

# Project 2

02 June 2024 09:15

## Project 2: Hosting the WildRydes application

AWS Services employed:



External Services:

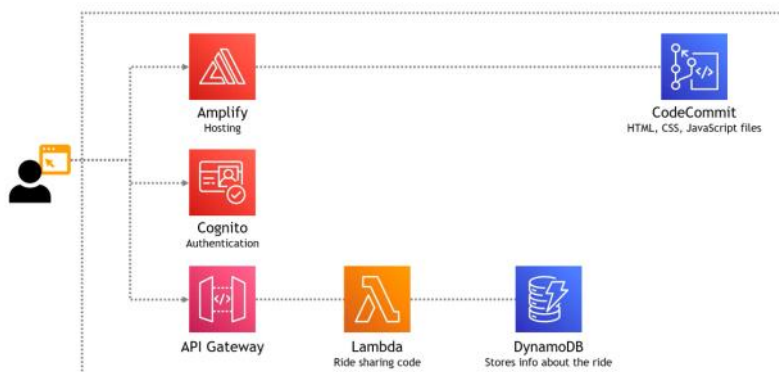


ArcGIS

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

Developer Tools > CodeCommit > Repositories > Create repository

## Create repository

Create a secure repository to store and share your code. Begin by typing a repository name and a description for your repository. Repository names are included in the URLs for that repository.

Repository settings

Repository name

wildrides-repo

100 characters maximum. Other limits apply.

Description - optional

1,000 characters maximum

The repository will be empty

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

Identity and Access Management (IAM)

Search IAM

Dashboard

Access management

User groups

Users

Roles

Policies

Identity providers

Account settings

Permissions
Groups
Tags (1)
Security credentials
Access Advisor

Permissions policies (1)

Remove Add permissions

Permissions are defined by policies attached to the user directly or through groups.

Filter by Type

Search All types

Policy name

Type

Attached via...

AdministratorAccess AWS managed - job ... Directly

Permissions boundary (not set)

Attach policies directly

Attach a managed policy directly to a user. As a best practice, we recommend attaching policies to a group instead. Then, add the user to the appropriate group.

Permissions policies (1/1200)

Filter by Type

awscodecommit All types 3 matches

Policy name

Type

Attached entities

AWSCodeCommitF...

AWS managed

0

AWSCodeCommitP...

AWS managed

0

AWSCodeCommitR...

AWS managed

0

Cancel Next

- Create git credentials for IAM user

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

HTTPS Git credentials for AWS CodeCommit (0)

Actions Generate credentials

Generate a user name and password you can use to authenticate HTTPS connections to AWS CodeCommit repositories. You can have a maximum of 2 sets of credentials (active or inactive) at a time. [Learn more](#)

User name

Created

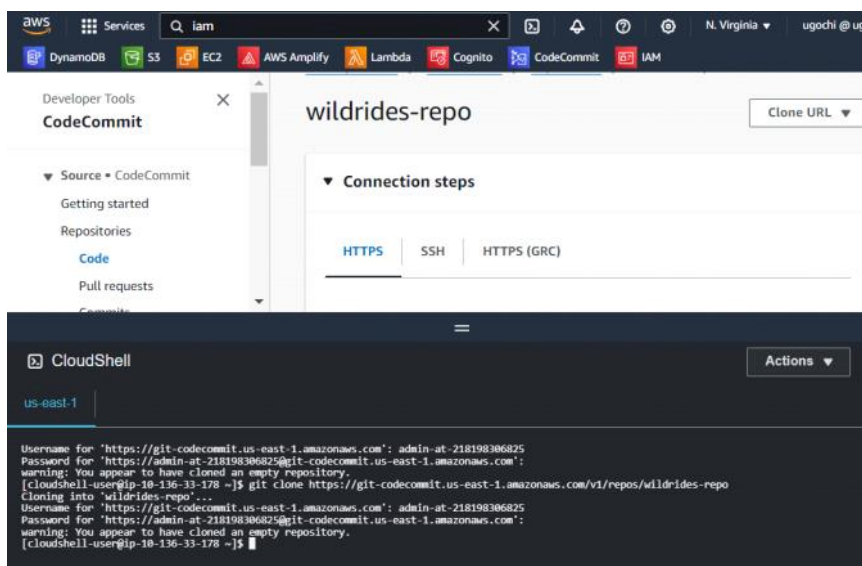
Status

No credentials

Generate credentials

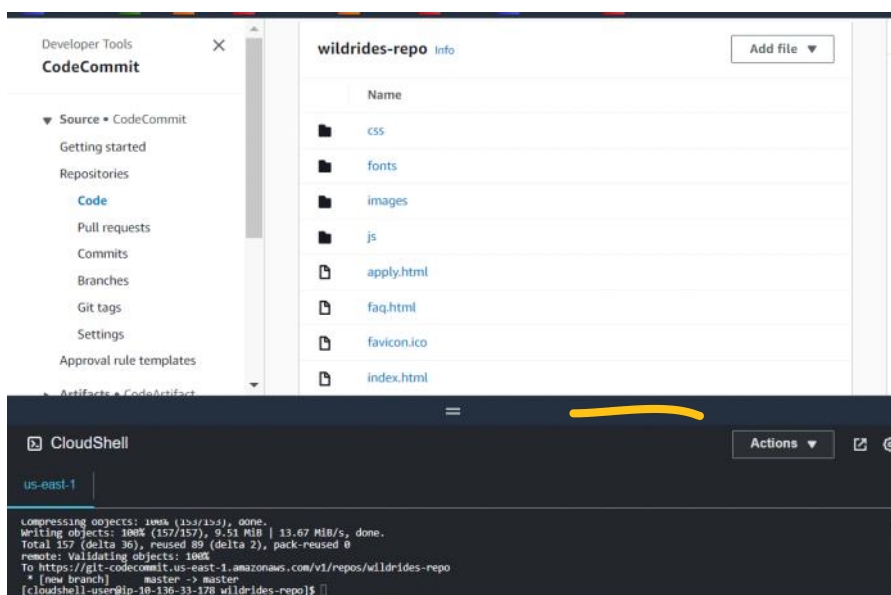
- Open cloudshell and clone the repository using the HTTPS to ensure credentials work
- Configure CloudShell with the username and password of the repo I just created.
- Cd into the repo folder
- The code for the website is stored in another s3 bucket and so I needed to copy the code from that bucket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

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



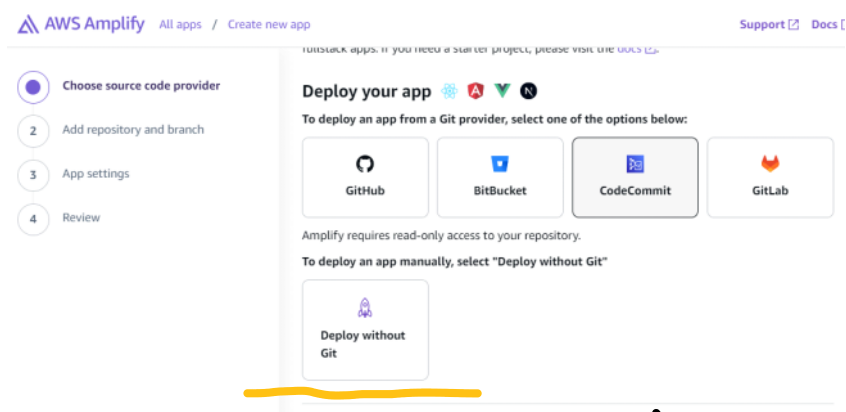
```
download: s3://ttt-wildrydes/wildrydes-site/js/vendor/unicorn-icon to js/vendor/unicorn-icon
[cloudshell-user@ip-10-136-33-178 wildrides-repo]$ ls
apply.html  faq.html  fonts  index.html  js  ride.html  signin.html  verify.html
css         favicon.ico  images  investors.html  register.html  robots.txt  unicorns.html
[cloudshell-user@ip-10-136-33-178 wildrides-repo]$
```

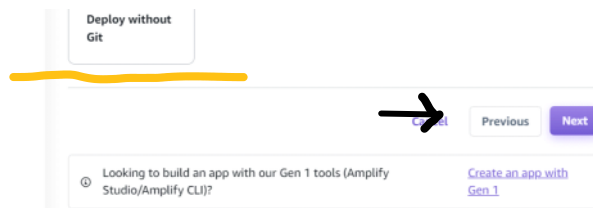
- I then used git commands (add, commit, push) to push the website files to the wild-rides repo



### Step 3: deploy the code on AWS Amplify

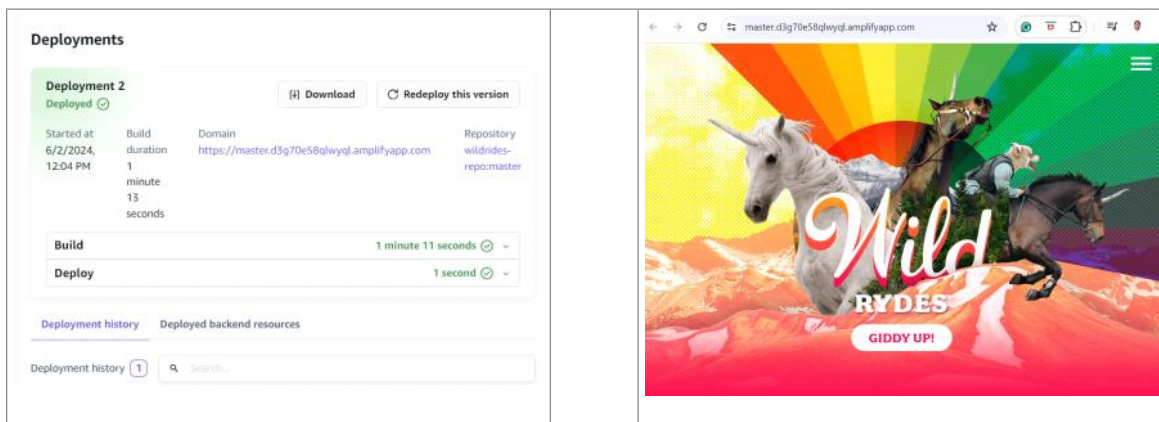
- Choose CodeCommit as the provider





- 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.

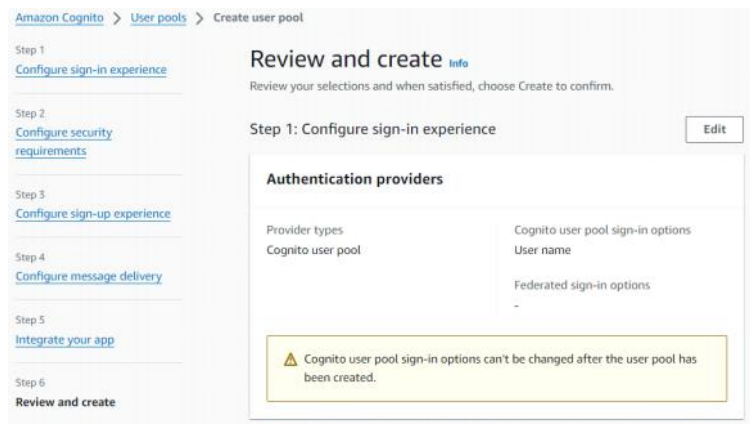
Clicking on the domain provided by the deployment shows the deployed app. At the moment, it's just a static html page.



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

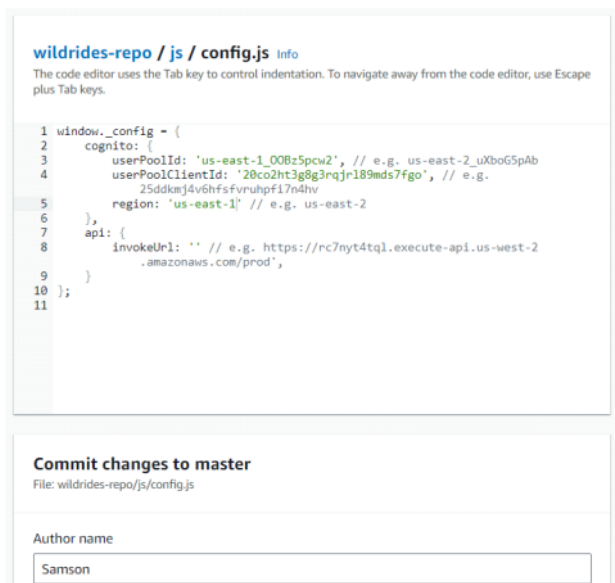
- Go to the Cognito console and create a user pool using just username. I named the user pool wildrides  
For the purpose of this project, I used just username as the sign-in option but obviously in production environments, email, phone numbers will be used. Also for the same purpose, I did not use MFA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

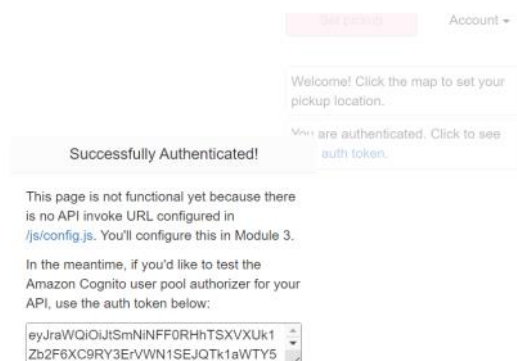
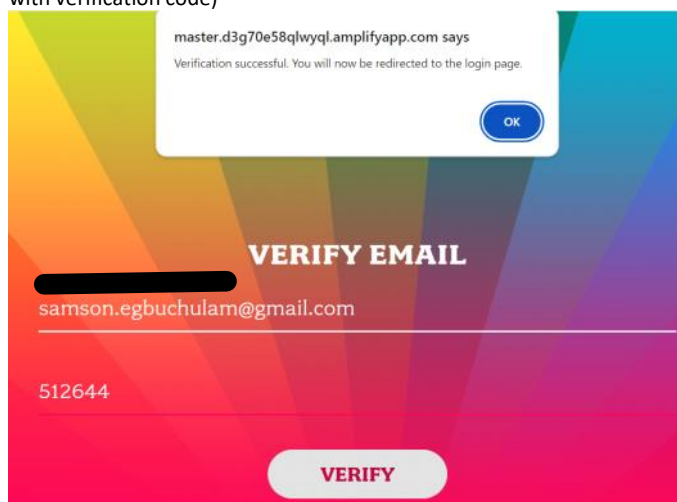
#### Step 5: configure the application to use Cognito

- 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'

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

- 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

**Code source** [Info](#)

Upload from

File Edit Find View Go Tools Window

Go to Anything (Ctrl-F)

Environment

index.js

```

1 const AWS = require('aws-sdk');
2 const db = new AWS.DynamoDB.DocumentClient();
3
4 const fleet = [
5   {
6     Name: 'Angel',
7     Color: 'White',
8     Gender: 'Female',
9   },
10  {
11    Name: 'Gil',
12    Color: 'White',
13    Gender: 'Male',
14  },
15  {
16    Name: 'Rocinante',
17    Color: 'Yellow',
18    Gender: 'Female',
19  },
20 ];
21
22 exports.handler = (event, context, callback) => {
23   if (!event.requestContext.authorizer) {
24     // ...
25   }
26 }

```

**Event sharing settings**

☒ Private  
This event is only available in the Lambda console and to the event creator. You can configure a total of 10. [Learn more](#)

☐ Shareable  
This event is available to IAM users within the same account who have permissions to access and use shareable events. [Learn more](#)

Template - optional

hello-world

**Event JSON**

Format JSON

```

1 {
2   "path": "/ride",
3   "httpMethod": "POST",
4   "headers": {
5     "Accept": "*/*",
6     "Authorization": "eyJ3aWQ1OjltRVRmZs",
7     "content-type": "application/json; charset=UTF-8"
8   },
9   "queryStringParameters": null,
10  "pathParameters": null,
11  "requestContext": {
12    "authorizer": {
13      "claims": {
14        "cognito:username": "the_username"
15      }
16    }
17  },
18  "body": "{\"PickupLocation\":{\"Latitude\":47.6174755835663,\"Longitude\":-122.288379666501"
19 }

```

Cancel Invoke Save

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

Select a table or index

Table - Rides2

Select attribute projection

All attributes

Filters

Run Reset

Completed. Read capacity units consumed: 0.5

Items returned (1)

Actions Create Item

1

Rideid (String)	me	Unicorn	User
lt_10S9ngZBrkp-h4x...	12T...	{ "Name": {...	the_username

### 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

**API details**

☒ **New API**  
Create a new REST API.

☐ **Clone existing API**  
Create a copy of an API in this AWS account.

☐ **Import API**  
Import an API from an OpenAPI definition.

☐ **Example API**  
Learn about API Gateway with an example API.

API name  
WildRydes2

Description - optional

Regional ☒   
 Edge-optimized   
 Private   
 Regional

Cancel **Create 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

**Create authorizer** [Info](#)

**Authorizer details**

Authorizer name  
WildRydes

Authorizer type [Info](#)  
Choose to authorize your API calls using one of your Lambda functions or a Cognito User Pool.

☐ Lambda  
☒ **Cognito**

Cognito user pool  
Select the Cognito user pool that will authenticate requests to your API.

us-west-2

Token source  
Enter the header that contains the authorization token.  
Authorization

Token validation - optional  
Enter a regular expression to validate tokens.

Cancel **Create authorizer**

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

API Gateway > APIs > Resources - WildRydes2 (Annpiep531) > Create resource

**Create resource**

**Resource details**

☒ **Proxy resource** [Info](#)  
Proxy resources handle requests to all sub-resources. To create a proxy resource use a path parameter that ends with a plus sign, for example {proxy+}.

Resource path  
/

Resource name  
ride

☒ **CORS (Cross Origin Resource Sharing)** [Info](#)  
Create an OPTIONS method that allows all origins, all methods, and several common headers.

Cancel **Create resource**

- 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

**Create method**

**Method details**

Method type  
POST

Integration type

☒ **Lambda function**  
Integrate your API with a Lambda function.

☐ HTTP  
Integrate with an existing HTTP endpoint.

☐ Mock  
Generate a response based on API Gateway mappings and transformations.

☐ AWS service  
Integrate with an AWS Service.

☐ VPC link  
Integrate with a resource that isn't accessible over the public internet.

[Lambda proxy integration](#)

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



API Gateway > APIs > Resources - WildRydes2 (knoopp9531) > Edit method request

### Edit method request

**Method request settings**

Authorization  
WildRydes

Authorization Scopes  
Add a scope Add

Request validator  
None

☐ API key required

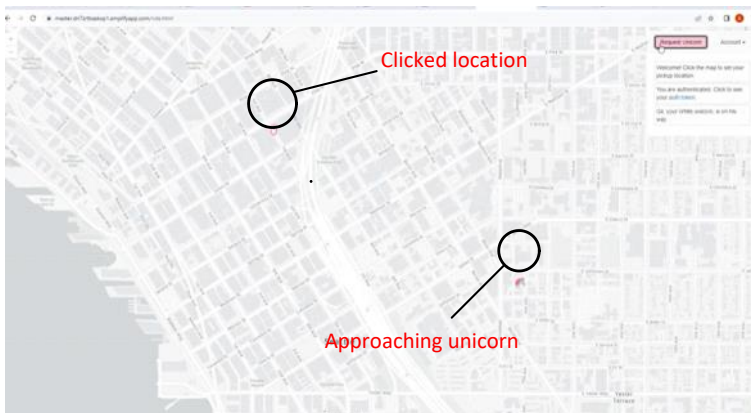
Operation name - optional  
GetPets

#### 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

Filters

Run Reset

Completed. Read capacity units consumed: 0.5

Items returned (2)

	RideId (String)	RequestTime	Unicorn	Use
<input type="checkbox"/>	lt_10S9ngZBrkp-h4x...	2023-11-02T...	{ "Name": { ...	the
<input type="checkbox"/>	7_pxUpE4MXf...	2023-11-02T...	{ "Name": { ...	aisr