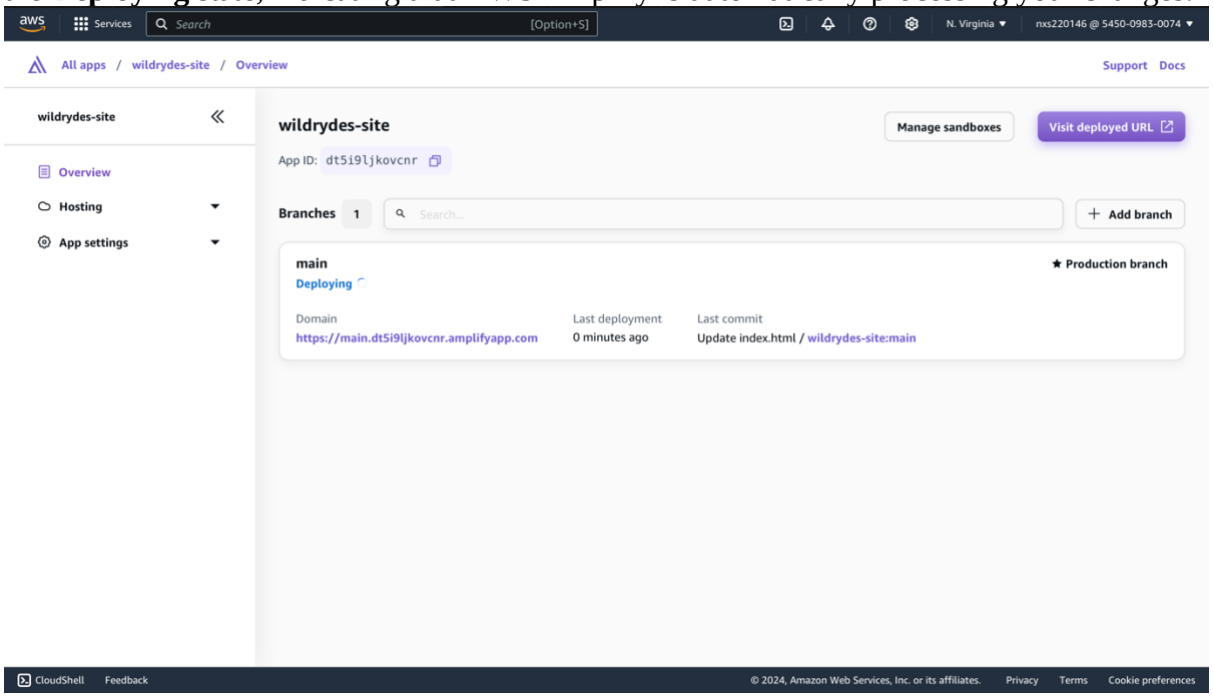
In the **index.html** file, locate **line 33**.
Replace the text **"How does this work"** with **"How does this Website work"**.



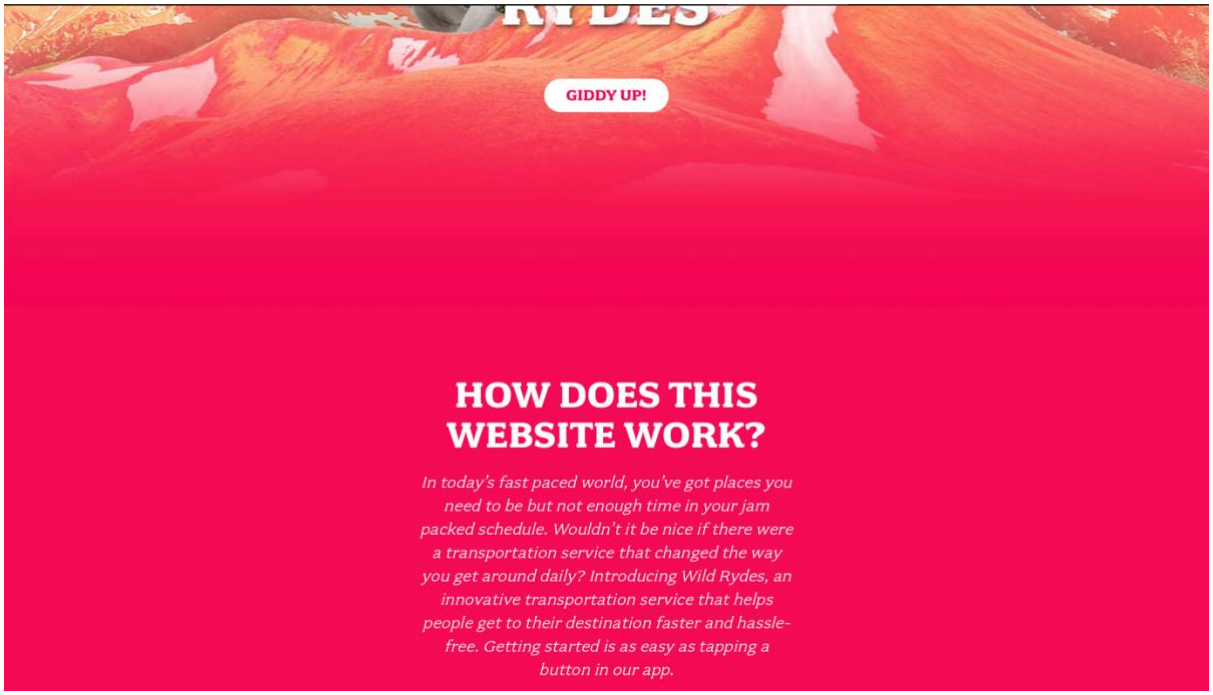
Save your changes by clicking **Commit changes.**



Switch back to your **AWS Amplify portal**, and you will see that the app is in the **Deploying** state, indicating that AWS Amplify is automatically processing your changes.

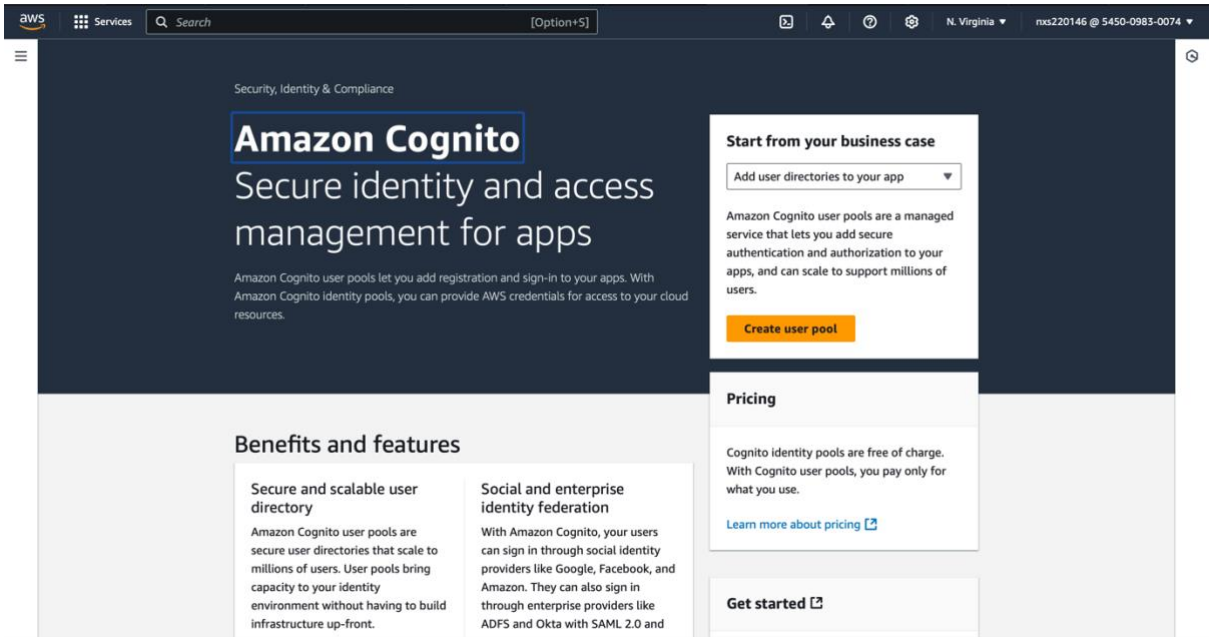


Once the deployment is complete, refresh your website or click on the **domain link** again to view the updated changes.

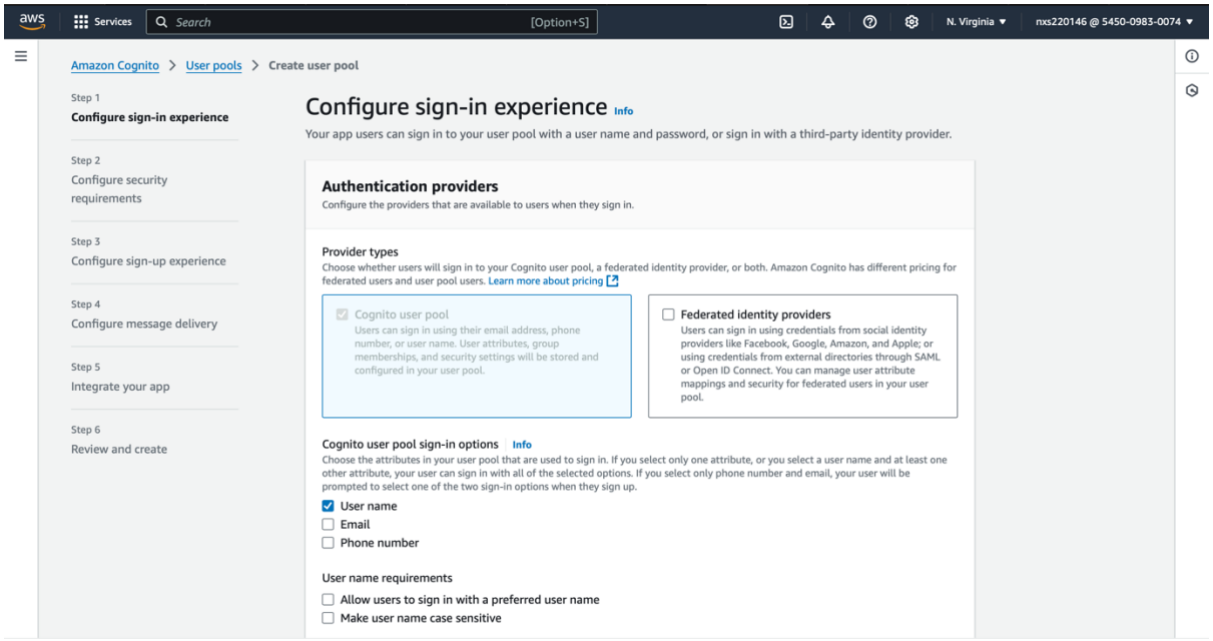


ii. Setting up Amazon Cognito for user authentication

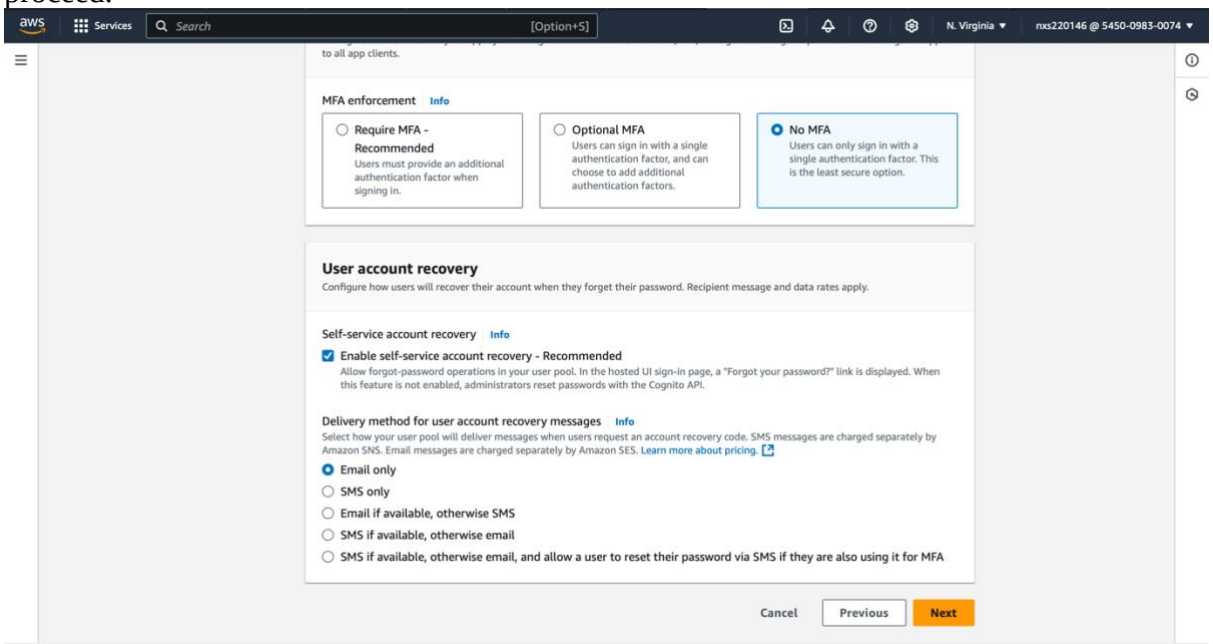
In the AWS Management Console, use the search bar to type **Cognito** and select it from the results. Once in the Cognito dashboard, click on **Create User Pool** to start setting up a new user pool.



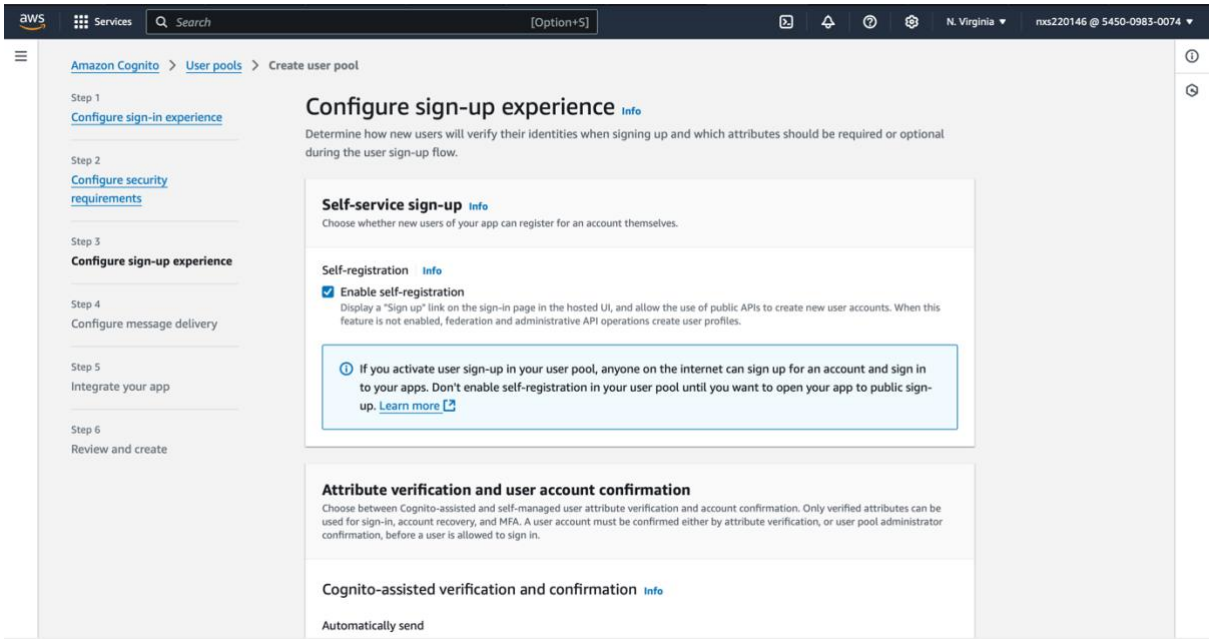
For the sign-in options, select **Username** as the preferred method. Proceed with the default settings provided by Cognito. Then, click **Next** to continue.

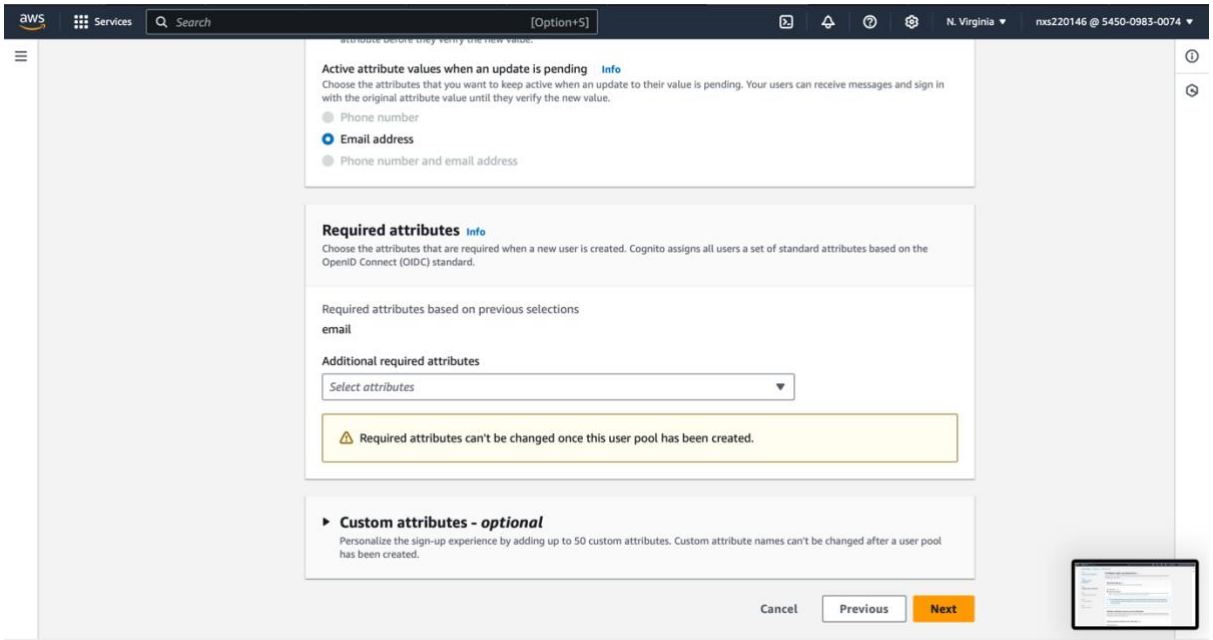


For the password policy, select **No MFA required**. While Cognito offers various customizable options here, we will stick with the default settings for simplicity. Click **Next** to proceed.

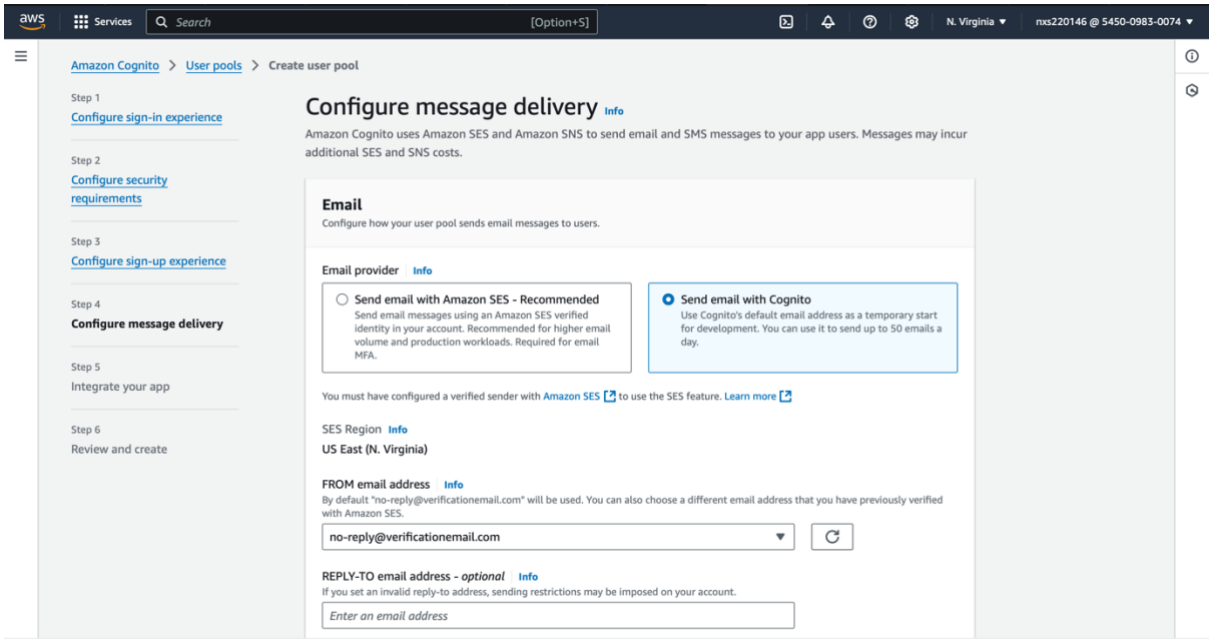


Under **Configure Sign-Up Experience**, enable the option for **Self Registration** to allow users to sign up for your application. Then, proceed to the next step.

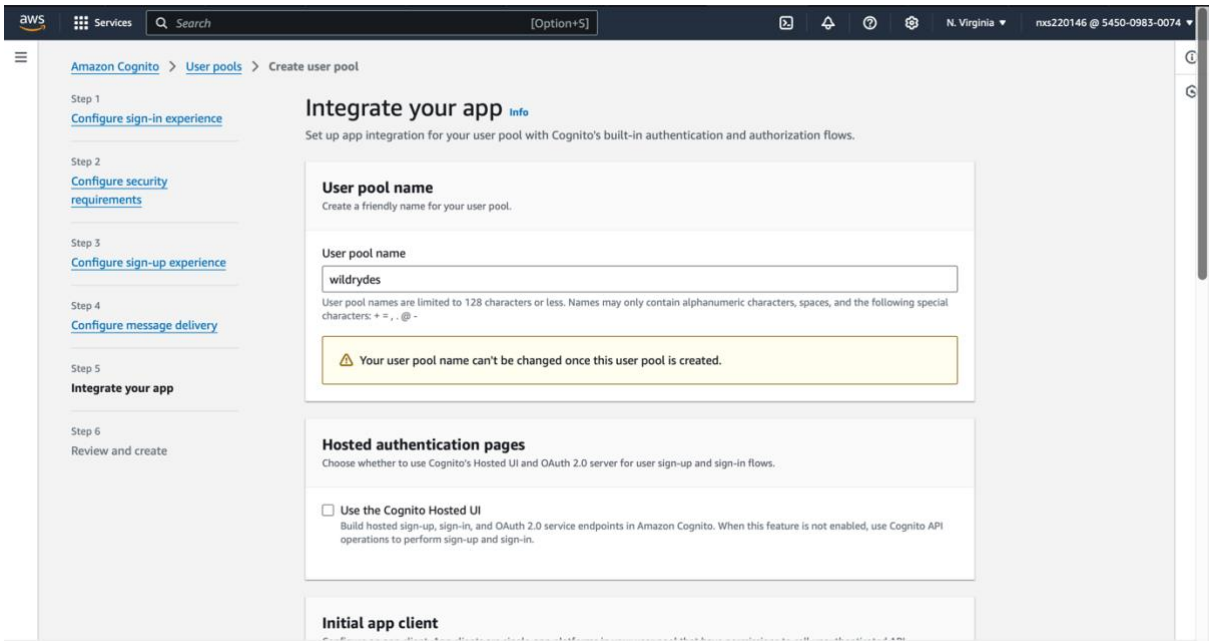




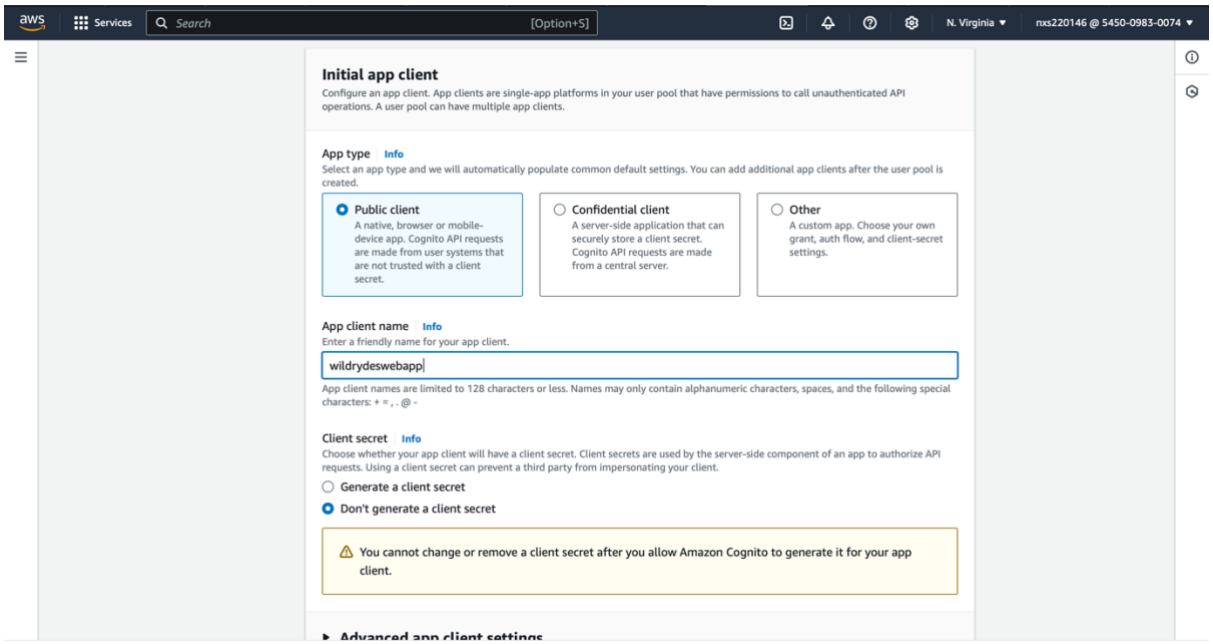
Select the option to send emails using **Cognito** instead of the **Simple Email Service (SES)**. Toggle this setting to enable it, and then proceed to the next step.



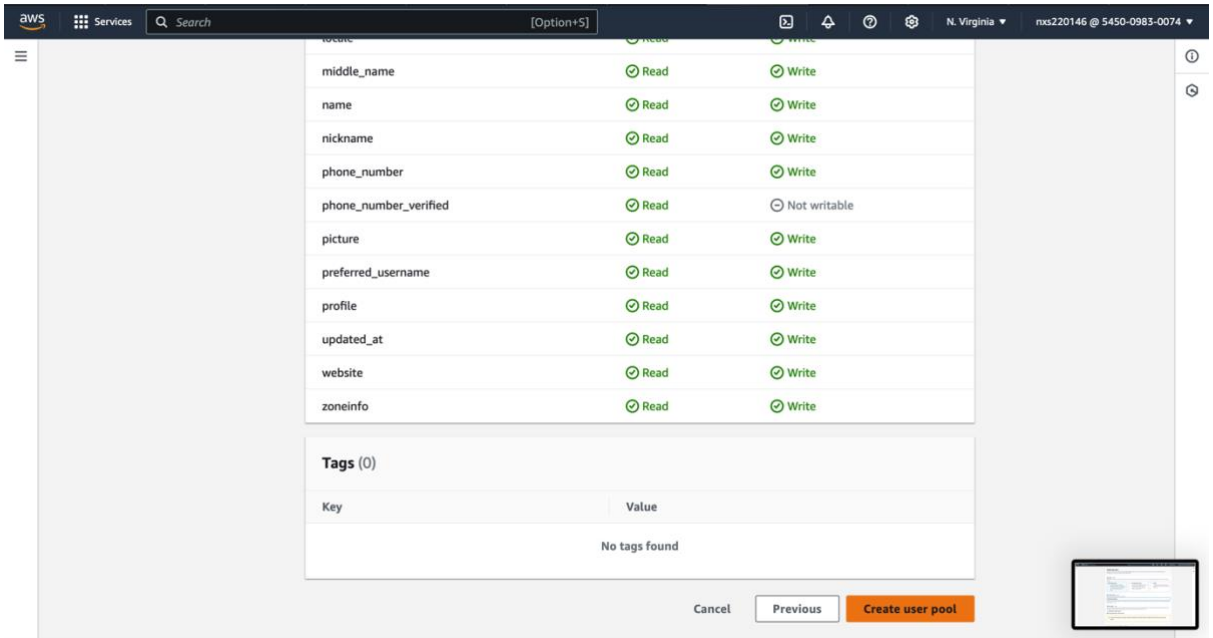
For the **User Pool Name**, enter **wildrydes**.

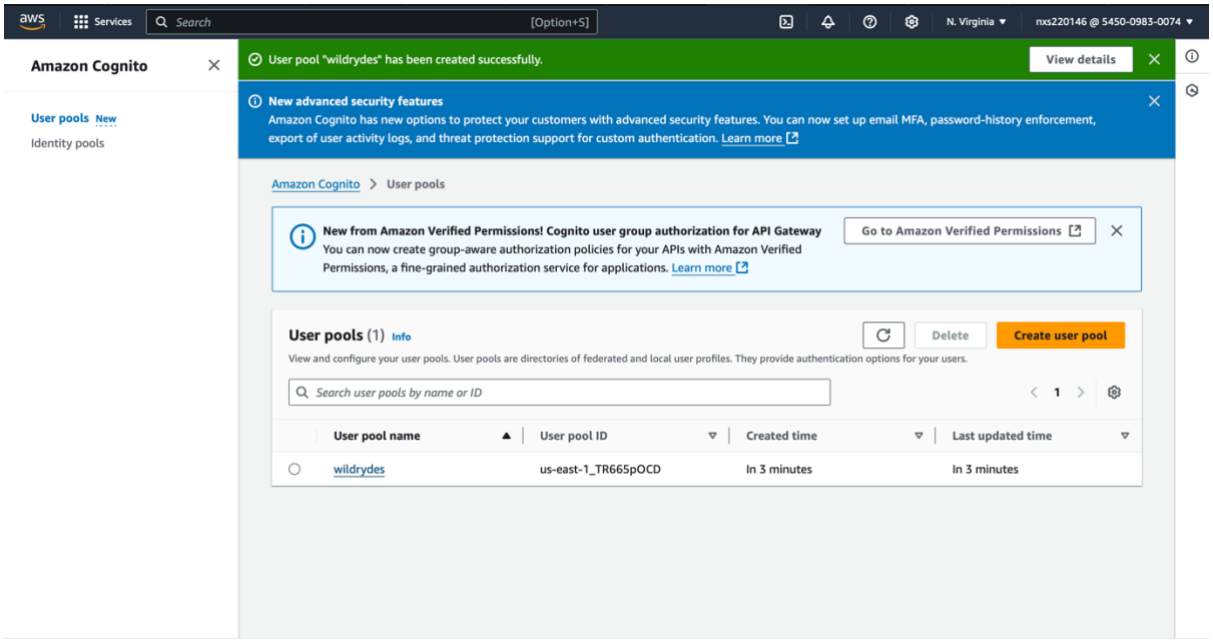


Provide a name for your initial app client. For the **Client Name**, enter **wildrydeswebapp**.



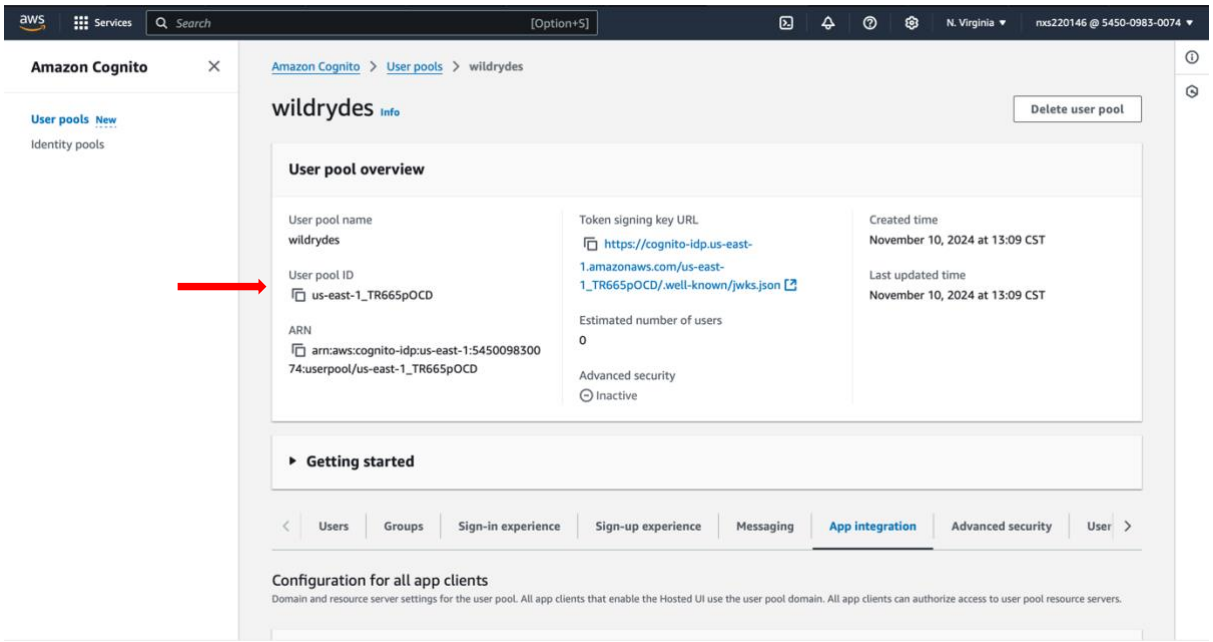
Click on next and create user pool.



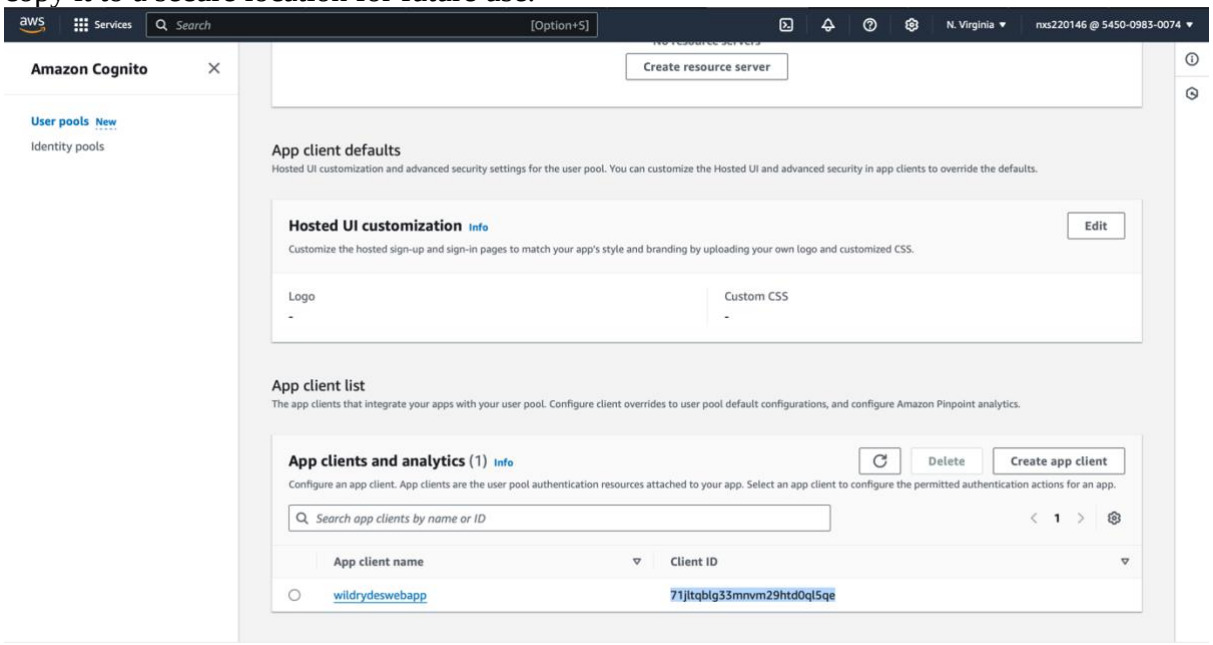


Note: The above screenshot is a deliverable

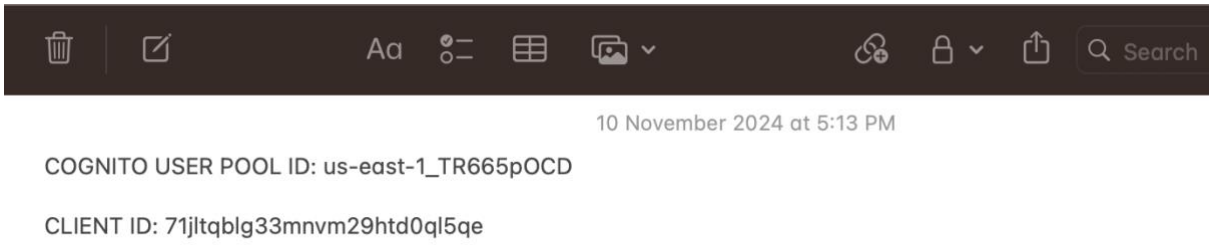
Open your newly created **User Pool** from the Cognito dashboard. Locate the **User Pool ID** and copy it to a safe place, such as a notepad or a secure document, for future reference.



In your **User Pool**, go to the **App Integration** section. Scroll down to find the **Client ID**, and copy it to a secure location for future use.

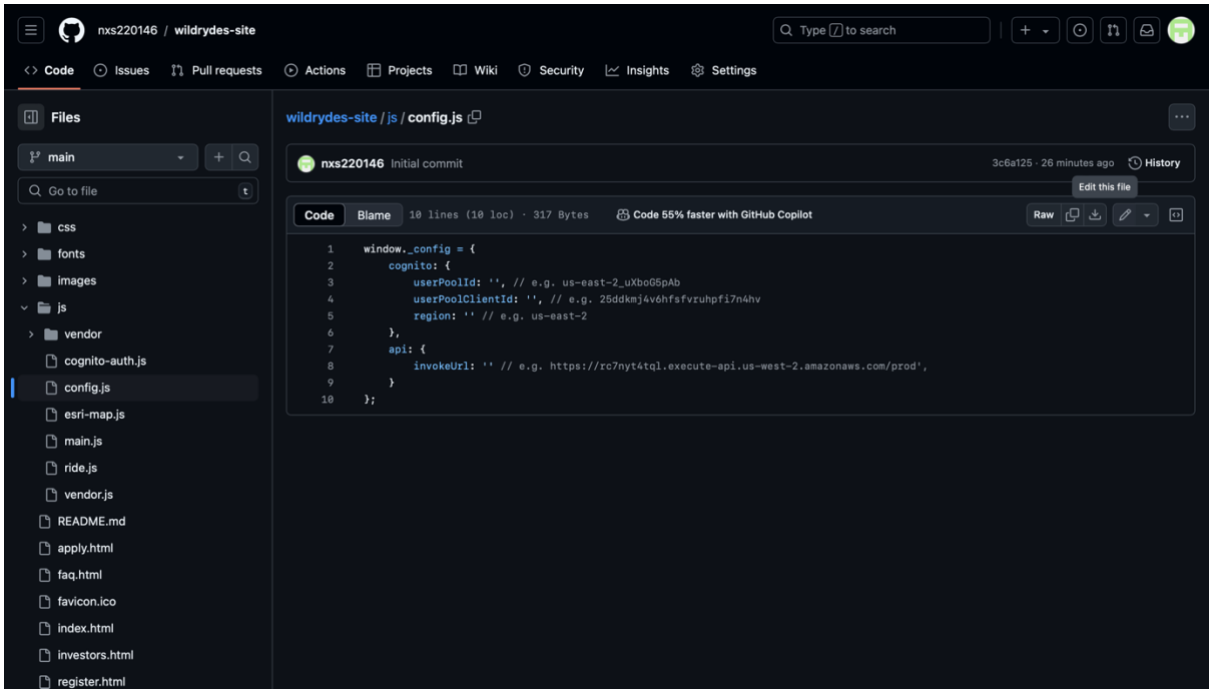


I have saved the **User Pool ID** and **Client ID** in my Notes app for easy access.



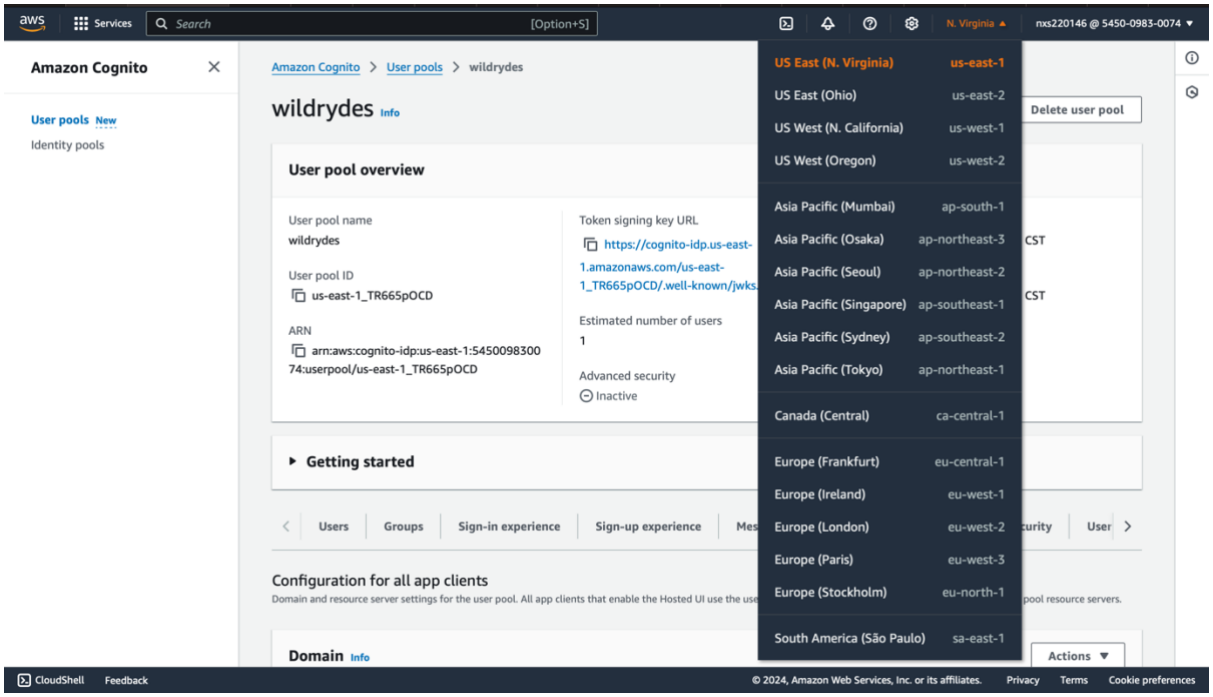
Updating the app configuration file to use the Amazon Cognito user pool

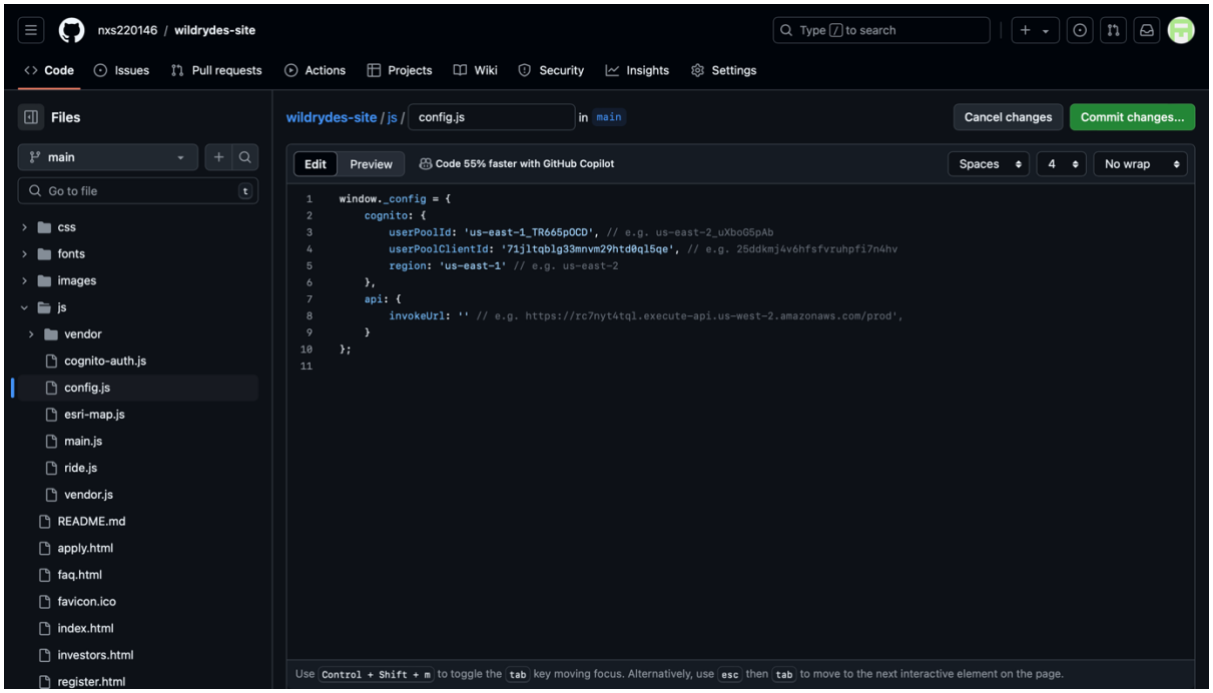
Go to your **GitHub repository** and open the **config.js** file. Click on the **Edit this file** button to make changes directly in the file.



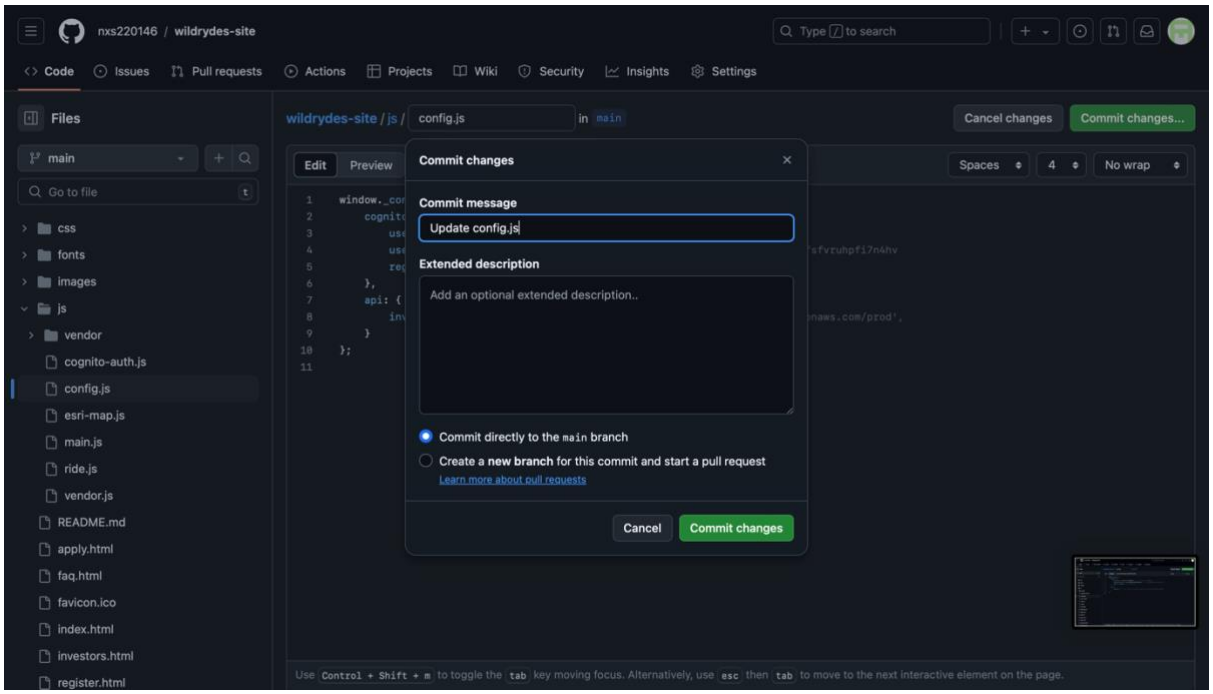
In the **config.js** file, paste your **User Pool ID** and **Client ID** between the quotes. Additionally, check which AWS region you are currently working in, and enter the corresponding region code in the appropriate field.

You can find your AWS region by going to the AWS Management Console. Look at the top-right corner and click the down arrow next to the region name. It will display your current region. For example, if you are in **North Virginia**, the region code will be **us-east-1**. Use this code when entering your region in the **config.js** file.

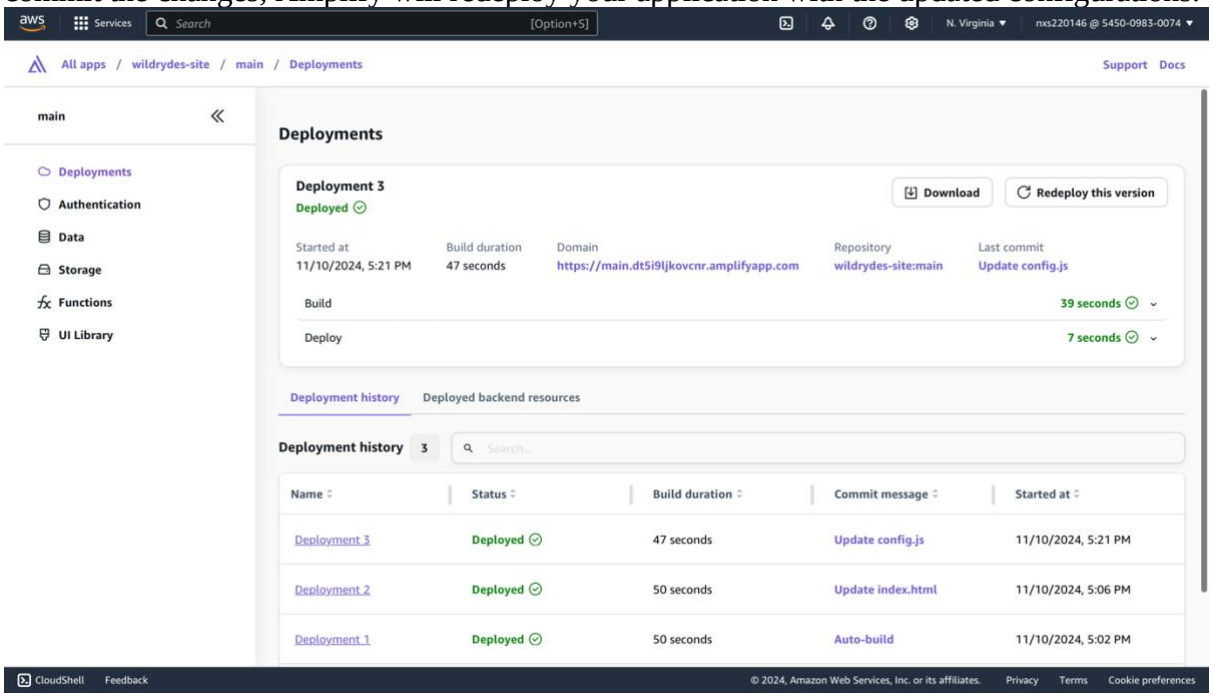




Click on commit changes.

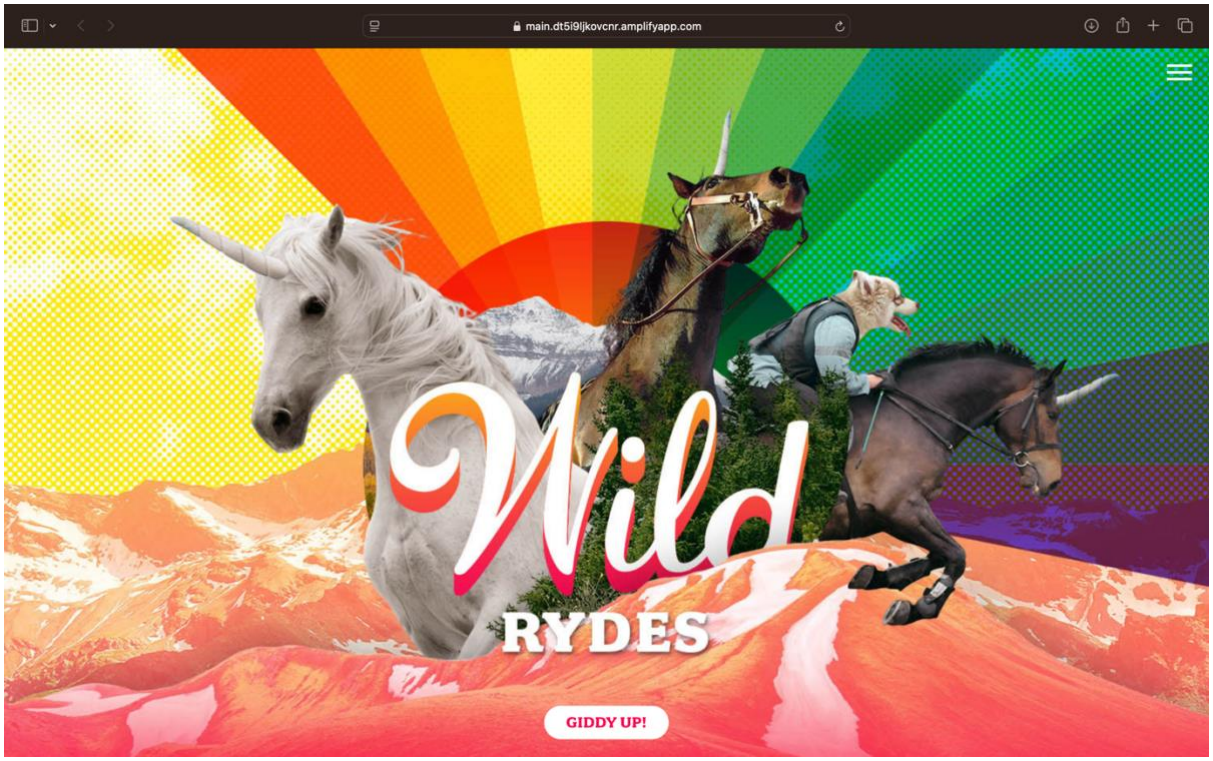


As you've seen before, **AWS Amplify** automatically detects changes in your code. Once you commit the changes, Amplify will redeploy your application with the updated configurations.

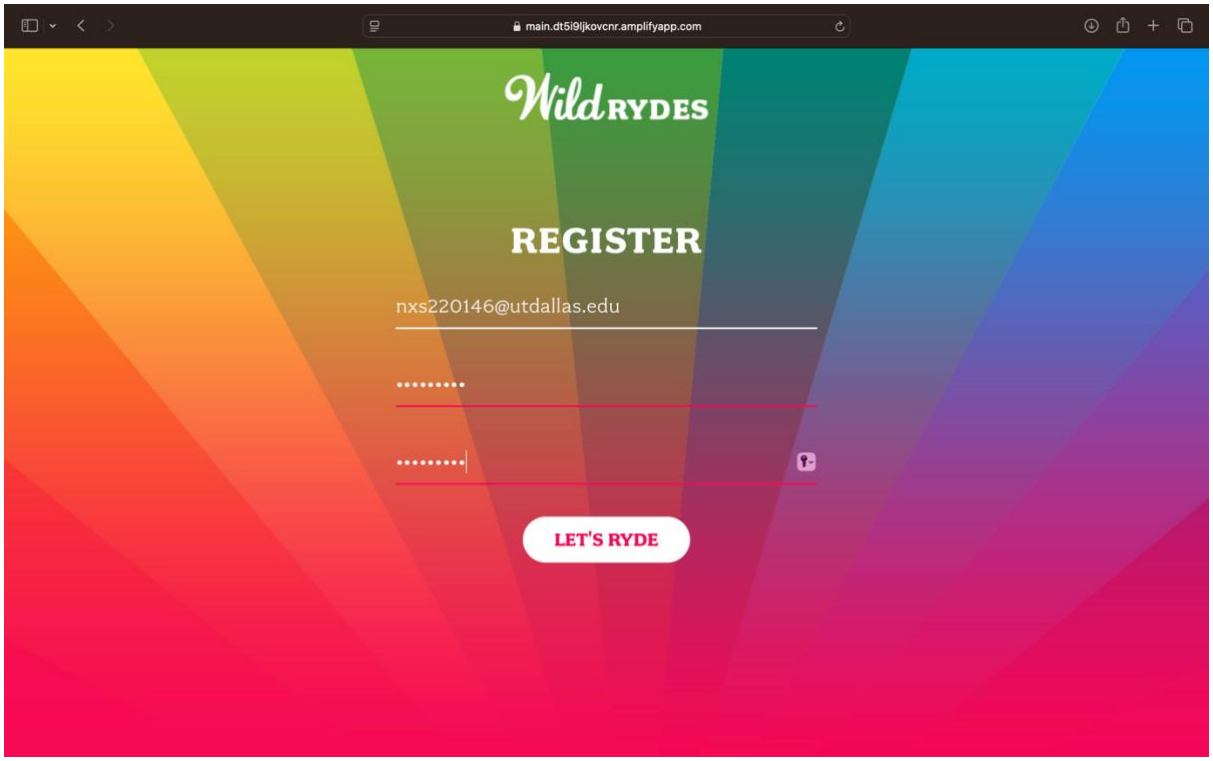


Click on the **domain URL** or refresh your website to see the updated changes reflected in your application.

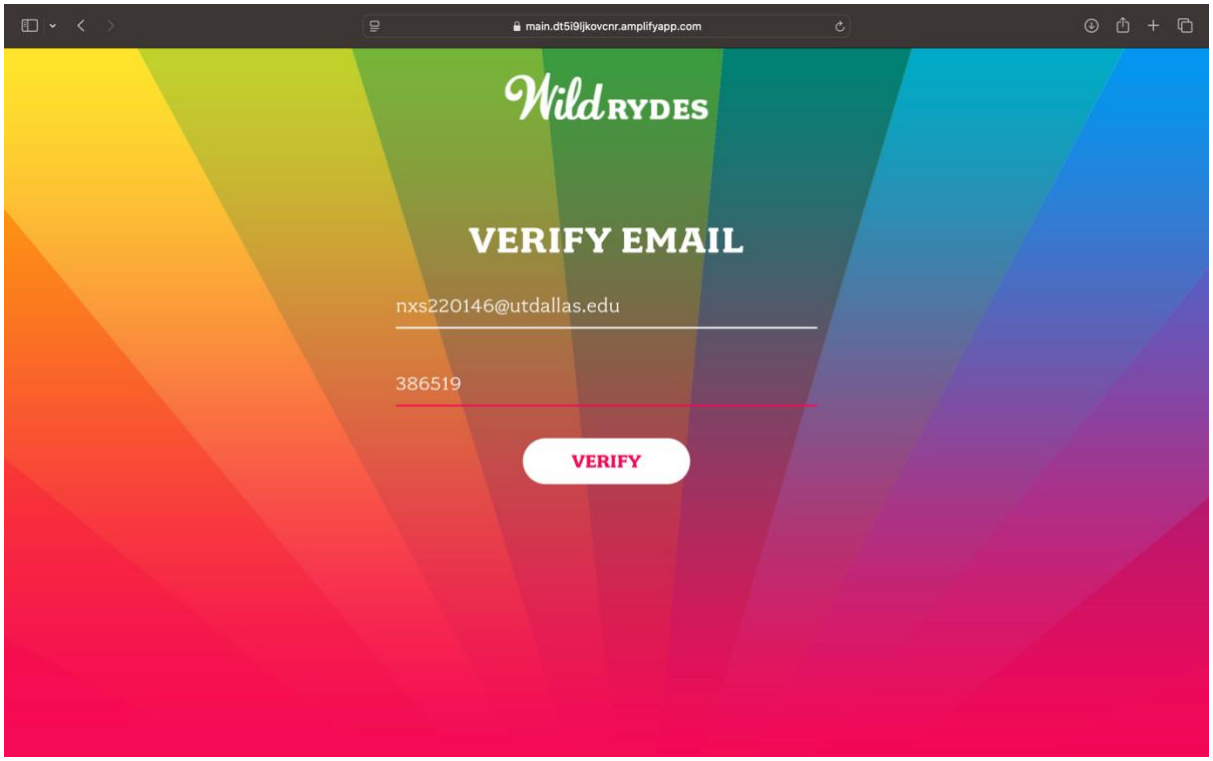
Testing Cognito integration by doing user registration and login



Click on **Giddy Up** to register for an account. Make sure to use a valid email address, as a verification code will be sent to this email for confirmation.



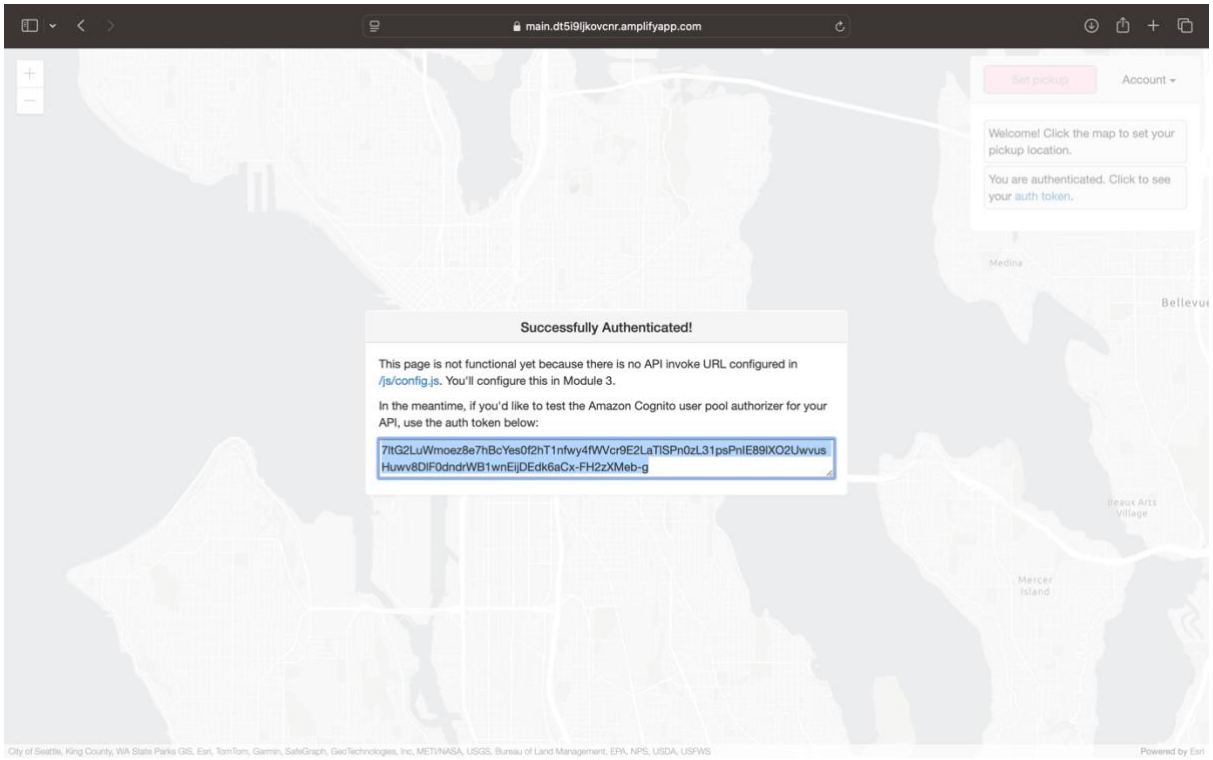
After completing the registration, you'll be prompted to verify your account. Check your email for the verification code sent by AWS Cognito, and enter the code to complete the verification process.



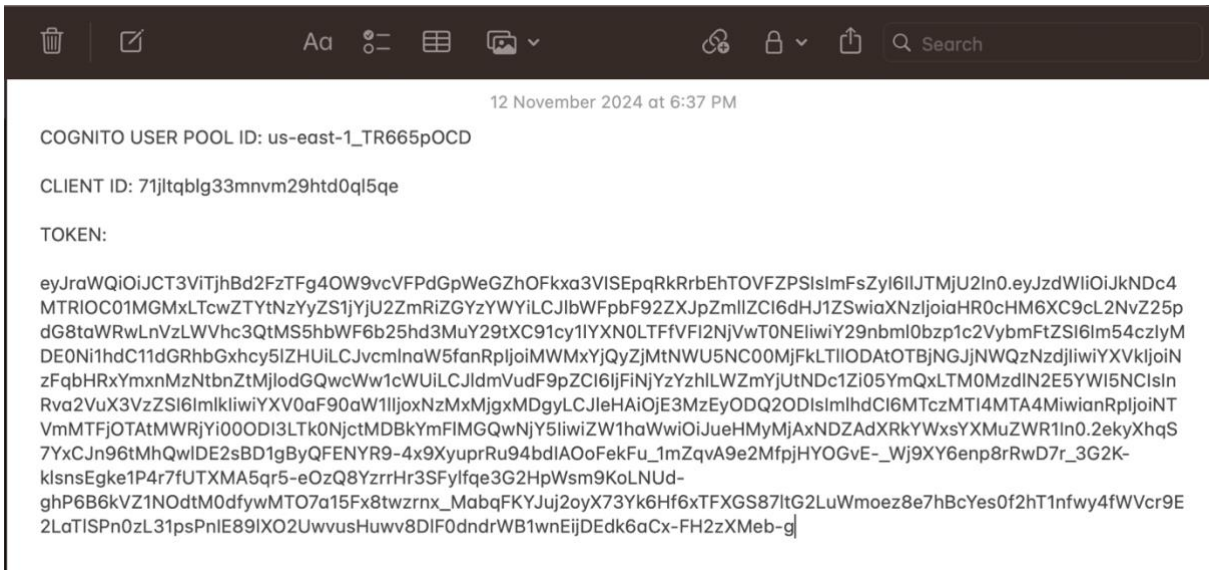
After verifying your account, proceed to **sign in** using your registered email address and password.

After successfully signing in, you might see a page you weren't expecting. This page will display a **token**. You'll need this token for testing your API later.

Select the entire token, copy it, and save it securely—for example, paste it into your **Notes** app or a text editor like Notepad. Make sure you don’t lose it, as it will be essential for the next steps.

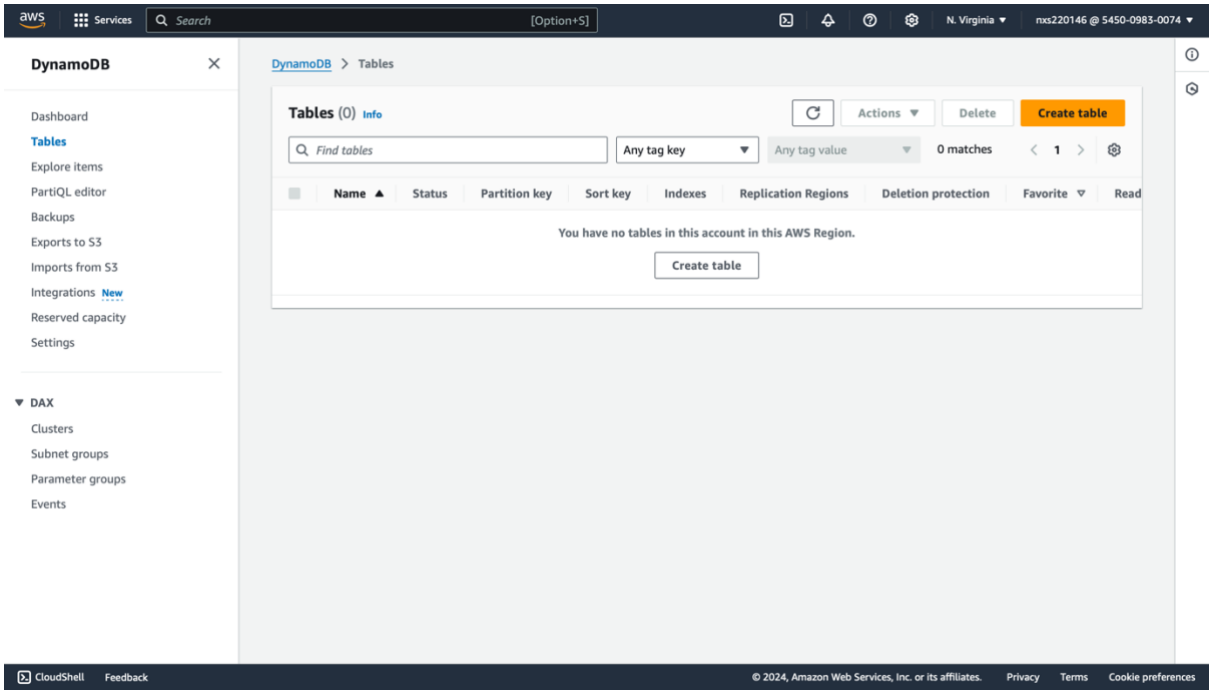


It would look something like this. Make sure to copy the entire token, as it may be long and span multiple lines.



iii. Creating a new DynamoDB table

In the AWS Management Console, search for **DynamoDB** and select it. Once you're in the DynamoDB dashboard, click on **Create Table** to start setting up a new table.

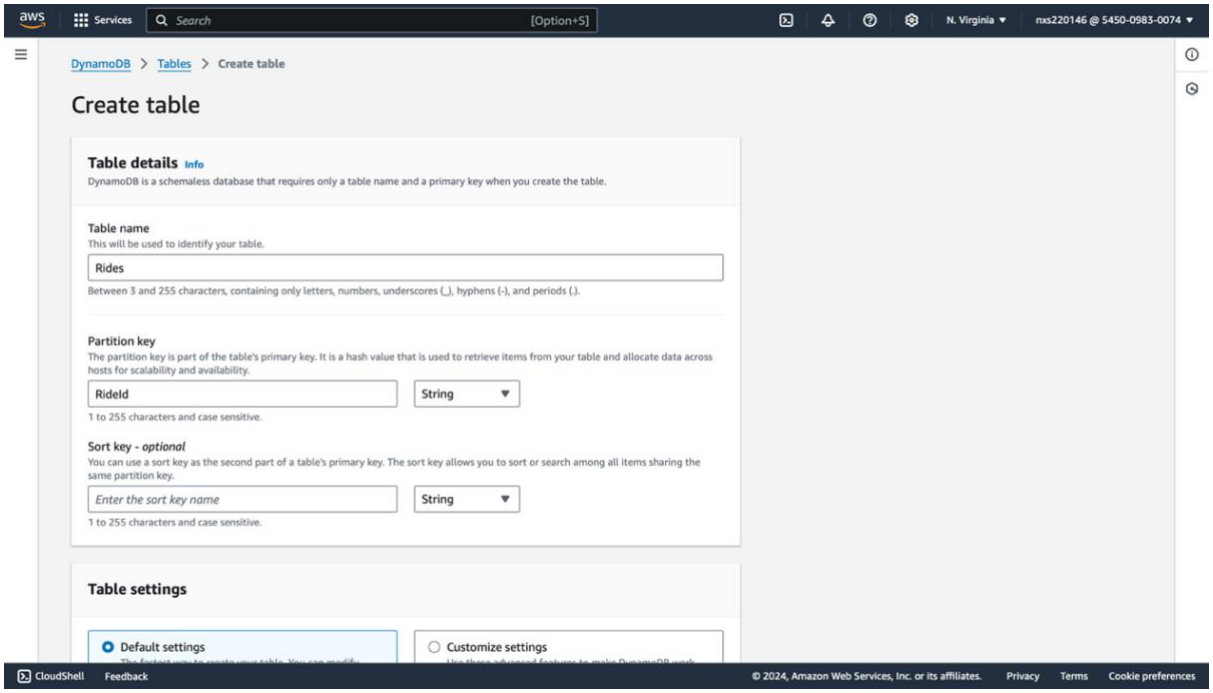


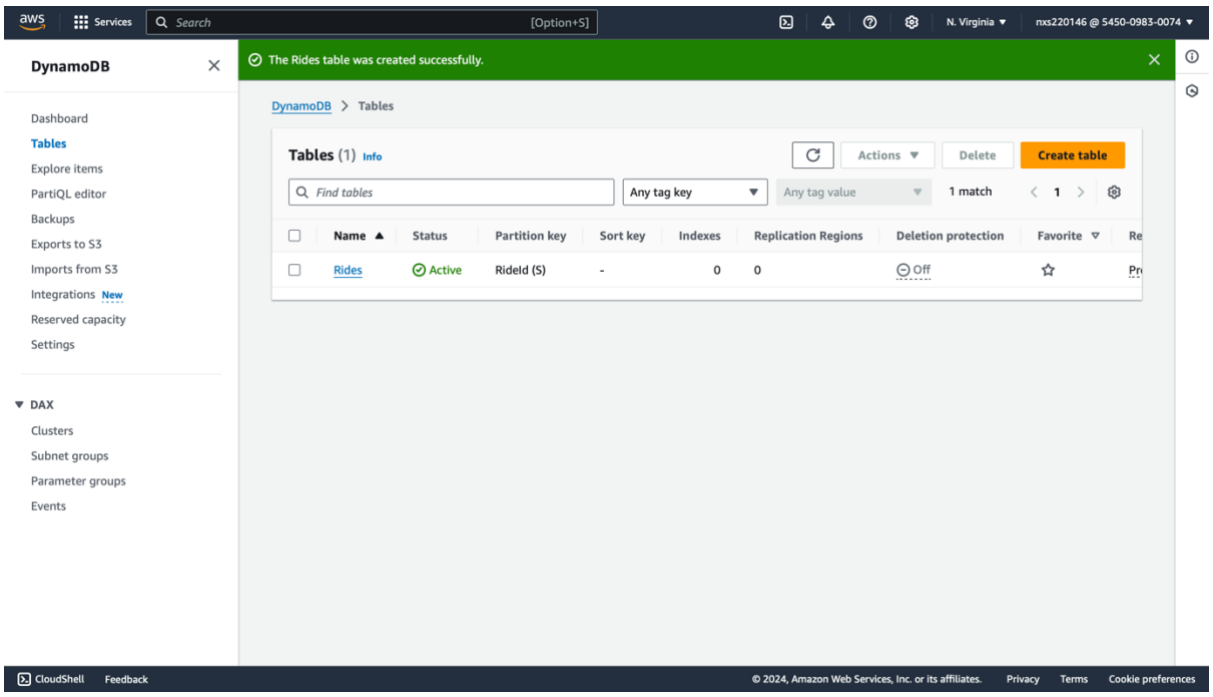
In the **Create Table** screen:

1. Set the **Table Name** to **Rides**.
2. For the **Partition Key**, enter **RideId**

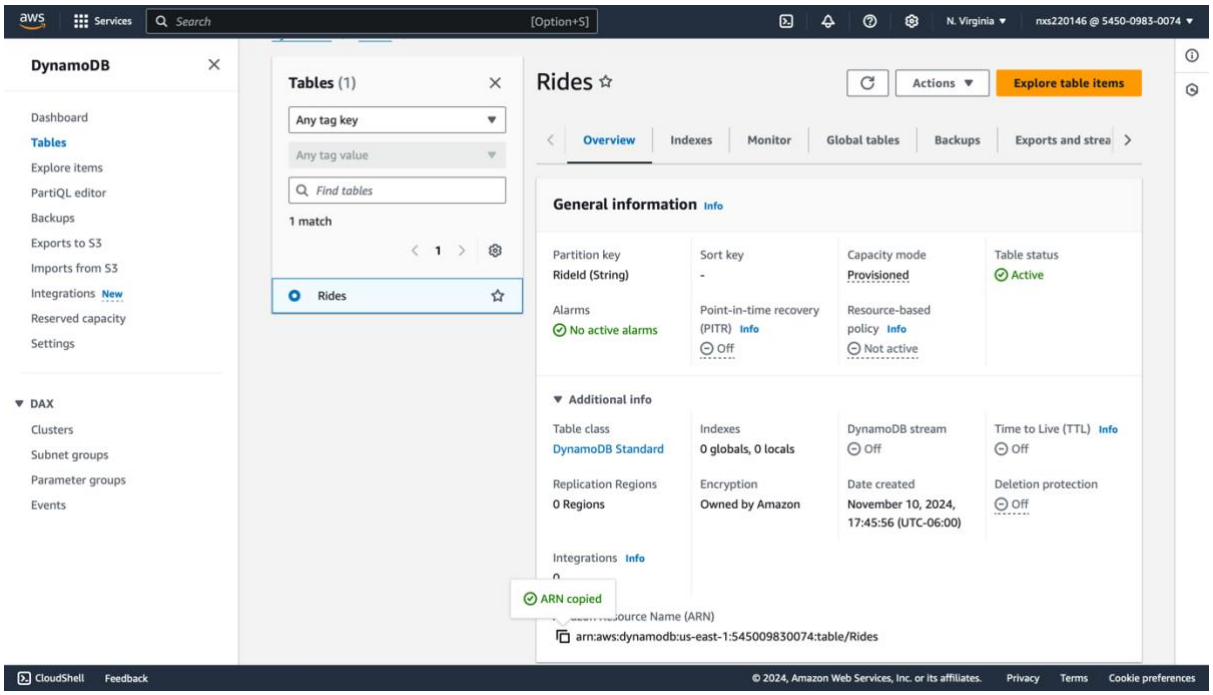
   Ensure the capitalization matches exactly - Capital R, small ide, Capital I, small d.

3. Leave all other settings as default.
4. Click **Create Table** to finalize the setup.


Services
Search
[Option+5]
N. Virginia
nxs220146 @ 5450-0983-0074
DynamoDB > Tables > Create table
Create table
Table details Info
DynamoDB is a schemaless database that requires only a table name and a primary key when you create the table.
Table name
This will be used to identify your table.
Rides
Between 3 and 255 characters, containing only letters, numbers, underscores (_), hyphens (-), and periods (.).
Partition key
The partition key is part of the table's primary key. It is a hash value that is used to retrieve items from your table and allocate data across hosts for scalability and availability.
RideId
String
1 to 255 characters and case sensitive.
Sort key - optional
You can use a sort key as the second part of a table's primary key. The sort key allows you to sort or search among all items sharing the same partition key.
Enter the sort key name
String
1 to 255 characters and case sensitive.
Table settings
Default settings
Customize settings
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Services
Search
[Option+5]
N. Virginia
nxs220146 @ 5450-0983-0074
DynamoDB
Dashboard
Tables
Explore items
PartiQL editor
Backups
Exports to S3
Imports from S3
Integrations New
Reserved capacity
Settings
DAX
Clusters
Subnet groups
Parameter groups
Events
The Rides table was created successfully.
DynamoDB > Tables
Tables (1) Info
Actions
Delete
Create table
Find tables
Any tag key
Any tag value
1 match
Name
Status
Partition key
Sort key
Indexes
Replication Regions
Deletion protection
Favorite
Rides
Active
RideId (S)
-
0
0
Off
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

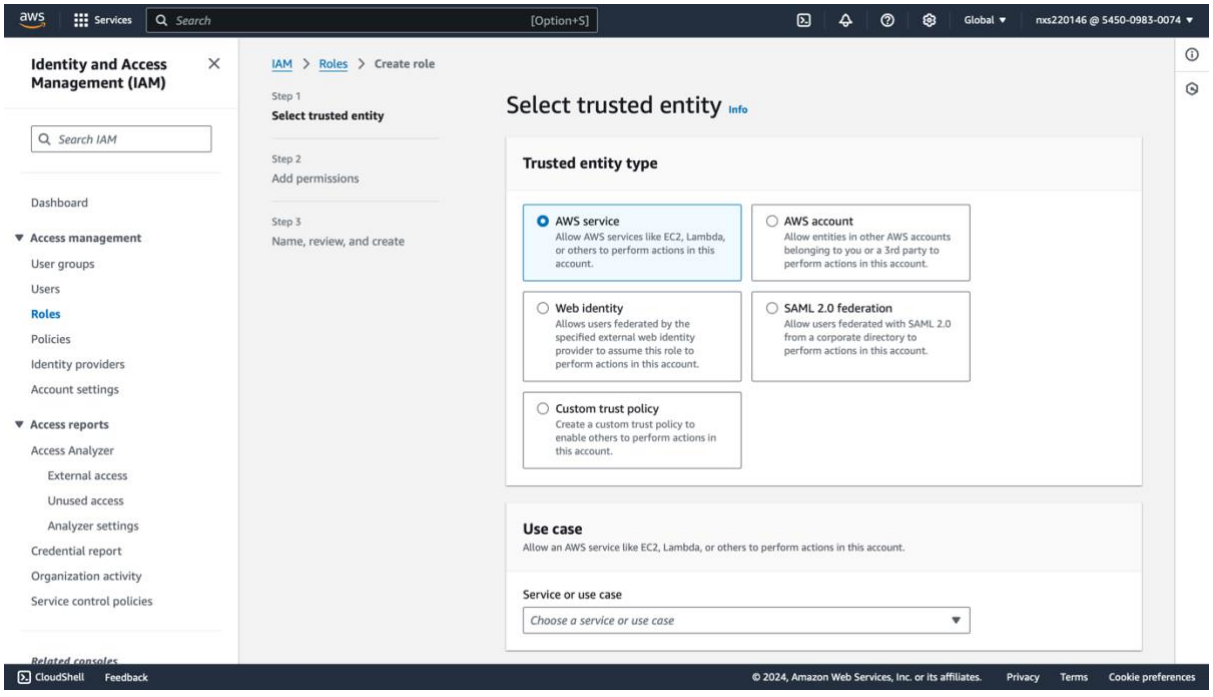
Open the **Rides** table in the DynamoDB console. Locate the **Amazon Resource Name (ARN)** for the table, copy it, and paste it into your **Notes** or a secure text editor for future reference.



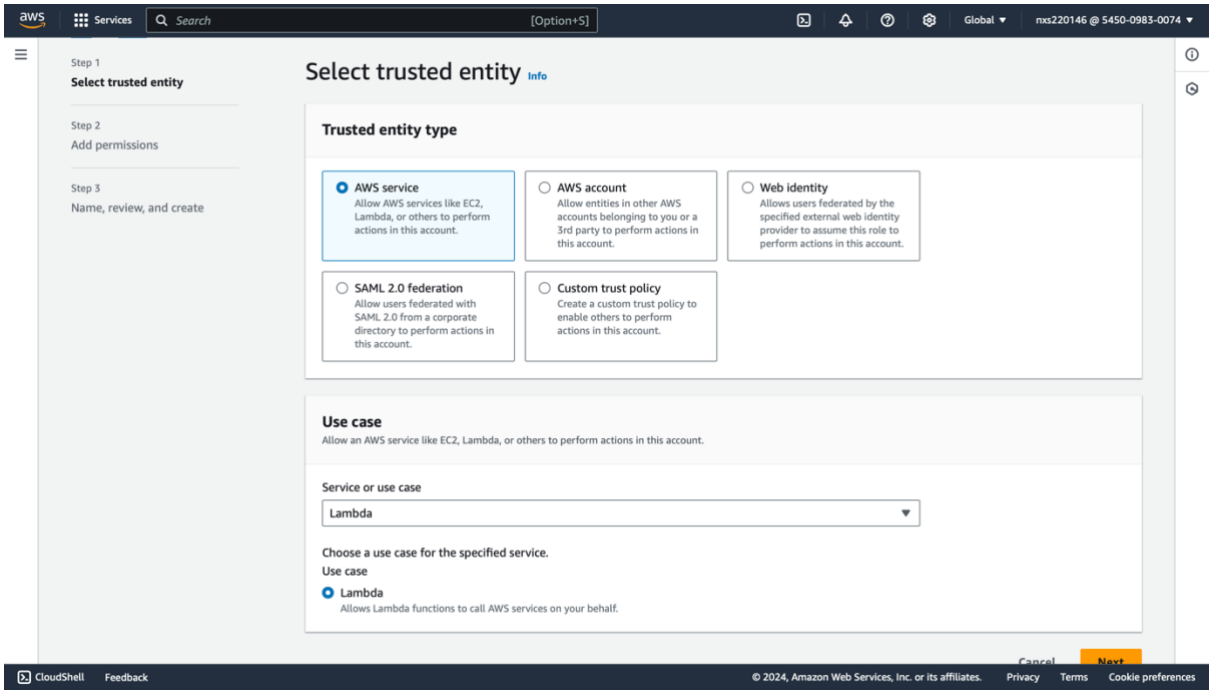
Note: The above screenshot is a deliverable

Creating an IAM Role for Lambda Execution with DynamoDB PutItem Permissions

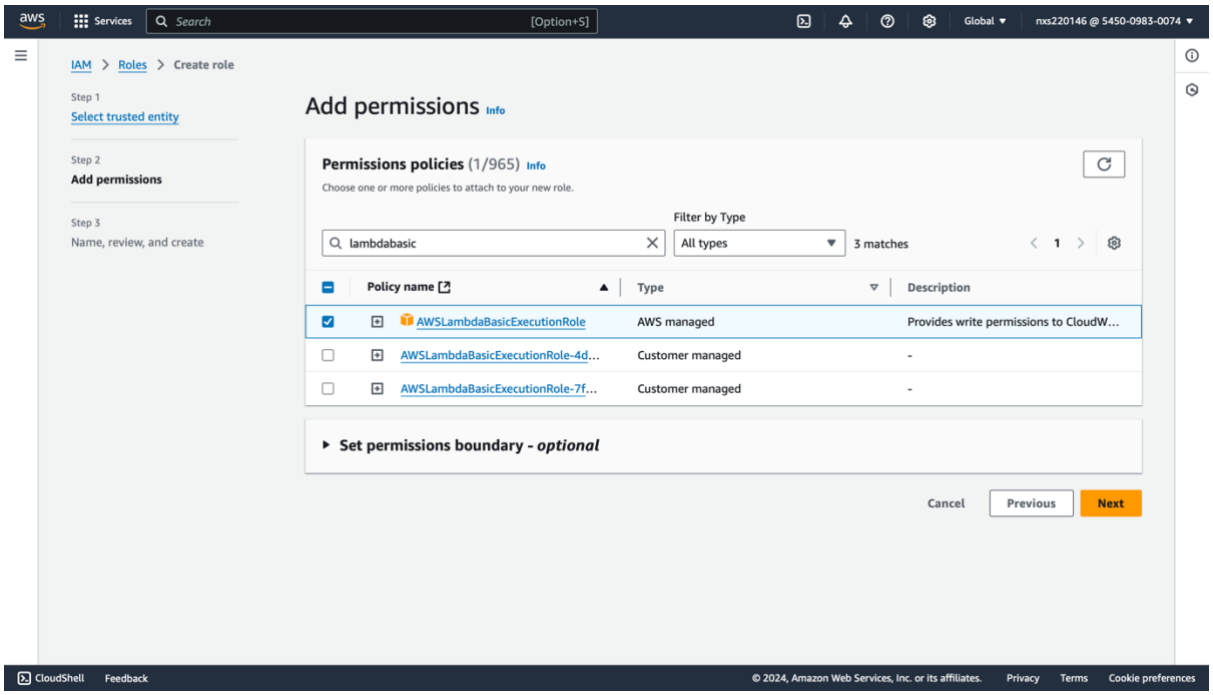
In the AWS Management Console, navigate to **IAM** and click on **Create Role** to start setting up a new role.



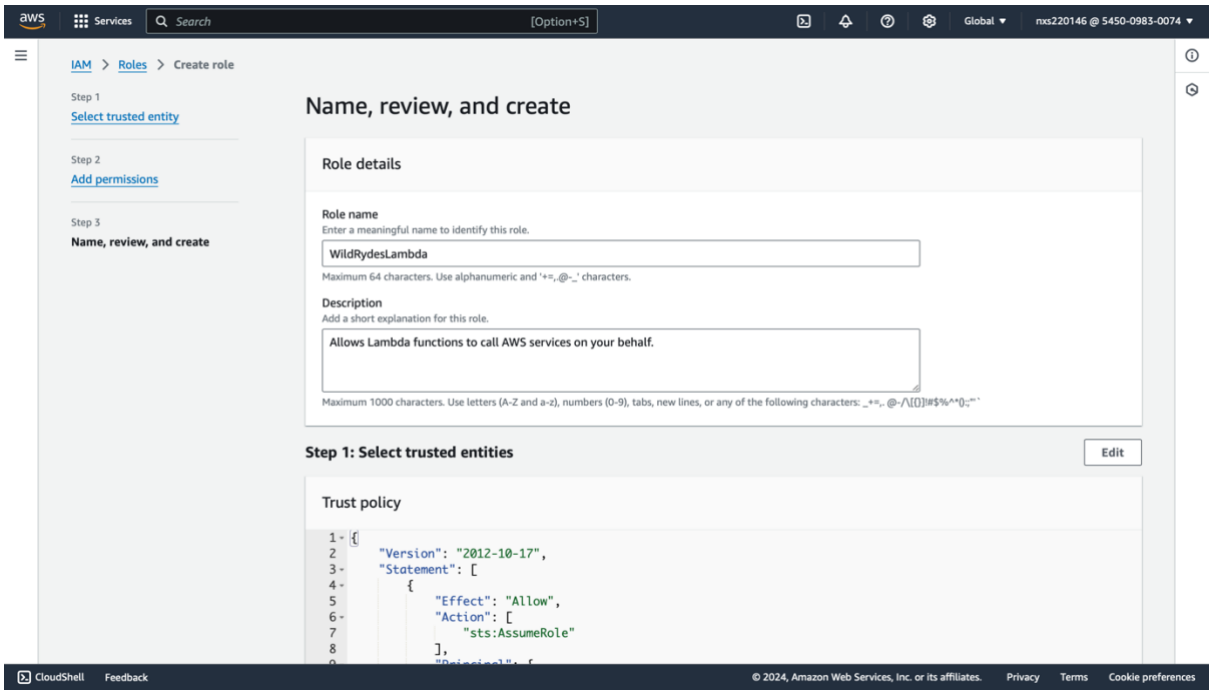
For the **use case**, select **Lambda** as the trusted entity and click **Next** to proceed.



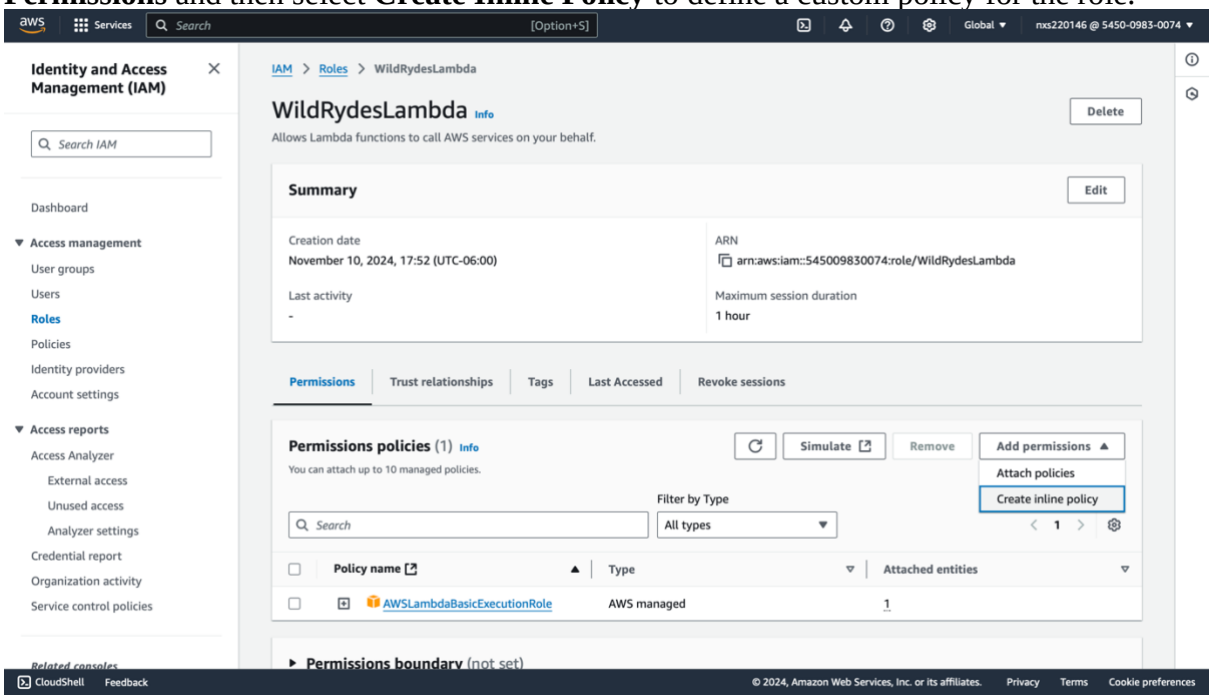
In the **Permissions policies** section, search for **AWSLambdaBasicExecutionRole**. Select it, and then click **Next** to continue.



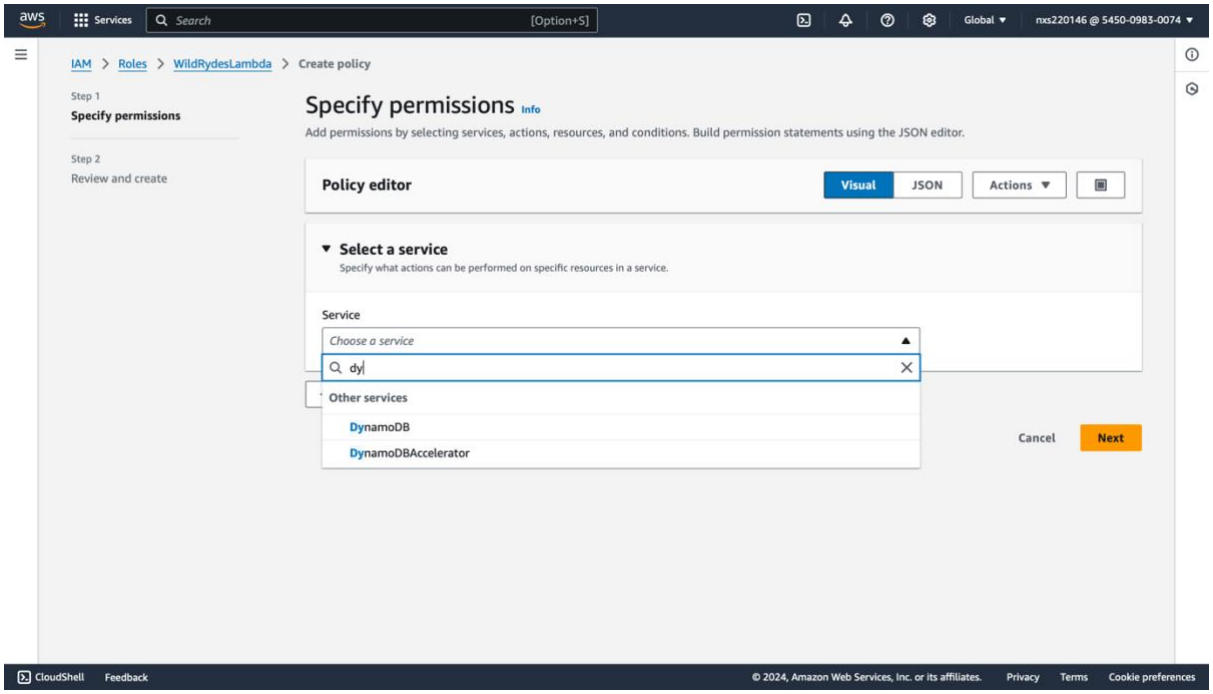
In the **Role Name** field, enter **WildRydesLambda**, and then click **Create Role** to finalize the setup.



Open the newly created **WildRydesLambda** role. Under the **Permissions** tab, click on **Add Permissions** and then select **Create Inline Policy** to define a custom policy for the role.

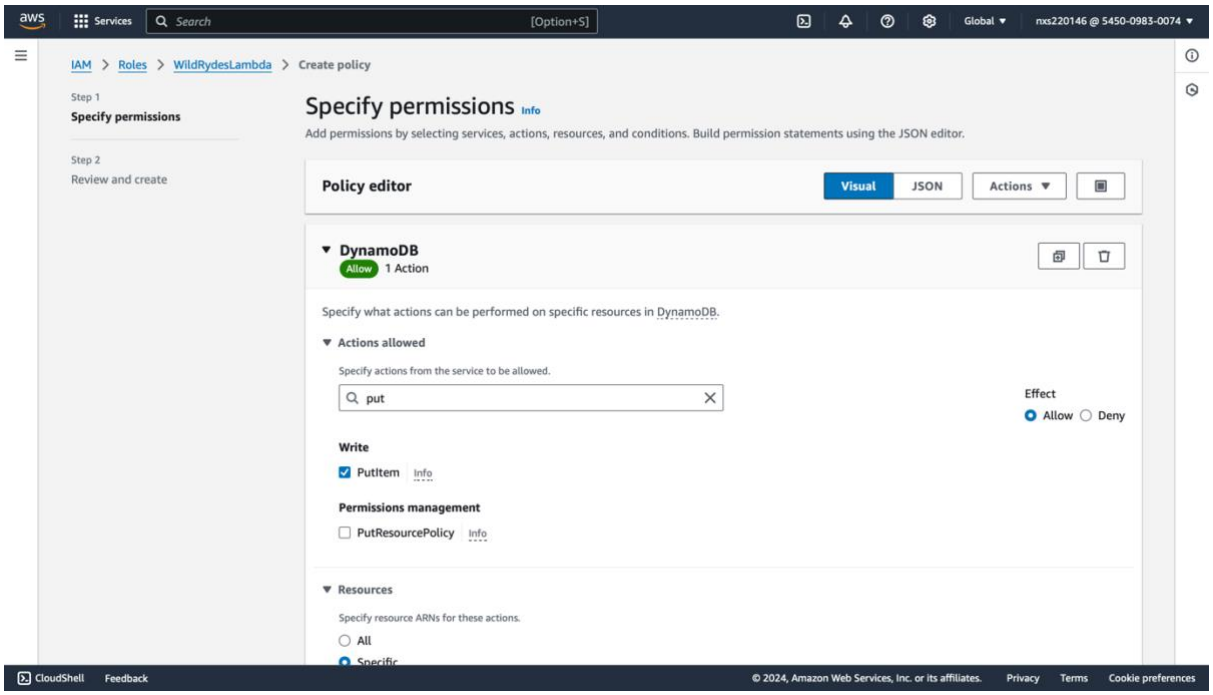


In the **Create Policy** screen, under the **Service** section, search for and select **DynamoDB** as the service.

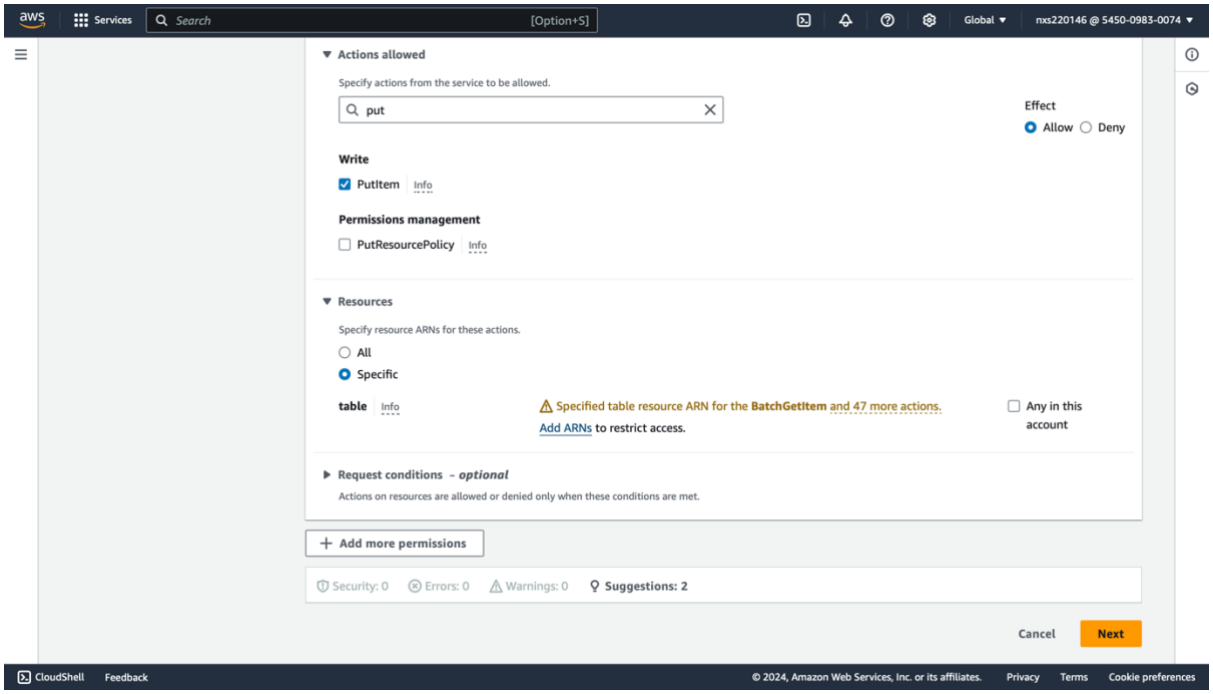


In the **Actions** section, search for **PutItem** and select it.

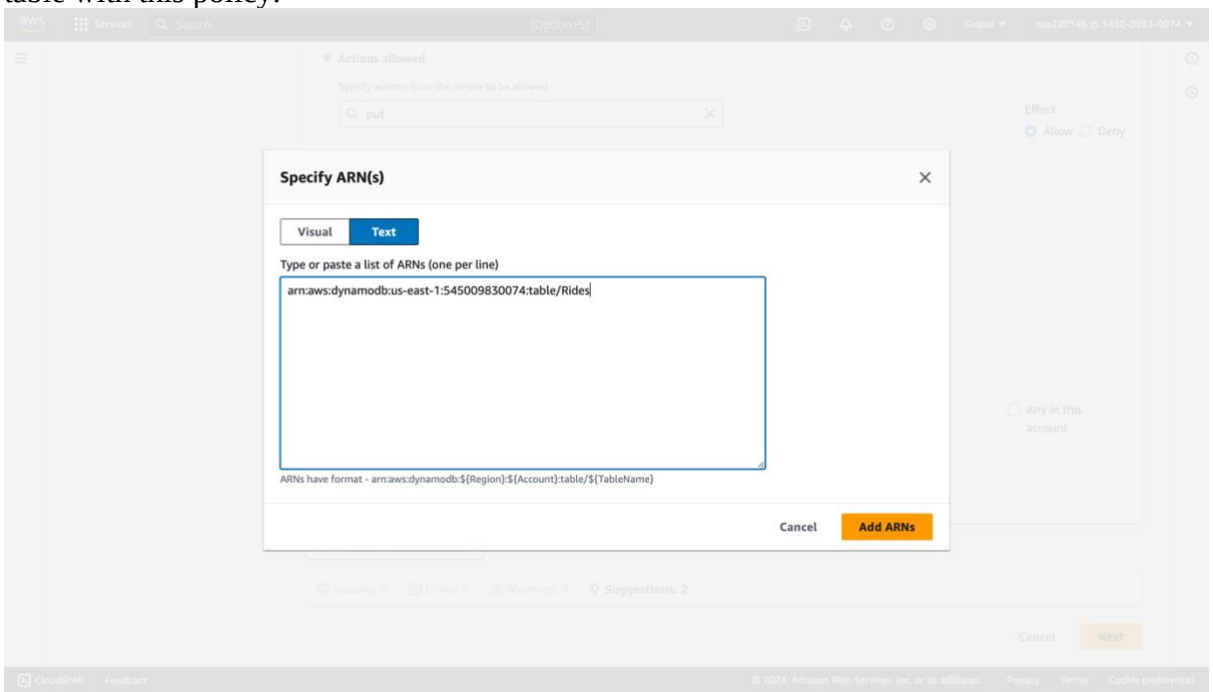
Under the **Resources** section, choose **Specific** to specify the DynamoDB table this policy will apply to.



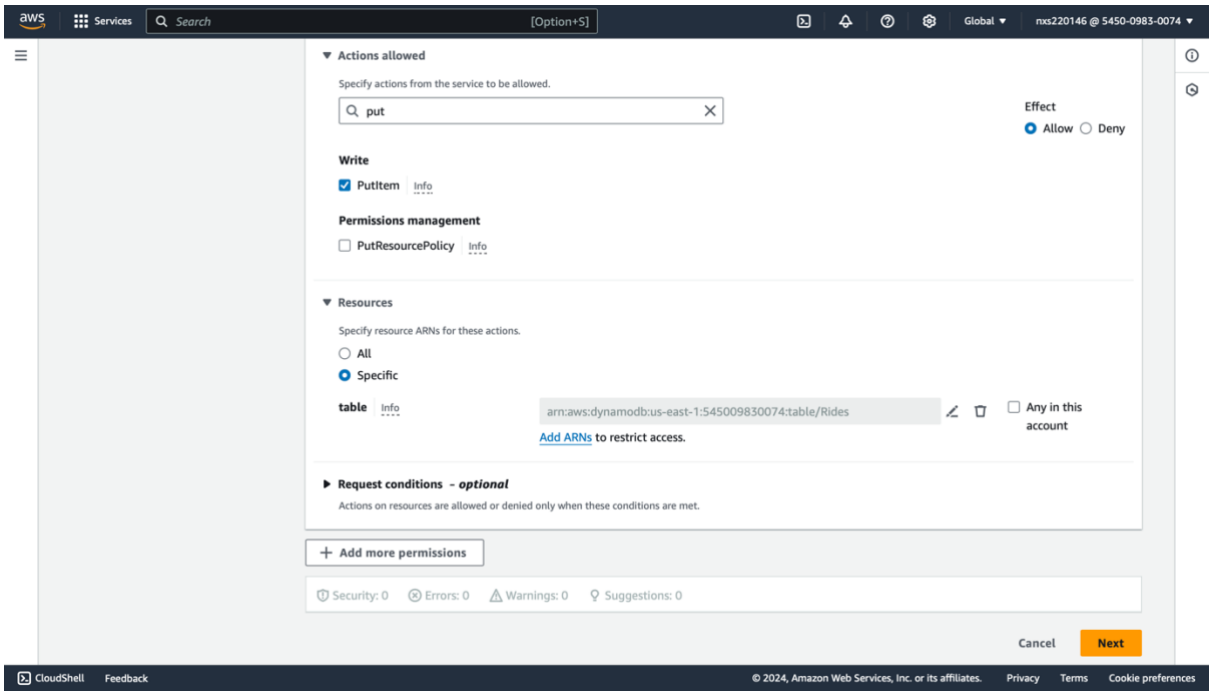
Click on **Add ARN** under the **Resources** section. This ensures the Lambda function will have permission to perform the **PutItem** operation only on the specific DynamoDB table.



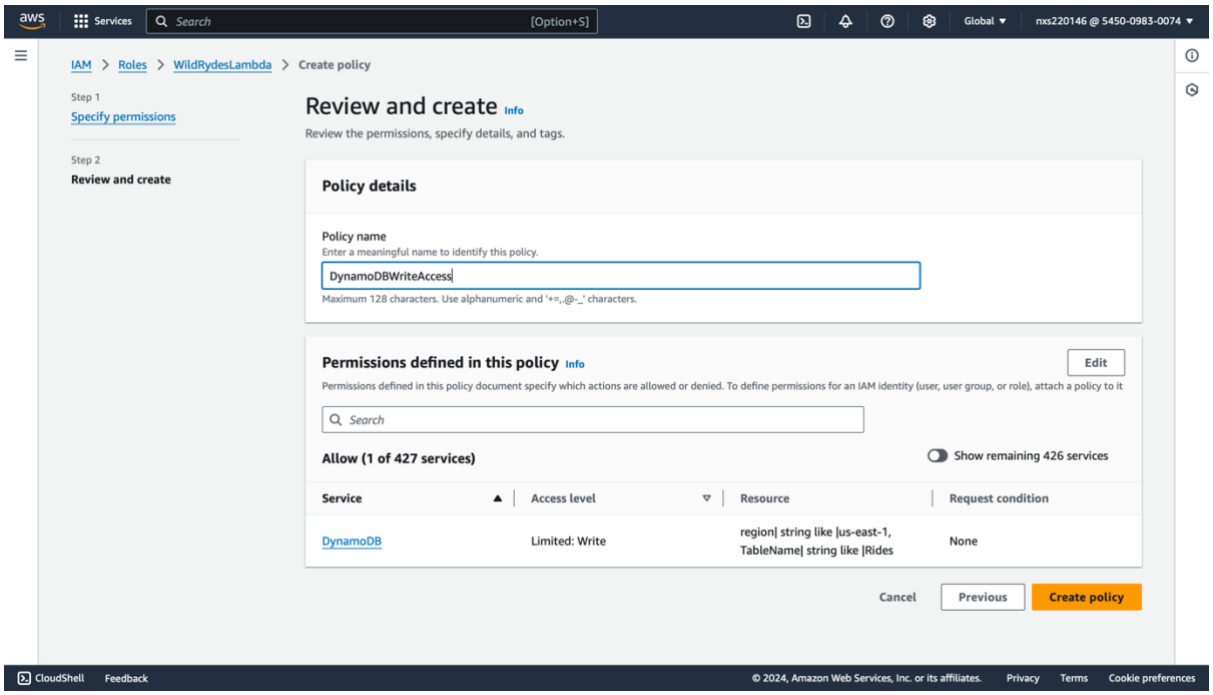
Switch to the **Text** view, and paste the **ARN** you previously copied from the DynamoDB **Rides** table into the specified field. Once done, click **Add ARN** to associate the table with this policy.



After adding the ARN, click **Next** to proceed to the review screen.



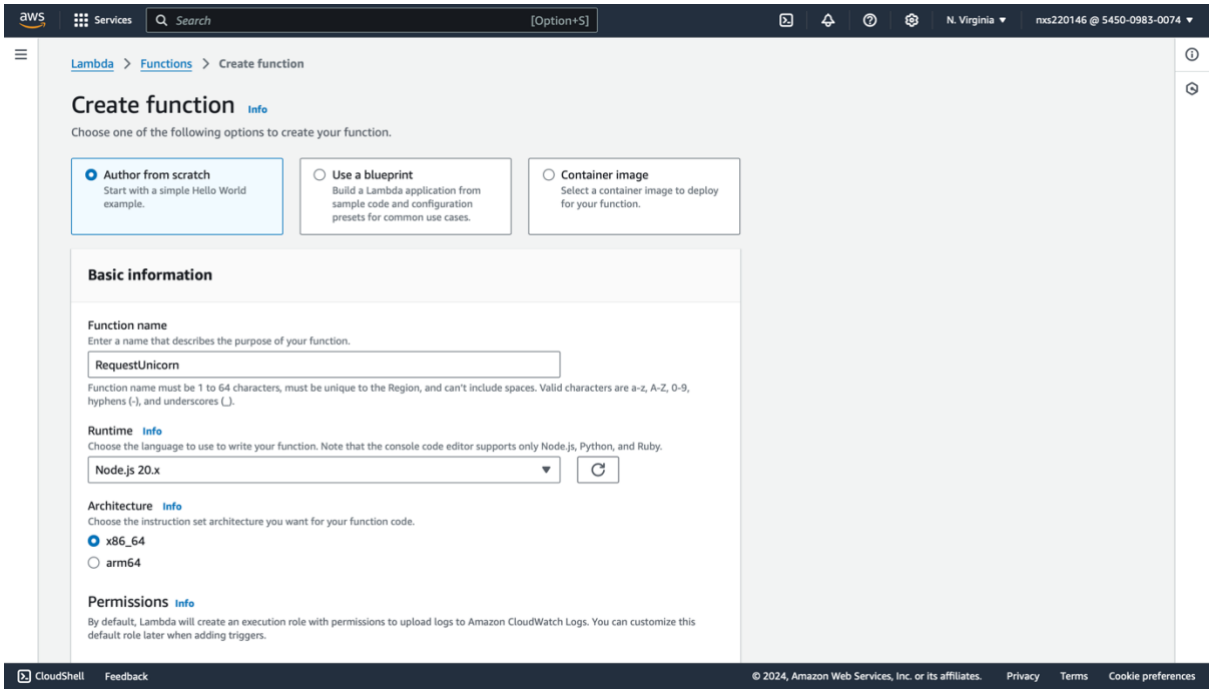
On the review screen, enter **DynamoDBWriteAccess** as the policy name. Then, click **Create Policy** to finalize and attach the policy to the role.



iv. Creating a New Lambda Function for Unicorn Selection and DynamoDB Integration

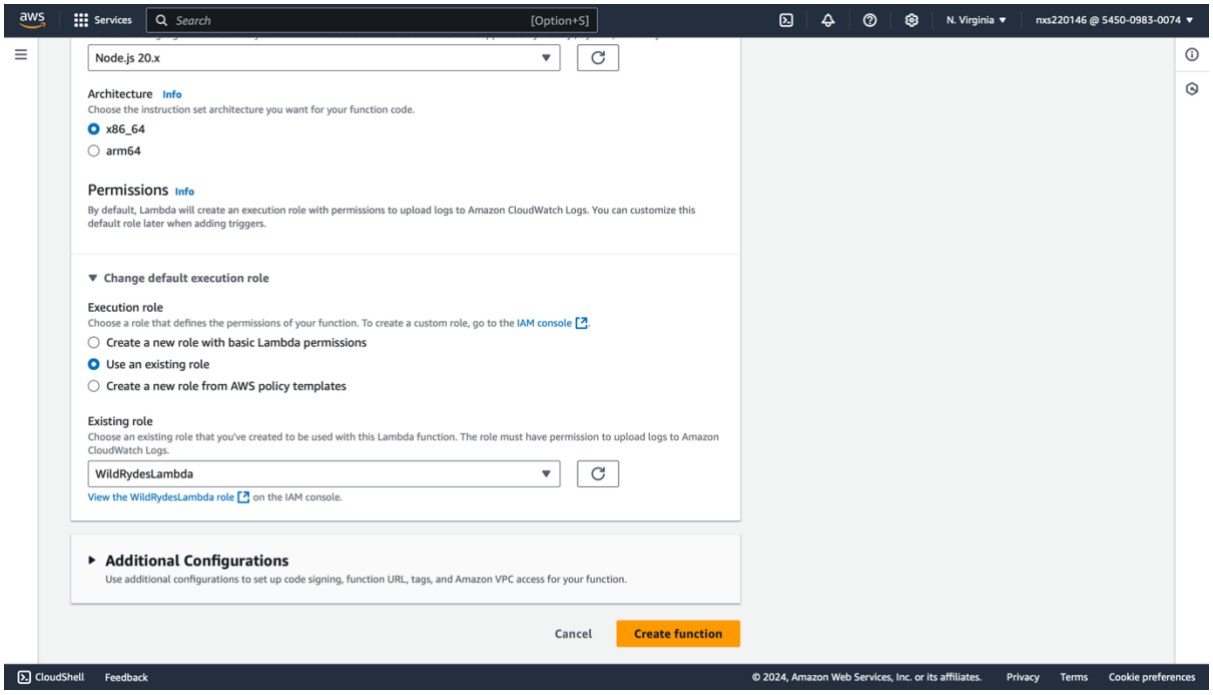
In the AWS Management Console, search for **Lambda** and select it. Click **Create Function** and follow these steps:

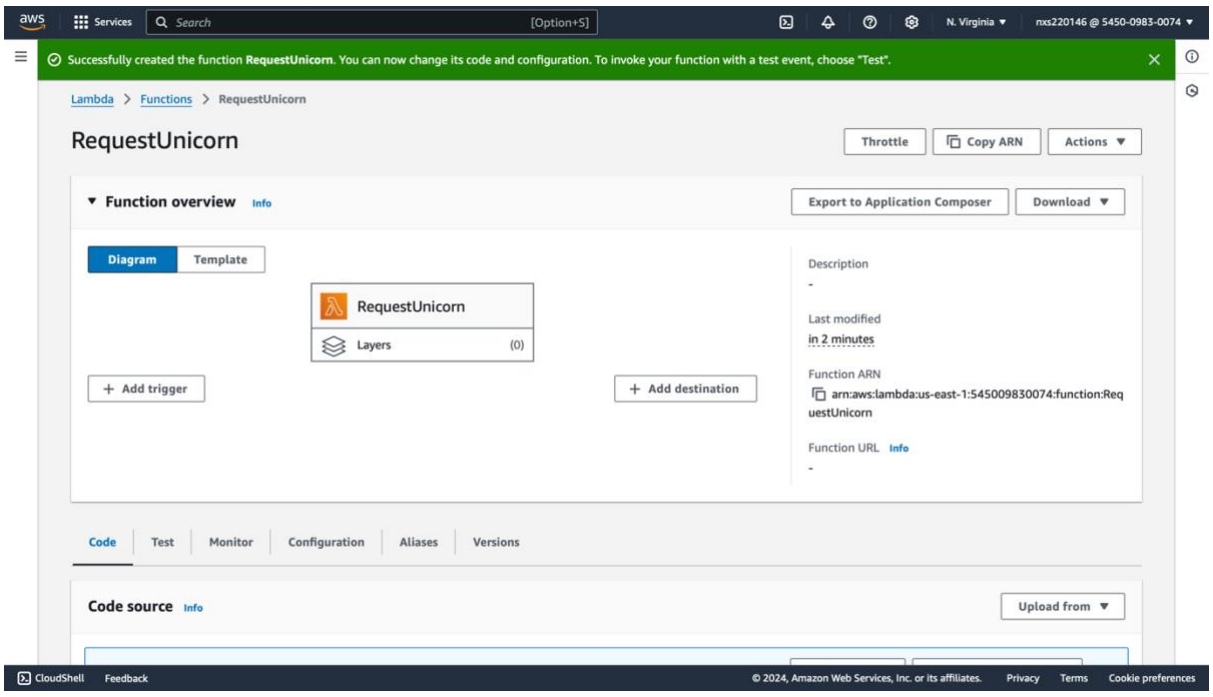
1. Choose **Author from scratch**.
2. Enter the **Function Name** as **RequestUnicorn**.
3. Select **Runtime** as **Node.js 20.x**.



Expand the **Change default execution role** section.

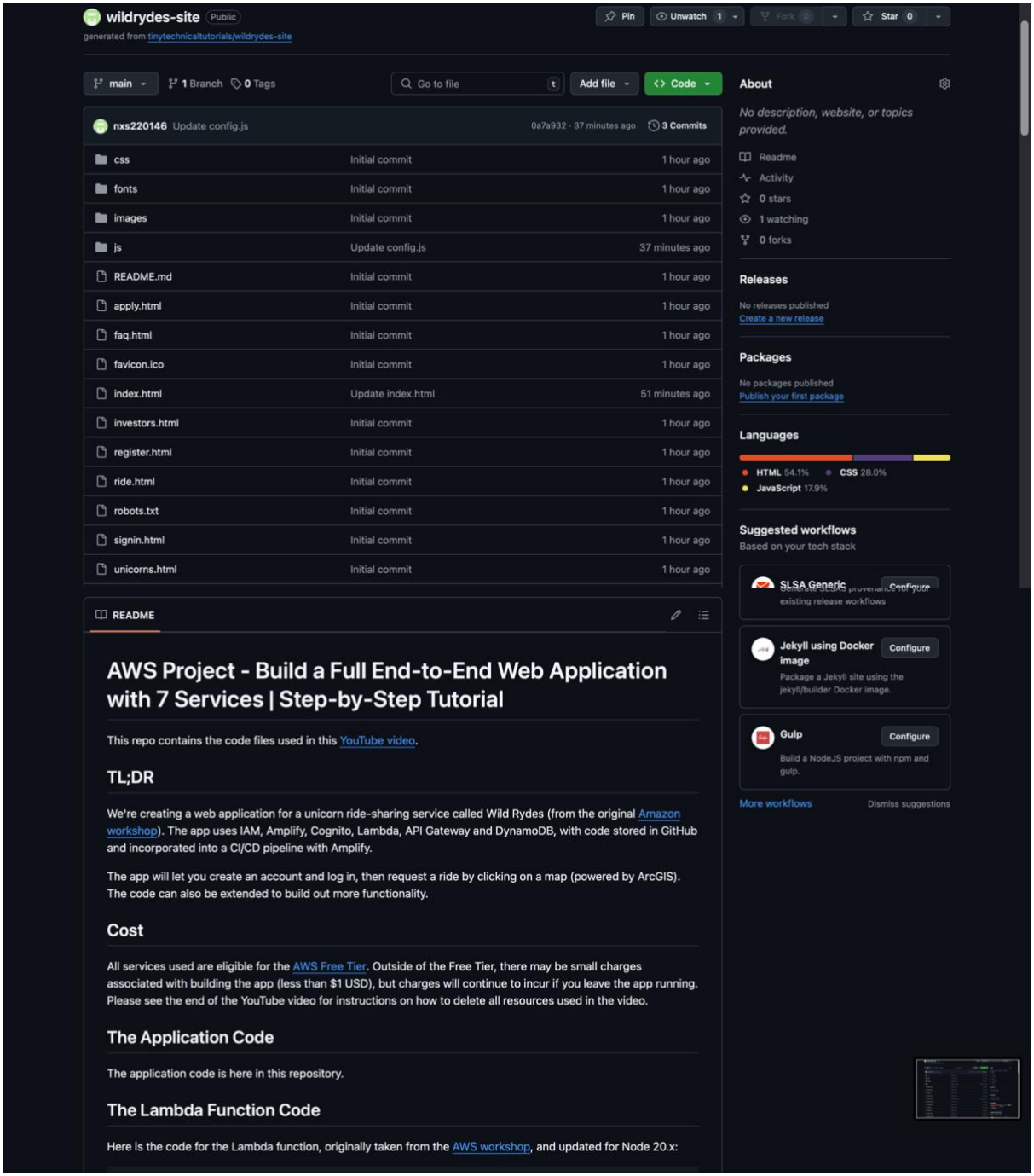
1. Choose the **Use an existing role** option.
2. From the dropdown, select the **WildRydesLambda** IAM role you created earlier.
3. Click **Create Function** to finalize the setup.





To add the Lambda function code:

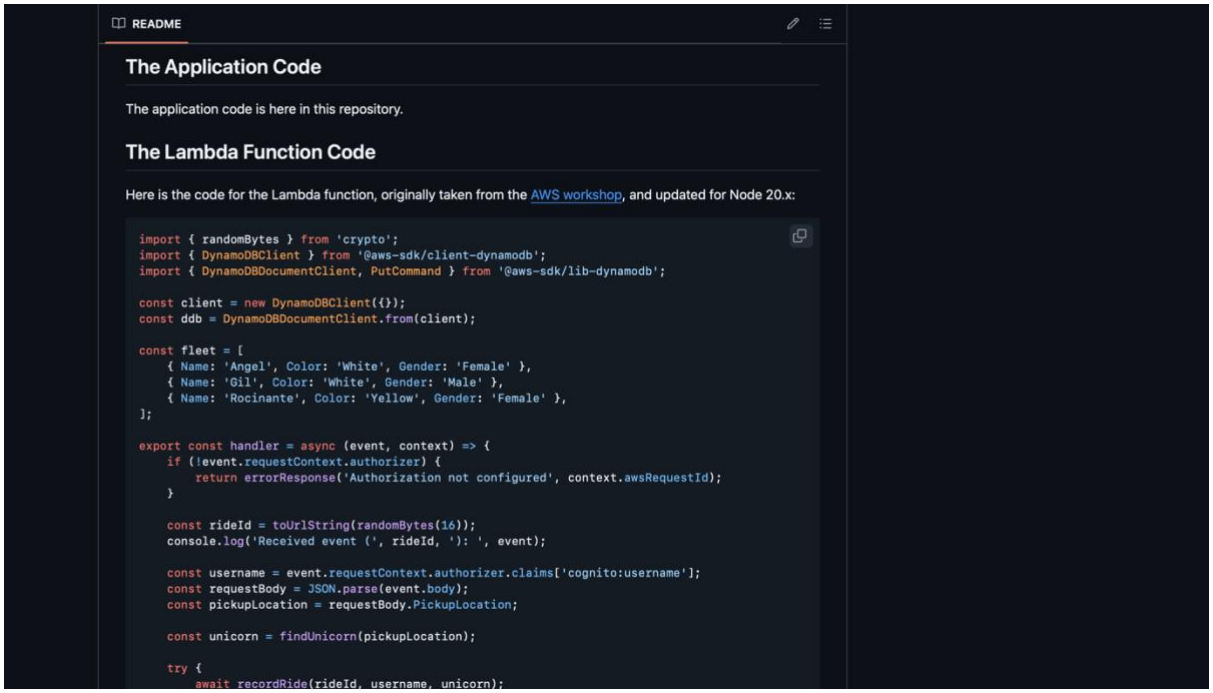
Go to your **GitHub repository** and scroll down to the **README** section.



On the READ me, you will find the lambda function code.

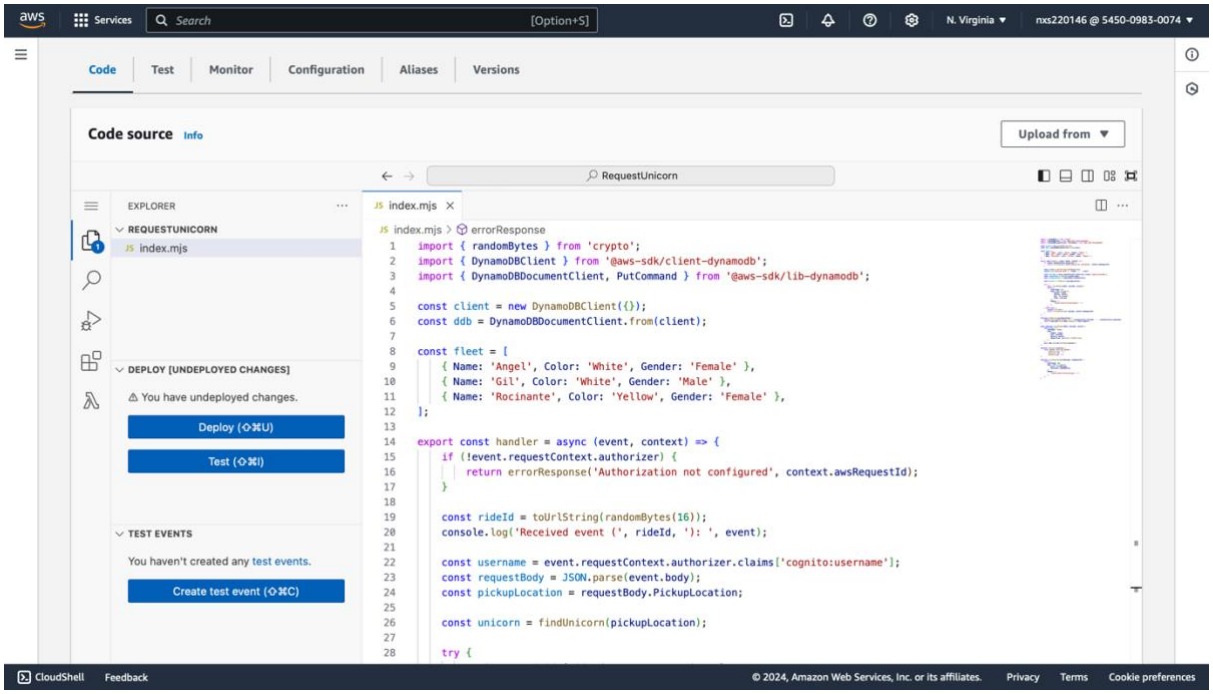
Locate the Lambda function code provided in the README.

Click on the **Copy** button next to the code to copy it to your clipboard.



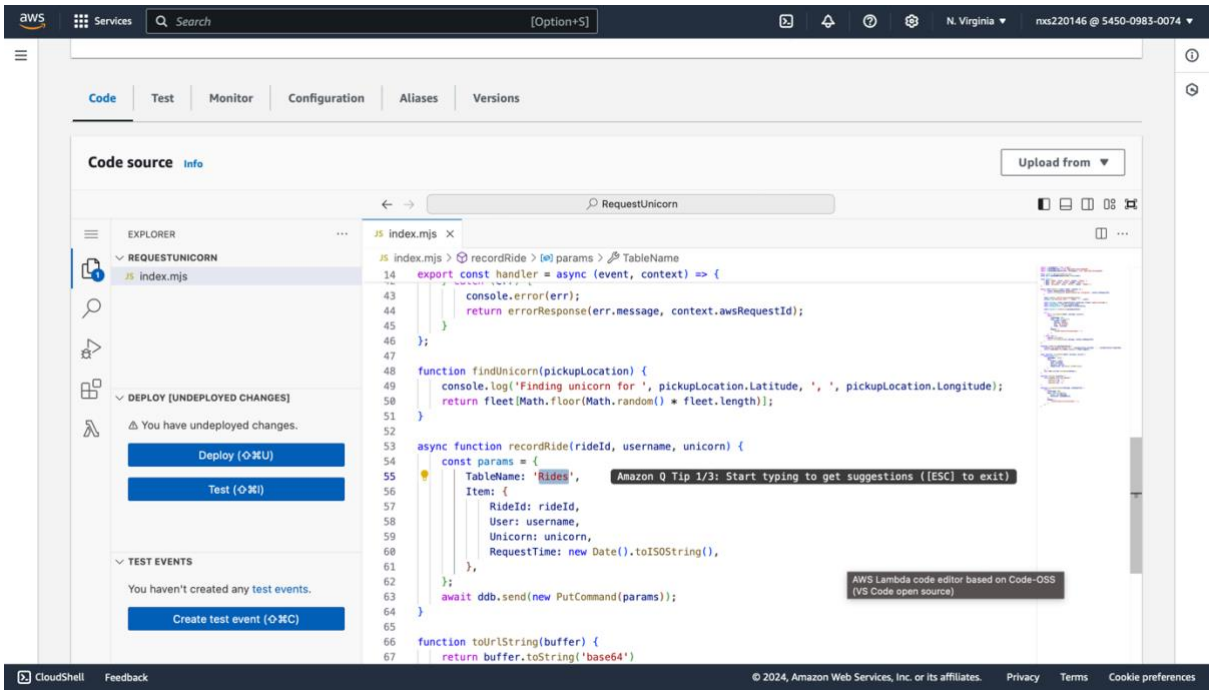
In the AWS Management Console, navigate to your **RequestUnicorn** Lambda function:

1. Open the **Code** tab.
2. Select the existing code and delete it.
3. Paste the new code " **The Lambda Function Code**" you copied from GitHub into the code editor.



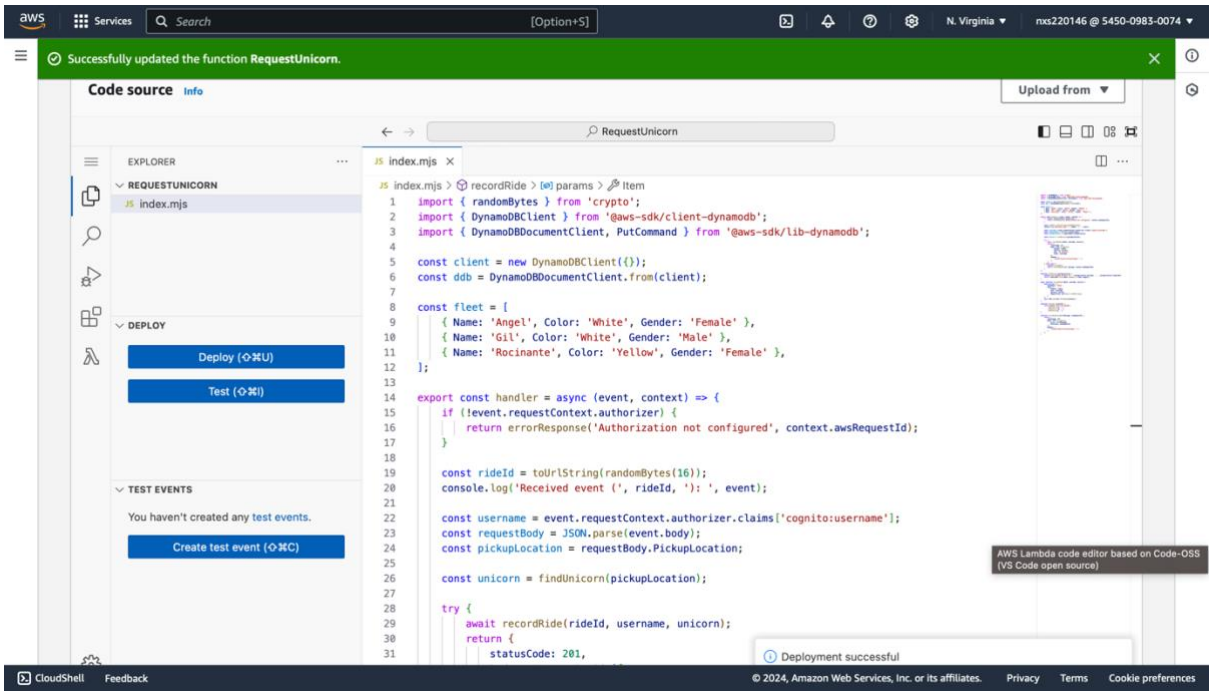
Note: In the Lambda code, locate **line 55** where the DynamoDB table name is defined. If you named your table something other than **Rides**, update the table name in the code to match your table's name exactly.

Ensure the name is accurate and case-sensitive, as a mismatch will prevent the function from working correctly.



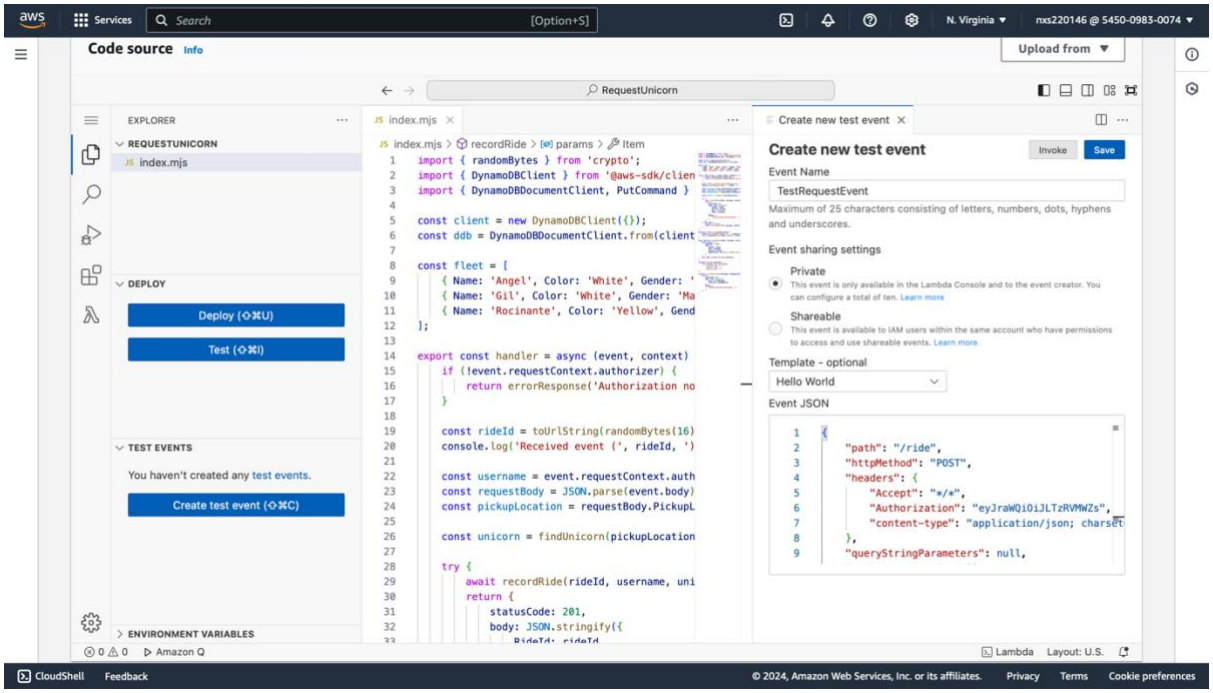
Note: The above screenshot is a deliverable

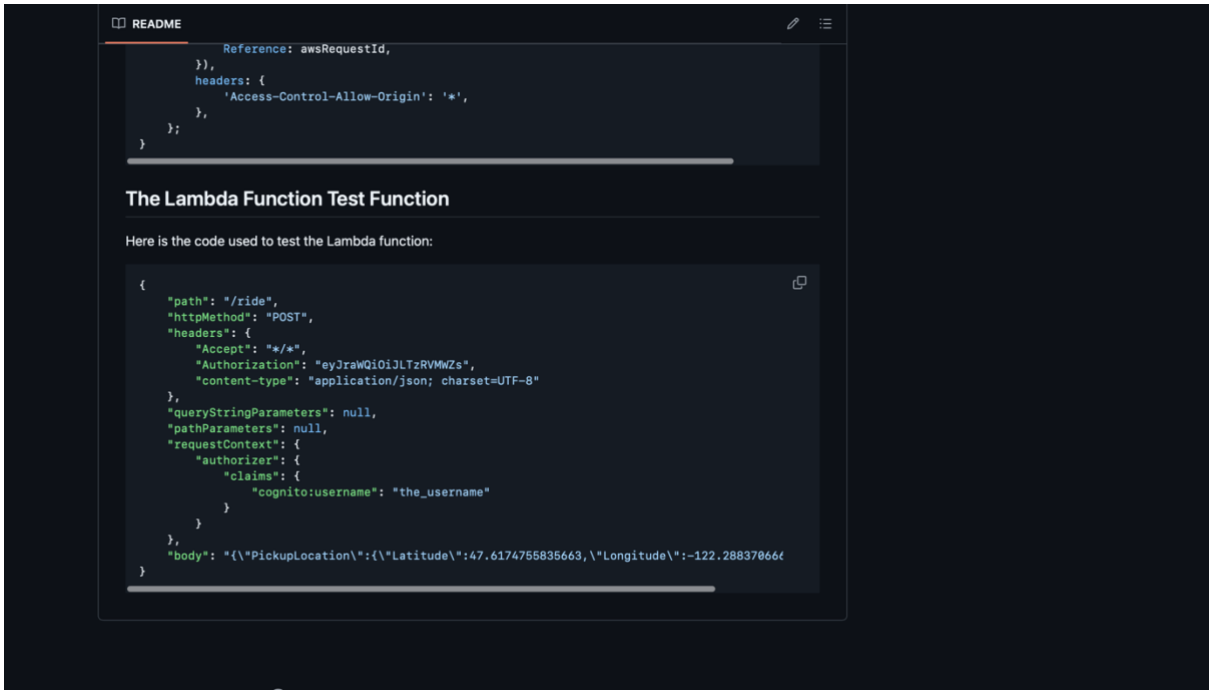
Click **Deploy** to save and apply the changes



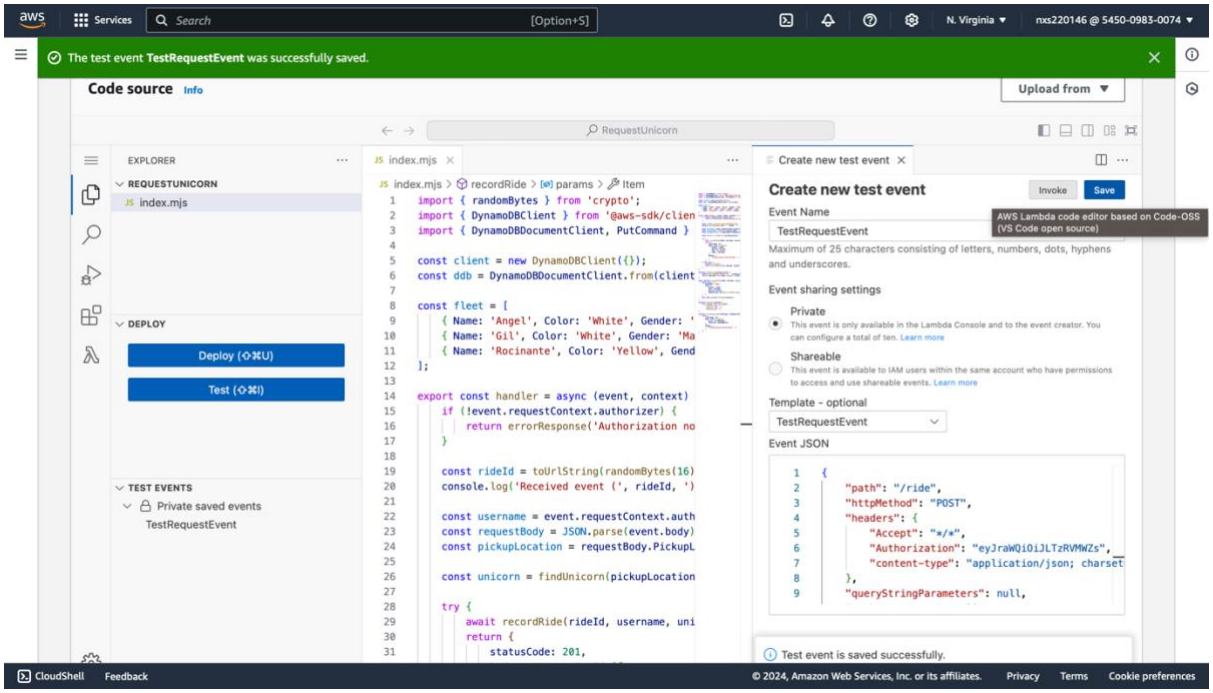
To create a test event and verify the Lambda function:

1. In the **RequestUnicorn** Lambda function console, click on **Create Test Event**.
2. Enter the name of the test event as **TestRequestEvent**.
3. Go to your **GitHub repository** and scroll down to the **README** section.
4. Locate the test event JSON "**The Lambda Function Test Function**" for the Lambda function.
5. Click the **Copy** button to copy the test event code.
6. Replace the existing JSON code in the test event editor with the copied code.

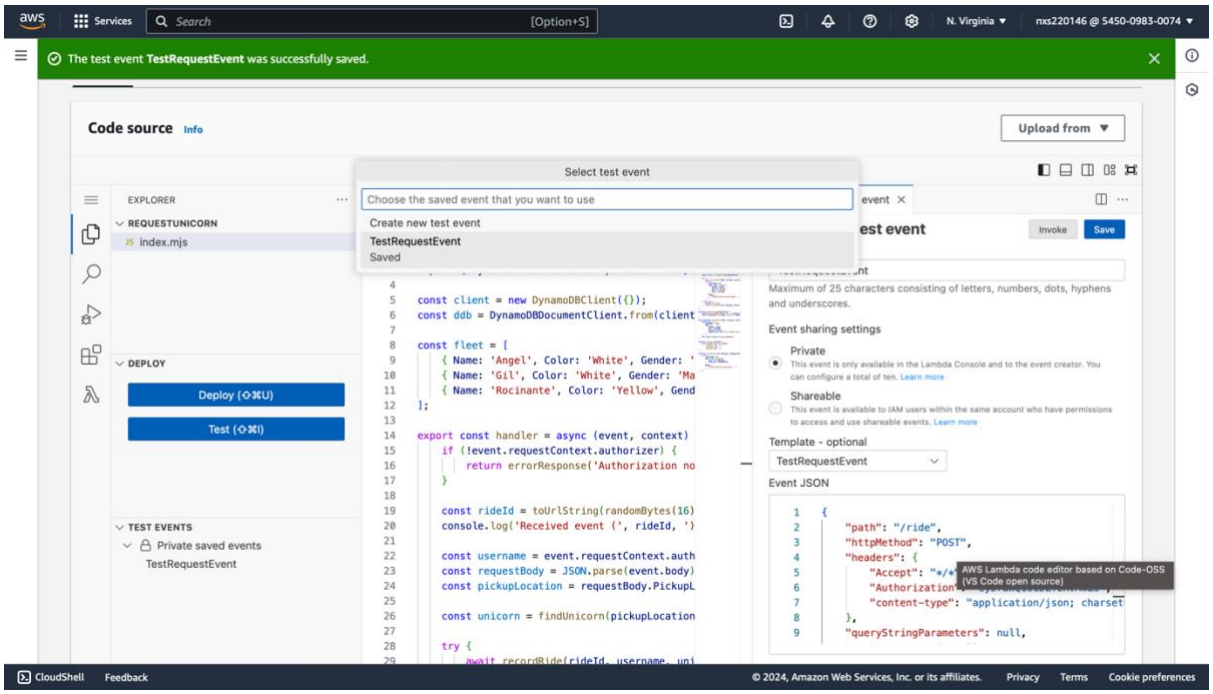




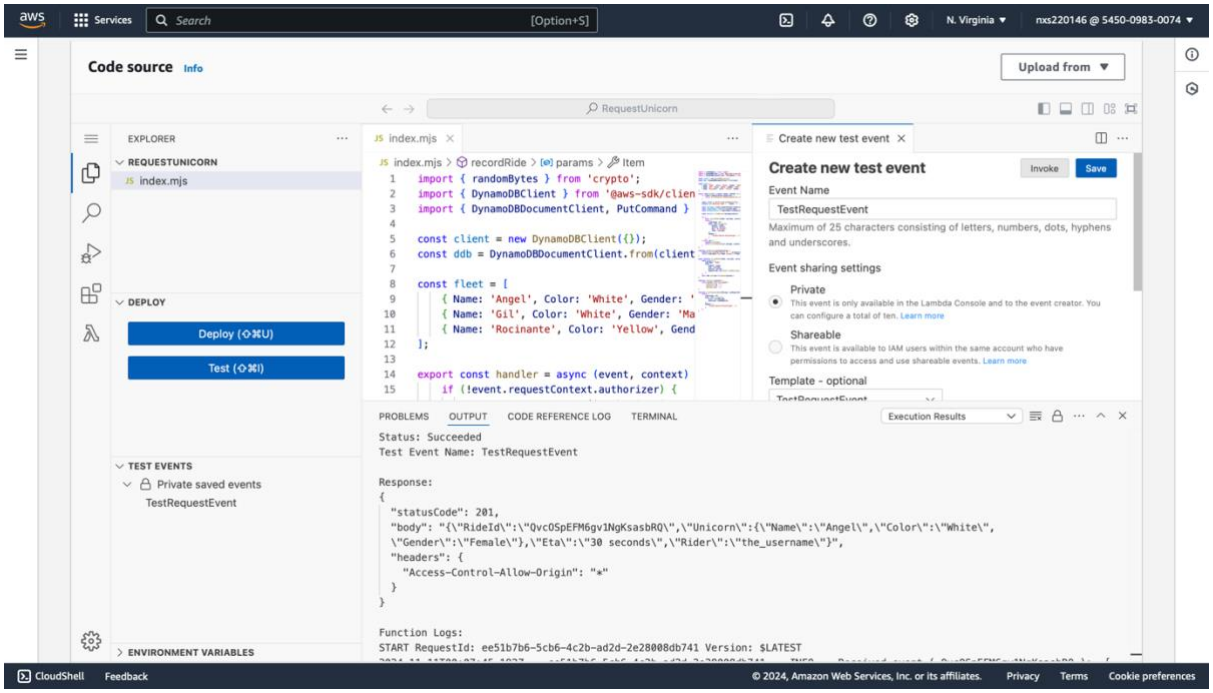
Click **Save**. Then click on **Test**.



Select **TestRequestEvent**.



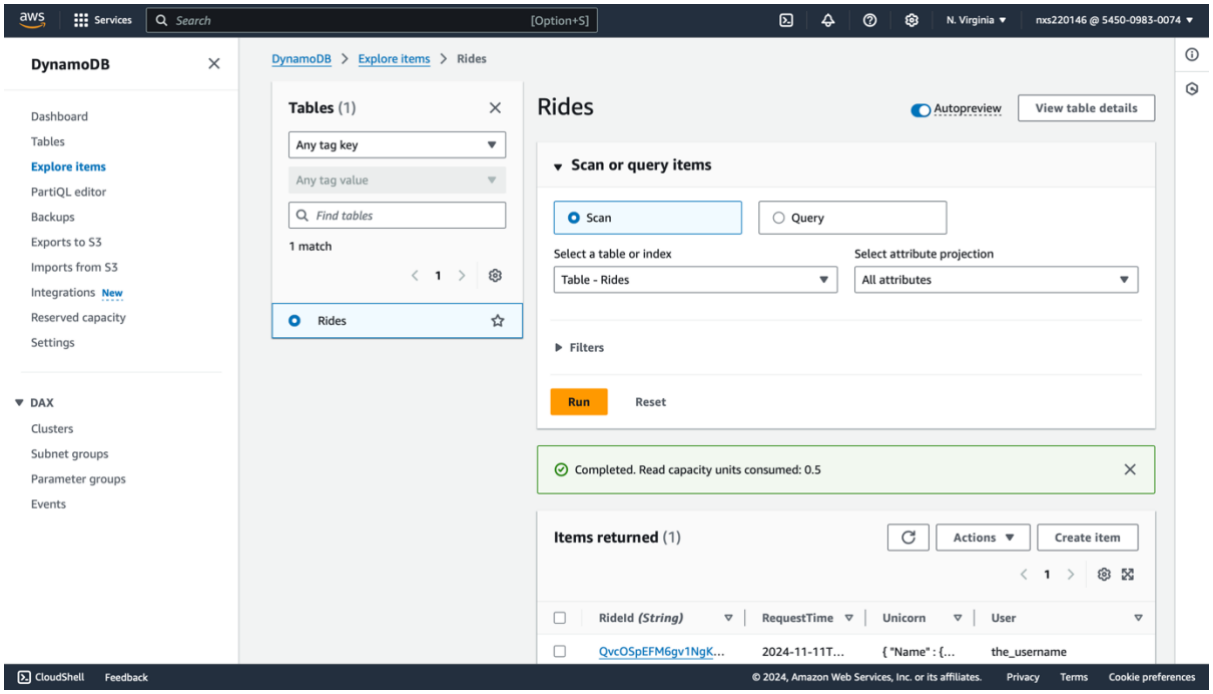
After running the test event, the **Execution results** section should display a **Succeeded** status. This confirms that your Lambda function executed successfully.

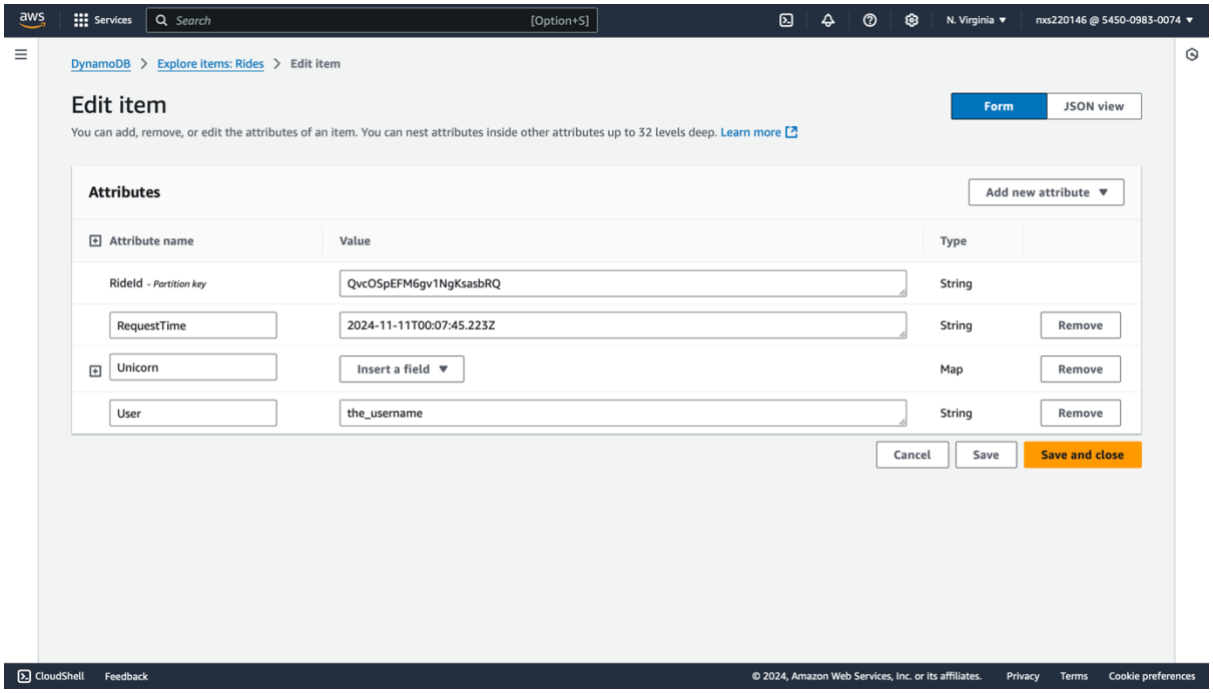


To verify the Lambda function's output:

1. Go to your **DynamoDB** console.
2. Open the **Rides** table (or the table you created).
3. Check the **Items** tab to see if a new record has been inserted.

This record confirms that the Lambda function successfully wrote data to the DynamoDB table



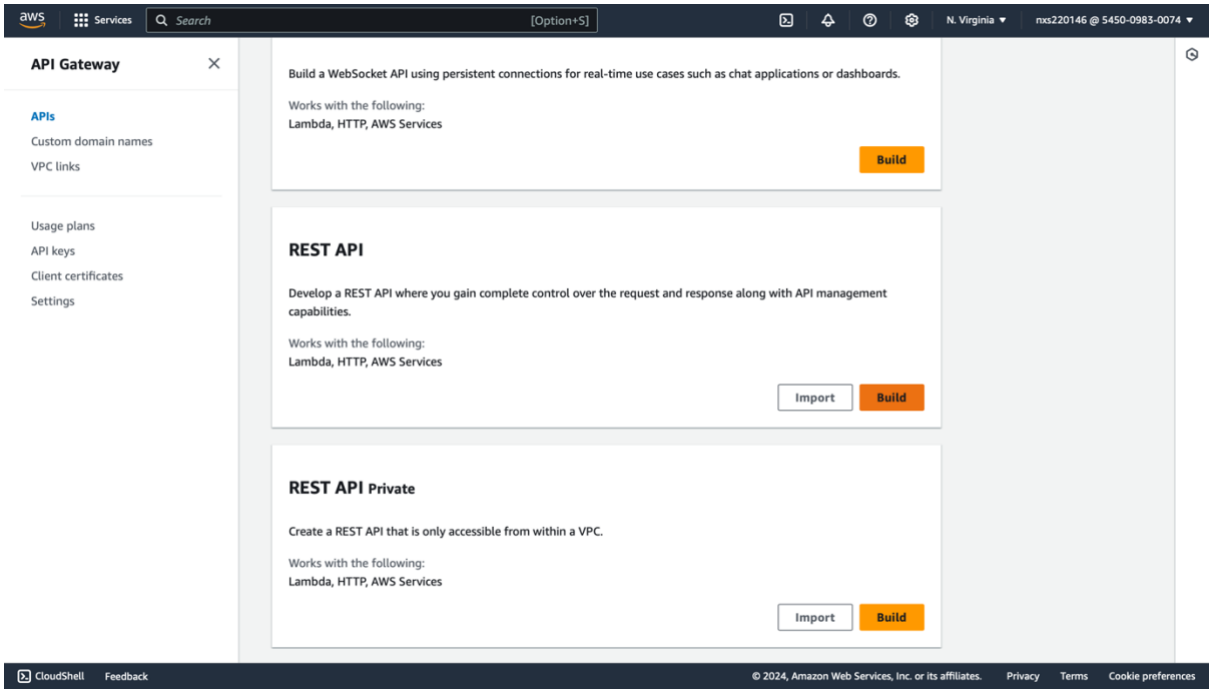


v. Setting Up API Gateway to Invoke the Ride-Sharing Functionality.

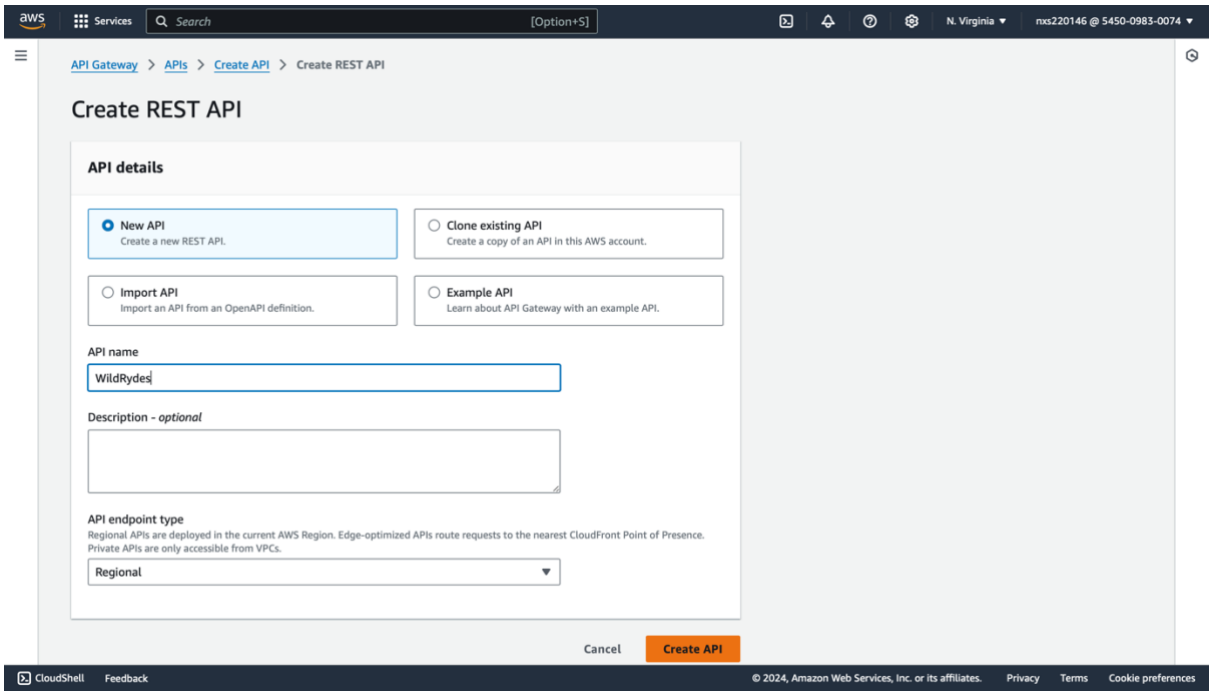
Create a new REST API in **API Gateway** to enable the invocation of your Lambda function for the ride-sharing feature. This will allow the application to trigger the **RequestUnicorn** Lambda function through a secure and scalable API endpoint.

In the AWS Management Console:

1. Search for **API Gateway** and open it.
2. Click on **Create an API**.
3. Select **REST API** from the options to create a RESTful interface for your Lambda function.



In the **Create API** screen: Enter **WildRydes** as the **API Name**. Click **Create API** to proceed.

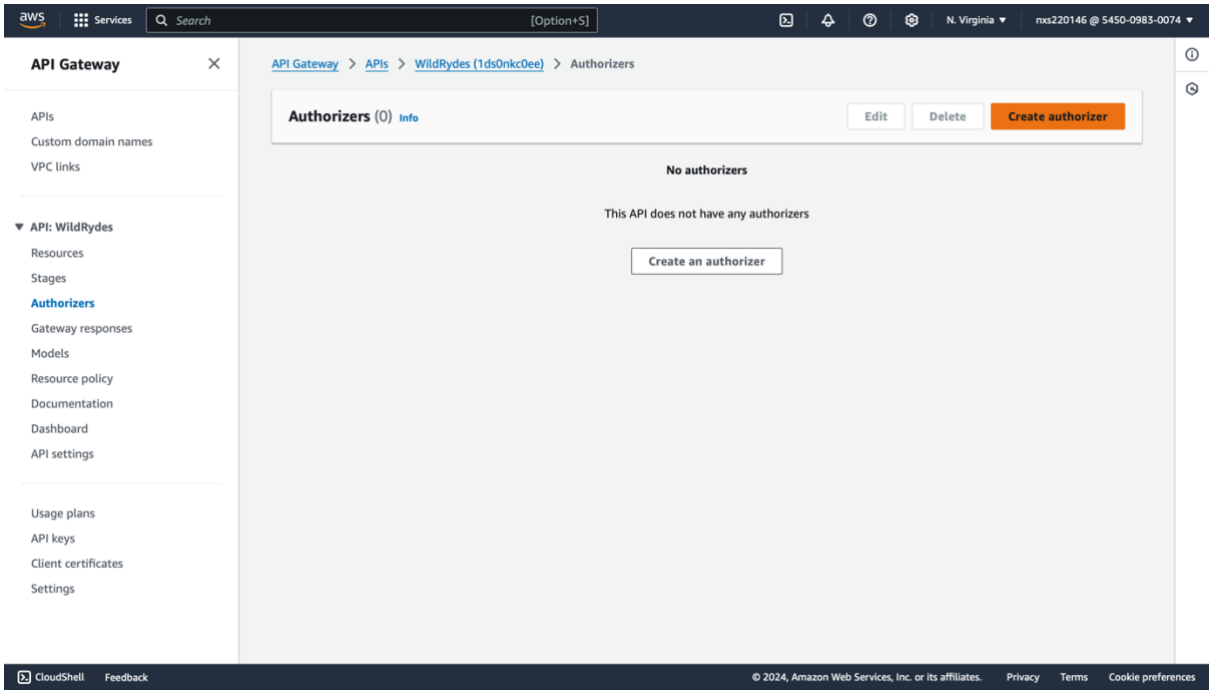


Creating an Authorizer to Enable API Gateway Integration with Cognito

Since we're using a **Cognito User Pool**, we need to create an authorizer in **API Gateway** to authenticate API calls. API Gateway will use **JSON Web Tokens (JWTs)** returned by

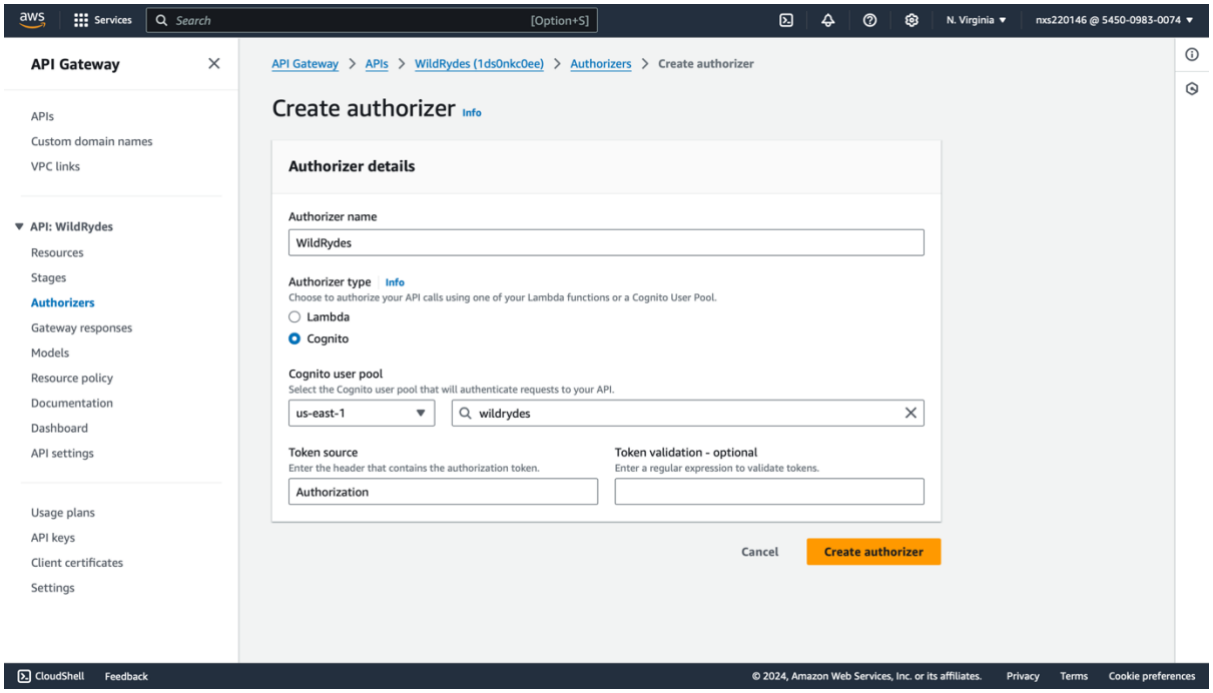
Cognito to verify and authorize requests. This step connects API Gateway with the Cognito User Pool for secure access.

1. In the **API Gateway** console, navigate to your API.
2. On the left pane, click on **Authorizers**.
3. Select **Create Authorizer** to set up the connection.



In the **Create Authorizer** screen:

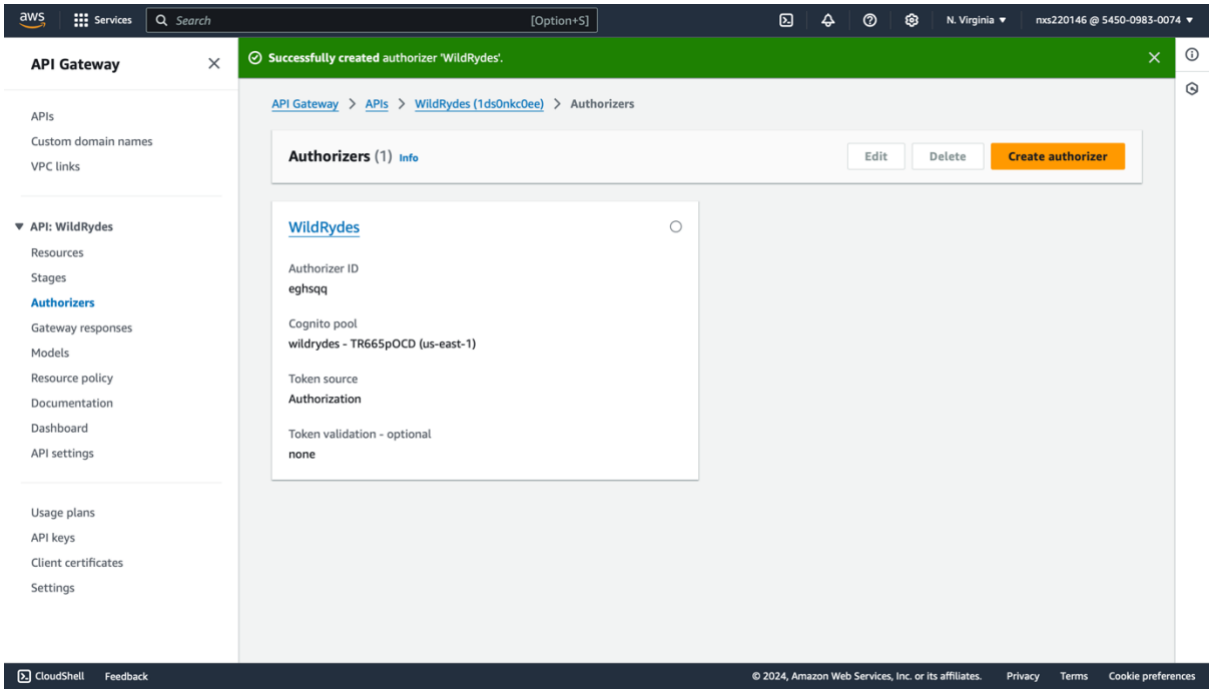
1. Set the **Name** to **WildRydes**.
2. For the **Authorizer Type**, select **Cognito**.
3. Choose your **Cognito User Pool** from the dropdown menu.
4. In the **Token Source** field, enter **Authorization**.
5. Click **Create Authorizer** to finalize the setup.

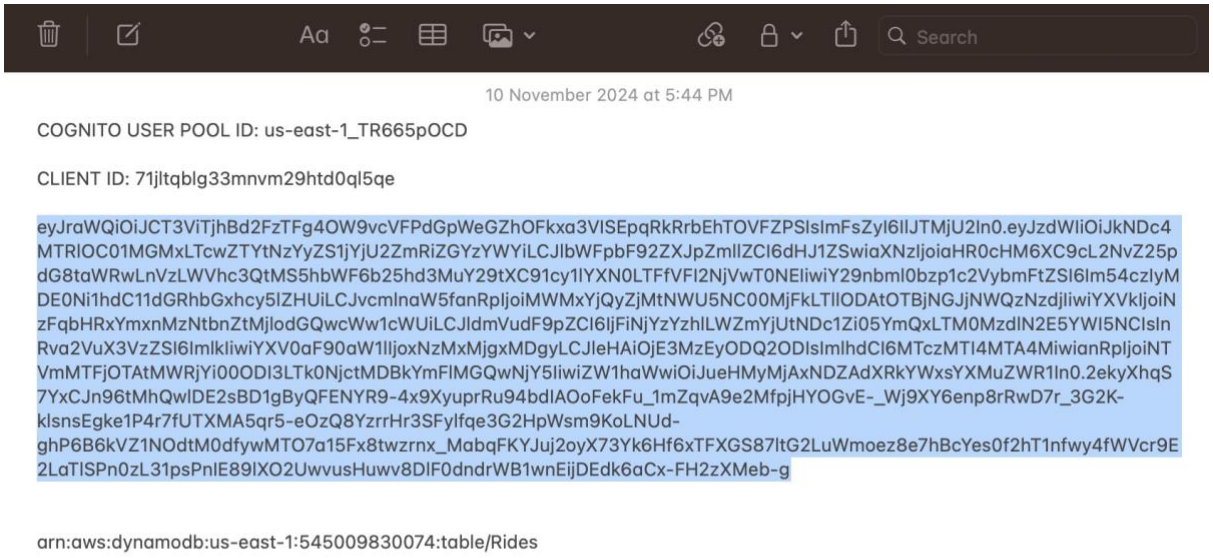


Open the **WildRydes** authorizer you just created.

In the **Token Value** field, paste the **token** you previously saved.

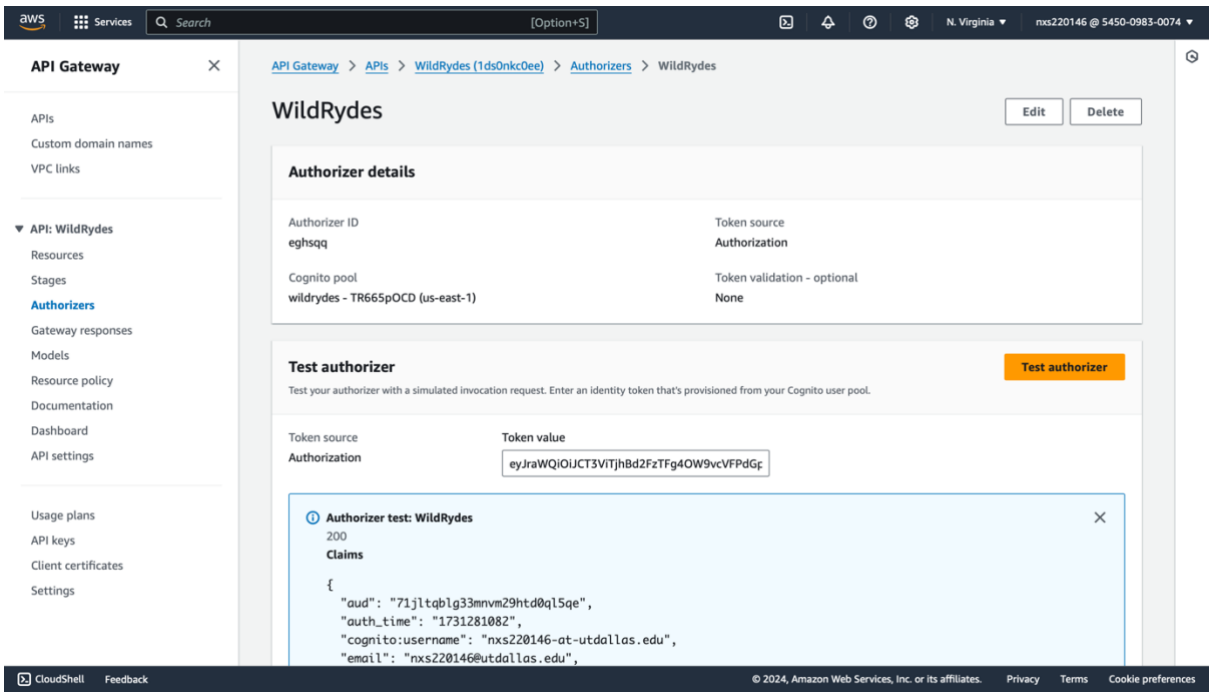
Note: Ensure there are no extra spaces at the beginning or end of the token.





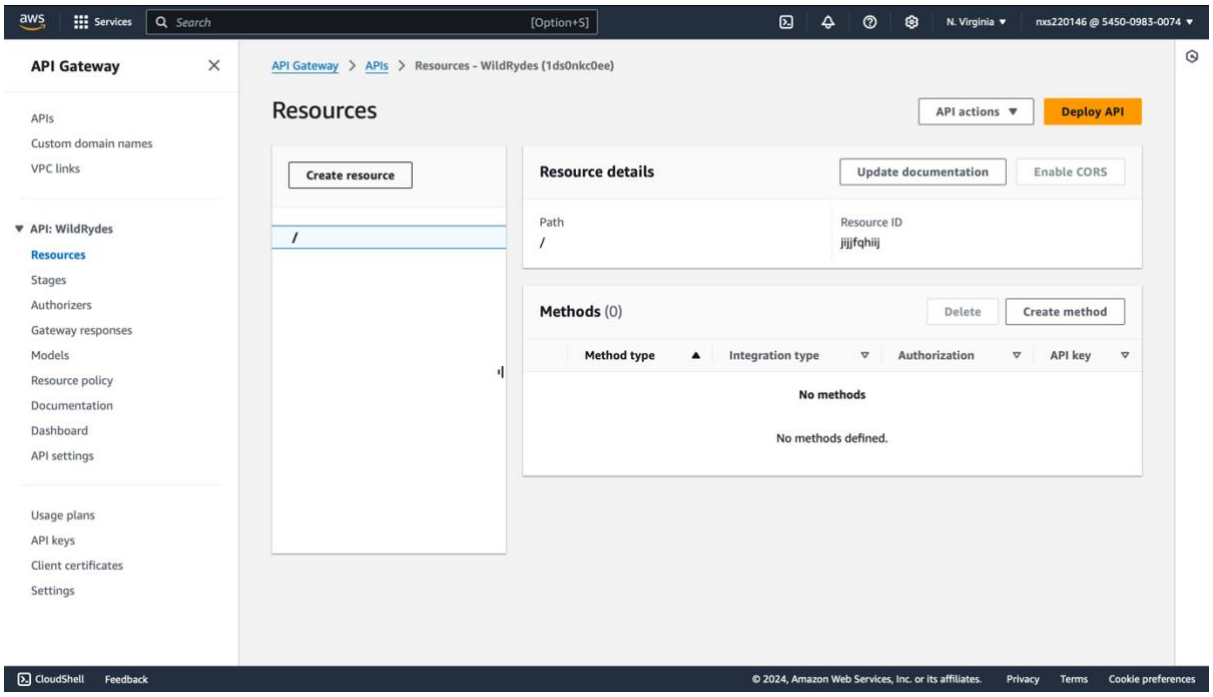
Click on **Test Authorizer** to validate the setup.

If everything is configured correctly, you should receive a **200 status code**, indicating the authorizer is working as expected.



Creating a Resource and POST Method in API Gateway for Lambda Integration

1. In the **API Gateway** console, select your **WildRydes** API.
2. On the left pane, click **Resources**.
3. Click **Create Resource** to define a new endpoint where you'll connect your Lambda function.

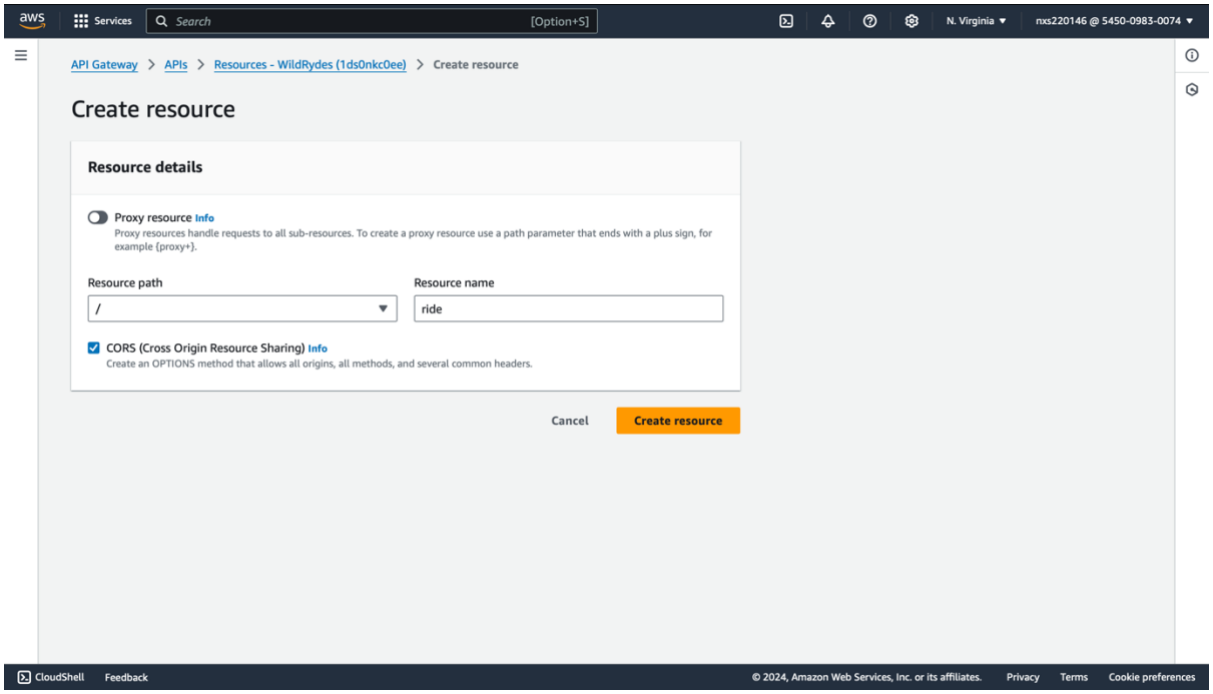


In the **Create Resource** screen: Enter **ride** as the **Resource Name**.

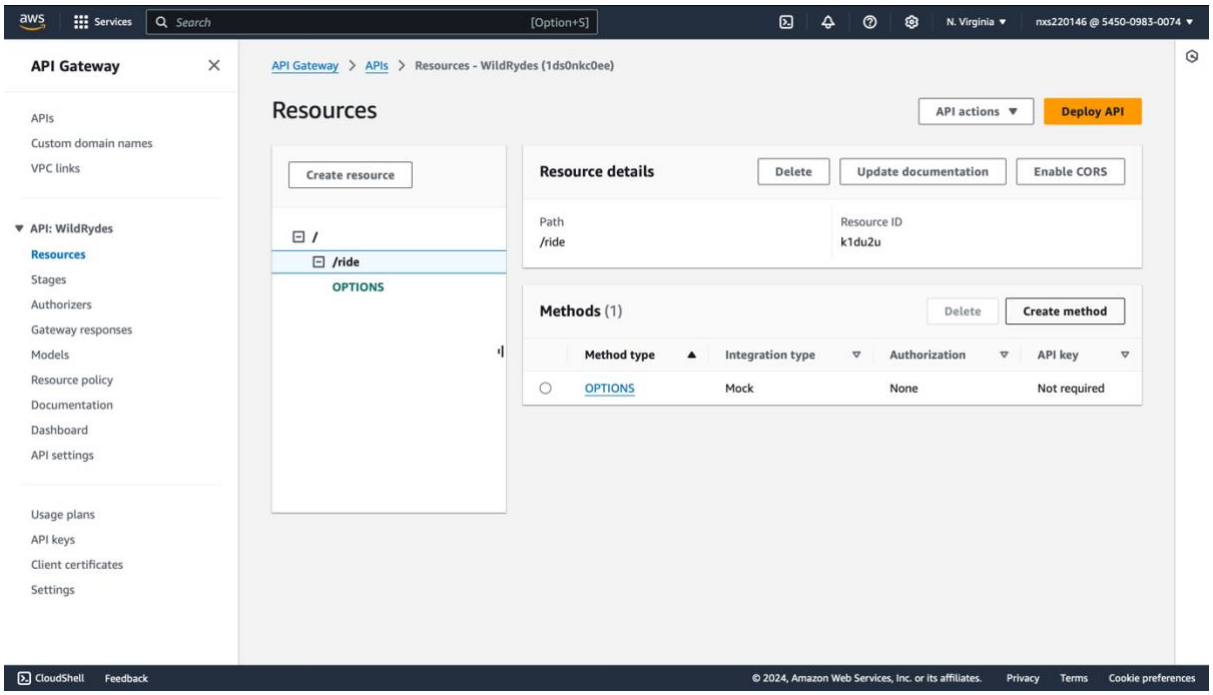
Check the box to enable CORS for the resource.

Enabling CORS (Cross Origin Resource Sharing) for the API Gateway
We do need to enable CORS, if you don't do that nothing's going to work and that's because the domain of the site on amplify is going to be different from the domain of API Gateway here so it's going cross origin.

Click on **create resource**.



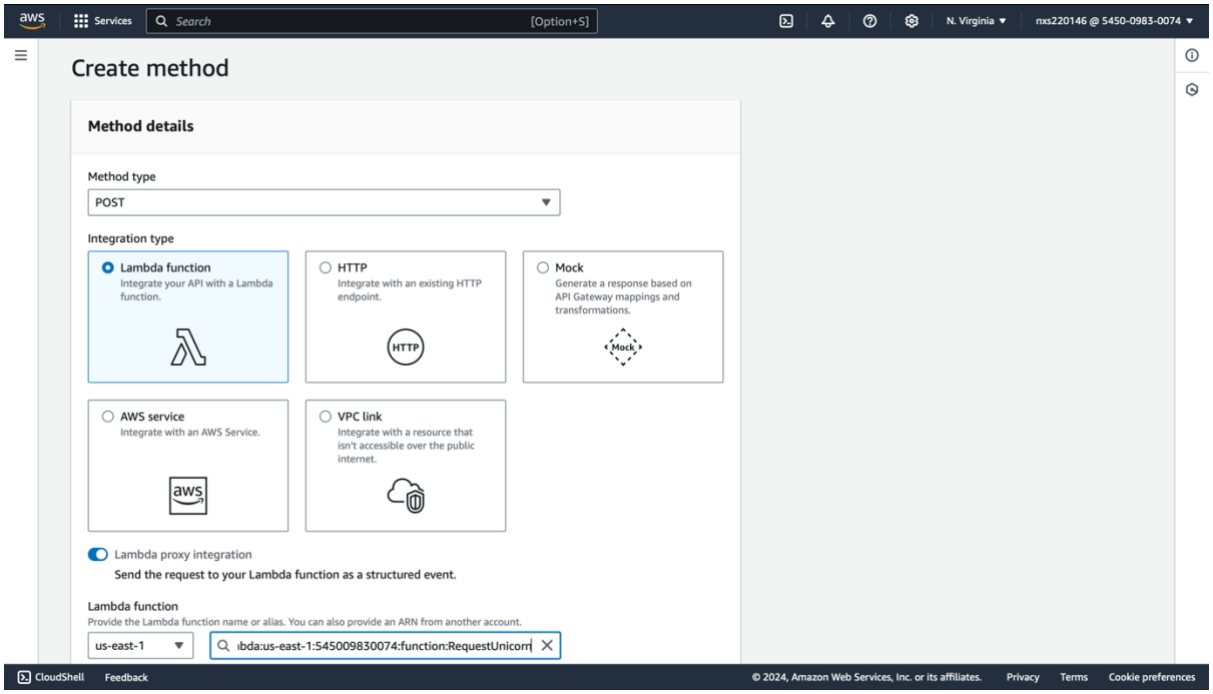
Choose **Create Method**.



To configure the method:

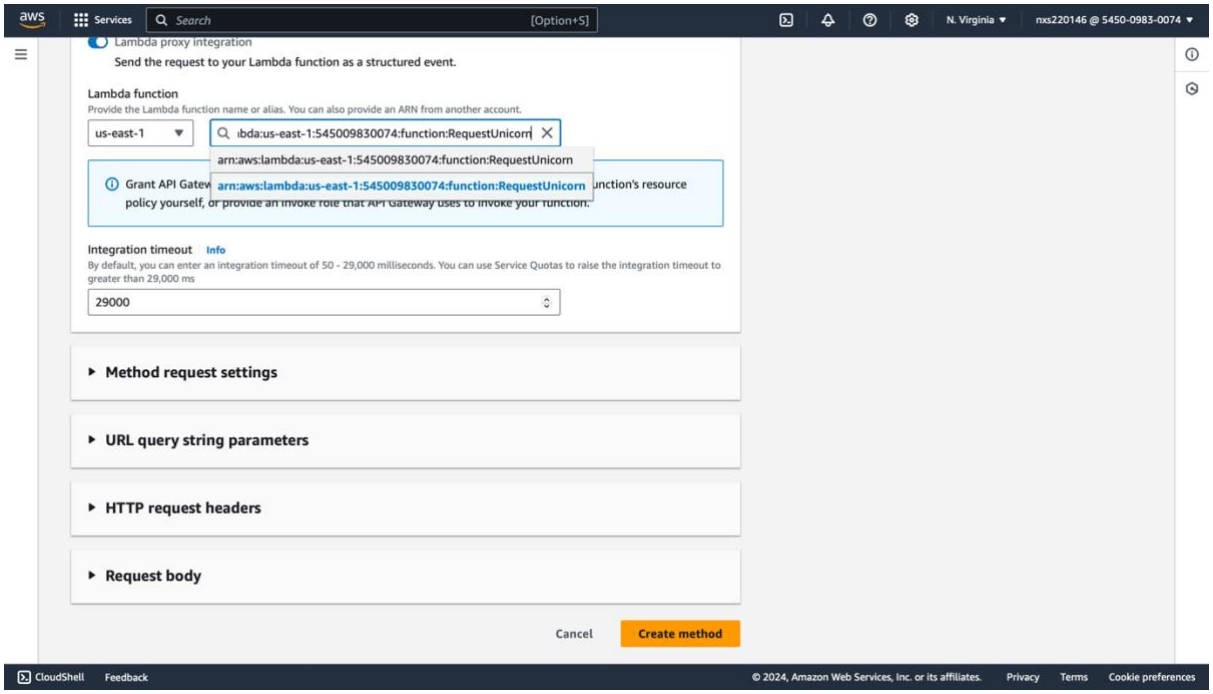
1. Select **POST** as the **Method Type** and click the checkmark to confirm.
2. For the **Integration Type**, choose **Lambda Function**.
3. Check the box for **Enable Lambda Proxy Integration** to allow API Gateway to pass requests directly to Lambda.
4. In the **Lambda Function** field, select the **RequestUnicorn** function you created earlier.

This will connect your API Gateway endpoint to the Lambda function.



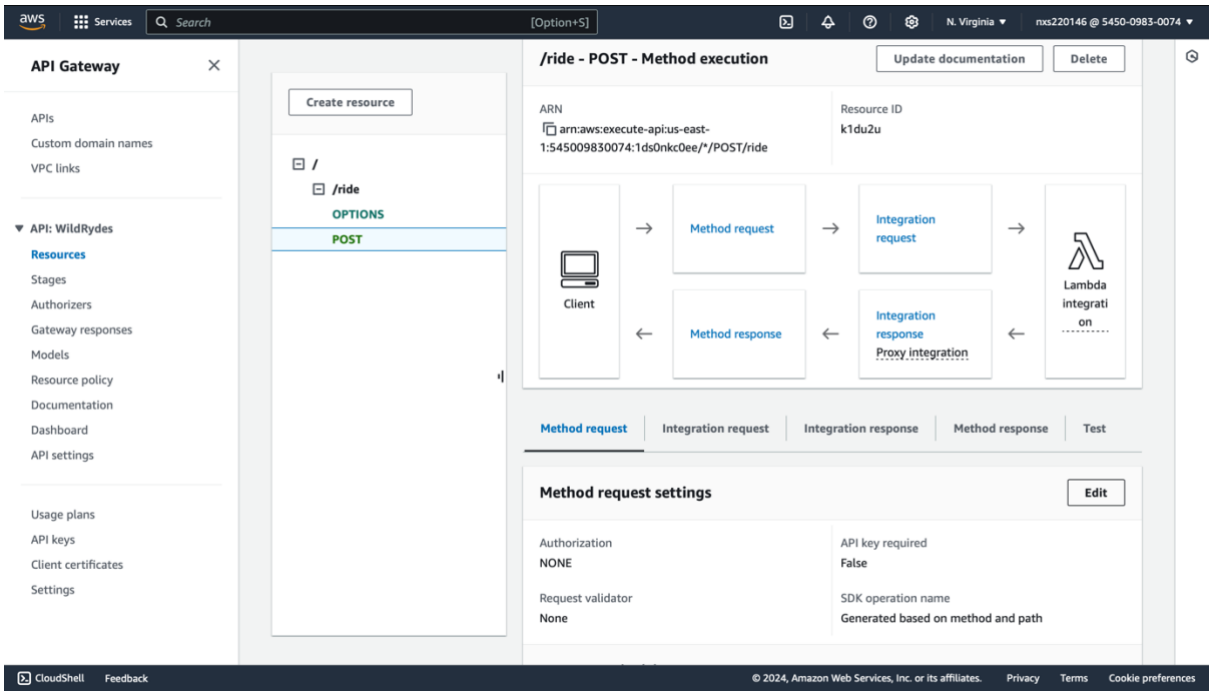
Click **Create Method** to complete the setup.

This will connect your API Gateway endpoint to the Lambda function.



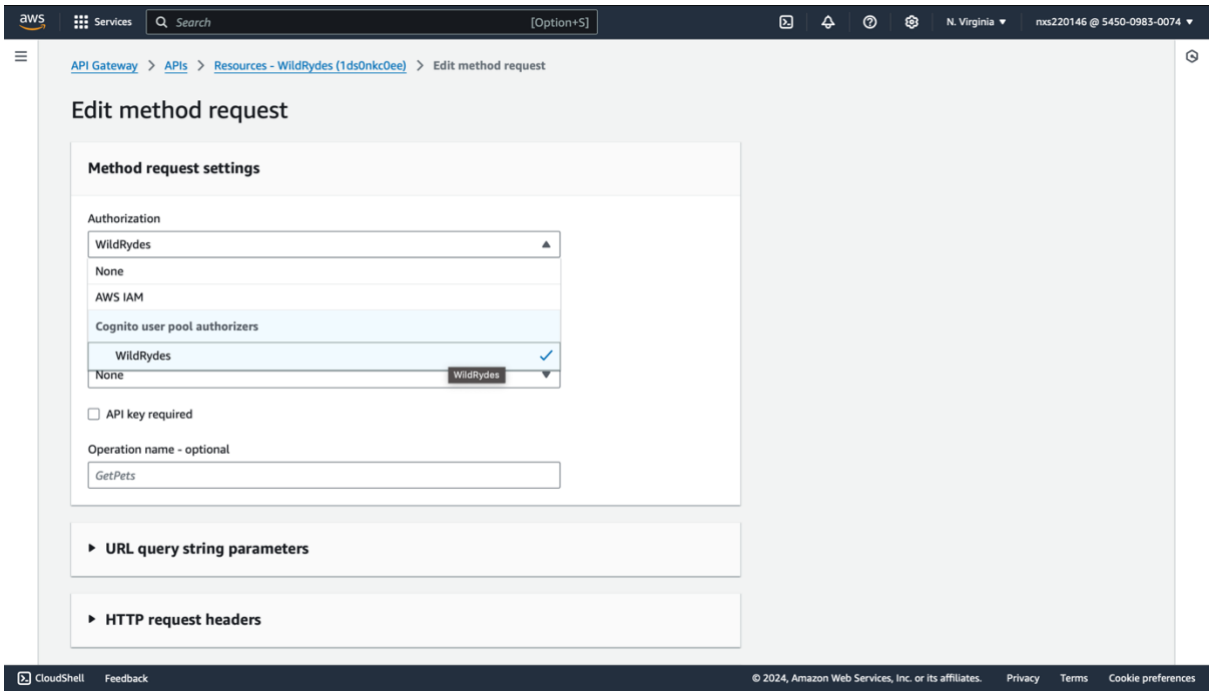
In the **POST** method configuration for your **ride** resource:

1. Go to the **Method Request** section.
2. Click **Edit** to modify the method settings as needed.

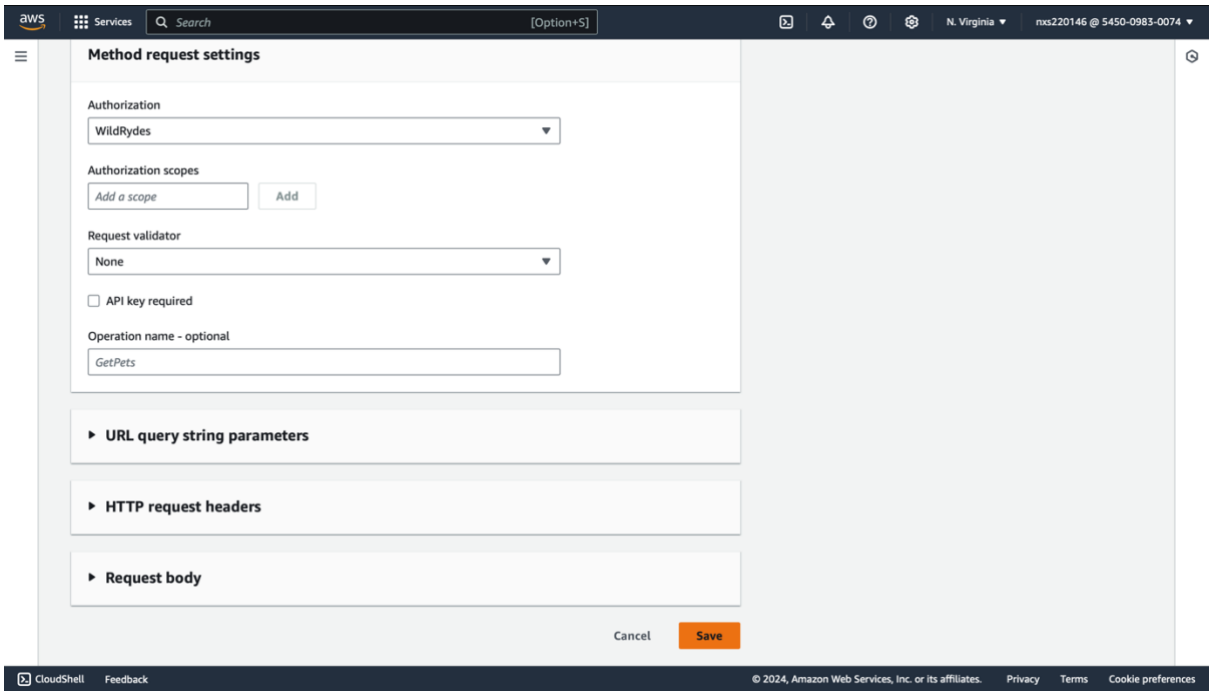


In the **Method Request** settings:

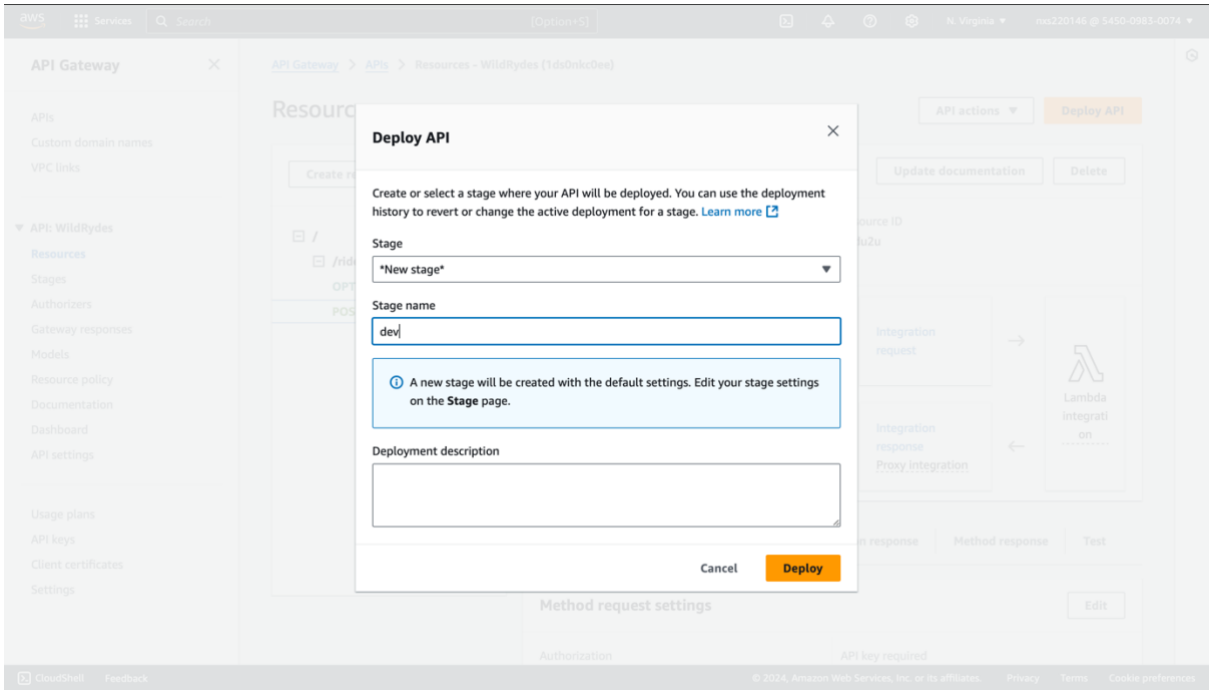
1. Locate the **Authorization** field and click **Edit**.
2. From the dropdown, select the **Cognito User Pool** authorizer you created earlier (e.g., **WildRydes**).



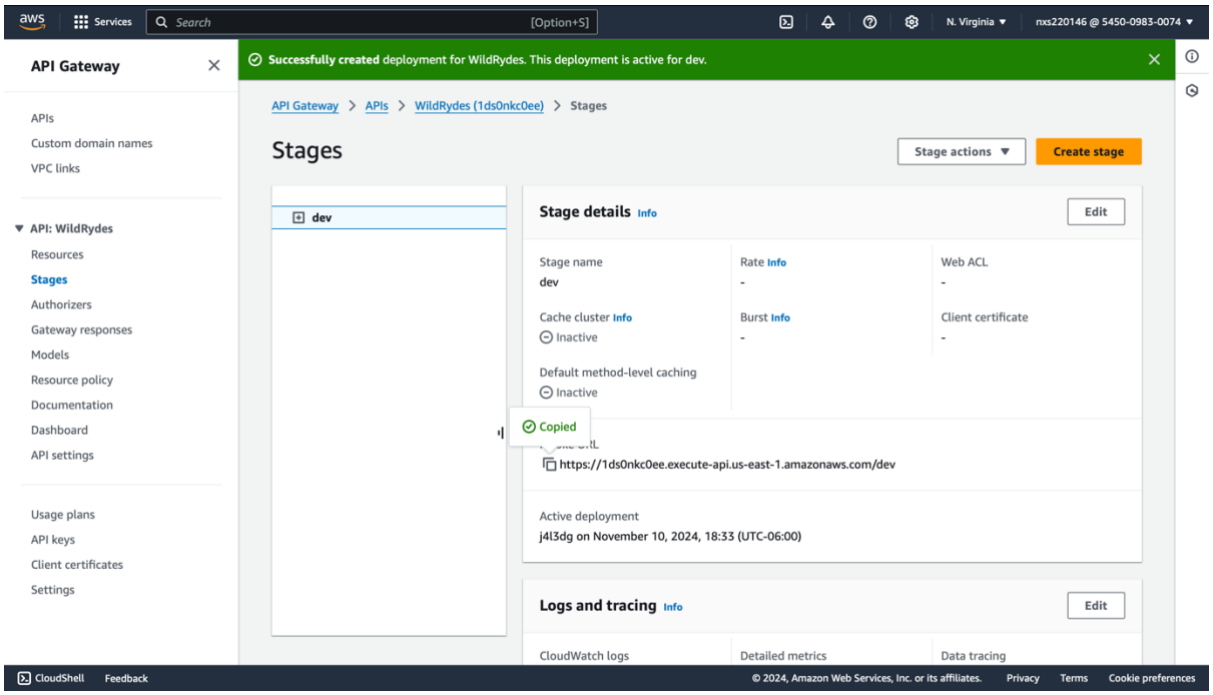
Leave the rest to default and click on **Save**.



Click on **Deploy API**, select **New Stage**, enter the stage name as **dev** and **deploy**.



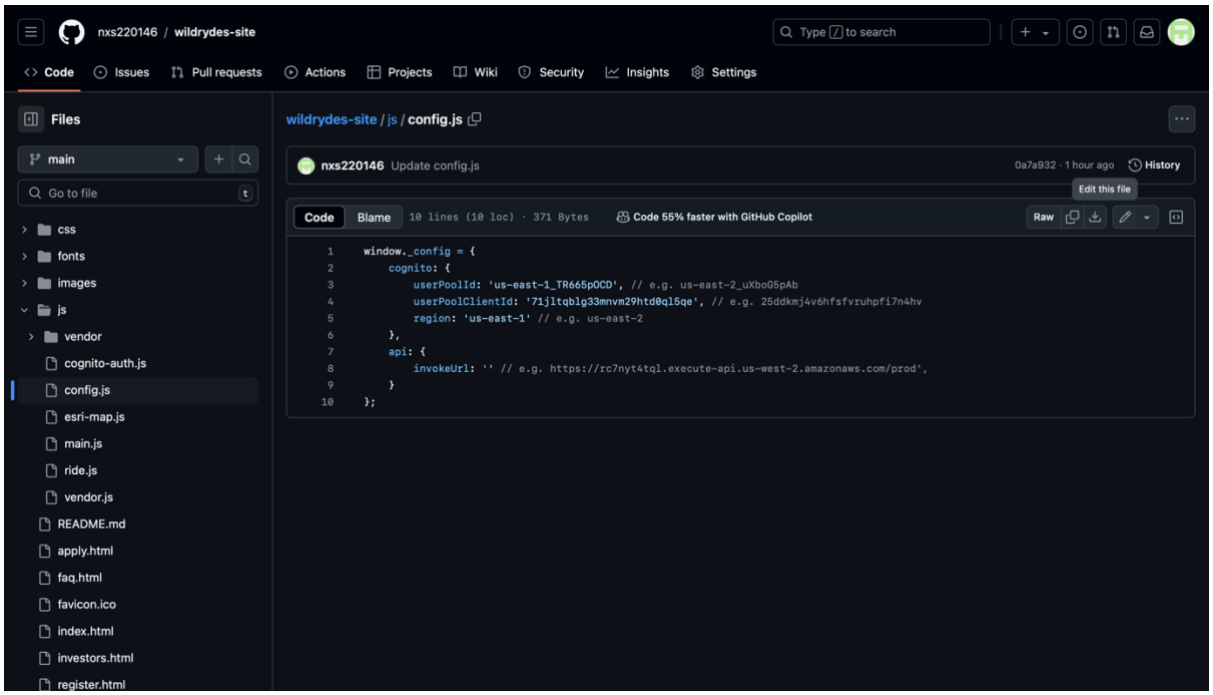
Copy the **invoke URL**, and paste it in your notepad.



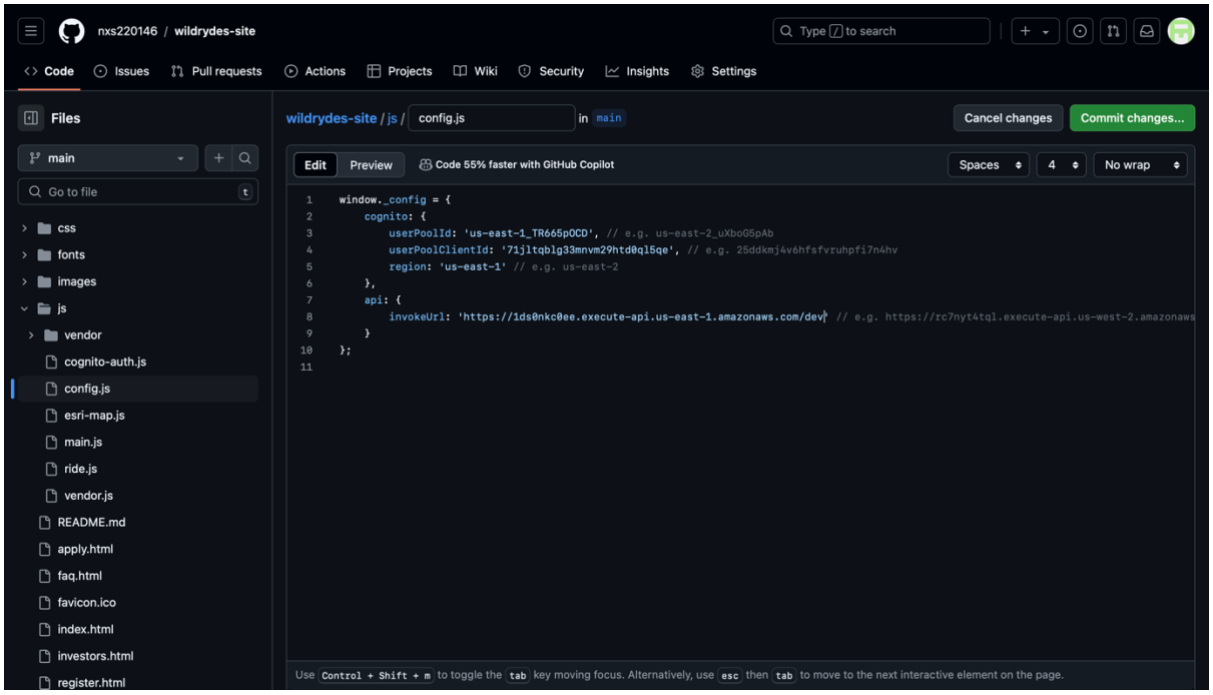
Note: The above screenshot is a deliverable

Updating the config file for the new Invoke URL from API Gateway

Open Config.js in github. Click on edit this file.



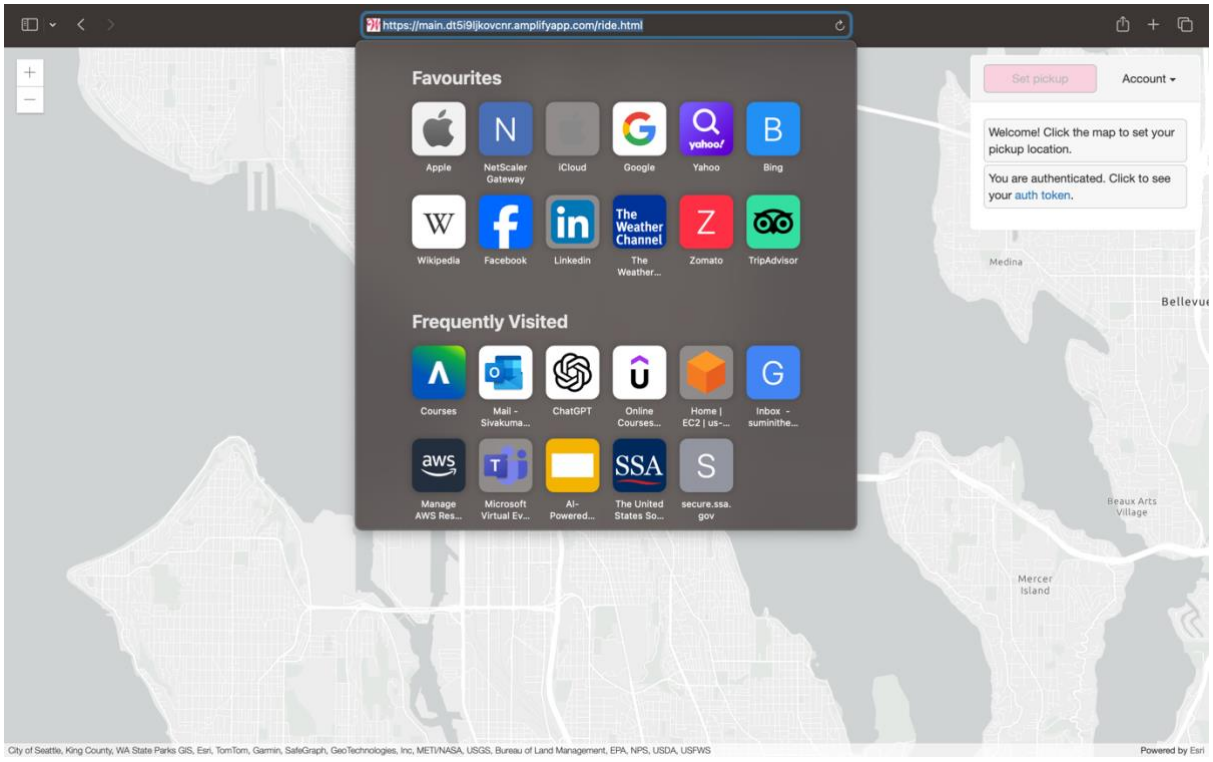
Paste the invoke url between the quotes ‘’ on line 8.



Click on Commit changes.

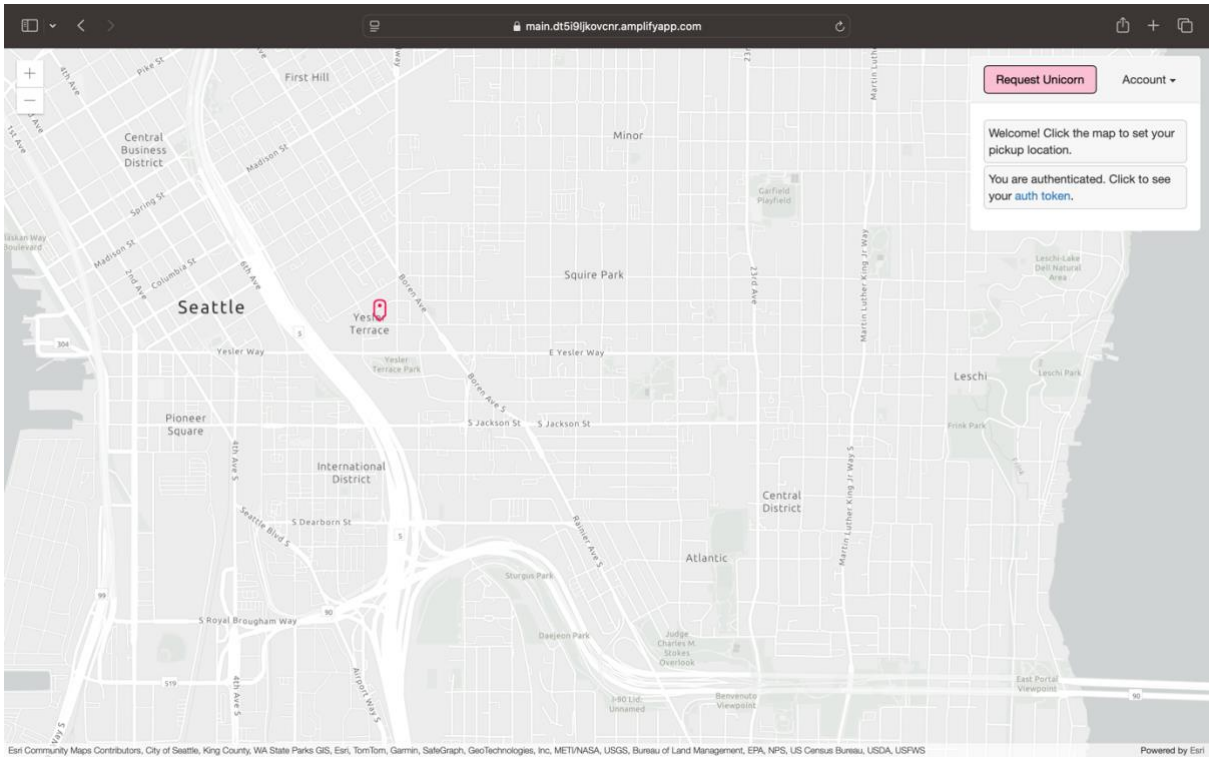
Testing our final application

Go to your website, and refresh once the new changes are deployed on Amplify. Navigate to **/ride.html** to access the ride functionality and verify that everything is working as expected.

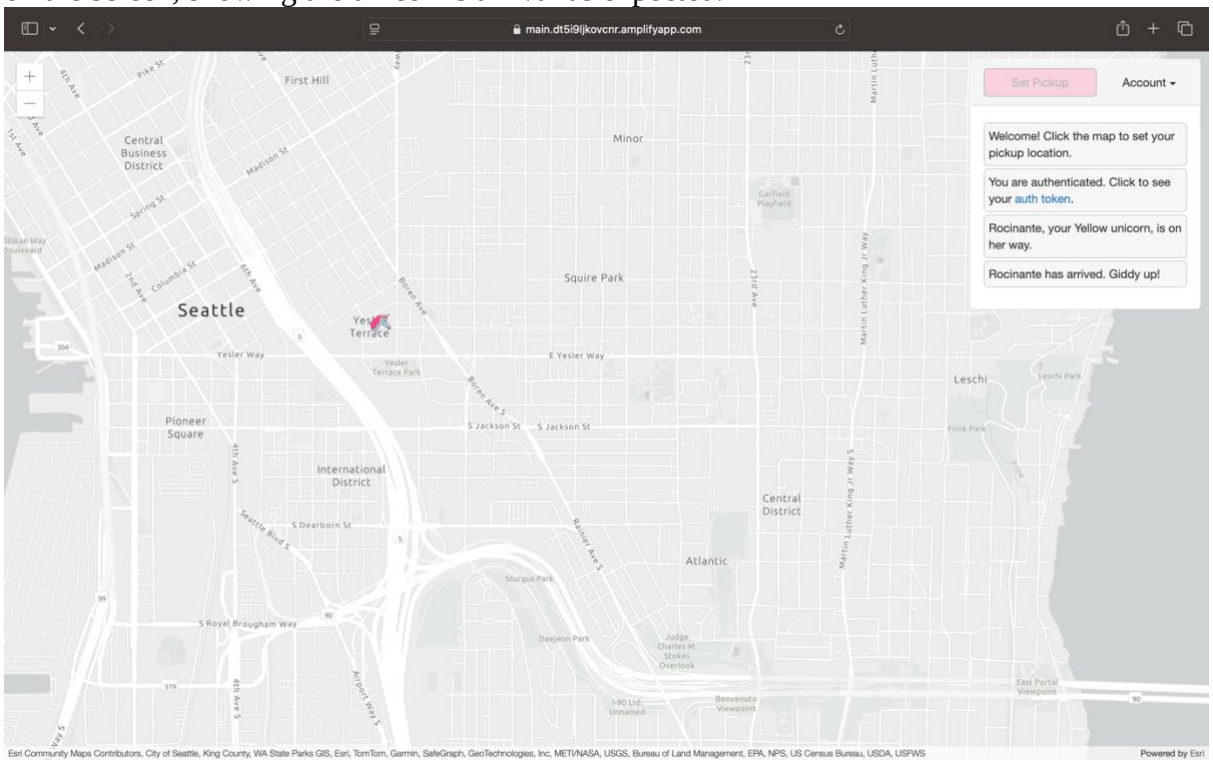


Click anywhere on the map to select a location.

Click on **Request Unicorn** to trigger the functionality and see if the ride-sharing process works as expected.



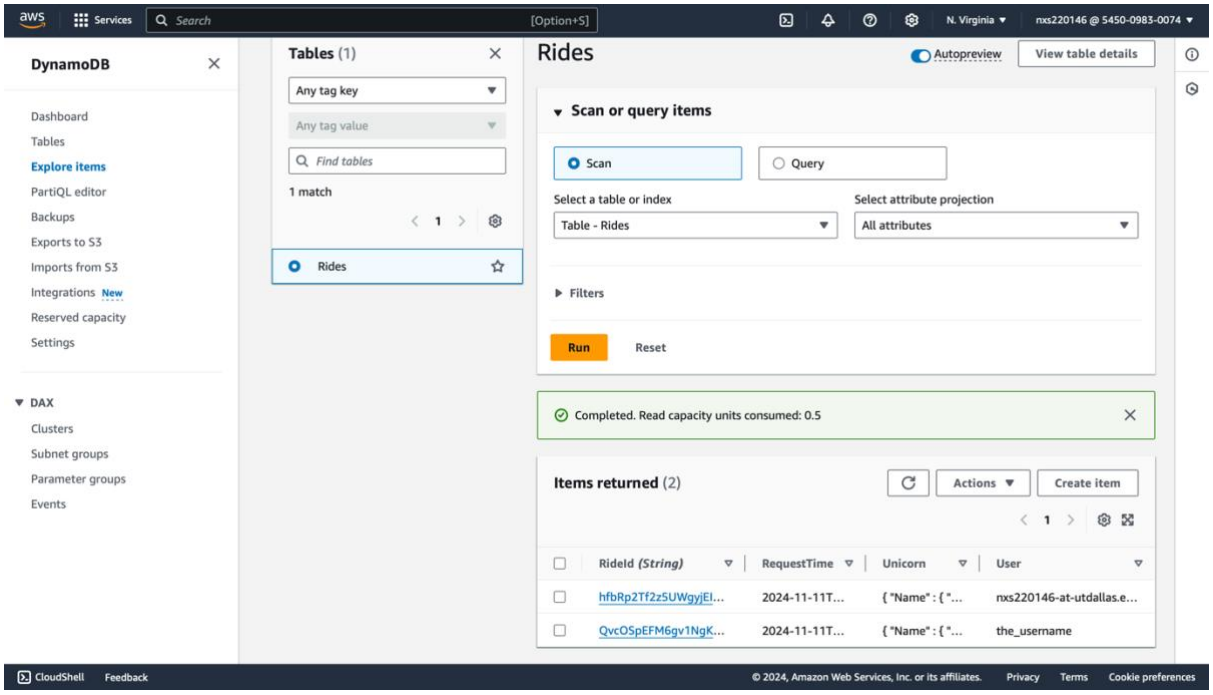
After clicking **Request Unicorn**, the unicorn should appear on the map, confirming that the ride request was successfully processed. You should see the updated information displayed on the screen, showing the unicorn's arrival as expected.

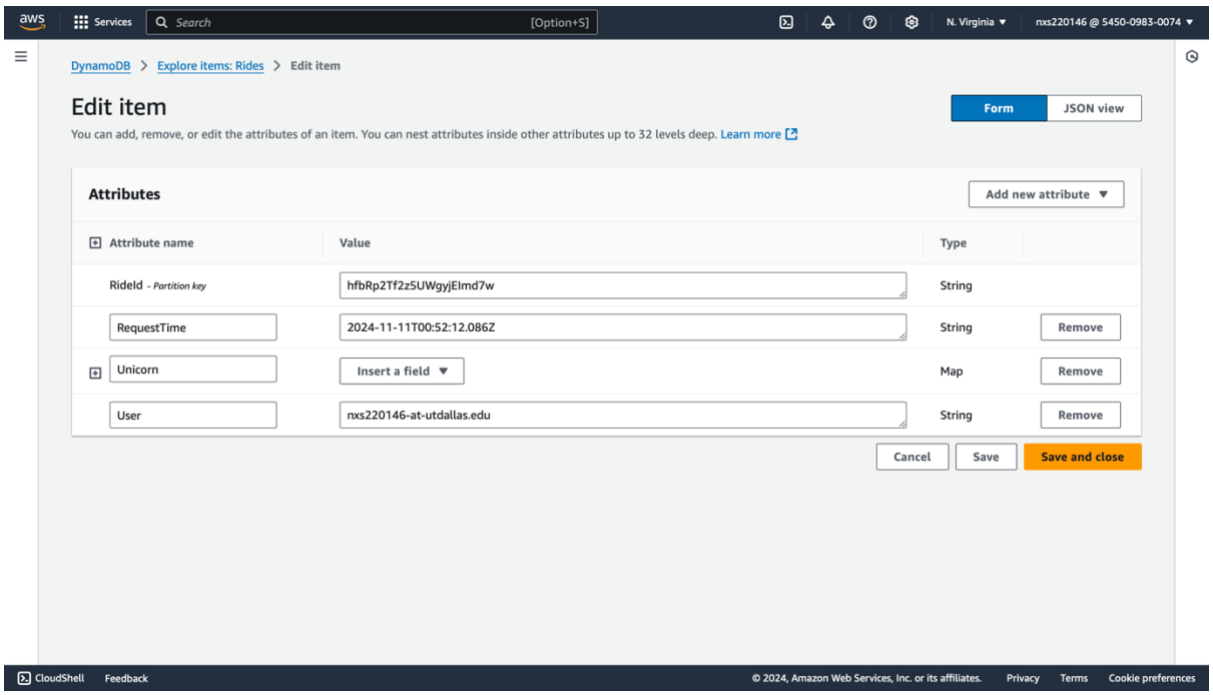


Note: The above screenshot is a deliverable (The URL must be visible)

Open your **DynamoDB** console and navigate to the **Rides** table.

Check the **Items** tab to see if a new record has been added, confirming that the ride-sharing information was successfully written to the table by the Lambda function.





To clean up after the lab and avoid any unnecessary charges, make sure to delete the following resources:

1. **Lambda Function**: Go to the Lambda console, select your RequestUnicorn function, and click Delete.
2. **API Gateway**: In the API Gateway console, delete the WildRydes API.
3. **DynamoDB Table**: Go to the DynamoDB console, select your Rides table, and click Delete Table.
4. **Cognito User Pool**: In the Cognito console, select the WildRydes user pool and delete it.
5. **IAM Role**: In the IAM console, delete the WildRydesLambda role.
6. **Amplify App**: Go to the Amplify console, select your app, and click Delete App.

Make sure to delete any other resources you've created during the lab to ensure you're not charged for unused services.