Project 2

02 June 2024 09:15

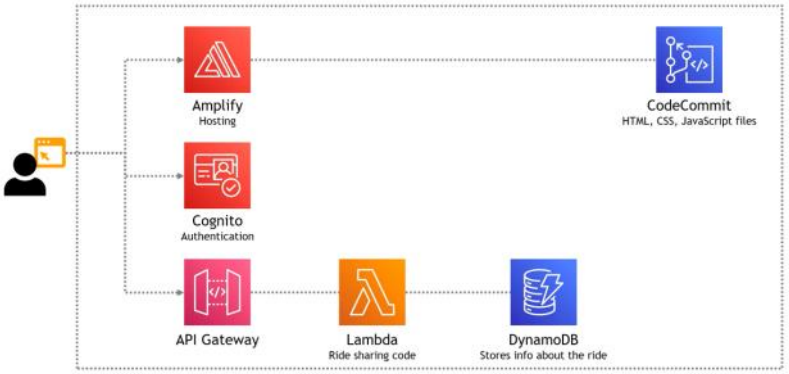
Project 2: Hosting the WildRydes application

AWS Services employed:

External Services:

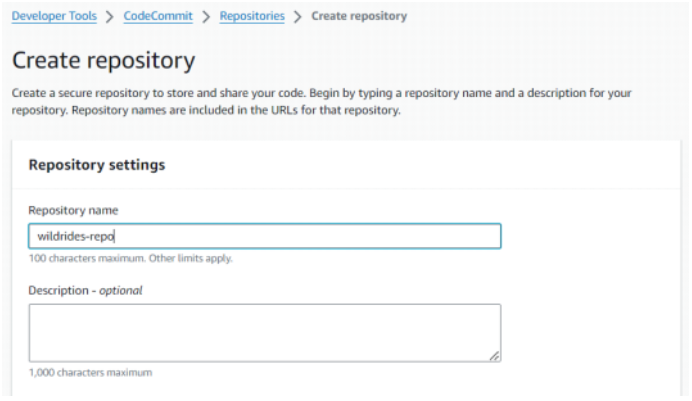
High level summary:
WildRydes is an application, just like the Uber taxi app that one can use to call for rides, but using unicorns instead of regular cars. The html and lambda function code for this app is from AWS.

Architectural Diagram



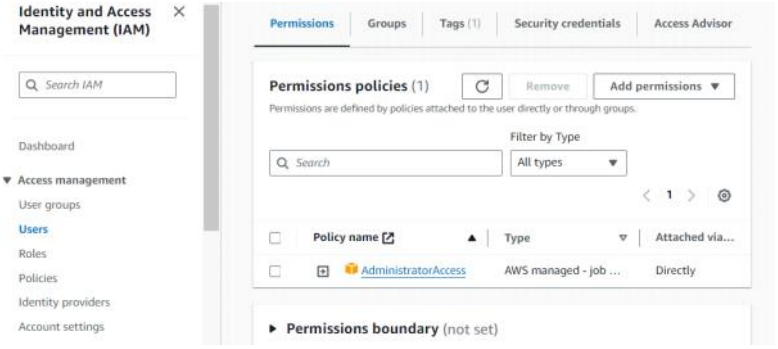
- Store/pull code and permissions to access the code
- Create an environment to host website and make updates
- Provide a way for users to authenticate and log in to the website
- Improve the website by including the ride sharing functionality
- Provide a database to store/return results
- Invoke the ride sharing functionality

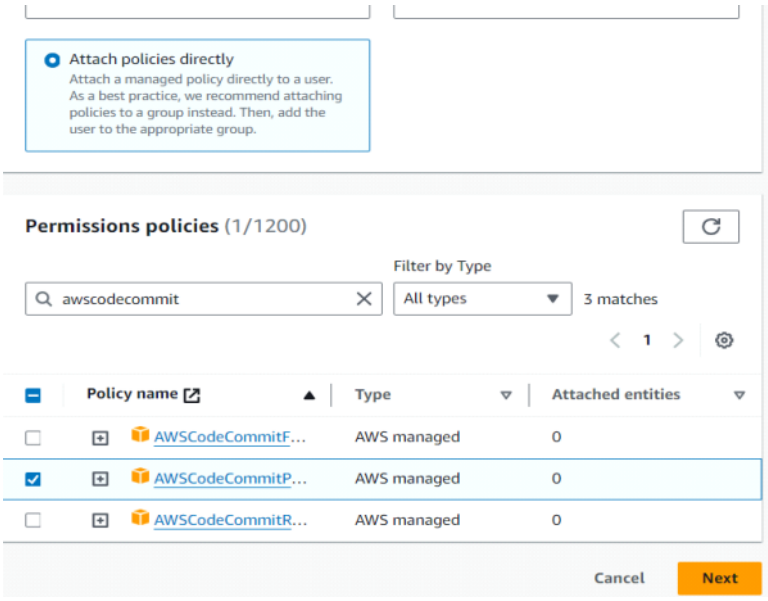
Step 1: create a code repository and create IAM policy



The repository will be empty

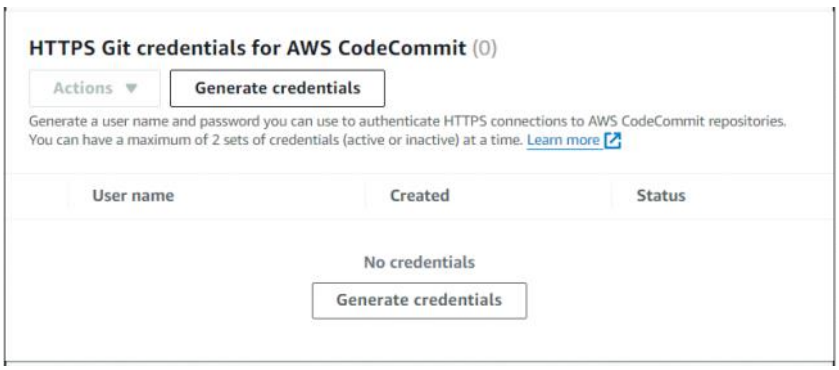
Step 2: Create an IAM user and give it necessary permissions





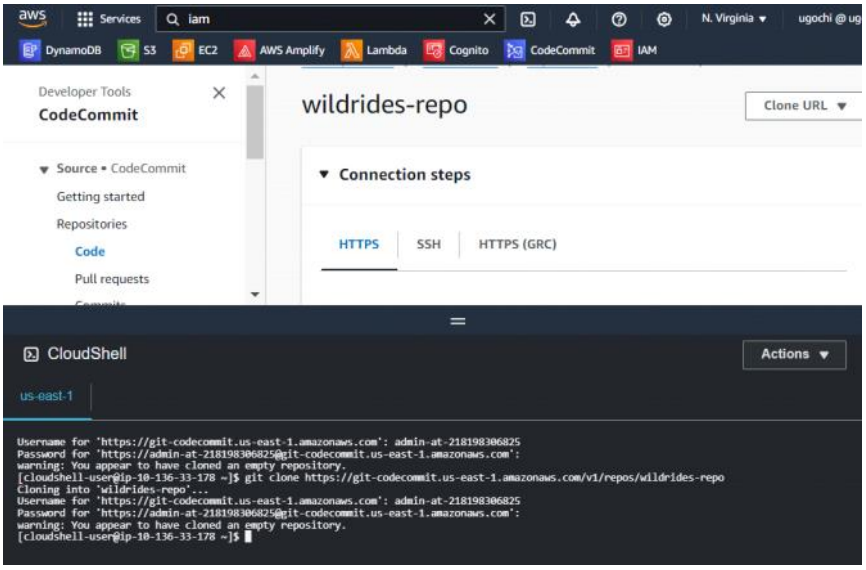
- Create git credentials for IAM user

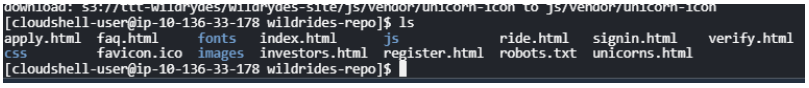
On the IAM admin user dashboard, click on the 'Security credentials' tab



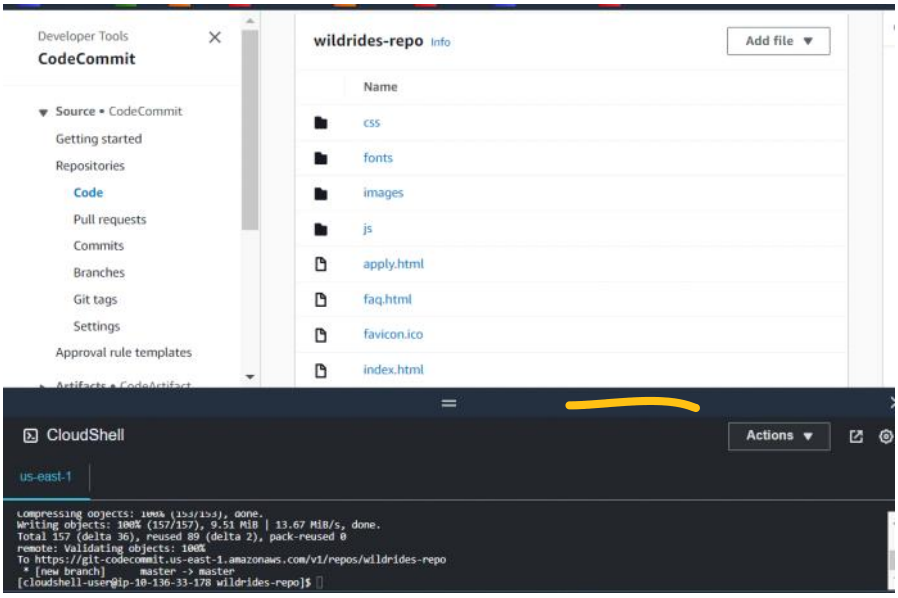
- Open cloudshell and clone the repository using the HTTPS to ensure credentials work
- Configure CloudShell with the username and password of the repo I just created.
- Cd into the repo folder
- The code for the website is stored in another s3 bucket and so I needed to copy the code from that bucket to my repository. I used this code to do just this:

  aws s3 cp s3://ttt-wildrydes/wildrydes-site ./ --recursive



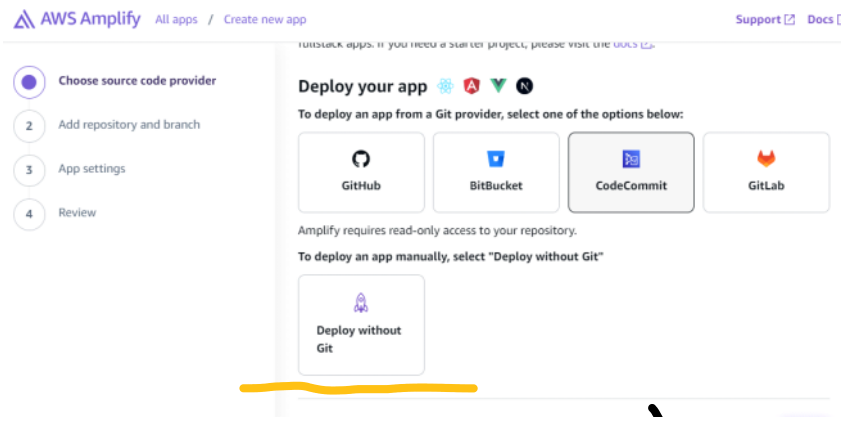


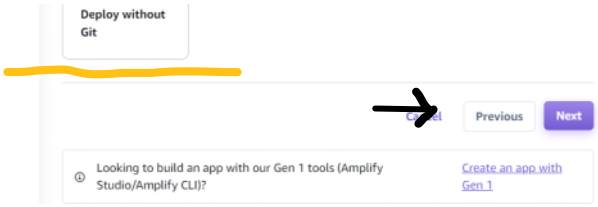
- I then used git commands (add, commit, push) to push the website files to the wild-rides repo



Step 3: deploy the code on AWS Amplify

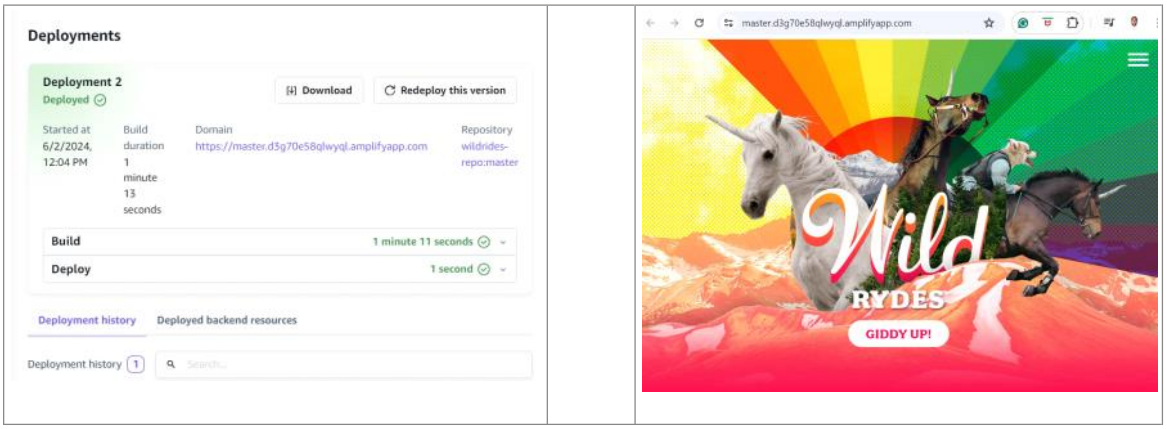
- Choose CodeCommit as the provider





- Follow the on-screen instructions to configure the app deployment with the repo.
  Note that an IAM role is required to allow Amplify access to CodeCommit. Due to changes in the UI, I didn't have the opportunity to create an IAM role during configuration of the Amplify's deployment. I had to create a role for this separately and edited the Amplify settings to make use of this role before having a successful deployment.

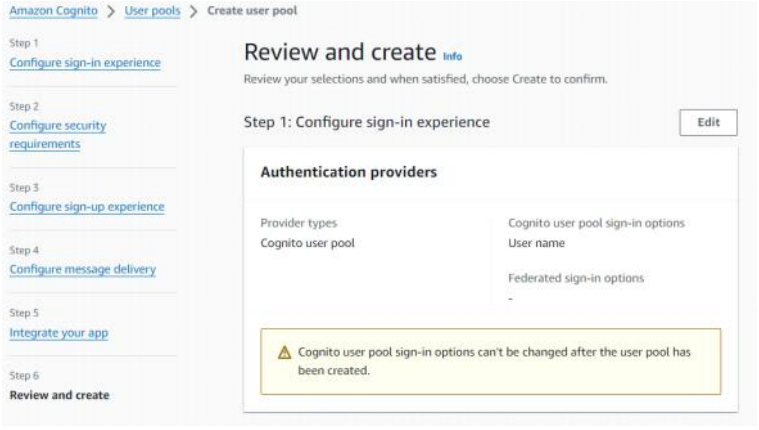
  Clicking on the domain provided by the deployment shows the deployed app. At the moment, it's just a static html page.



  The great thing about Amplify is that it redeploys the code each time there are changes to the codecommit widerides-repo.

Step 4: setup Cognito for authenticating users to the application

- Go to the Cognito console and create a user pool using just username. I named the user pool wildrides
  For the purpose of this project, I used just username as the sign-in option but obviously in production envrinoments, email, phone numbers will be used. Also for the same purpose, I did not use MFA and used just the default configuration.



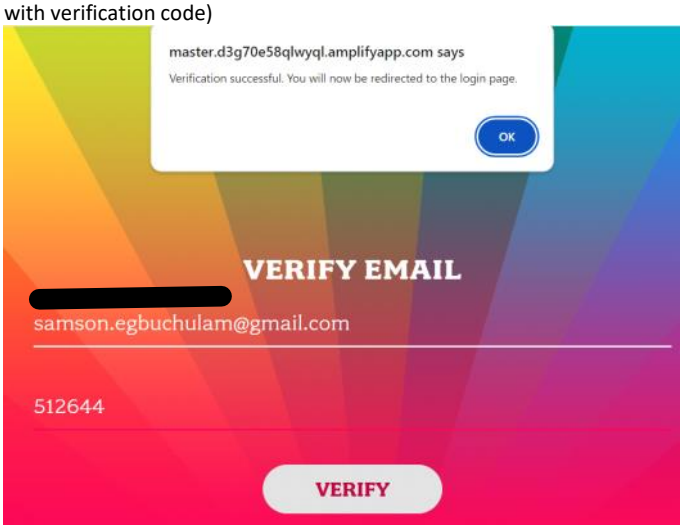
After the creation of the pool, I copied the user pool id of cognito and the client id (under the app integration tab)

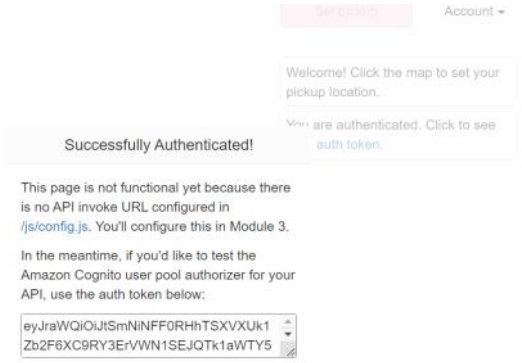
Step 5: configure the application to use Cognito

- Head back to the wildrides repo and edit the js/config.js file with the user pool id and client id copied in the previous step.
- The invokeUrl is edited later. The change is saved and committed to the master branch of the repo.

This change triggers Amplify to deploy the changes, and now we have

The new changes allows us to register on the site. Cognito behind the scenes provides the mechanism for registration and authentication (sends a mail with verification code)

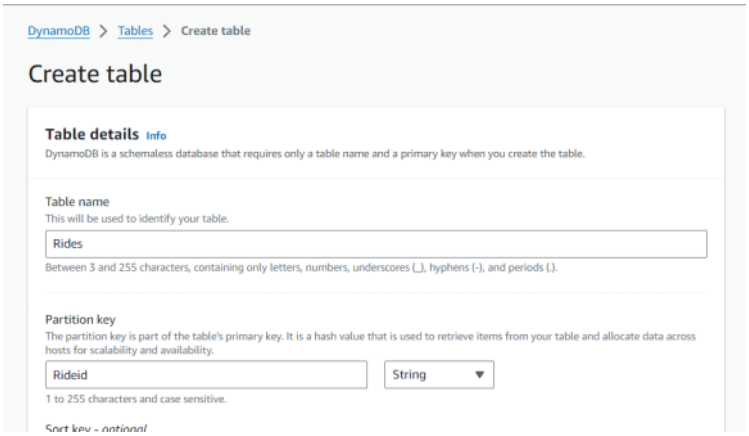




Copy the authentication code shown

Step 6: configure DynamoDB table

- Navigate to DynamoDB console and create table with name, 'Rides' and Partition key 'Rideid'

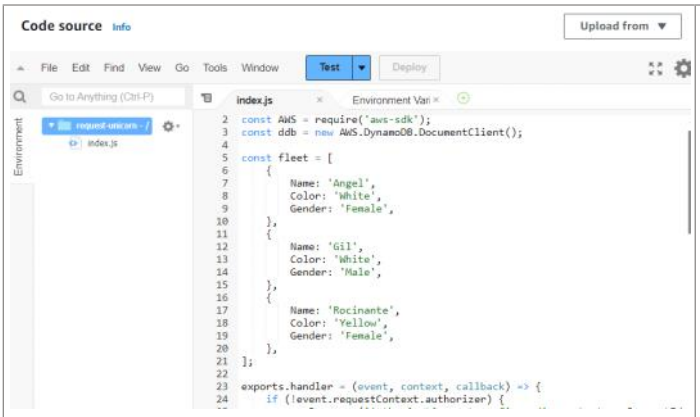


- After creation, I copied the arn of the table

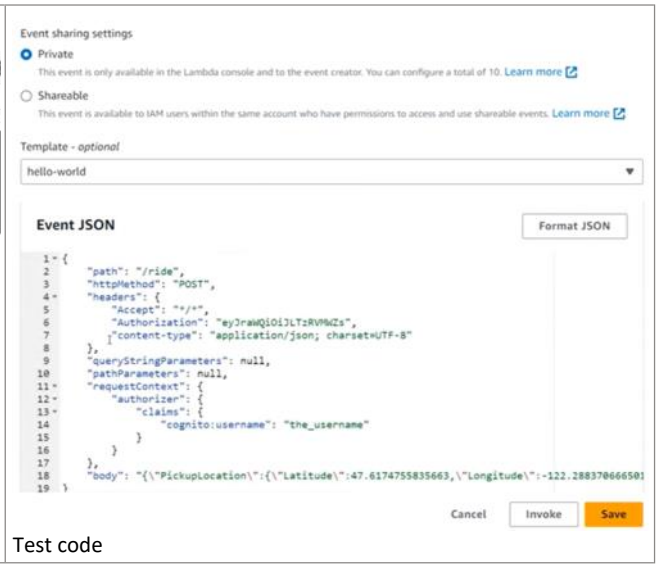
Step 7:configure lambda function

Lambda needs permission to write to the dynamo db table, so I had to create a role for lambda before creating the function. I could have also created the lambda function and edited the role to have access to the created DynamoDB table

- Create lambda function using Nodejs 16.x environment
- The code I used for this lambda is taken from aws. The code(including the test code) is also hosted on this github repo
- Deploy the code and test it

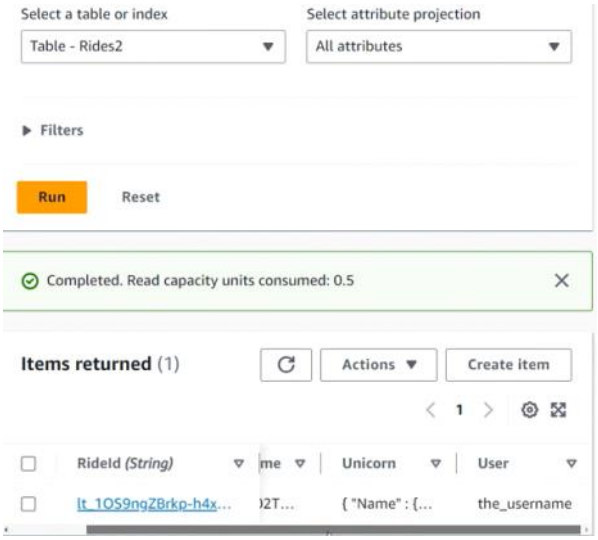


Lambda code



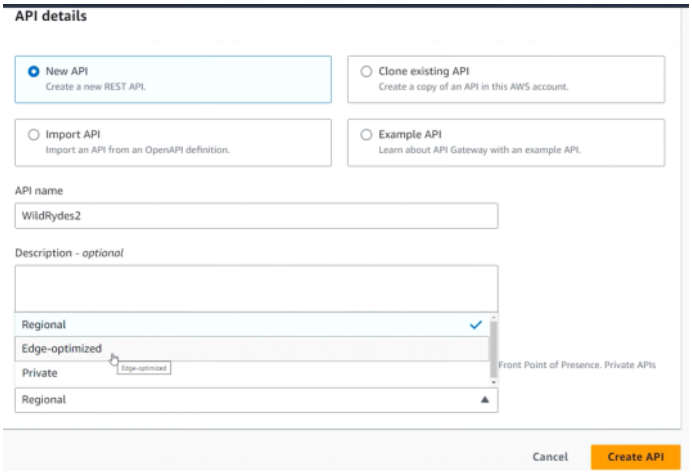
Test code

Everything should work and the function should return a status code of 201. also there should be an entry in our DynamoDB table

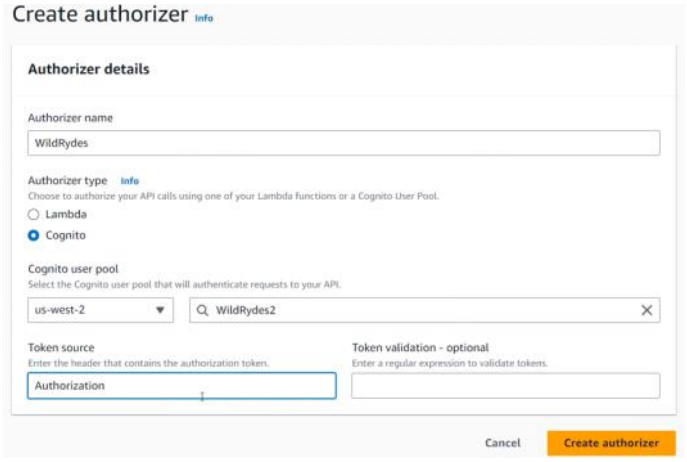


Step 8: Create a REST API and configure the Authorizer

- I created an edge-optimized API. This ensures the API is accessible globally, as against Regional or even private API



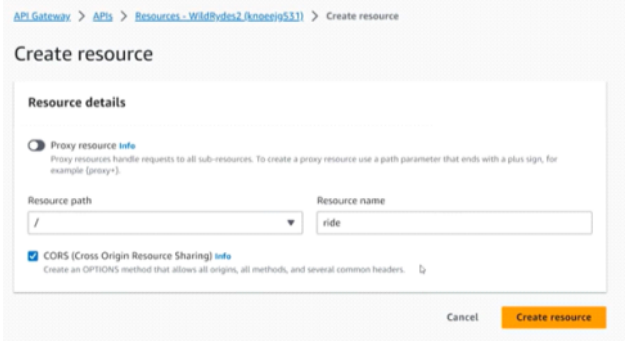
We need to create an authorizer from the API to use the json web tokens (jwt) returned by cognito. This jwt will be used to invoke the API



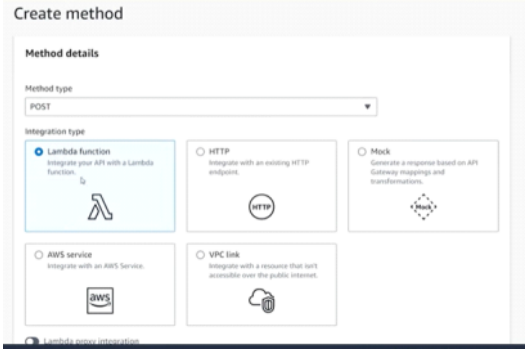
The token source, Authorization, is the name of the header that will be returned from Cognito.

- After creation, click on the WildRydes authorizer. In the Token value, copy and paste the Cognito token copied from a previous step
- Test authorizer. I get a status code of 200

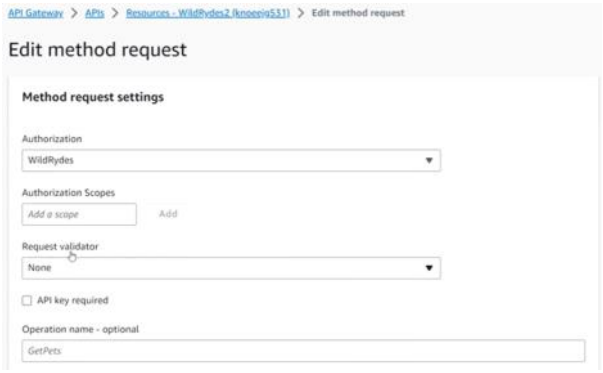
- Create a resource, named ride with CORS enabled



- With the ride resource selected, I created a POST method that will be used to invoke the lambda function
- The lambda function is selected when created the POST method



- After creating the POST method, I edited its Method Request tab to use the WildRydes Cognito User Pool Authorization



Step 9: Final code edits and app deployment

- Deploy the API (create a new stage named dev)
- Grab the API url
- Edit the config.js file in CodeCommit with the url of the API just copied. Commit the changes
- I also edited the ride.html in CodeCommit to use arcgis 4.6 instead of 4.3

AWS Amplify should push out the changes automatically.

Ensure you have setup the arcgis account and that it is open in your browser. Once the domain (from Amplify) is clicked this time, you should see the app open with a map from arcgis

The DynamoDB table also updates with the latest request