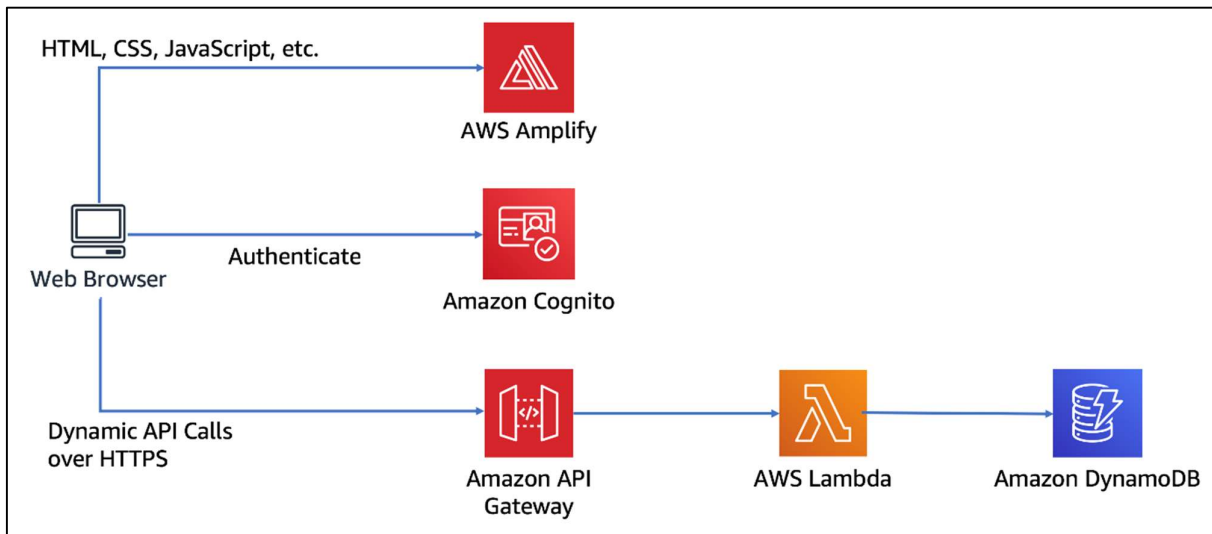


SETUP GUIDE

WildRydes is a simple web application that enables users to request unicorn rides. The application presents users with an HTML based user interface for indicating the location where they would like to be picked up and will interface on the backend with a RESTful web service to submit the request and dispatch a nearby unicorn. The application will also provide facilities for users to register with the service and log in before requesting rides.

ARCHITECTURE DIAGRAM

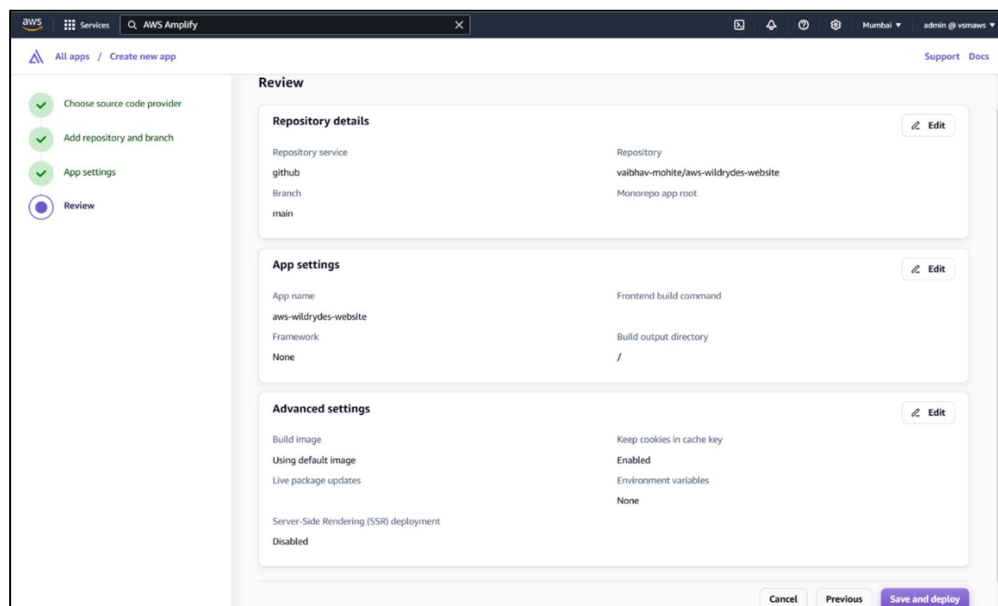


GIT REPOSITORY

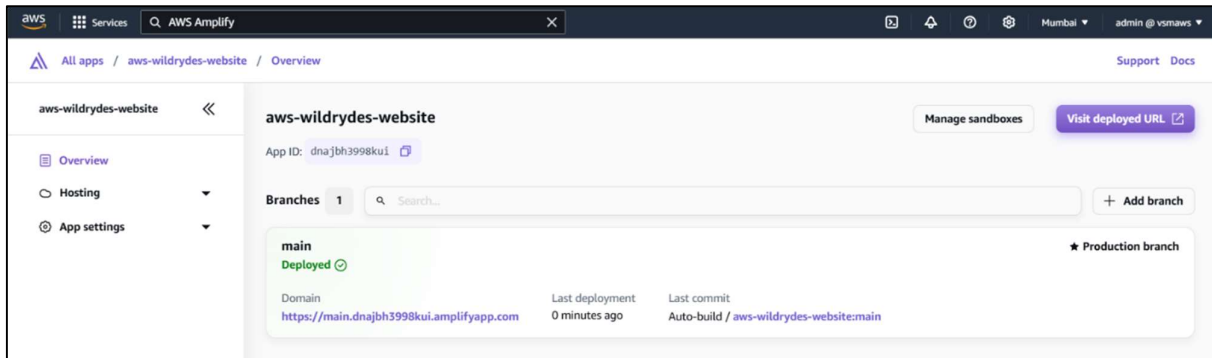
1. The repository can be found at <https://github.com/vaibhav-mohite/aws-wildrydes-website>

AMPLIFY

1. Launch the Amplify Console
2. Click on Deploy an App
3. Select GitHub as Git provider



4. Connect your GitHub Account to AWS. Select the GitHub Repository.
5. Give App name as per your choice and click Save and Deploy
6. Application will be Deployed.

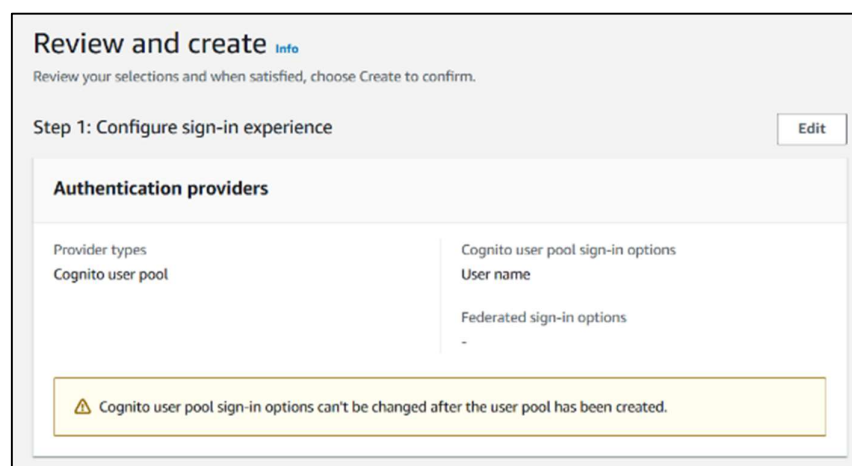


7. Click on Domain to open the website.



COGNITO

1. Launch Cognito Console
2. Click on "Create User Pool"
3. Select "Cognito user pool sign-in options" as Username



4. Select No MFA in Multi-factor authentication
5. Select Email Provider as "Send email with Cognito"

Step 4: Configure message delivery Edit


Email [Info](#)

Email provider	FROM email address
Send email with Cognito	no-reply@verificationemail.com
SES Region	REPLY-TO email address
Asia Pacific (Mumbai)	-

Step 5: Integrate your app Edit

User pool name

User pool name
wildrydes-cognito-userpool

 Your user pool name can't be changed once this user pool is created.






6. Keep rest options as default and Create Userpool
7. Once the Userpool is created, note down User pool ID and App Client ID. App Client ID can be found at User Pool → App integration → App client list

[Amazon Cognito](#) > [User pools](#) > wildrydes-cognito-userpool

wildrydes-cognito-userpool [Info](#)

Delete user pool

User pool overview

User pool name wildrydes-cognito-userpool	Token signing key URL  https://cognito-idp.ap-south-1.amazonaws.com/ap-south-1_ivrsEjYpF/.well-known/jwks.json 	Created time November 10, 2024 at 15:55 GMT+5:30
User pool ID  ap-south-1_ivrsEjYpF	Estimated number of users 0	Last updated time November 10, 2024 at 15:55 GMT+5:30
ARN  arn:aws:cognito-idp:ap-south-1:723330070958:userpool/ap-south-1_ivrsEjYpF	Advanced security  Inactive	

App client list

The app clients that integrate your apps with your user pool. Configure client overrides to user pool default configurations, and configure Amazon Pinpoint analytics.

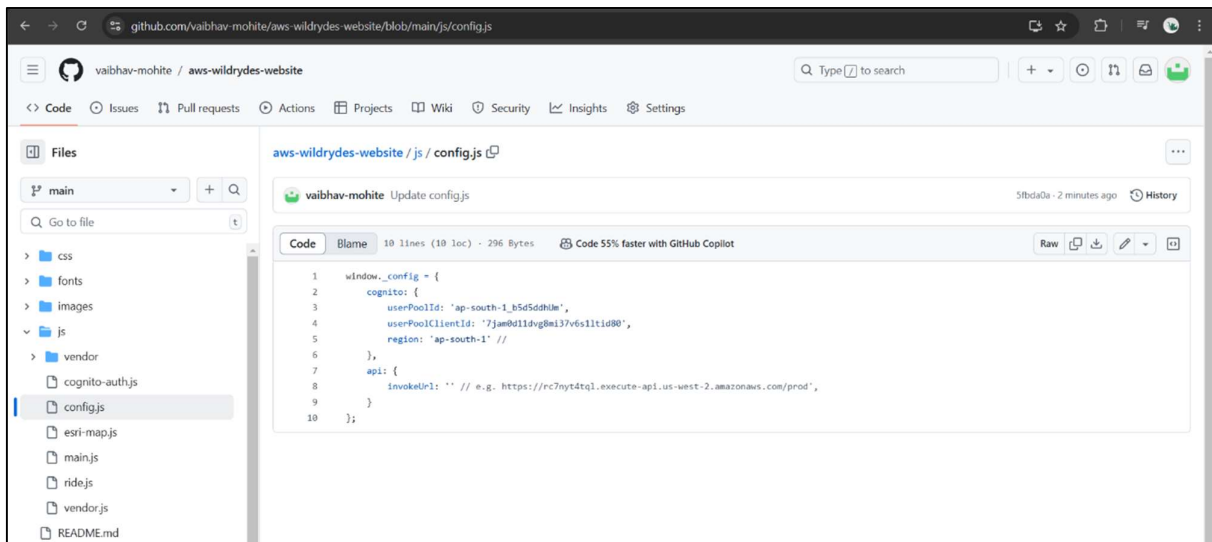
App clients and analytics (1) [Info](#)

Refresh Delete Create app client

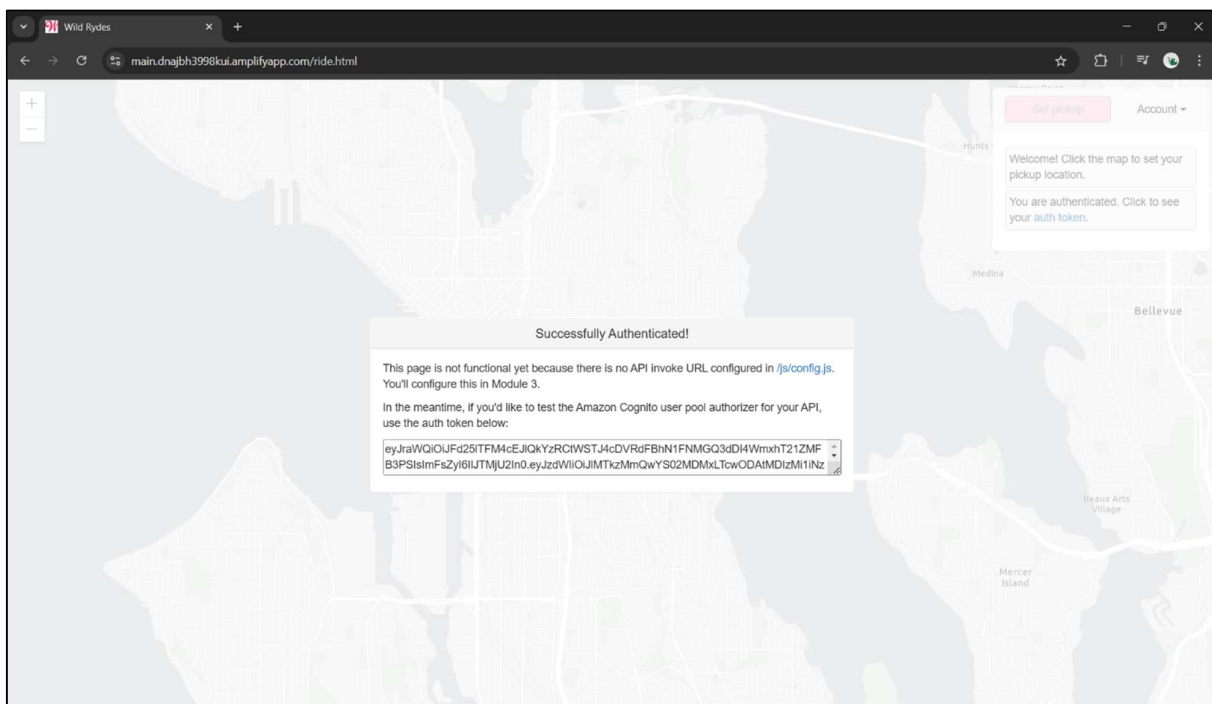
Configure an app client. App clients are the user pool authentication resources attached to your app. Select an app client to configure the permitted authentication actions for an app.

App client name	Client ID
<input type="radio"/> wildrydes-app-client	6v8uhsda4tn69da8dfmbeebuj

8. Open GitHub Repo. Navigate to js → vendor → config.js
9. Edit UserPoolId, UserPoolClientId and region with the values copied in Step 7 of Cognito.

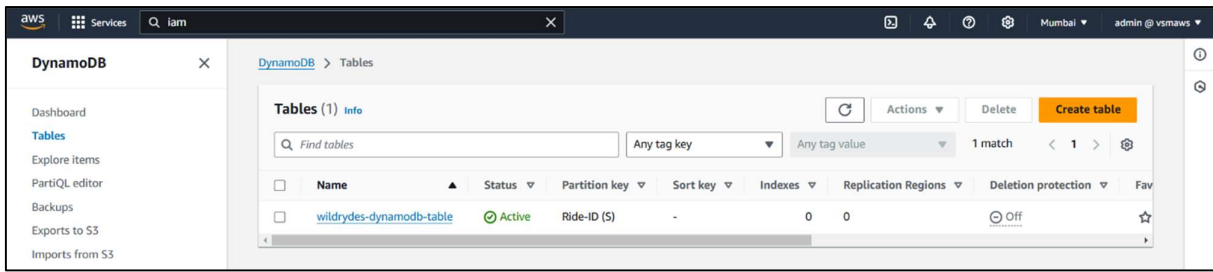


10. Commit the changes. This will start a new Deployment.
11. Once App is Deployed. Open the site. Click on Giddy Up to register a new user. Enter Email ID and Password and verify the same.
12. Login in again. Copy the auth code. It will be used later.



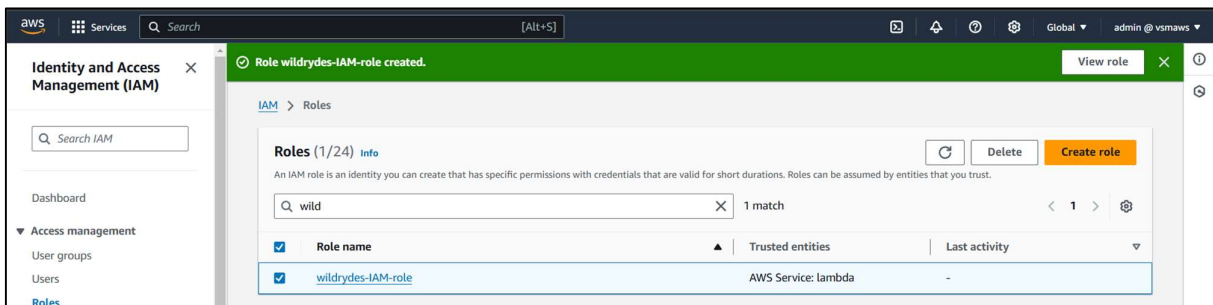
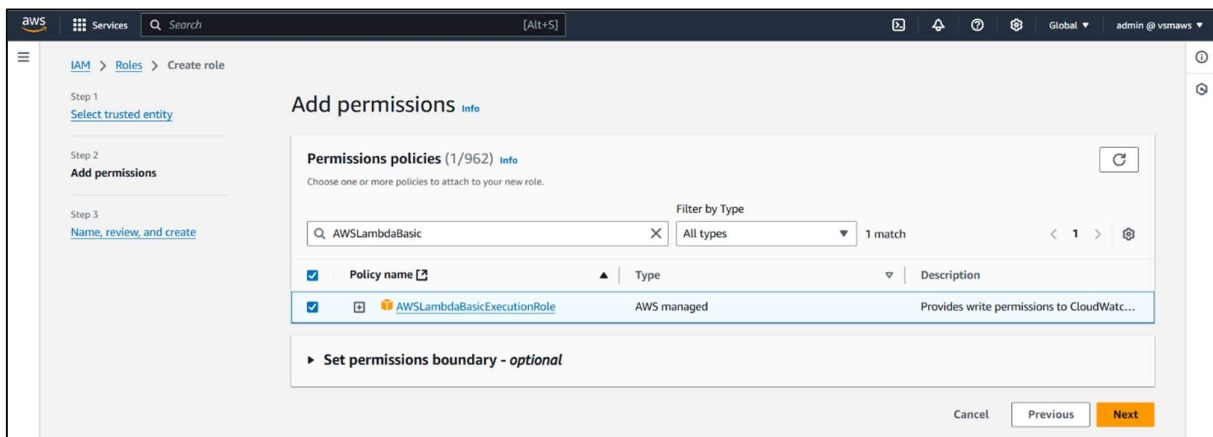
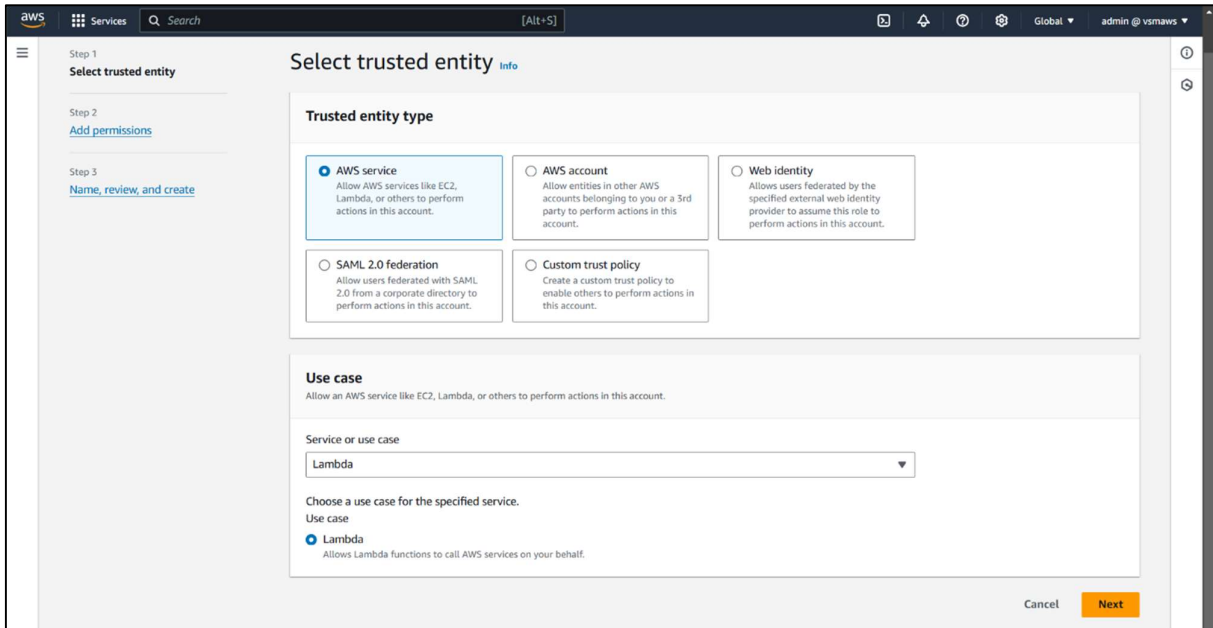
DYNAMODB

1. Launch DynamoDB Console
2. Create a new Table.
3. Table Name : wildrydes-dynamodb-table
Partition Key : RideID (String)



IAM

1. Create New Role as seen in Images below.



2. Once role is created, add DynamoDB Put Item permission in the IAM Role.

The screenshot shows the AWS IAM console for the role 'wildrydes-IAM-role'. The role's summary indicates it was created on November 10, 2024, and allows Lambda functions to call AWS services. Under the 'Permissions policies' tab, two policies are listed:

Policy name	Type	Attached entities
AWSLambdaBasicExecutionRole	AWS managed	1
wildryde-DynamoDB-put-access	Customer inline	0

This screenshot shows the same IAM role after the 'wildryde-DynamoDB-put-access' policy has been removed. Only the 'AWSLambdaBasicExecutionRole' is now listed under the 'Permissions policies' tab.

The screenshot shows the 'Specify permissions' page for the 'wildrydes-IAM-role'. The 'Policy editor' is in 'Visual' mode. Under the 'DynamoDB' section, the 'Allow' radio button is selected, and the 'PutItem' action is chosen under 'Actions allowed'. The 'Resources' section is set to 'Specific', with the resource ARN 'arn:aws:dynamodb:ap-south-1:723330070958:table/wildrydes-dynamodb-table' entered. The 'Effect' is set to 'Allow'.

LAMBDA

1. Create a New Lambda Function.

[Lambda](#) > [Functions](#) > Create function

Create function [Info](#)

Choose one of the following options to create your 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.

Basic information

Function name
Enter a name that describes the purpose of your function.

wildryde-lambda-function

Function name must be 1 to 64 characters, must be unique to the Region, and can't include spaces. Valid characters are a-z, A-Z, 0-9, hyphens (-), and underscores (_).

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 20.x

↻

Architecture [Info](#)
Choose the instruction set architecture you want for your function code.

☒ x86_64

☐ arm64

2. Select IAM Role created earlier.

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.

wildrydes-IAM-role

↻

[View the wildrydes-IAM-role role](#) on the IAM console.

► **Additional Configurations**
Use additional configurations to set up code signing, function URL, tags, and Amazon VPC access for your function.

Cancel **Create function**

3. Edit the Lambda Code and replace it with Lambda Function Code which could be found out in Readme File of GitHub Repo.

API GATEWAY

1. Click on Create API
2. Select REST API
3. Create a New API named “wildrydes-REST-API”

The screenshot shows the 'Create REST API' page in the AWS API Gateway console. The breadcrumb trail at the top is 'API Gateway > APIs > Create API > Create REST API'. The page title is 'Create REST API'. Under the 'API details' section, there are four radio button options: 'New API' (selected), 'Clone existing API', 'Import API', and 'Example API'. Below these, the 'API name' field contains 'wildrydes-REST-API'. The 'Description - optional' field is empty. The 'API endpoint type' dropdown is set to 'Regional'. At the bottom right, there are 'Cancel' and 'Create API' buttons.

4. Click on Authorizers in Left Pane of API Gateway.
5. Click on Create Authorizer
6. Select Authorizer Type as Cognito. Select “wildrydes-cognito-userpool” userPool which was created earlier.
7. Type Token Source as Authorization.

The screenshot shows the 'Create authorizer' page in the AWS API Gateway console. The breadcrumb trail at the top is 'API Gateway > APIs > wildrydes-REST-API (i3drdmmhii) > Authorizers > Create authorizer'. The page title is 'Create authorizer' with an 'Info' link. Under the 'Authorizer details' section, the 'Authorizer name' field contains 'wildrydes-authorizer'. The 'Authorizer type' is set to 'Cognito' (selected). Below this, the 'Cognito user pool' section shows a dropdown with 'ap-south-1' and a search bar containing 'wildrydes-cognito-userpool'. The 'Token source' field contains 'Authorization'. The 'Token validation - optional' field is empty. At the bottom right, there are 'Cancel' and 'Create authorizer' buttons.

8. To test if authorizer is working as expected, click on newly created Authorizer and paste the token Value which was copied in Step 12 of Cognito.
9. Test the authorizer. The result should output 200 Code.

API Gateway > APIs > wildrydes-REST-API (i3drdmhii) > Authorizers > wildrydes-authorizer

wildrydes-authorizer

Edit Delete

Authorizer details

Authorizer ID	Token source
gcbr7d	Authorization
Cognito pool	Token validation - optional
wildrydes-cognito-userpool - b5d5ddhUm (ap-south-1)	None

Test authorizer

Test your authorizer with a simulated invocation request. Enter an identity token that's provisioned from your Cognito user pool.

Token source: Authorization

Token value: eyJraWQiOiJFd25lTFM4cEJIQkYzRCtWSTJ4cDVRdFE

Authorizer test: wildrydes-authorizer

200

Claims

```
{
  "aud": "7jam0d11dvg8mi37v6s1ltid80",
  "auth_time": "1731236648",
  "cognito:username": "xjaddfif-at-telegmail.com",
  "email": "xjaddfif@telegmail.com",
  "email_verified": "true",
  "event_id": "b53b12d3-38dd-431d-8dcf-07f74198e09e",
  "exp": "Sun Nov 10 12:04:08 UTC 2024",
  "iat": "Sun Nov 10 11:04:08 UTC 2024",
  "iss": "https://cognito-idp.ap-south-1.amazonaws.com/ap-south-1_b5d5ddhUm",
  "jti": "9242e227-9240-4d0f-b03d-d4a77ca4d0c4",
}
```

10. Click on Resources Tab in Left Pane
11. Click on Create Resource. Enable CORS.

API Gateway > APIs > Resources - wildrydes-REST-API (i3drdmhii) > 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

12. One resource is created, click on “Create Method” Select Method type as POST. Select Lambda Function created earlier.

[API Gateway](#) > [APIs](#) > [Resources - wildrydes-REST-API \(i3drdmmhii\)](#) > [Create 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**
Send the request to your Lambda function as a structured event.

Lambda function
Provide the Lambda function name or alias. You can also provide an ARN from another account.

ap-south-1 ▼ 🔍 arn:aws:lambda:ap-south-1:723330070958:function:wil ✕

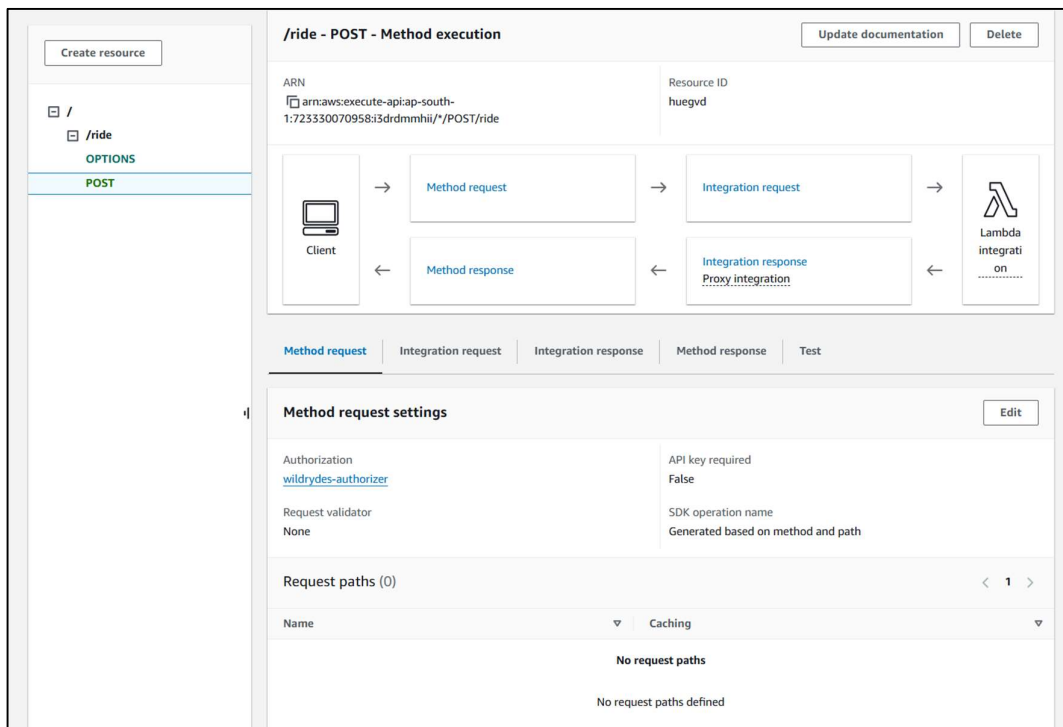
i Grant API Gateway permission to invoke your Lambda function. To turn off, update the function's resource policy yourself, or provide an invoke role that API Gateway uses to invoke your function.

Integration timeout | [Info](#)
By default, you can enter an integration timeout of 50 - 29,000 milliseconds. You can use Service Quotas to raise the integration timeout to greater than 29,000 ms

29000

13. Create Method

14. Edit Method Request



15. Select Authorization as Cognito User Pool Authorizers

API Gateway > APIs > Resources - wildrydes-REST-API (i3drdmmhii) > Edit method request

Edit method request

Method request settings

Authorization
wildrydes-authorizer ▼

Authorization scopes

Request validator
None ▼

☐ API key required

Operation name - optional

► URL query string parameters

► HTTP request headers

► Request body

16. Save and Deploy the API

The screenshot shows the AWS API Gateway console for the 'wildrydes-REST-API (i3drdmhii)'. The 'Stages' tab is selected, showing a list of stages with 'dev' highlighted. The 'Stage details' for 'dev' are displayed, including the stage name, rate limit, cache cluster, default method-level caching, invoke URL, and active deployment.

Stages

Stage actions ▼ Create stage

Stage details Info Edit

Stage name dev	Rate Info -	Web ACL -
Cache cluster Info <input type="radio"/> inactive	Burst Info -	Client certificate -
Default method-level caching <input type="radio"/> inactive		
Invoke URL https://i3drdmhii.execute-api.ap-south-1.amazonaws.com/dev		
Active deployment cvw0ti on November 10, 2024, 17:10 (UTC+05:30)		

17. Copy the Invoke API and Paste it in GitHub Repo → js → vendor → config.js and paste it in invokeUrl. Commit Changes. These changes will trigger new deployment.

The screenshot shows the GitHub repository 'vaibhav-mohite / aws-wildrydes-website'. The file 'config.js' is selected, showing its contents. The file is a JavaScript configuration for a web application, including a window object with a config property and an api property.

github.com/vaibhav-mohite/aws-wildrydes-website/blob/main/js/config.js

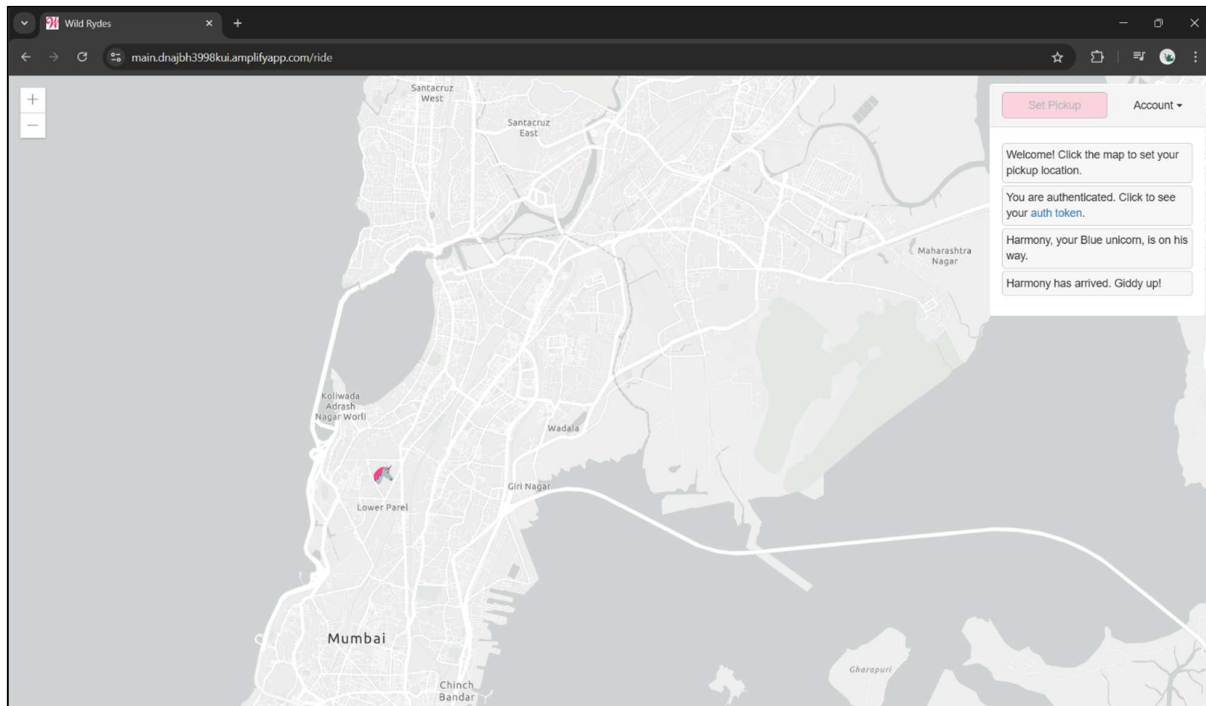
vaibhav-mohite / aws-wildrydes-website

Update config.js ed93440 · now History

```
1 window.config = {
2   cognito: {
3     userPoolId: 'ap-south-1_355d8d4e',
4     userPoolClientId: '7jam81ldvg8m137v6s1lti080',
5     region: 'ap-south-1' //
6   },
7   api: {
8     invokeUrl: 'https://i3drdmhii.execute-api.ap-south-1.amazonaws.com/dev'
9   }
10  };
```

WEBSITE

1. Add /ride.html to the Domain url. This will load the APP. Click on Any Place on the Map and request the unicorn.



Please check “final-website-images” Folder in GitHub Repo for more Images of the Application