Cloud Plus Plus Services

Tutorial to build a serverless web application

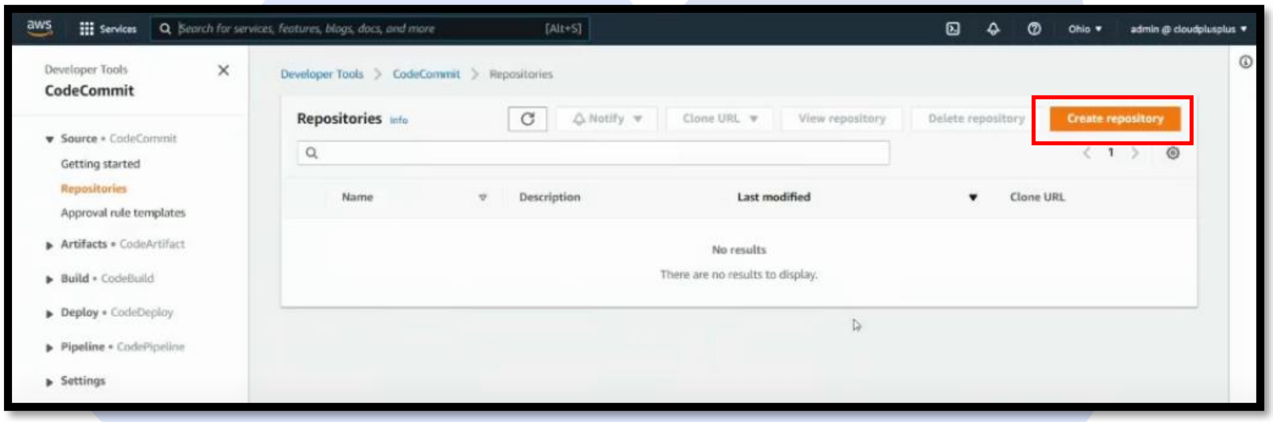
Tutorial Objectives:

1. Learn to build a serverless web application using Lambda, API Gateway, DynamoDB, Cognito and Amplify.

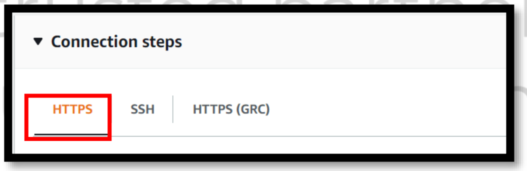
Step 1: Host a Static Website

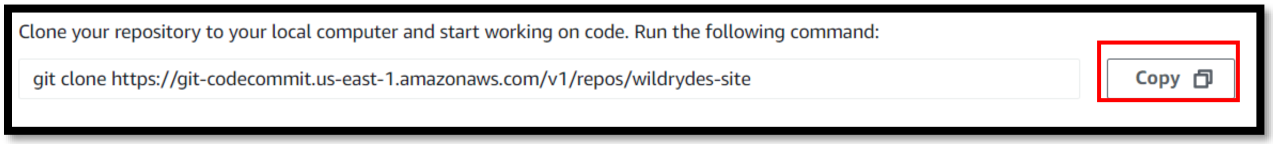
1. Log on to your AWS Management Console and Select a Region: **N. Virginia**.

- Search for AWS CodeCommit and open the console
- Click on Create Repository



- Repository Name: **wildrydes-site** and create Repository and copy the URL.



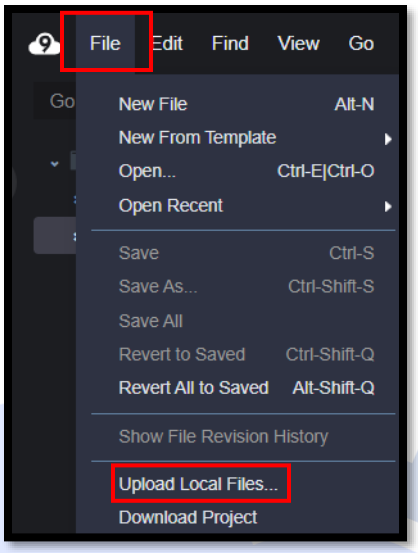

Clone your repository to your local computer and start working on code. Run the following command:

```
git clone https://git-codecommit.us-east-1.amazonaws.com/v1/repos/wildrydes-site
```

2. Open your AWS management console in another new tab and navigate to Cloud9.Create environment

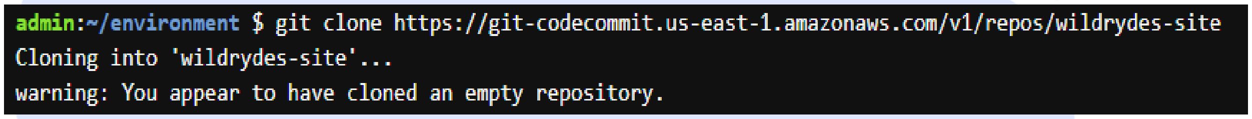
- Name: **MyCloud9env**, click on Next Step->Next Step->Create environment.
- Download **wildrydes-site.zip** from here into your local machine.

- Once the environment is up and running, Click on the File->Upload local files…



And select downloaded wildrydes-site.zip folder and click upload.

- Now run the previously copied git clone command in the cloud9 terminal.

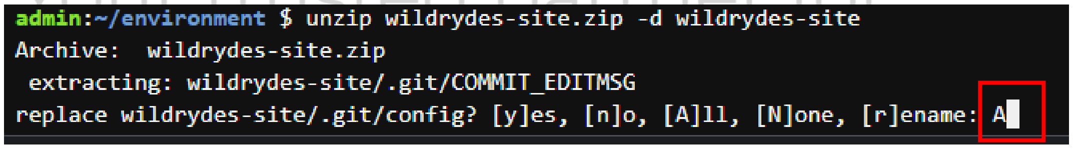


```
admin:~/environment $ git clone https://git-codecommit.us-east-1.amazonaws.com/v1/repos/wildrydes-site
Cloning into 'wildrydes-site'...
warning: You appear to have cloned an empty repository.
```

- Run the following command to unzip the wildrydes-site.zip file.

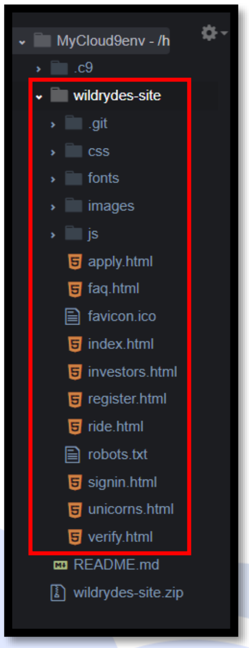
unzip wildrydes-site.zip -d wildrydes-site

- When prompted enter **A** and click enter



```
admin:~/environment $ unzip wildrydes-site.zip -d wildrydes-site
Archive:  wildrydes-site.zip
 extracting: wildrydes-site/.git/COMMIT_EDITMSG
replace wildrydes-site/.git/config? [y]es, [n]o, [A]ll, [N]one, [r]ename: A
```

- Once the process finished, click on the wildrydes-site folder and check the folder.



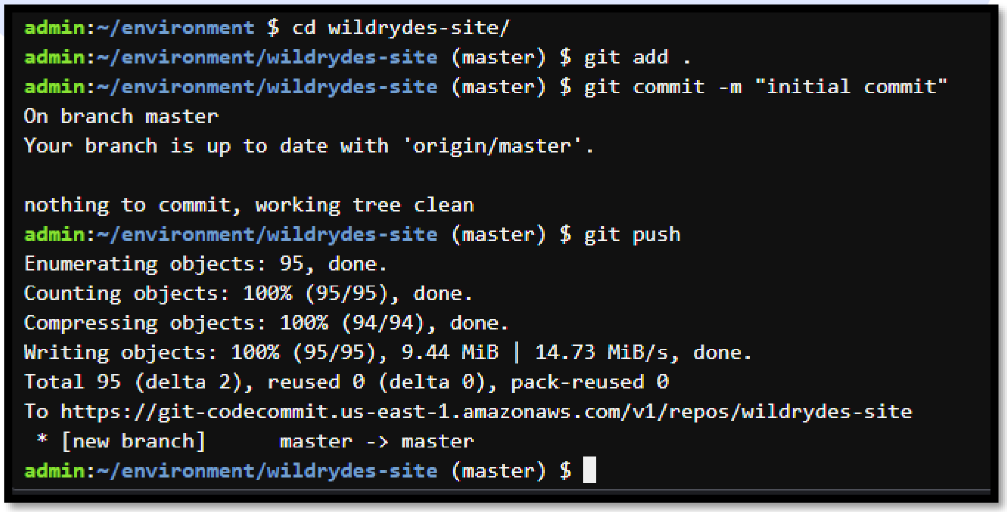
Now, Run the following command

cd wildrydes-site/

git add .

git commit -m "initial commit"

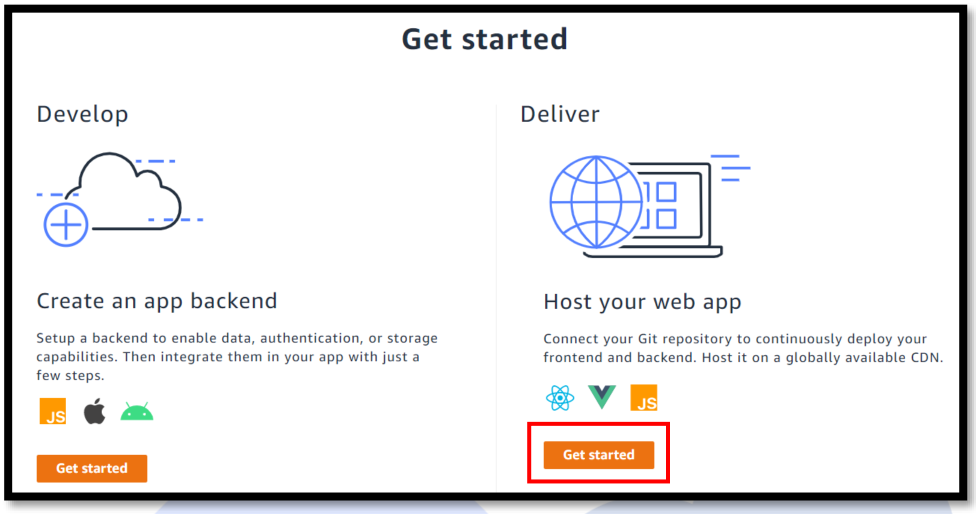
git push



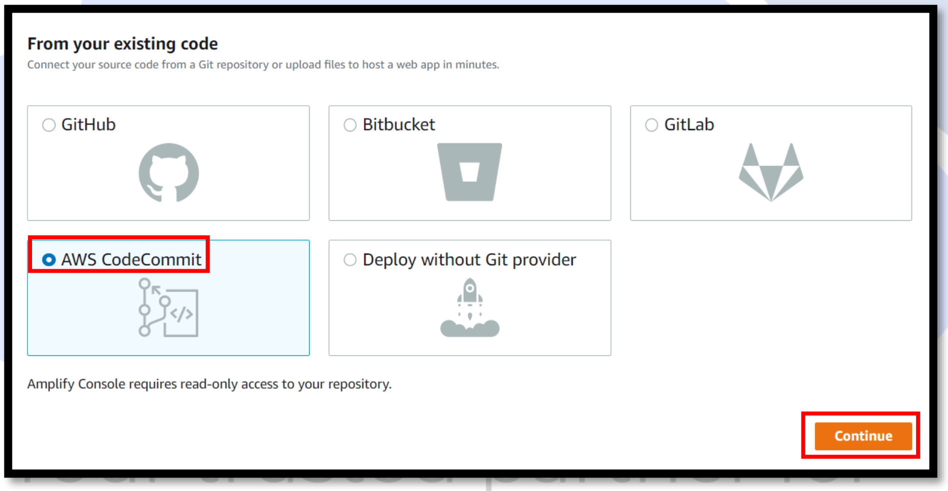
```
admin:~/environment $ cd wildrydes-site/
admin:~/environment/wildrydes-site (master) $ git add .
admin:~/environment/wildrydes-site (master) $ git commit -m "initial commit"
On branch master
Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean
admin:~/environment/wildrydes-site (master) $ git push
Enumerating objects: 95, done.
Counting objects: 100% (95/95), done.
Compressing objects: 100% (94/94), done.
Writing objects: 100% (95/95), 9.44 MiB | 14.73 MiB/s, done.
Total 95 (delta 2), reused 0 (delta 0), pack-reused 0
To https://git-codecommit.us-east-1.amazonaws.com/v1/repos/wildrydes-site
 * [new branch]      master -> master
admin:~/environment/wildrydes-site (master) $
```

3. In the new tab open AWS Amplify console, under **Deliver** click on **Get Started.**


Get started
Develop
Create an app backend
Setup a backend to enable data, authentication, or storage capabilities. Then integrate them in your app with just a few steps.
Get started
Deliver
Host your web app
Connect your Git repository to continuously deploy your frontend and backend. Host it on a globally available CDN.
Get started

- Select **AWS CodeCommit** and click Continue.

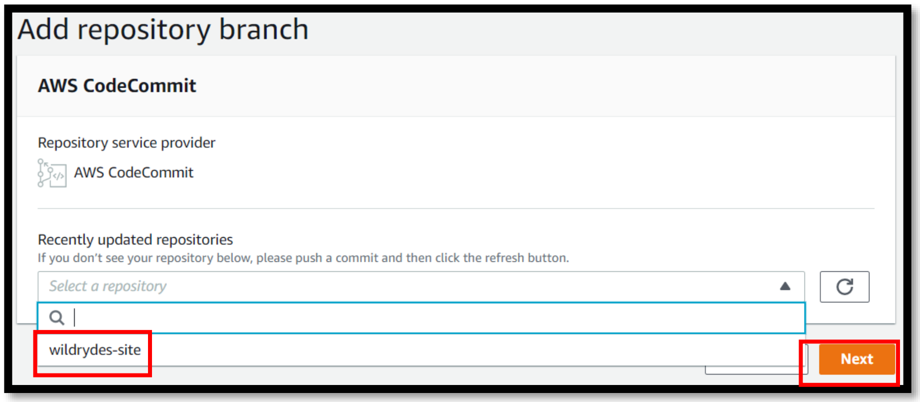

From your existing code

Connect your source code from a Git repository or upload files to host a web app in minutes.

GitHub | Bitbucket | GitLab | [✓ AWS CodeCommit] | Deploy without Git provider

Amplify Console requires read-only access to your repository.

- Select repo from the dropdown and click **Next**. Keep branch as **master** and **Next**.


Add repository branch

AWS CodeCommit

Repository service provider

AWS CodeCommit

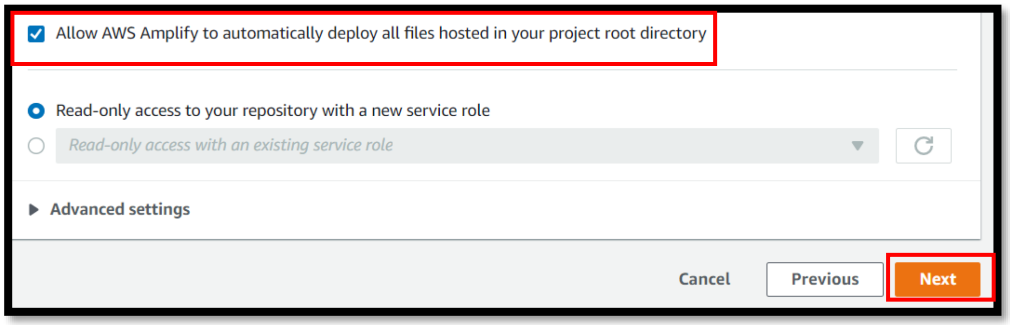
Recently updated repositories

If you don't see your repository below, please push a commit and then click the refresh button.

Select a repository

wildrydes-site

- Enable **Allow AWS Amplify to automatically deploy all files hosted in your project root directory** and Next->Save and Deploy.

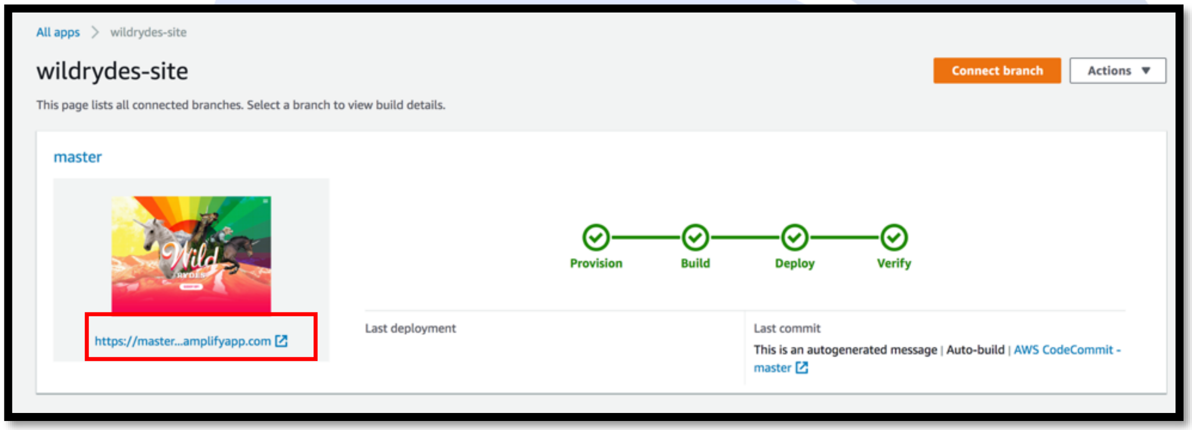

☑ Allow AWS Amplify to automatically deploy all files hosted in your project root directory

● Read-only access to your repository with a new service role
○ Read-only access with an existing service role

▸ Advanced settings

It will take a couple of minutes for Amplify Console to create the necessary resources and to deploy your code.

- Once completed, click on the site image to launch your Wild Rydes site.


All apps > wildrydes-site

wildrydes-site

This page lists all connected branches. Select a branch to view build details.

master

https://master...amplifyapp.com

Last deployment

Last commit
This is an autogenerated message | Auto-build | AWS CodeCommit - master

<u>Step 2:</u> Create an Amazon Cognito User Pool

1. In the new tab, Open **Amazon Cognito** console

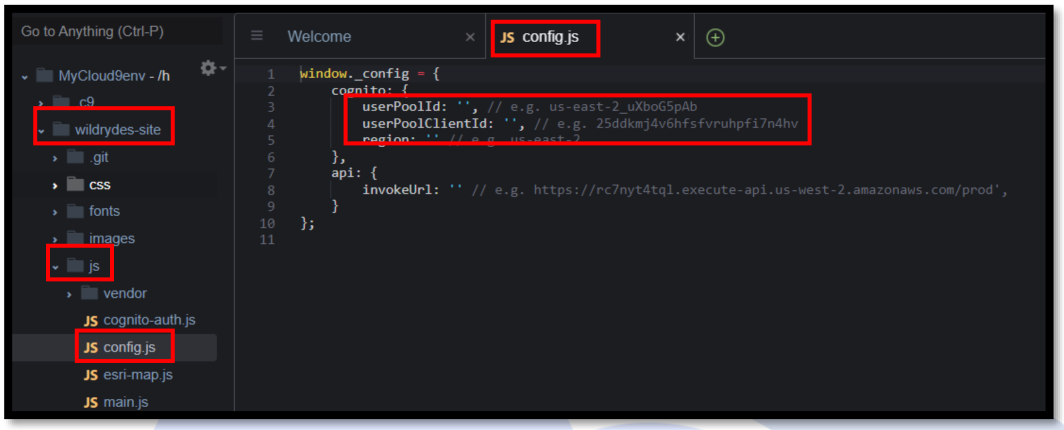
- Choose **Manage your User Pools**
- Choose **Create a User Pool**
- User pool name: **WildRydes**, then select **Review Defaults**
- On the review page, click **Create pool**
- Note the **Pool Id**.

2. To Add an App to user pool

- Select **App clients** from the left General Settings section in the navigation bar.
- Choose **Add an app client**.
- App client name: **WildRydesWebApp**.

- Uncheck the Generate client secret option.
- Choose **Create app client**.
- Note the **App client id**.

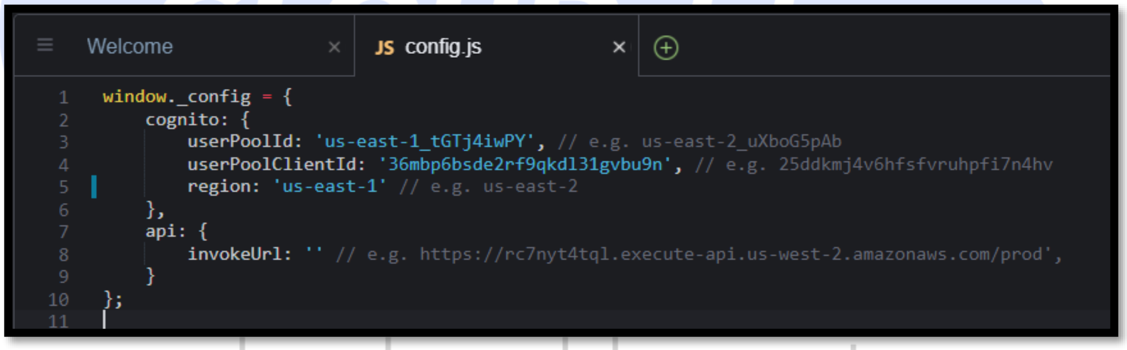
In the Cloud9 console, open the open '**wild-ryde-site/js/config.js'**



```
window._config = {
    cognito: {
        userPoolId: '', // e.g. us-east-2_uXboG5pAb
        userPoolClientId: '', // e.g. 25ddkmj4v6hfsfvruhpfi7n4hv
        region: '' // e.g. us-east-2
    },
    api: {
        invokeUrl: '' // e.g. https://rc7nyt4tql.execute-api.us-west-2.amazonaws.com/prod',
    }
};
```

- Update the cognito section with the correct values for the region, user pool id, userpoolClientId, you just created and click control+s.

Updated config.js file should look like this:



```
window._config = {
    cognito: {
        userPoolId: 'us-east-1_tGTj4iwPY', // e.g. us-east-2_uXboG5pAb
        userPoolClientId: '36mbp6bsde2rf9qkdl31gvbu9n', // e.g. 25ddkmj4v6hfsfvruhpfi7n4hv
        region: 'us-east-1' // e.g. us-east-2
    },
    api: {
        invokeUrl: '' // e.g. https://rc7nyt4tql.execute-api.us-west-2.amazonaws.com/prod',
    }
};
```

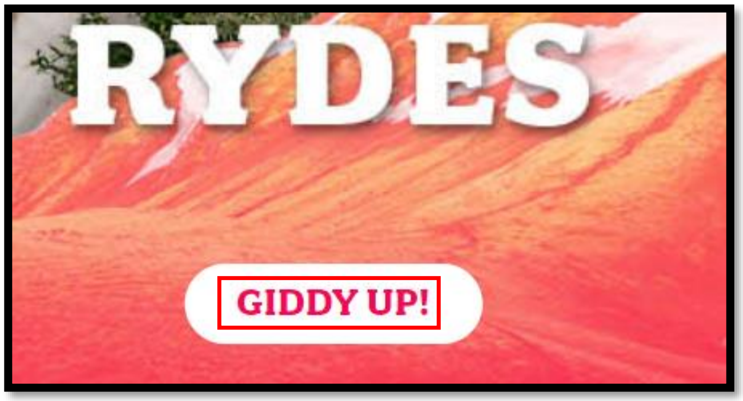
- Save the modified file and push it to repository to have it automatically deploy to Amplify Console

git add .

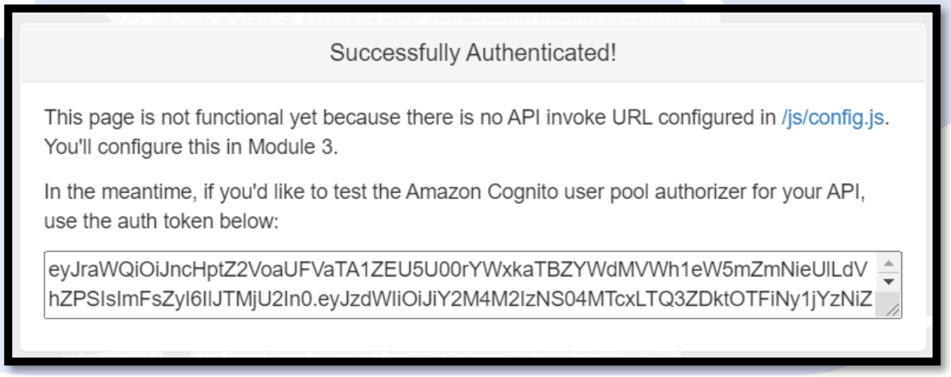
git commit -m "Updated config.js"

git push

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



- Complete the registration form by entering Email, Password and confirm the password and click to **Lets RYde!**
- Verify email by entering code.
- After successful verification Sign in using email and password you entered.
- If successful you should be redirected to /ride.html. You should see a notification that the API is not configured.



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:

```
eyJraWQiOiJncHptZ2VoaUFVaTA1ZEU5U00rYWxkaTBZYWdMVWh1eW5mZmNieUlLdV
hZPSIsImFsZyI6IlJTMjU2In0.eyJzdWIiOiJiY2M4M2IzNS04MTcxLTQ3ZDktOTFiNy1jYzNiZ
```

Step 3: **Serverless Service Backend**

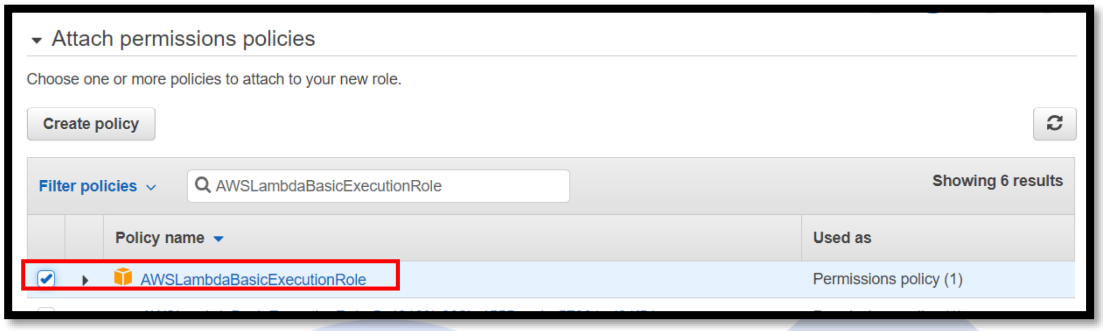
1. Create Amazon DynamoDB Table

- Open **Amazon DynamoDB** Service in new tab
- Choose **Create table**
- Table Name: **Rides**
- Enter **RideId** for the Partition key and select String for the key type.
- Check the Use default settings box and choose Create.
- Scroll to the bottom of the Overview section of new table and **note the ARN**. We will use this in the next section.

2. Create an IAM Role for Lambda function

- Open Amazon **IAM** Service in new tab
- Select **Roles** and create new role

- Select **Lambda** for the role type from the AWS service group, then click Next: Permissions.
- In the attach permission policy, search for **AWSLambdaBasicExecutionRole** and click the check box.



Attach permissions policies

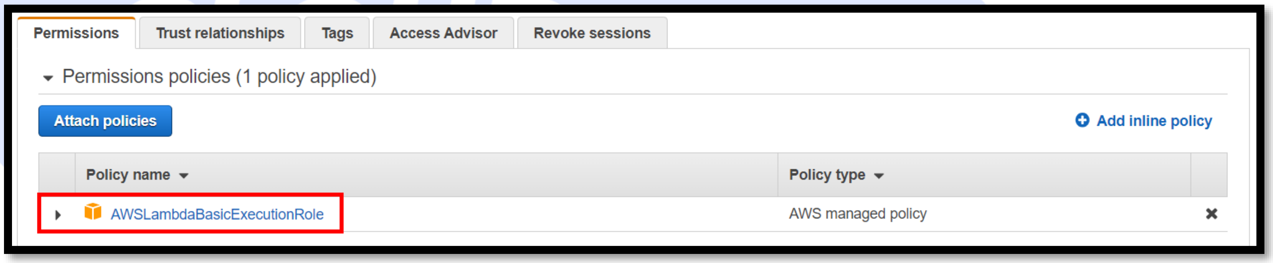
Choose one or more policies to attach to your new role.

Create policy

Filter policies AWSLambdaBasicExecutionRole Showing 6 results

	Policy name	Used as
✓	AWSLambdaBasicExecutionRole	Permissions policy (1)

- Choose Next Step.
- Role Name: **WildRydesLambda**
- Choose Create Role.
- Type **WildRydesLambda** into the filter box on the Roles page and choose the role.
- On the Permissions tab, choose the Add inline policy link to create a new inline policy.



Permissions | Trust relationships | Tags | Access Advisor | Revoke sessions

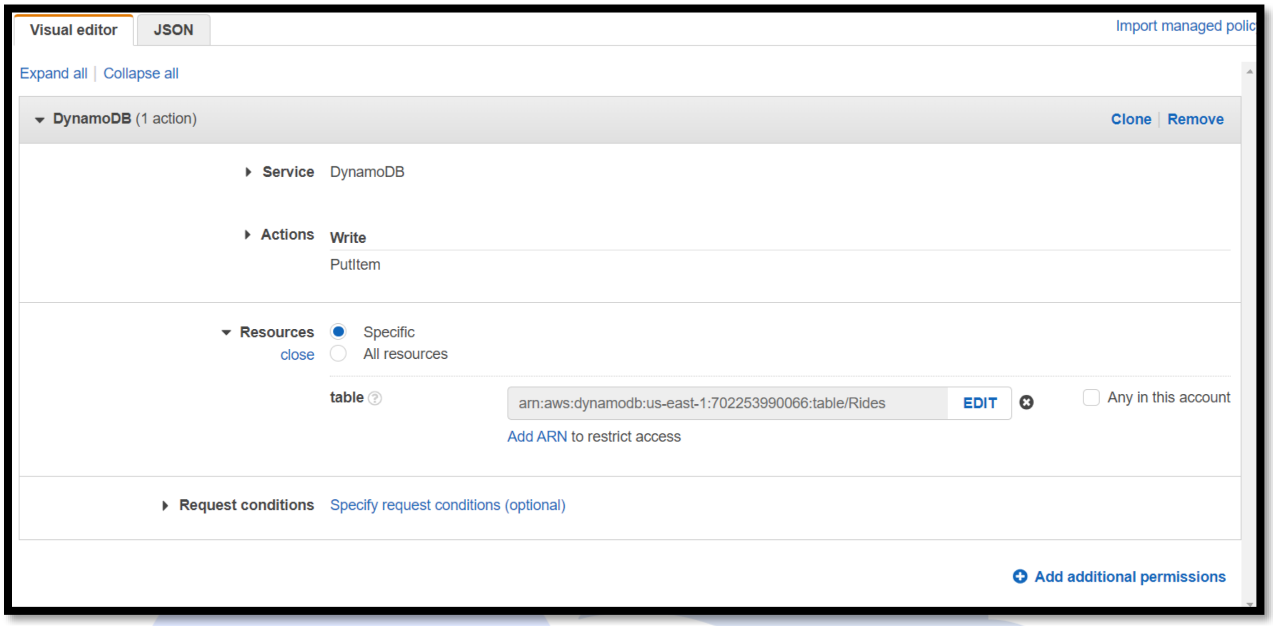
Permissions policies (1 policy applied)

Attach policies

Add inline policy

Policy name	Policy type
AWSLambdaBasicExecutionRole	AWS managed policy

- Select **Choose a service**.
- Type DynamoDB into the search box labeled Find a service and select **DynamoDB**.
- Choose **Select actions**.
- Type **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 you created in the previous section in the Specify ARN for table field, and choose **Add**.
- Choose Review Policy.
- Enter **Policy name: DynamoDBWriteAccess** and choose **Create policy**.



Visual editor | JSON

Import managed polic

Expand all | Collapse all

DynamoDB (1 action) — Clone | Remove

Service: DynamoDB

Actions: Write — PutItem

Resources: [✓ Specific] All resources

table: arn:aws:dynamodb:us-east-1:702253990066:table/Rides — EDIT — Any in this account

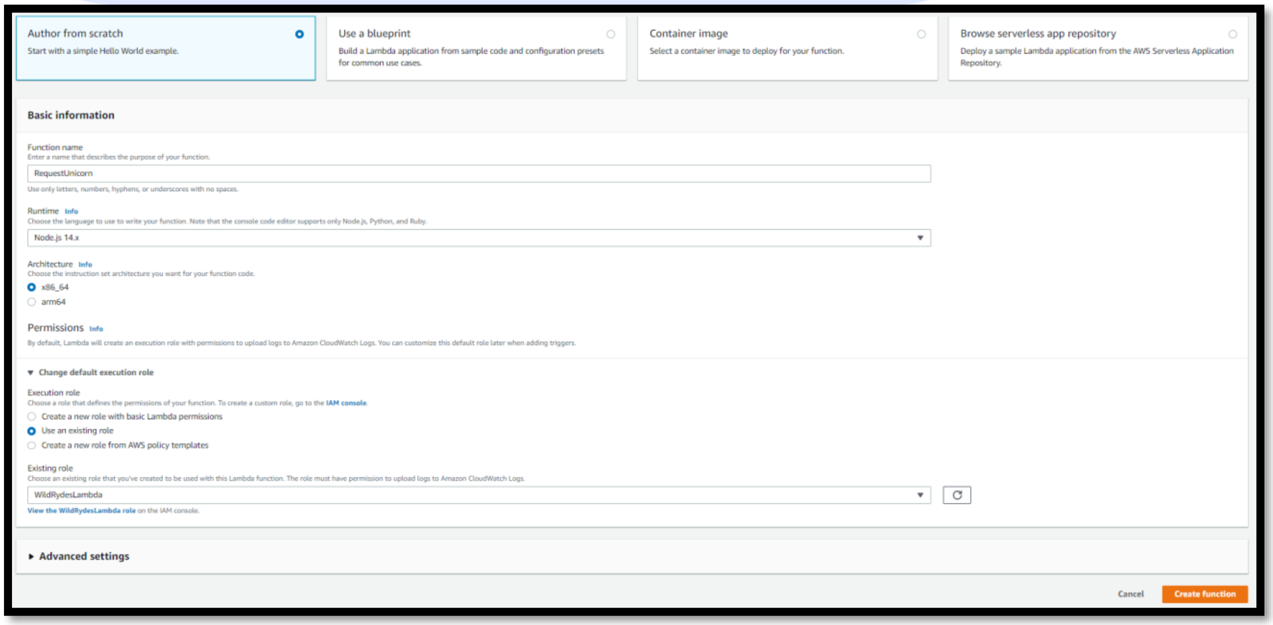
Add ARN to restrict access

Request conditions: Specify request conditions (optional)

Add additional permissions

3. Create a Lambda Function for Handling Requests

- Open **AWS Lambda** service in new tab
- Choose **Create Function**
- Keep the **default Author from scratch** card selected.
- Enter **RequestUnicorn** in the Name field.
- Select **Node.js 14.x** for the Runtime.
- Choose **use an existing role** from the Role dropdown.
- Select **WildRydesLambda** from the Existing Role dropdown.
- Click on Create function.



[✓ Author from scratch — Start with a simple Hello World example.]

Use a blueprint — Build a Lambda application from sample code and configuration presets for common use cases.

Container image — Select a container image to deploy for your function.

Browse serverless app repository — Deploy a sample Lambda application from the AWS Serverless Application Repository.

Basic information

Function name
Enter a name that describes the purpose of your function.
RequestUnicorn
Use only letters, numbers, hyphens, or underscores with no spaces.

Runtime Info
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.
Node.js 14.x

Architecture Info
Choose the instruction set architecture you want for your function code.
[✓ x86_64] arm64

Permissions Info
By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

Change default execution role

Execution role
Choose a role that defines the permissions of your function. To create a custom role, go to the IAM console.
Create a new role with basic Lambda permissions
[✓ Use an existing role]
Create a new role from AWS policy templates

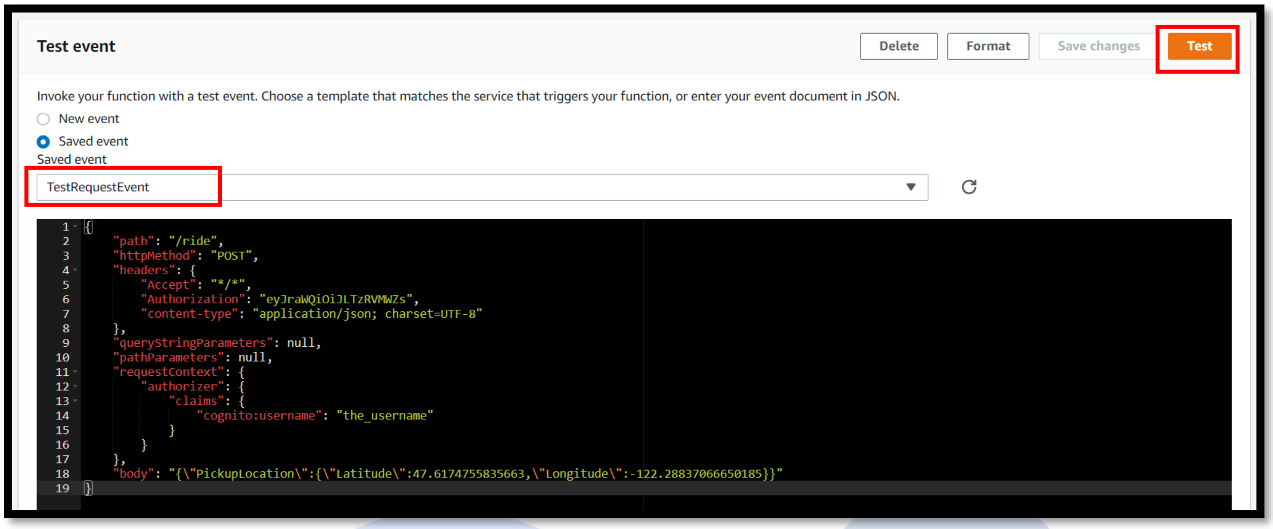
Existing role
Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.
WildRydesLambda
View the WildRydesLambda role on the IAM console.

Advanced settings

Cancel | Create function

- Scroll down to the Function code section and replace the existing code in the index.js code editor with the code in the requestUnicorn.txt. Download **requestUnicorn.txt** file from **here**
- Click "**Save**" in the upper right corner of the page.
- Click to **test**, choose **new event**
- Named it as **TestRequestEvent**
- 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}}"
}
```



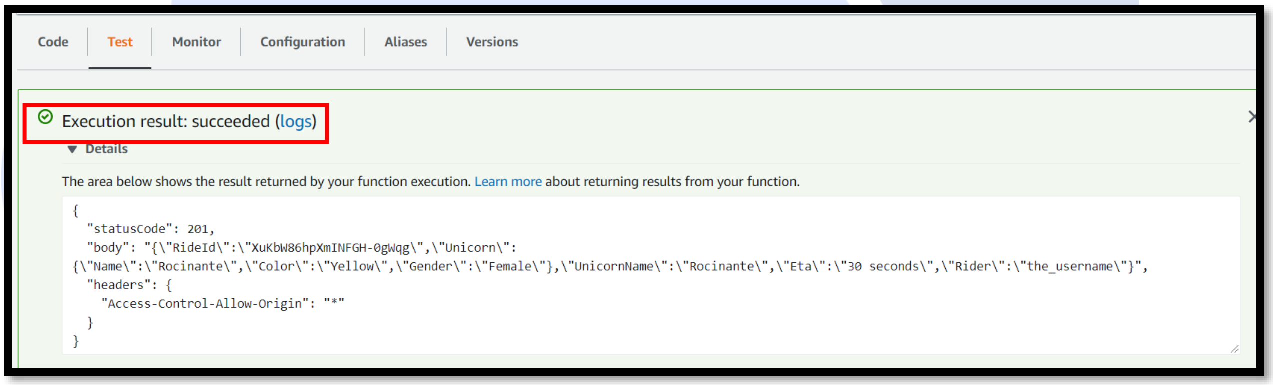
Invoke your function with a test event. Choose a template that matches the service that triggers your function, or enter your event document in JSON.

( ) New event
(●) Saved event

Saved event: TestRequestEvent

```
{
    "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}}"
}
```

- Save the changes and **Test**
- You will get the Execution result as succeeded as below.
- Deploy the code.



Execution result: succeeded (logs)

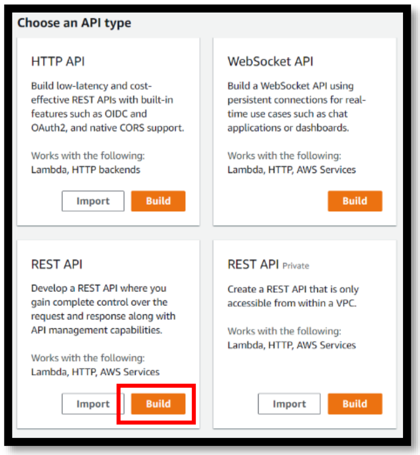
The area below shows the result returned by your function execution. Learn more about returning results from your function.

```
{
  "statusCode": 201,
  "body": "{\"RideId\":\"XuKbW86hpXmINFGH-0gWqg\",\"Unicorn\":
{\"Name\":\"Rocinante\",\"Color\":\"Yellow\",\"Gender\":\"Female\"},\"UnicornName\":\"Rocinante\",\"Eta\":\"30 seconds\",\"Rider\":\"the_username\"}",
  "headers": {
    "Access-Control-Allow-Origin": "*"
  }
}
```

Step 4: Deploy a RESTful API

1. Create a new REST API

- In the AWS Management Console, click Services then select **API Gateway** under Application Services.
- Choose an API type as **REST API** Click on **Build**



Choose an API type

HTTP API

Build low-latency and cost-effective REST APIs with built-in features such as OIDC and OAuth2, and native CORS support.

Works with the following: Lambda, HTTP backends

WebSocket API

Build a WebSocket API using persistent connections for real-time use cases such as chat applications or dashboards.

Works with the following: Lambda, HTTP, AWS Services

REST API

Develop a REST API where you gain complete control over the request and response along with API management capabilities.

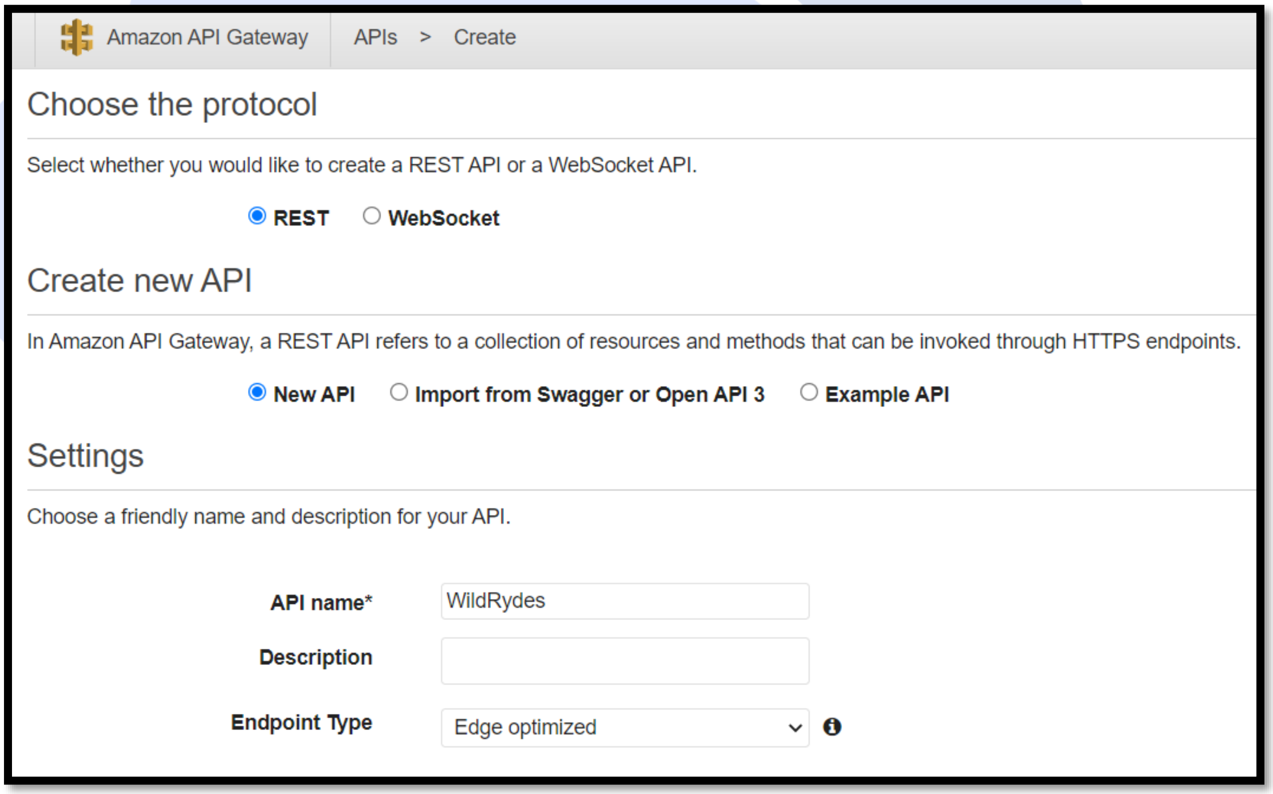
Works with the following: Lambda, HTTP, AWS Services

REST API Private

Create a REST API that is only accessible from within a VPC.

Works with the following: Lambda, HTTP, AWS Services

- Choose the **Protocol: REST**
- Create **New API: New API**
- **API Name: WildRydes**
- **Endpoint Type: Edge optimized**
- Click **Create API**



Choose the protocol

Select whether you would like to create a REST API or a WebSocket API.

[✓ REST] WebSocket

Create new API

In Amazon API Gateway, a REST API refers to a collection of resources and methods that can be invoked through HTTPS endpoints.

[✓ New API] Import from Swagger or Open API 3 Example API

Settings

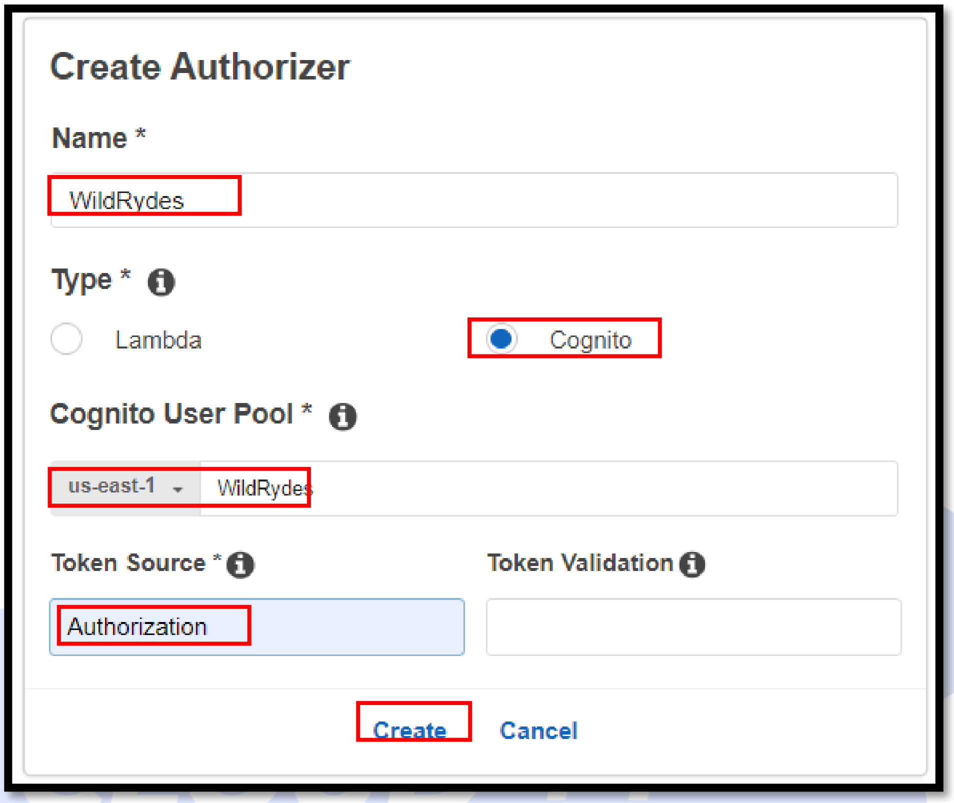
Choose a friendly name and description for your API.

API name*	WildRydes
Description	
Endpoint Type	Edge optimized

2. Create a Cognito User Pools Authorizer

- Under your newly created API, choose **Authorizers**.
- Choose Create New Authorizer
- **Authorizer name: WildRydes**
- **Type: Cognito**

- **Region: us-east-1**
- Enter **WildRydes** in the Cognito User Pool input.
- **Token Source: Authorization** and click on **create**.



Create Authorizer

Name *: WildRydes

Type *: Lambda [✓ Cognito]

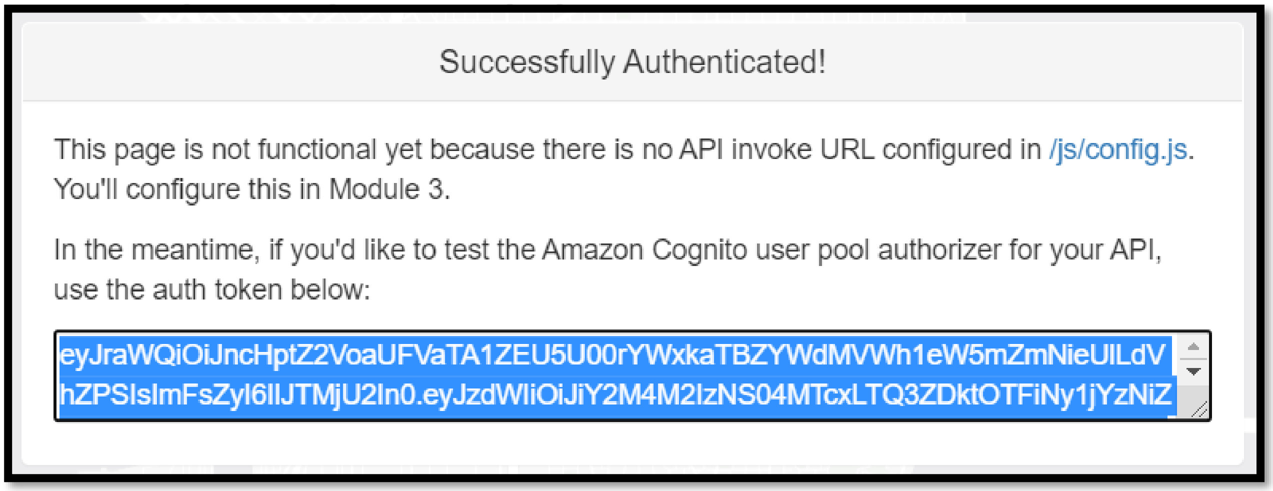
Cognito User Pool *: us-east-1 WildRydes

Token Source *: Authorization

Token Validation:

To Verify Authorizer configuration

- Open a new browser tab and visit **/ride.html**
- If you are redirected to the sign-in page, sign in with the user you created in the last module. You will be redirected back to /ride.html.
- Copy the **auth token** from the notification on the /ride.html



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.

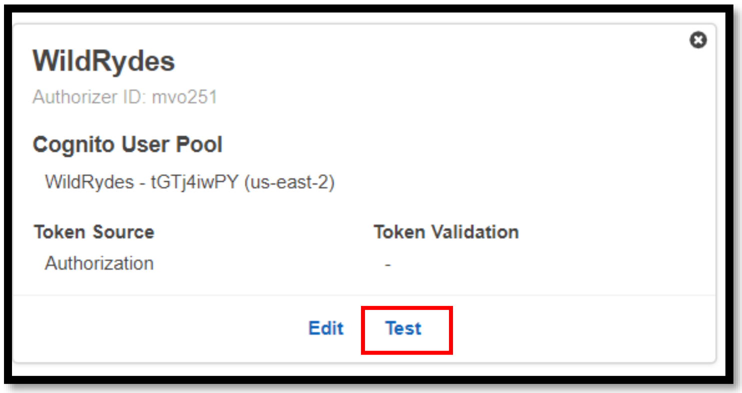
In the meantime, if you'd like to test the Amazon Cognito user pool authorizer for your API, use the auth token below:

```
eyJraWQiOiJncHptZ2VoaUFVaTA1ZEU5U00rYWxkaTBZYWdMVWh1eW5mZmNieUlLdV
hZPSIsImFsZyI6IlJTMjU2In0.eyJzdWIiOiJiY2M4M2IzNS04MTcxLTQ3ZDktOTFiNy1jYzNiZ
```

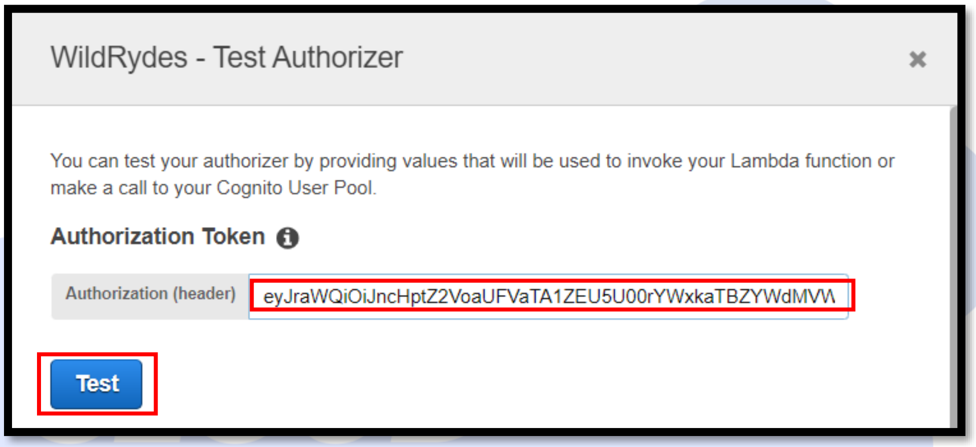
- Go back to API Gateway tab

www.cloud-plusplus.com/aws-training

- Click **Test** at the bottom of the card for the authorizer.



- Paste the auth token into the Authorization Token field in the popup dialog.



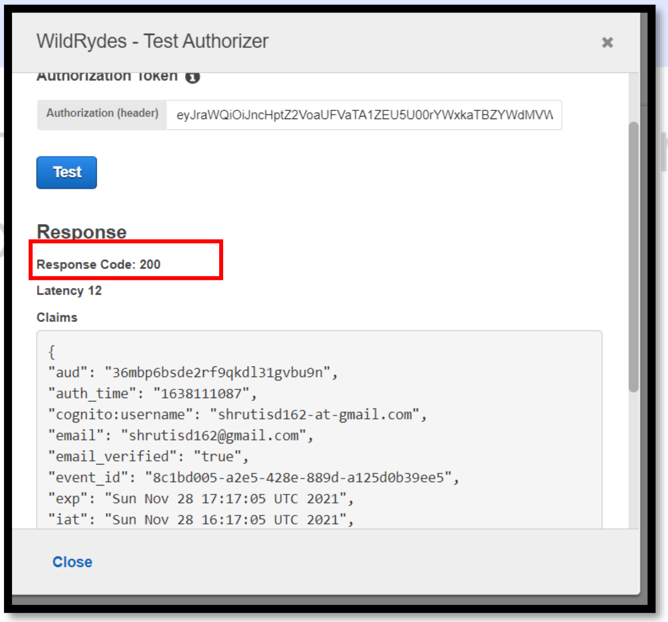
WildRydes - Test Authorizer

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) eyJraWQiOiJncHptZ2VoaUFVaTA1ZEU5U00rYWxkaTBZYWdMVW

- Click **Test** button and verify that the response code is 200.



WildRydes - Test Authorizer

Authorization Token

Authorization (header) eyJraWQiOiJncHptZ2VoaUFVaTA1ZEU5U00rYWxkaTBZYWdMVW

Response

Response Code: 200

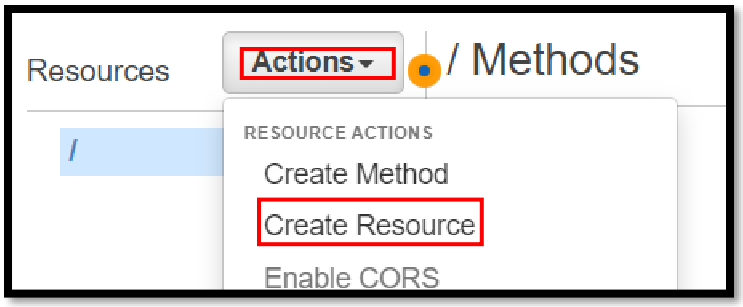
Latency 12

Claims

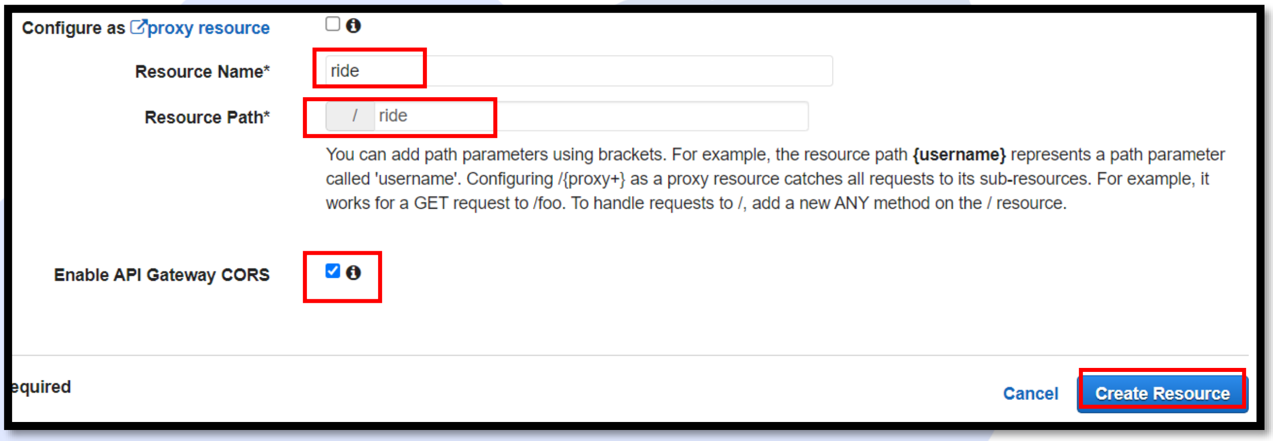
```
{
"aud": "36mbp6bsde2rf9qkdl31gvbu9n",
"auth_time": "1638111087",
"cognito:username": "shrutisd162-at-gmail.com",
"email": "shrutisd162@gmail.com",
"email_verified": "true",
"event_id": "8c1bd005-a2e5-428e-889d-a125d0b39ee5",
"exp": "Sun Nov 28 17:17:05 UTC 2021",
"iat": "Sun Nov 28 16:17:05 UTC 2021",
```

3. Create a new resource and method

- click on **Resources** under your WildRydes API.
- From the Actions dropdown select **Create Resource**.



- **Resource Name: ride**
- Ensure the Resource Path is set to ride.
- Select **Enable API Gateway CORS** for the resource.
- Click Create Resource.



Configure as proxy resource ☐

Resource Name* ride

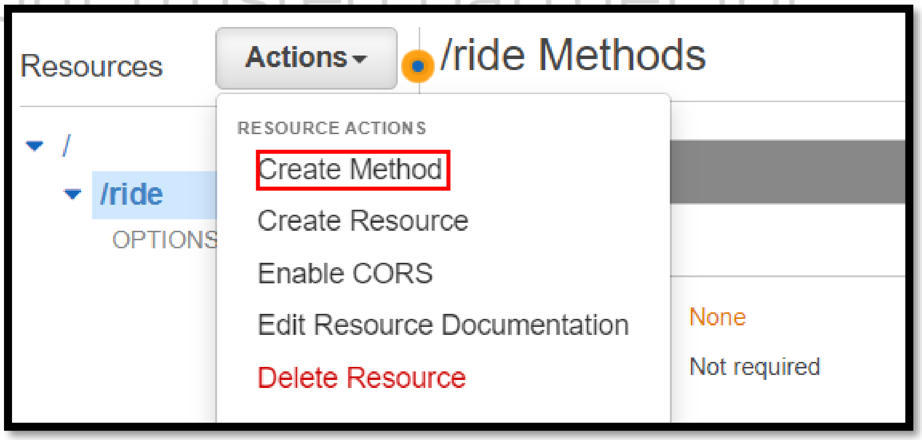
Resource Path* / ride

You can add path parameters using brackets. For example, the resource path **{username}** represents a path parameter called 'username'. Configuring /{proxy+} as a proxy resource catches all requests to its sub-resources. For example, it works for a GET request to /foo. To handle requests to /, add a new ANY method on the / resource.

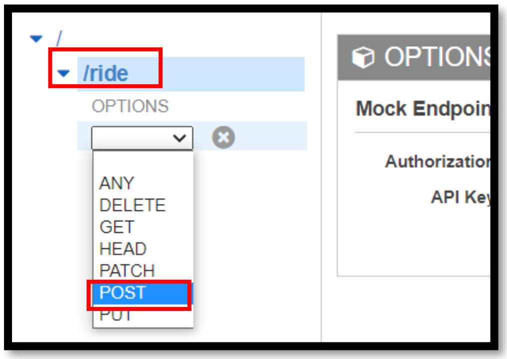
Enable API Gateway CORS ☑

equired

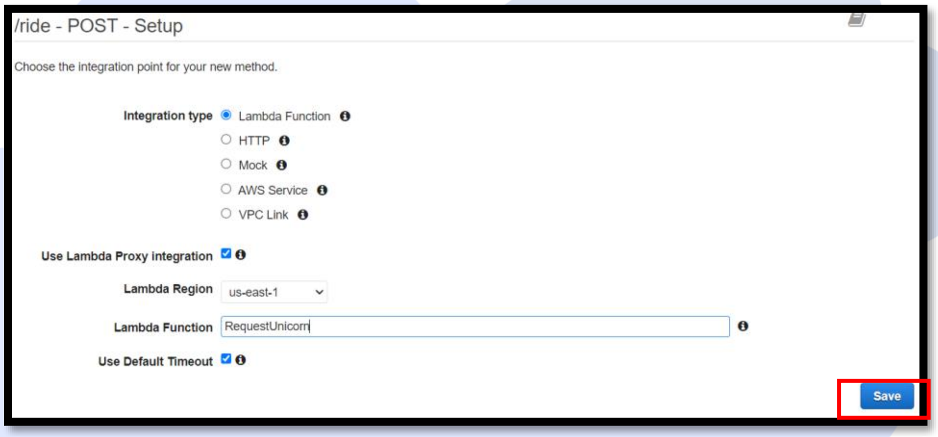
- With the newly created /ride resource selected, from the Action dropdown select **Create Method**.



- Select **POST** from the new dropdown, then click the **checkmark**.



- **Integration type: Lambda Function**
- Check the box for Use Lambda Proxy integration
- **Lambda Region: us-east-1**
- **Lambda Function: RequestUnicorn**
- Click Save.



/ride - POST - Setup

Choose the integration point for your new method.

Integration type: ● Lambda Function ○ HTTP ○ Mock ○ AWS Service ○ VPC Link

Use Lambda Proxy integration ✓

Lambda Region: us-east-1

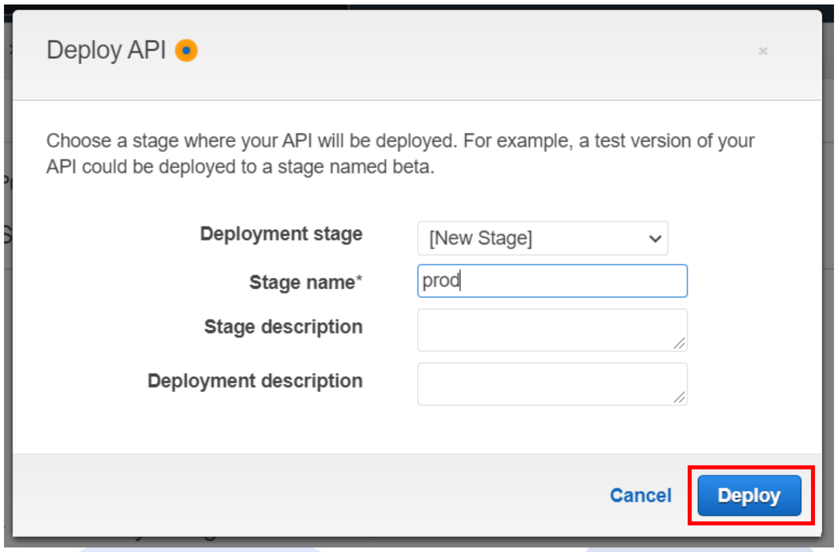
Lambda Function: RequestUnicorn

Use Default Timeout ✓

- When prompted to give Amazon API Gateway permission to invoke your 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.

4. Deploy API

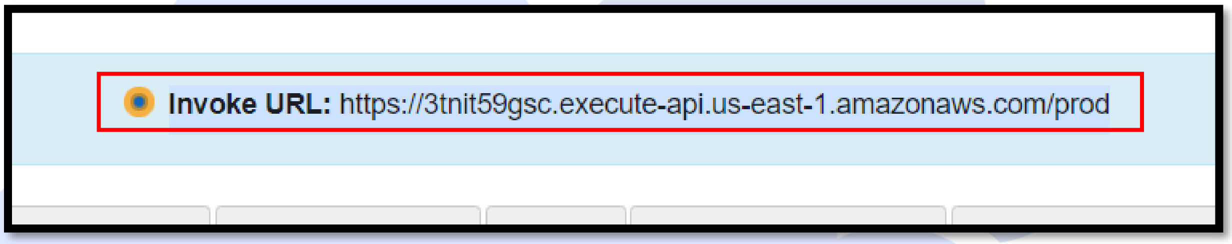
- In the Actions drop-down list select **Deploy API**.
- Select [New Stage] in the Deployment stage drop-down list.
- Enter **Stage Name: prod**
- Choose Deploy.



Choose a stage where your API will be deployed. For example, a test version of your API could be deployed to a stage named beta.

Deployment stage	[New Stage]
Stage name*	prod
Stage description	
Deployment description	

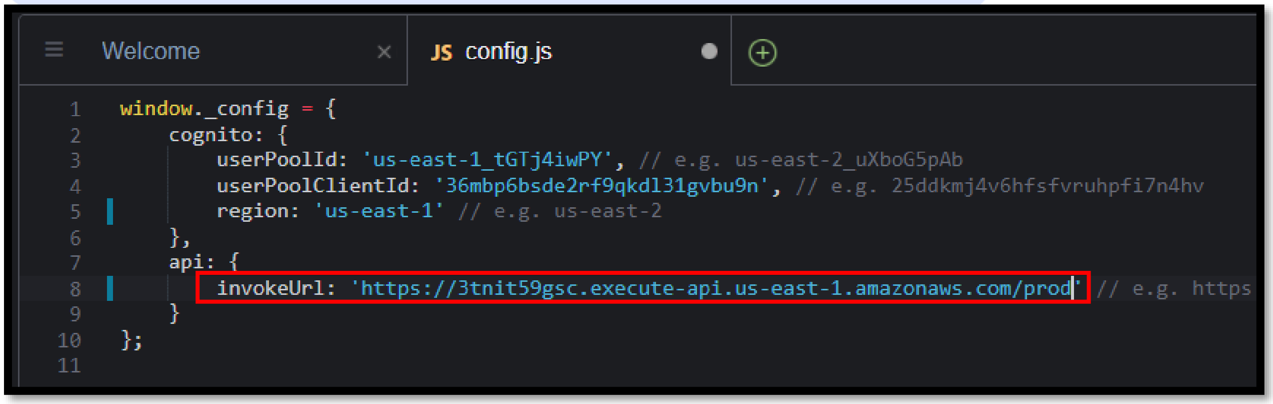
- Note the **Invoke URL**. You will use it in the next section.



Invoke URL: https://3tnit59gsc.execute-api.us-east-1.amazonaws.com/prod

5. Update the website config

- Open the **config.js** file in a cloud9 editor.
- Update the **invokeUrl** setting under the api key in the config.js file.



```
window._config = {
    cognito: {
        userPoolId: 'us-east-1_tGTj4iwPY', // e.g. us-east-2_uXboG5pAb
        userPoolClientId: '36mbp6bsde2rf9qkdl31gvbu9n', // e.g. 25ddkmj4v6hfsfvruhpfi7n4hv
        region: 'us-east-1' // e.g. us-east-2
    },
    api: {
        invokeUrl: 'https://3tnit59gsc.execute-api.us-east-1.amazonaws.com/prod' // e.g. https
    }
};
```

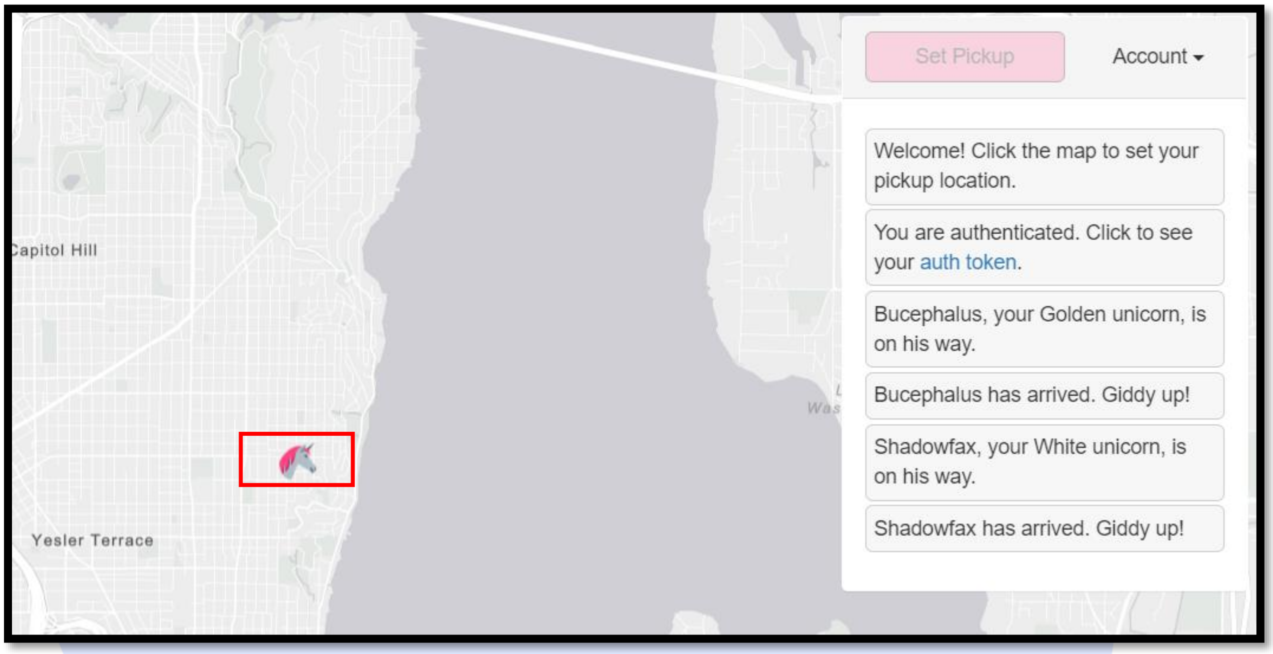
- Save the changes by **ctrl+s** and run the following command

git add .

git commit -m "Updated config.js"

git push

- Now Visit **/ride.html** under your website domain.
- If you are redirected to the sign in page, sign in with the user you created in the previous module.
- After the map has loaded, click **anywhere** on the map to set a pickup location.
- Choose **Request Unicorn**. You should see a notification in the right sidebar that a unicorn is on its way and then see a unicorn icon.



Welcome! Click the map to set your pickup location.

You are authenticated. Click to see your auth token.

Bucephalus, your Golden unicorn, is on his way.

Bucephalus has arrived. Giddy up!

Shadowfax, your White unicorn, is on his way.

Shadowfax has arrived. Giddy up!

Note: If you no longer need the resources, delete DynamoDB Table, API, Lambda Function, Role, User Pool, Repository, Cloud9 environment and wildrydes-site app (Amazon Amplify).

Document Created by	Version
Shruti Dhongade	1-December-2021