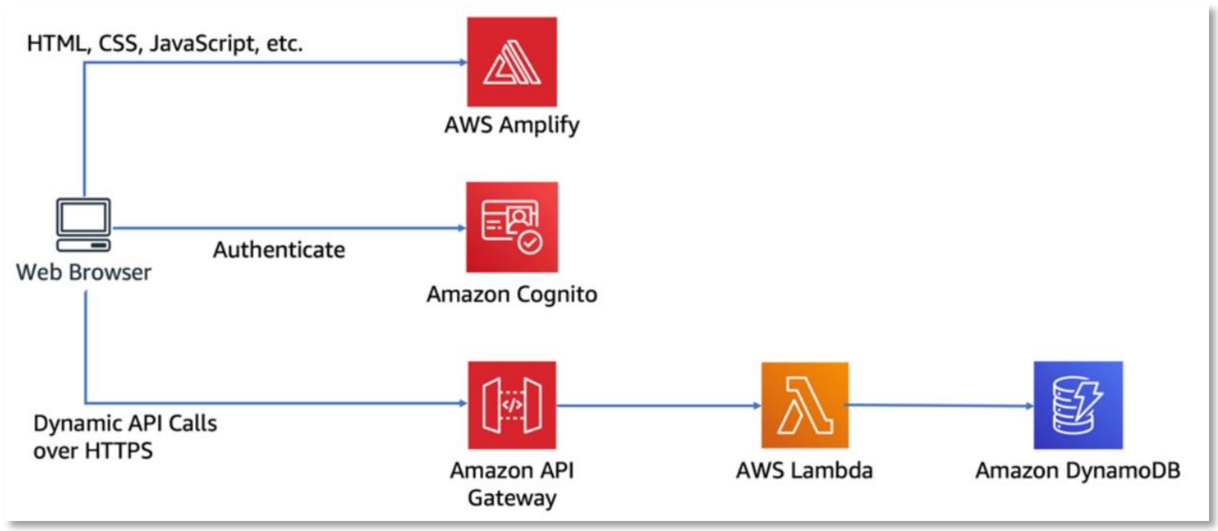
Capstone Project 03
Build a serverless web application using Amplify, Cognito, API Gateway, Lambda & DynamoDB

This capstone project will be executed in 4 different modules:

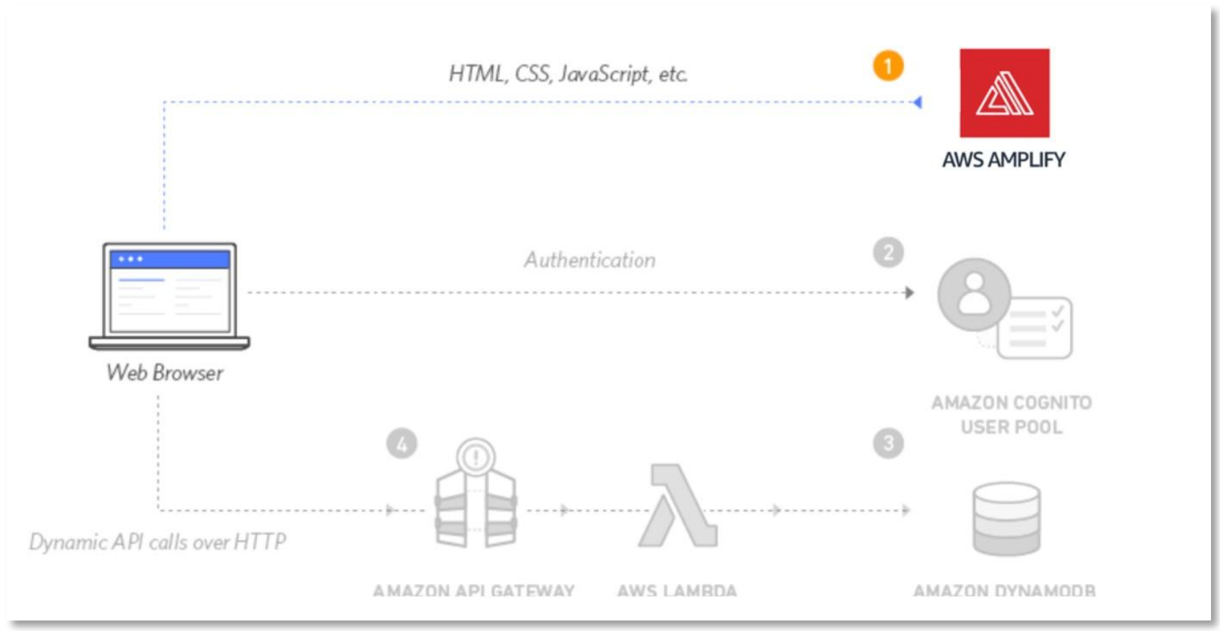
1. **Static Web Hosting** - Configure AWS Amplify to host the static resources for web application with continuous deployment built-in
2. **User Management** - Create an Amazon Cognito user pool to manage users' accounts
3. **Serverless Backend** - Build a backend process for handling requests for web application
4. **RESTful API** - Use Amazon API Gateway to expose the Lambda function built in the previous module as a RESTful API

Architectural Diagram



Module 1 – Host Static Website

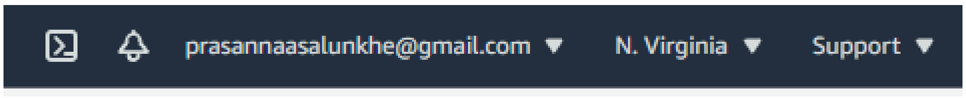
Architectural Overview:



All the static web content including HTML, CSS, JavaScript, images and other files will be managed by AWS Amplify Console.
End-users will then access the site using the public website URL exposed by AWS Amplify Console. Our end users will then access r site using the public website URL exposed by AWS Amplify Console

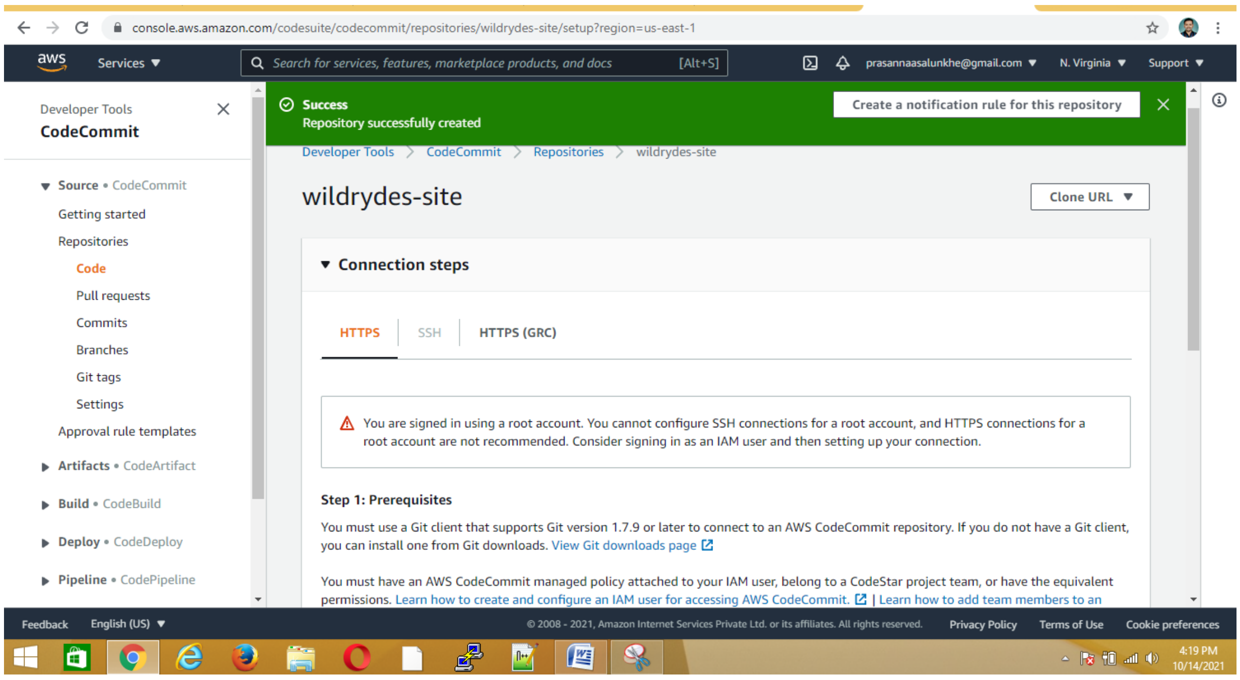
Step 1 – Select a Region

For this project region – US-east-1 (N Virginia) is selected. (default region.



Step 2 – Create a git repository

We have two options to manage the source code for this module
in this project; CodeCommit will be used to store application code.
A CodeCommit repository named “wildrydes-site” is created.
Once the repository is created, appropriate IAM user with GIT credential is created.


Success
Repository successfully created

wildrydes-site

Connection steps

You are signed in using a root account. You cannot configure SSH connections for a root account, and HTTPS connections for a root account are not recommended. Consider signing in as an IAM user and then setting up your connection.

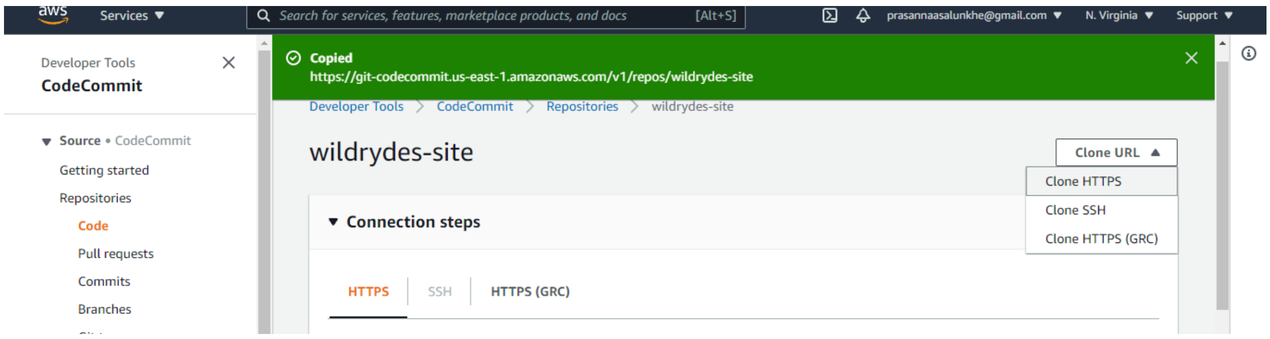
Step 1: Prerequisites

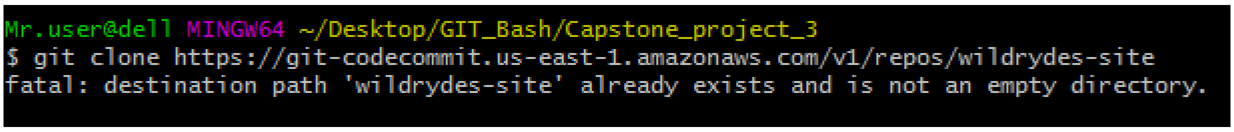
You must use a Git client that supports Git version 1.7.9 or later to connect to an AWS CodeCommit repository. If you do not have a Git client, you can install one from Git downloads. View Git downloads page

You must have an AWS CodeCommit managed policy attached to your IAM user, belong to a CodeStar project team, or have the equivalent permissions. Learn how to create and configure an IAM user for accessing AWS CodeCommit. | Learn how to add team members to an

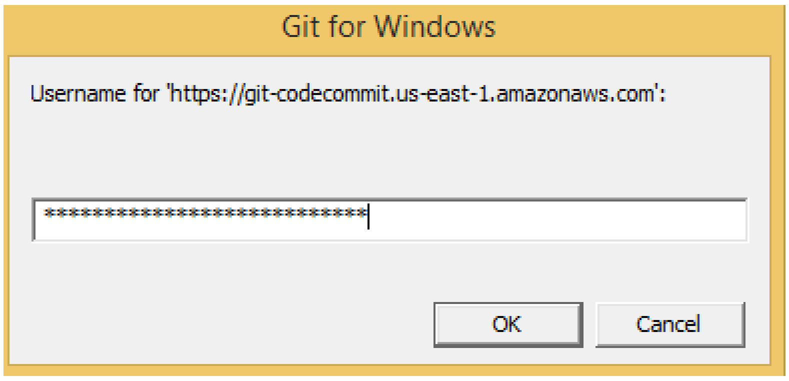
console.aws.amazon.com/codesuite/codecommit/repositories/wildrydes-site/setup?region=us-east-1
Services
Search for services, features, marketplace products, and docs [Alt+S]
prasannaasalunkhe@gmail.com
N. Virginia
Support
Developer Tools
CodeCommit
Source • CodeCommit
Getting started
Repositories
Code
Pull requests
Commits
Branches
Git tags
Settings
Approval rule templates
Artifacts • CodeArtifact
Build • CodeBuild
Deploy • CodeDeploy
Pipeline • CodePipeline
Create a notification rule for this repository
Developer Tools > CodeCommit > Repositories > wildrydes-site
Clone URL
HTTPS
SSH
HTTPS (GRC)
Feedback
English (US)
© 2008 - 2021, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use
Cookie preferences
4:19 PM
10/14/2021

Next step is to clone URL using “Clone HTTPS” link as show below


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

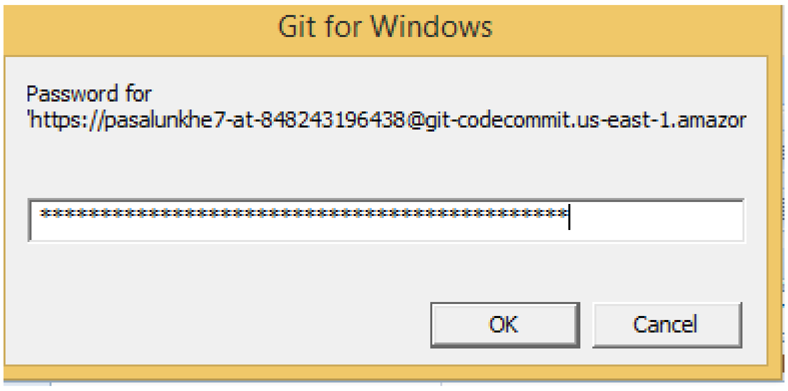

```
Mr.user@dell MINGW64 ~/Desktop/GIT_Bash/Capstone_project_3
$ git clone https://git-codecommit.us-east-1.amazonaws.com/v1/repos/wildrydes-site
fatal: destination path 'wildrydes-site' already exists and is not an empty directory.
```

Entered Username and password for a code commit credentials. This credential s care generated from IAM user for HTTPS connections.


Git for Windows

Username for 'https://git-codecommit.us-east-1.amazonaws.com':

****



Password for
'https://pasalunkhe7-at-848243196438@git-codecommit.us-east-1.amazor

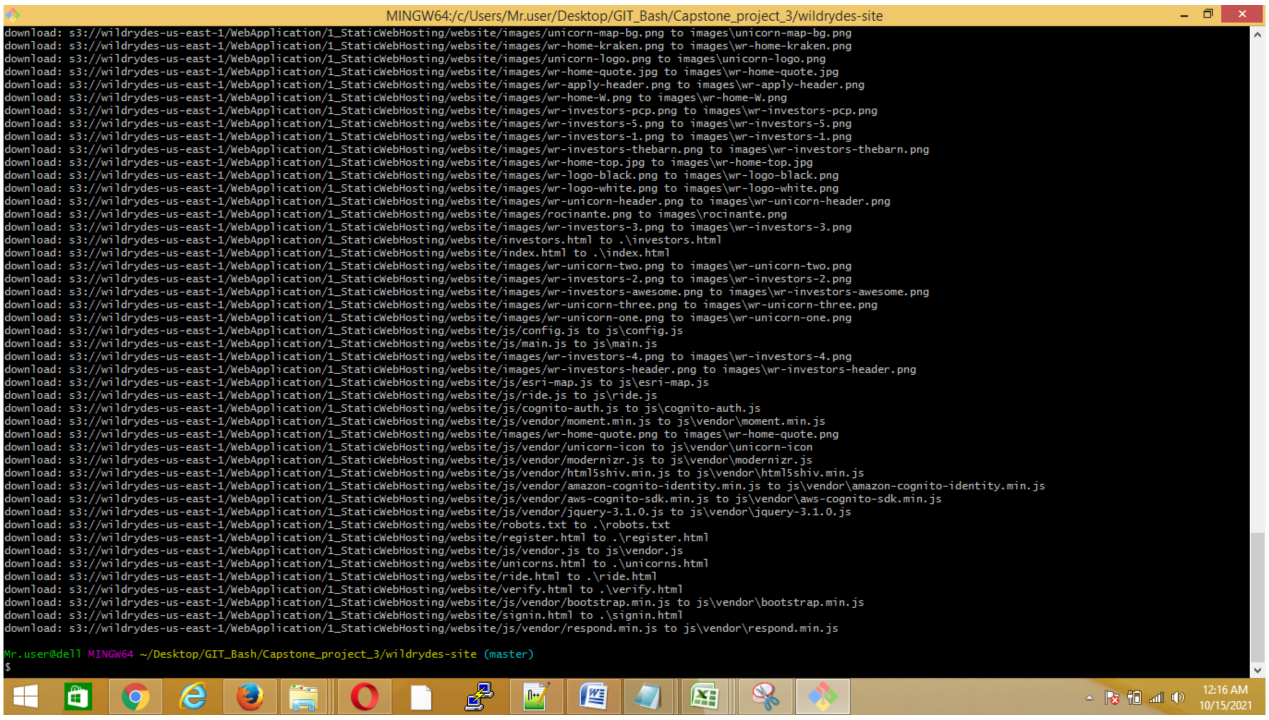
```
**********************************************
```

Step 3 – Populate the git repository

Now we have to copy the web site data. The web site content is copied from an existing publicly accessible S3 bucket associated and added the content to this repository.

Command to download the content:

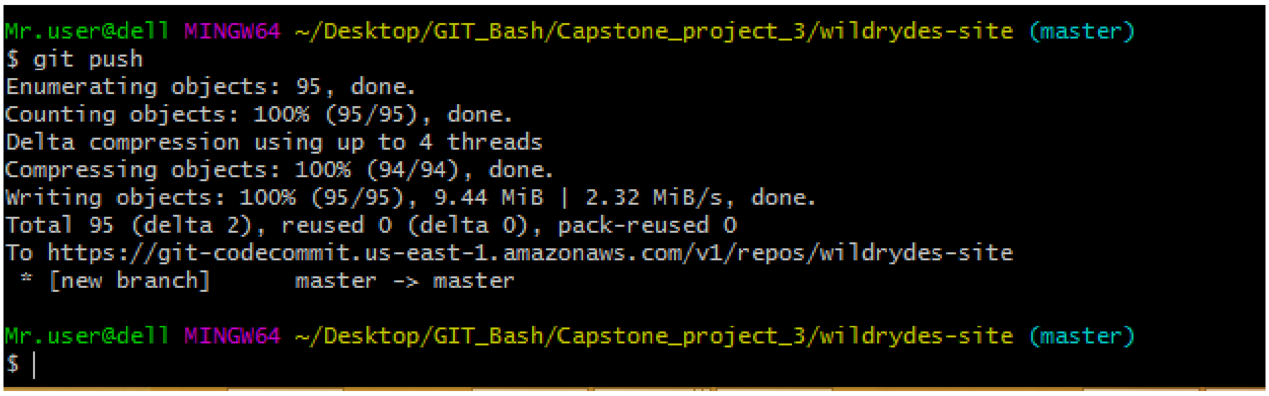
```
cd wildrydes-site

aws s3 cp s3://wildrydes-us-east-1/WebApplication/1_StaticWebHosting/website ./ --
recursive
```


MINGW64:/c/Users/Mr.user/Desktop/GIT_Bash/Capstone_project_3/wildrydes-site

```
download: s3://wildrydes-us-east-1/WebApplication/1_StaticWebHosting/website/images/unicorn-map-bg.png to images\unicorn-map-bg.png
download: s3://wildrydes-us-east-1/WebApplication/1_StaticWebHosting/website/images/wr-home-kraken.png to images\wr-home-kraken.png
download: s3://wildrydes-us-east-1/WebApplication/1_StaticWebHosting/website/images/unicorn-logo.png to images\unicorn-logo.png
download: s3://wildrydes-us-east-1/WebApplication/1_StaticWebHosting/website/images/wr-home-quote.jpg to images\wr-home-quote.jpg
download: s3://wildrydes-us-east-1/WebApplication/1_StaticWebHosting/website/images/wr-apply-header.png to images\wr-apply-header.png
download: s3://wildrydes-us-east-1/WebApplication/1_StaticWebHosting/website/images/wr-home-W.png to images\wr-home-W.png
download: s3://wildrydes-us-east-1/WebApplication/1_StaticWebHosting/website/images/wr-investors-pcp.png to images\wr-investors-pcp.png
download: s3://wildrydes-us-east-1/WebApplication/1_StaticWebHosting/website/images/wr-investors-5.png to images\wr-investors-5.png
download: s3://wildrydes-us-east-1/WebApplication/1_StaticWebHosting/website/images/wr-investors-1.png to images\wr-investors-1.png
download: s3://wildrydes-us-east-1/WebApplication/1_StaticWebHosting/website/images/wr-investors-thebarn.png to images\wr-investors-thebarn.png
download: s3://wildrydes-us-east-1/WebApplication/1_StaticWebHosting/website/images/wr-home-top.jpg to images\wr-home-top.jpg
download: s3://wildrydes-us-east-1/WebApplication/1_StaticWebHosting/website/images/wr-logo-black.png to images\wr-logo-black.png
download: s3://wildrydes-us-east-1/WebApplication/1_StaticWebHosting/website/images/wr-logo-white.png to images\wr-logo-white.png
download: s3://wildrydes-us-east-1/WebApplication/1_StaticWebHosting/website/images/wr-unicorn-header.png to images\wr-unicorn-header.png
download: s3://wildrydes-us-east-1/WebApplication/1_StaticWebHosting/website/images/rocinante.png to images\rocinante.png
download: s3://wildrydes-us-east-1/WebApplication/1_StaticWebHosting/website/images/wr-investors-3.png to images\wr-investors-3.png
download: s3://wildrydes-us-east-1/WebApplication/1_StaticWebHosting/website/investors.html to .\investors.html
download: s3://wildrydes-us-east-1/WebApplication/1_StaticWebHosting/website/index.html to .\index.html
download: s3://wildrydes-us-east-1/WebApplication/1_StaticWebHosting/website/images/wr-unicorn-two.png to images\wr-unicorn-two.png
download: s3://wildrydes-us-east-1/WebApplication/1_StaticWebHosting/website/images/wr-investors-2.png to images\wr-investors-2.png
download: s3://wildrydes-us-east-1/WebApplication/1_StaticWebHosting/website/images/wr-investors-awesome.png to images\wr-investors-awesome.png
download: s3://wildrydes-us-east-1/WebApplication/1_StaticWebHosting/website/images/wr-unicorn-three.png to images\wr-unicorn-three.png
download: s3://wildrydes-us-east-1/WebApplication/1_StaticWebHosting/website/images/wr-unicorn-one.png to images\wr-unicorn-one.png
download: s3://wildrydes-us-east-1/WebApplication/1_StaticWebHosting/website/js/config.js to js\config.js
download: s3://wildrydes-us-east-1/WebApplication/1_StaticWebHosting/website/js/main.js to js\main.js
download: s3://wildrydes-us-east-1/WebApplication/1_StaticWebHosting/website/images/wr-investors-4.png to images\wr-investors-4.png
download: s3://wildrydes-us-east-1/WebApplication/1_StaticWebHosting/website/images/wr-investors-header.png to images\wr-investors-header.png
download: s3://wildrydes-us-east-1/WebApplication/1_StaticWebHosting/website/js/esri-map.js to js\esri-map.js
download: s3://wildrydes-us-east-1/WebApplication/1_StaticWebHosting/website/js/ride.js to js\ride.js
download: s3://wildrydes-us-east-1/WebApplication/1_StaticWebHosting/website/js/cognito-auth.js to js\cognito-auth.js
download: s3://wildrydes-us-east-1/WebApplication/1_StaticWebHosting/website/js/vendor/moment.min.js to js\vendor\moment.min.js
download: s3://wildrydes-us-east-1/WebApplication/1_StaticWebHosting/website/images/wr-home-quote.png to images\wr-home-quote.png
download: s3://wildrydes-us-east-1/WebApplication/1_StaticWebHosting/website/js/vendor/unicorn-icon to js\vendor\unicorn-icon
download: s3://wildrydes-us-east-1/WebApplication/1_StaticWebHosting/website/js/vendor/modernizr.js to js\vendor\modernizr.js
download: s3://wildrydes-us-east-1/WebApplication/1_StaticWebHosting/website/js/vendor/html5shiv.min.js to js\vendor\html5shiv.min.js
download: s3://wildrydes-us-east-1/WebApplication/1_StaticWebHosting/website/js/vendor/amazon-cognito-identity.min.js to js\vendor\amazon-cognito-identity.min.js
download: s3://wildrydes-us-east-1/WebApplication/1_StaticWebHosting/website/js/vendor/aws-cognito-sdk.min.js to js\vendor\aws-cognito-sdk.min.js
download: s3://wildrydes-us-east-1/WebApplication/1_StaticWebHosting/website/js/vendor/jquery-3.1.0.js to js\vendor\jquery-3.1.0.js
download: s3://wildrydes-us-east-1/WebApplication/1_StaticWebHosting/website/robots.txt to .\robots.txt
download: s3://wildrydes-us-east-1/WebApplication/1_StaticWebHosting/website/register.html to .\register.html
download: s3://wildrydes-us-east-1/WebApplication/1_StaticWebHosting/website/js/vendor.js to js\vendor.js
download: s3://wildrydes-us-east-1/WebApplication/1_StaticWebHosting/website/unicorns.html to .\unicorns.html
download: s3://wildrydes-us-east-1/WebApplication/1_StaticWebHosting/website/ride.html to .\ride.html
download: s3://wildrydes-us-east-1/WebApplication/1_StaticWebHosting/website/verify.html to .\verify.html
download: s3://wildrydes-us-east-1/WebApplication/1_StaticWebHosting/website/js/vendor/bootstrap.min.js to js\vendor\bootstrap.min.js
download: s3://wildrydes-us-east-1/WebApplication/1_StaticWebHosting/website/signin.html to .\signin.html
download: s3://wildrydes-us-east-1/WebApplication/1_StaticWebHosting/website/js/vendor/respond.min.js to js\vendor\respond.min.js

Mr.user@dell MINGW64 ~/Desktop/GIT_Bash/Capstone_project_3/wildrydes-site (master)
$
```

Now perform a commit operation. these files are committed using following commands:

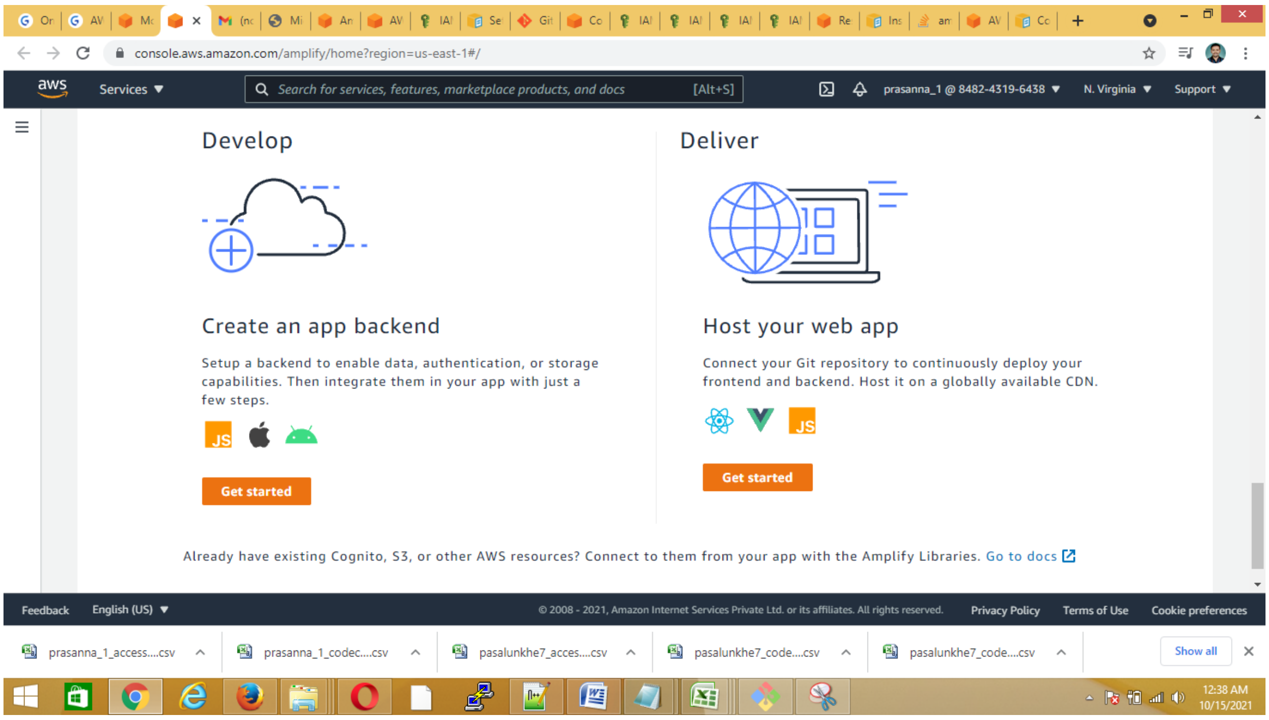
```
$ git add .
$ git commit -m 'new'
$ git push
```



```
Mr.user@dell MINGW64 ~/Desktop/GIT_Bash/Capstone_project_3/wildrydes-site (master)
$ git push
Enumerating objects: 95, done.
Counting objects: 100% (95/95), done.
Delta compression using up to 4 threads
Compressing objects: 100% (94/94), done.
Writing objects: 100% (95/95), 9.44 MiB | 2.32 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

Mr.user@dell MINGW64 ~/Desktop/GIT_Bash/Capstone_project_3/wildrydes-site (master)
$ |
```

Step 4 – Enable web hosting with AWS Amplify

Next used the AWS Amplify Console to deploy the website we just committed to git. For this, On the AWS Amplify service, Get started under “Deliver” is selected.



console.aws.amazon.com/amplify/home?region=us-east-1#/

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

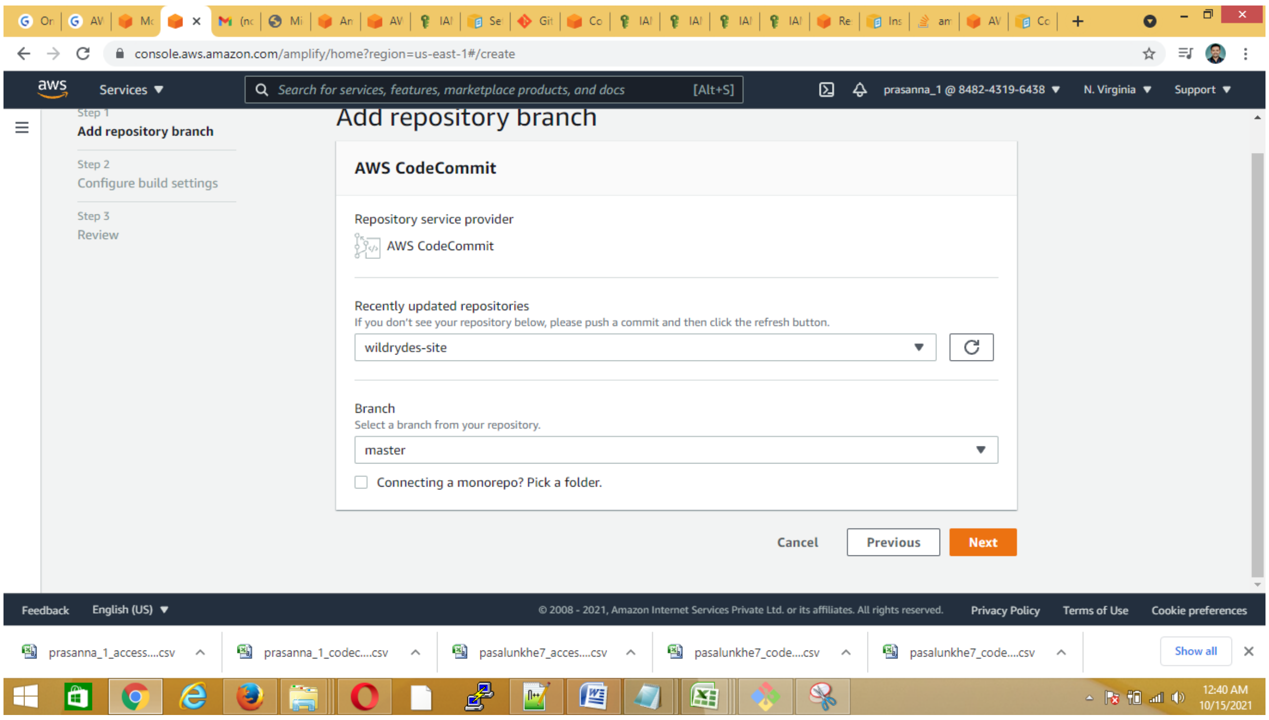
Host your web app

Connect your Git repository to continuously deploy your frontend and backend. Host it on a globally available CDN.

Get started

Already have existing Cognito, S3, or other AWS resources? Connect to them from your app with the Amplify Libraries. Go to docs

Selected the Repository service provider as AWS CodeCommit on the next page, and appropriate repository is selected fromthe drop-down menu and branch as a “master”



Add repository branch

AWS CodeCommit

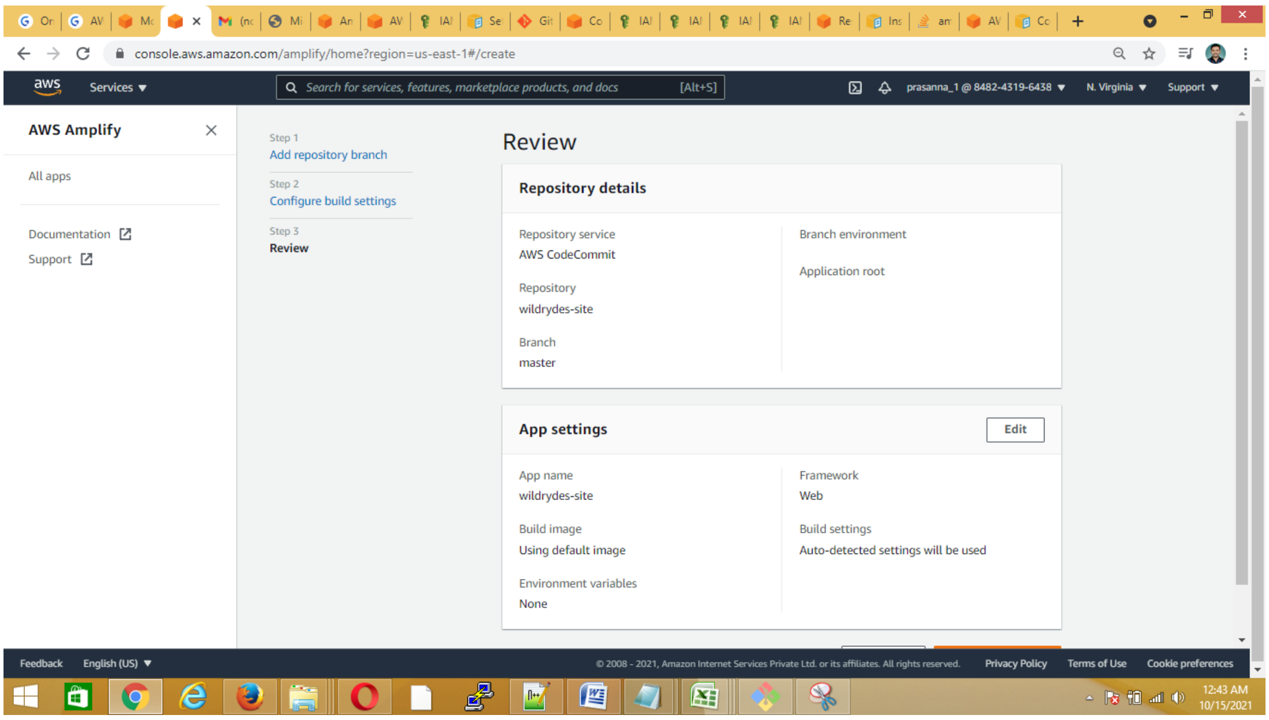
Repository service provider
AWS CodeCommit

Recently updated repositories
If you don't see your repository below, please push a commit and then click the refresh button.
wildrydes-site

Branch
Select a branch from your repository.
master

Connecting a monorepo? Pick a folder.

Now next on the "Configure build settings" page all the fields are left defaults and next is selected.On the "Review" page Save and deploy is selected.



Review

Repository details

Repository service
AWS CodeCommit

Repository
wildrydes-site

Branch
master

Branch environment

Application root

App settings

App name
wildrydes-site

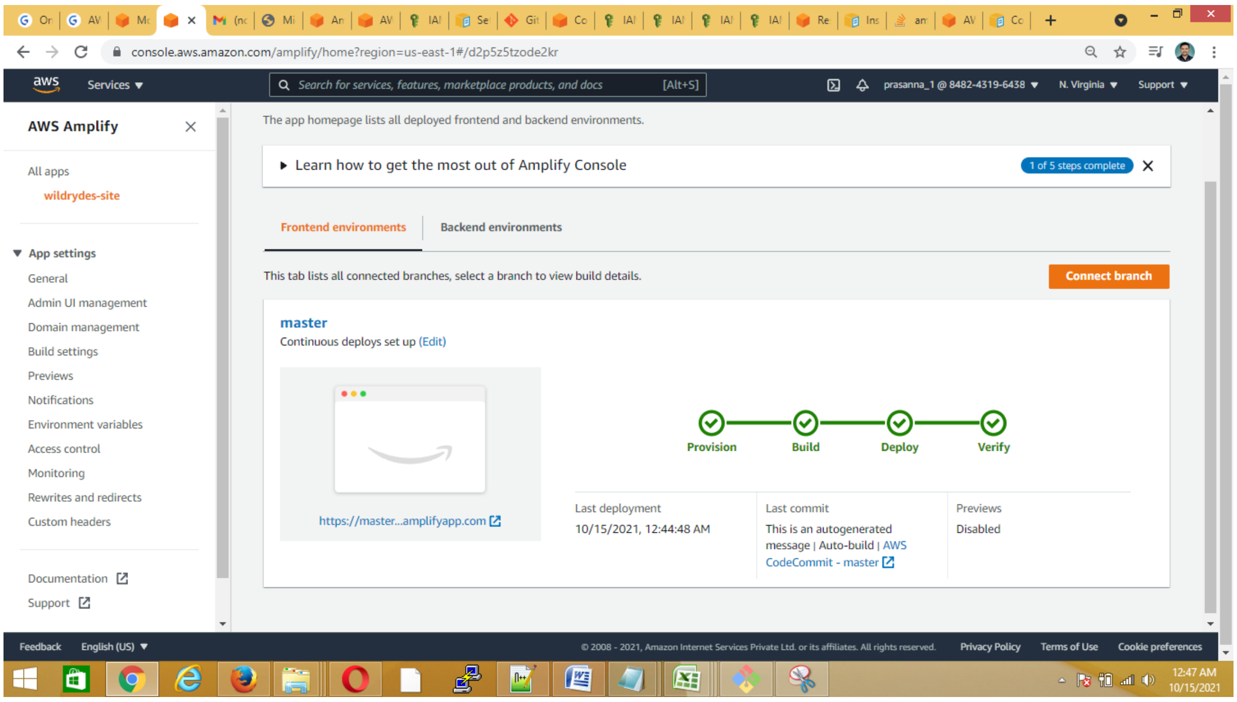
Build image
Using default image

Environment variables
None

Framework
Web

Build settings
Auto-detected settings will be used

It took some time to Amplify Console to create the necessary resources and to deploy r code. The final static site is up now, and can be viewed as shown below


console.aws.amazon.com/amplify/home?region=us-east-1#/d2p5z5tzode2kr

AWS Amplify

All apps
wildrydes-site

App settings
General
Admin UI management
Domain management
Build settings
Previews
Notifications
Environment variables
Access control
Monitoring
Rewrites and redirects
Custom headers

Documentation
Support

The app homepage lists all deployed frontend and backend environments.

Learn how to get the most out of Amplify Console — 1 of 5 steps complete

Frontend environments | Backend environments

This tab lists all connected branches, select a branch to view build details. — Connect branch

master
Continuous deploys set up (Edit)

https://master...amplifyapp.com

Provision — Build — Deploy — Verify

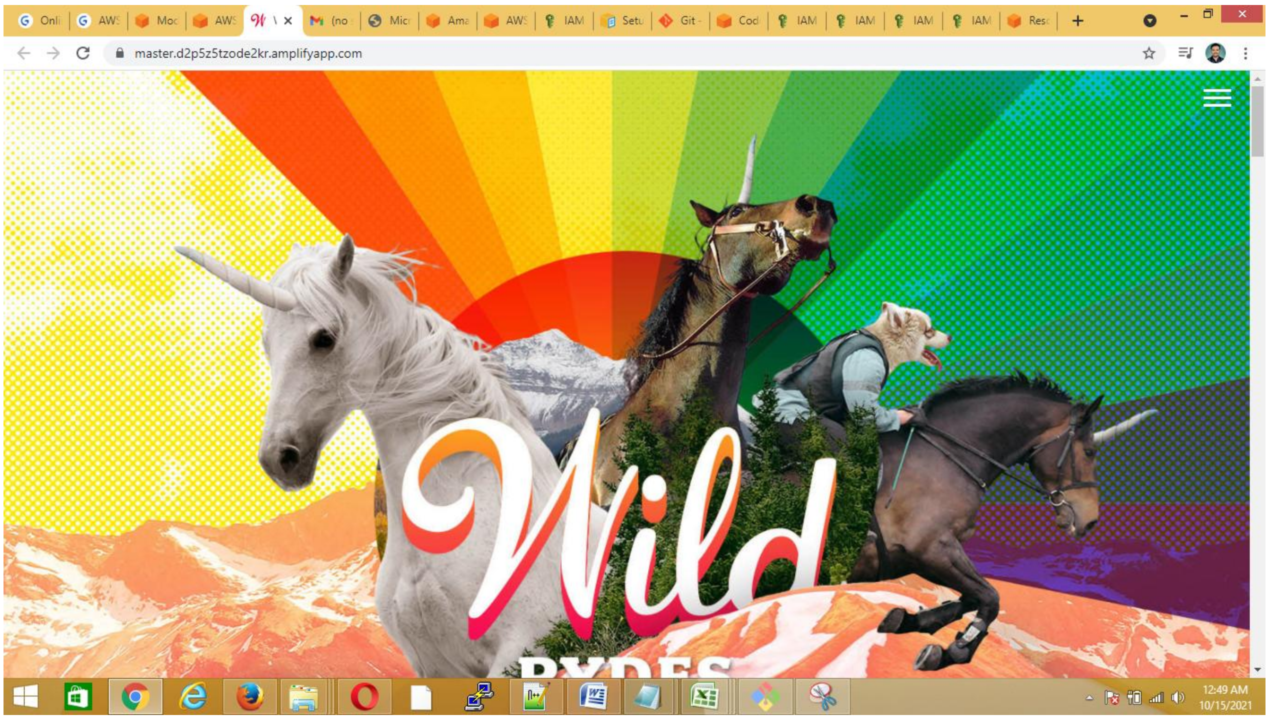
Last deployment
10/15/2021, 12:44:48 AM

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

Previews
Disabled

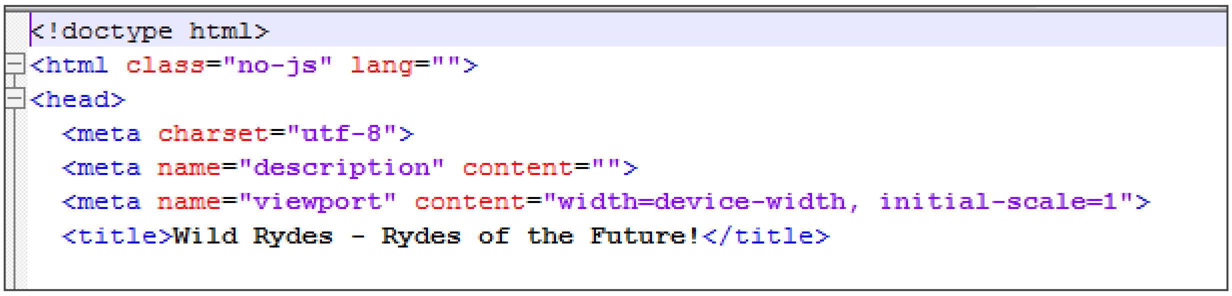
© 2008 - 2021, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved. Privacy Policy Terms of Use Cookie preferences

Once completed, click on the site image to launch r Wild Rides site.



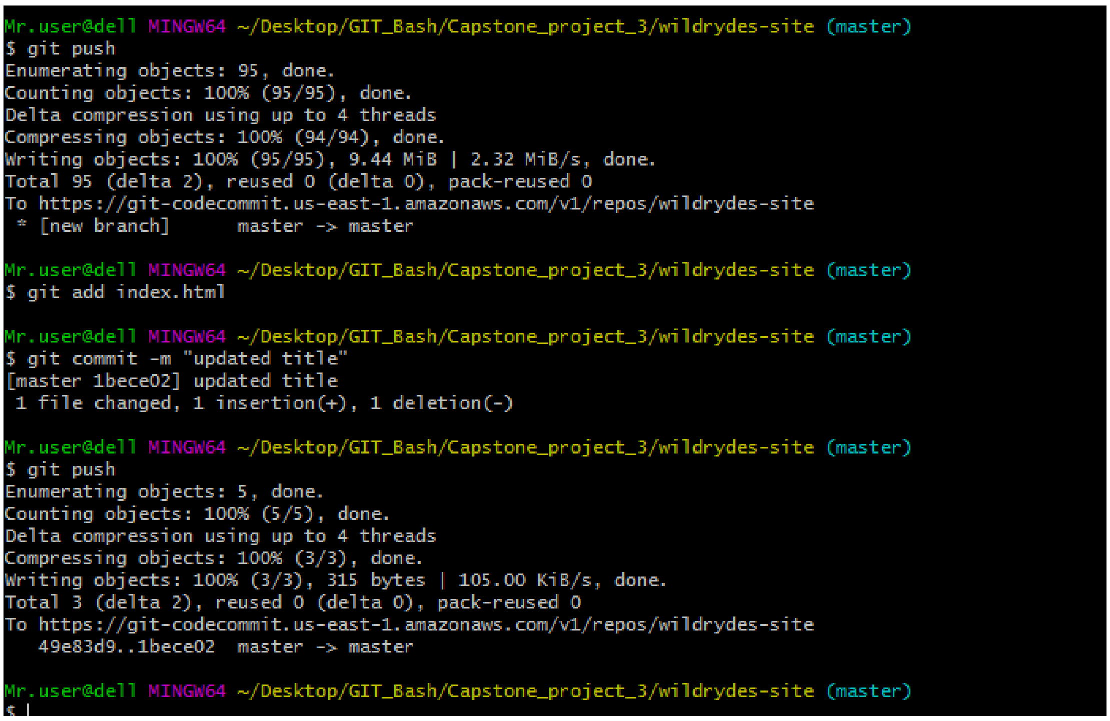
Step 5 – Modify the site

Now made a change to the main page to test out this process.
"index.html" is edited as shown below:



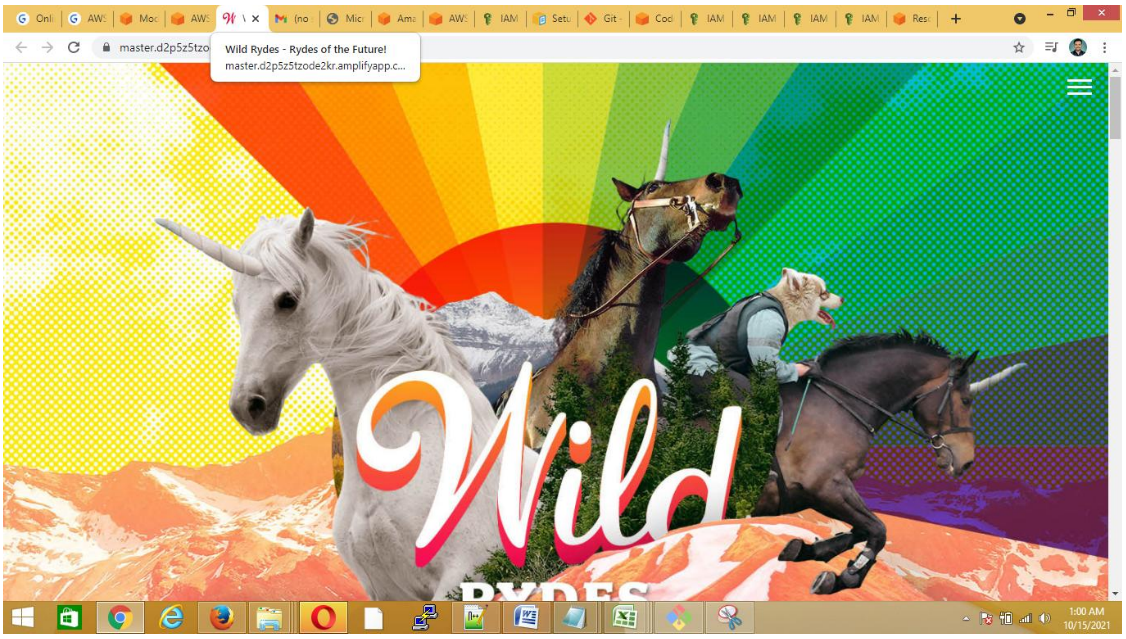
```
<!doctype html>
<html class="no-js" lang="">
<head>
  <meta charset="utf-8">
  <meta name="description" content="">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>Wild Rydes - Rydes of the Future!</title>
```

Saved the file and commit to r git repository again, and it's time to push the file to the CodeCommit repository.



```
Mr.user@dell MINGW64 ~/Desktop/GIT_Bash/Capstone_project_3/wildrydes-site (master)
$ git push
Enumerating objects: 95, done.
Counting objects: 100% (95/95), done.
Delta compression using up to 4 threads
Compressing objects: 100% (94/94), done.
Writing objects: 100% (95/95), 9.44 MiB | 2.32 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

Mr.user@dell MINGW64 ~/Desktop/GIT_Bash/Capstone_project_3/wildrydes-site (master)
$ git add index.html

Mr.user@dell MINGW64 ~/Desktop/GIT_Bash/Capstone_project_3/wildrydes-site (master)
$ git commit -m "updated title"
[master 1bece02] updated title
 1 file changed, 1 insertion(+), 1 deletion(-)

Mr.user@dell MINGW64 ~/Desktop/GIT_Bash/Capstone_project_3/wildrydes-site (master)
$ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 315 bytes | 105.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0
To https://git-codecommit.us-east-1.amazonaws.com/v1/repos/wildrydes-site
   49e83d9..1bece02  master -> master

Mr.user@dell MINGW64 ~/Desktop/GIT_Bash/Capstone_project_3/wildrydes-site (master)
$
```

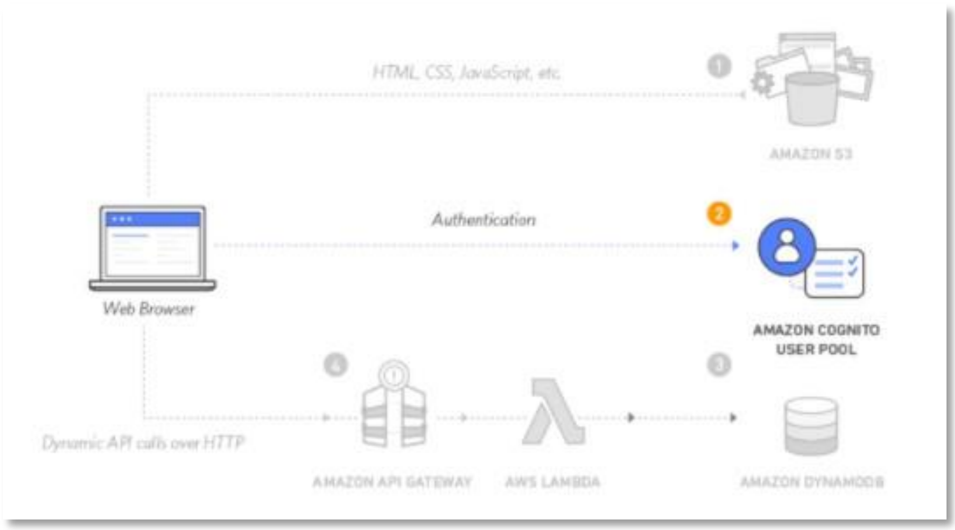
Once completed, re-opened the Wild Rydes site and notice the title change. The title can be viewed updated.


Wild Rydes - Rydes of the Future!
master.d2p5z5tzode2kr.amplifyapp.c...

Module 2 – Manage users

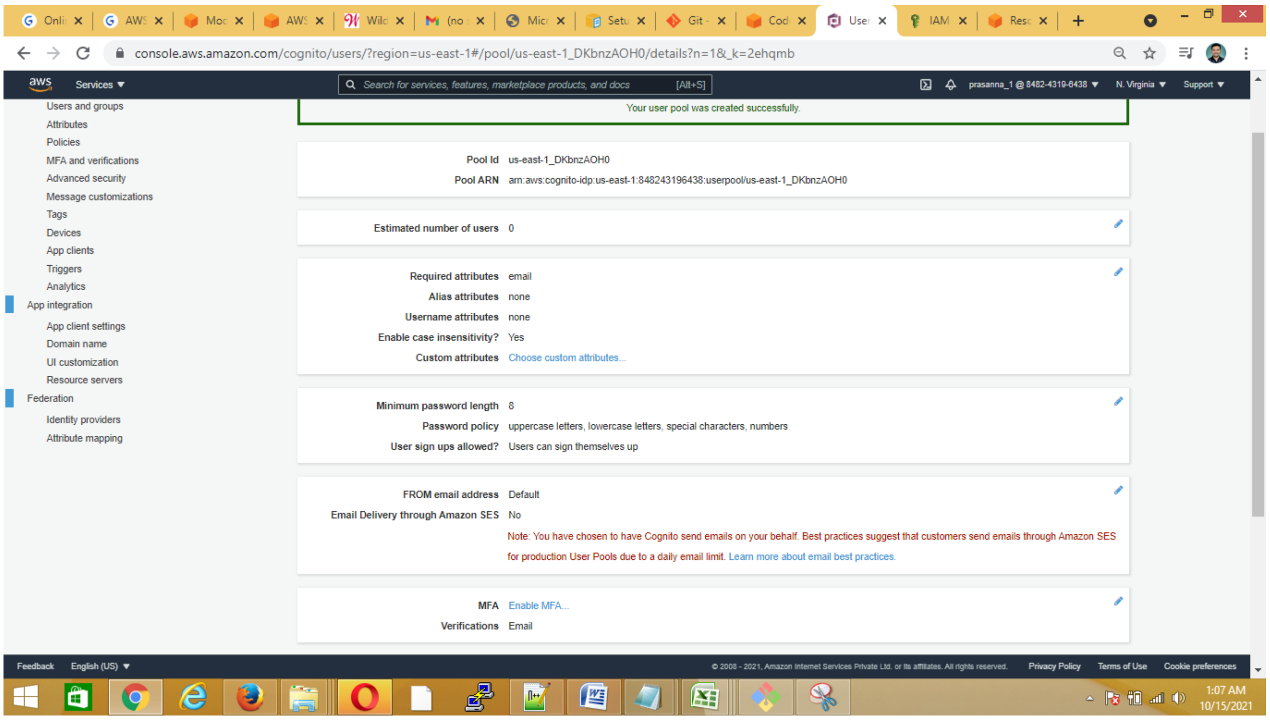
In this module we have created an Amazon Cognito user pool to manage r users' accounts

Architectural Overview:



Step 1 – Create an Amazon Cognito User Pool

- From the AWS Console Cognito is selected under Mobile ServicesManage r User Pools is chosen.
- Created a User Pool is chosen.
- A name is provided for the user pool as “WildRydes”.
- On the review page, click Create pool.



console.aws.amazon.com/cognito/users/?region=us-east-1#/pool/us-east-1_DKbnzAOH0/details?n=1&_k=2ehqmb

Your user pool was created successfully.

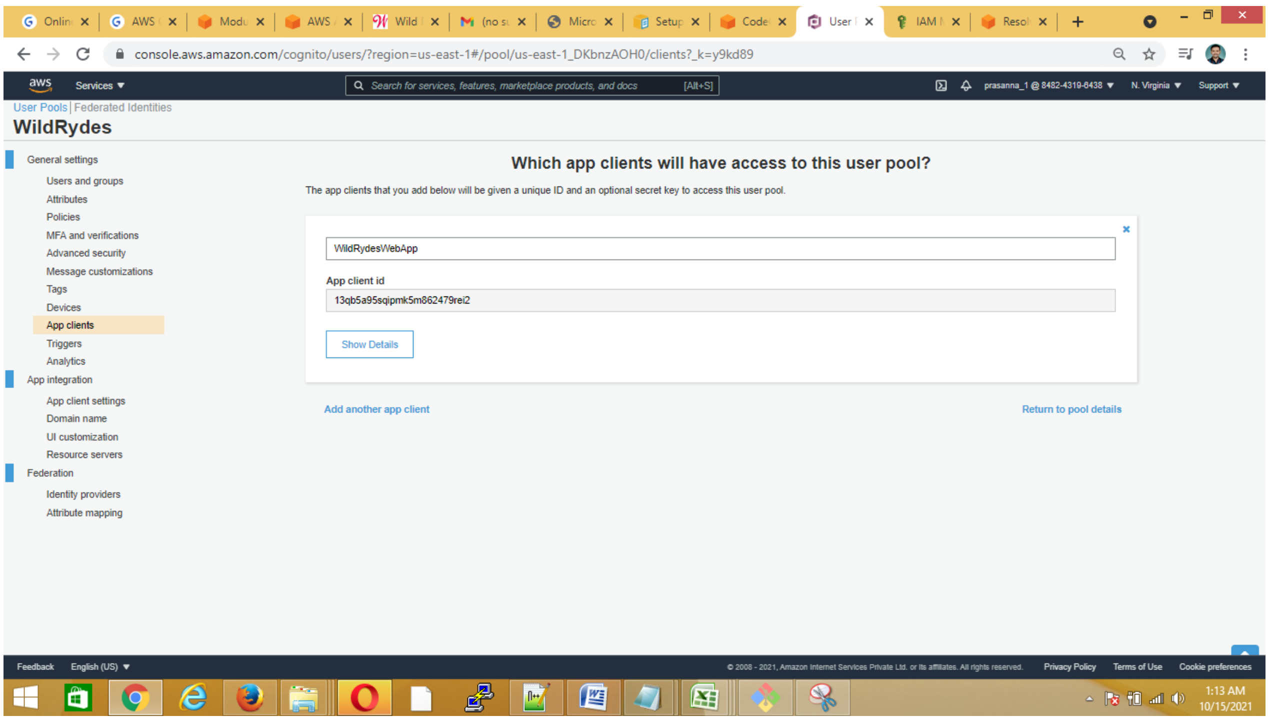
Pool Id	us-east-1_DKbnzAOH0
Pool ARN	arn:aws:cognito-idp:us-east-1:848243196438:userpool/us-east-1_DKbnzAOH0
Estimated number of users	0
Required attributes	email
Alias attributes	none
Username attributes	none
Enable case insensitivity?	Yes
Custom attributes	Choose custom attributes...
Minimum password length	8
Password policy	uppercase letters, lowercase letters, special characters, numbers
User sign ups allowed?	Users can sign themselves up
FROM email address	Default
Email Delivery through Amazon SES	No
	Note: You have chosen to have Cognito send emails on your behalf. Best practices suggest that customers send emails through Amazon SES for production User Pools due to a daily email limit. Learn more about email best practices.
MFA	Enable MFA...
Verifications	Email

Step 2 – Add an App to the User Pool

From the Amazon Cognito console the user pool is selected and then the App clients section is selected. A new app client is added. The Generate client secret option is deselected. Client secrets aren't currently supported with the JavaScript SDK.

- From the Pool Details page for r user pool, selected App clients.
- Choose Add an app client.
- Given the app client a name as "WildRydesWebApp."
- Unchecked the Generate client secret option.
- Chosen Create app client.

"WildRydesWebApp App Client" is created as shown in the screenshot below:


console.aws.amazon.com/cognito/users/?region=us-east-1#/pool/us-east-1_DKbnzAOH0/clients?_k=y9kd89

User Pools | Federated Identities

WildRydes

Which app clients will have access to this user pool?

The app clients that you add below will be given a unique ID and an optional secret key to access this user pool.

WildRydesWebApp

App client id

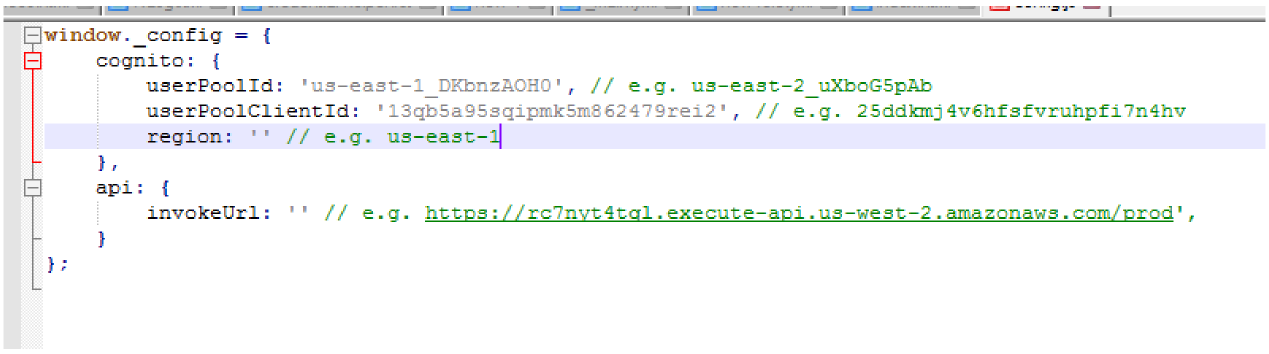
13qb5a95sqipmk5m862479rei2

Step 3 – Update website config

The /js/config.js file contains settings for the **user pool ID, app client ID and Region.** Updated this file with the settings from the user pool and app created in the previous steps

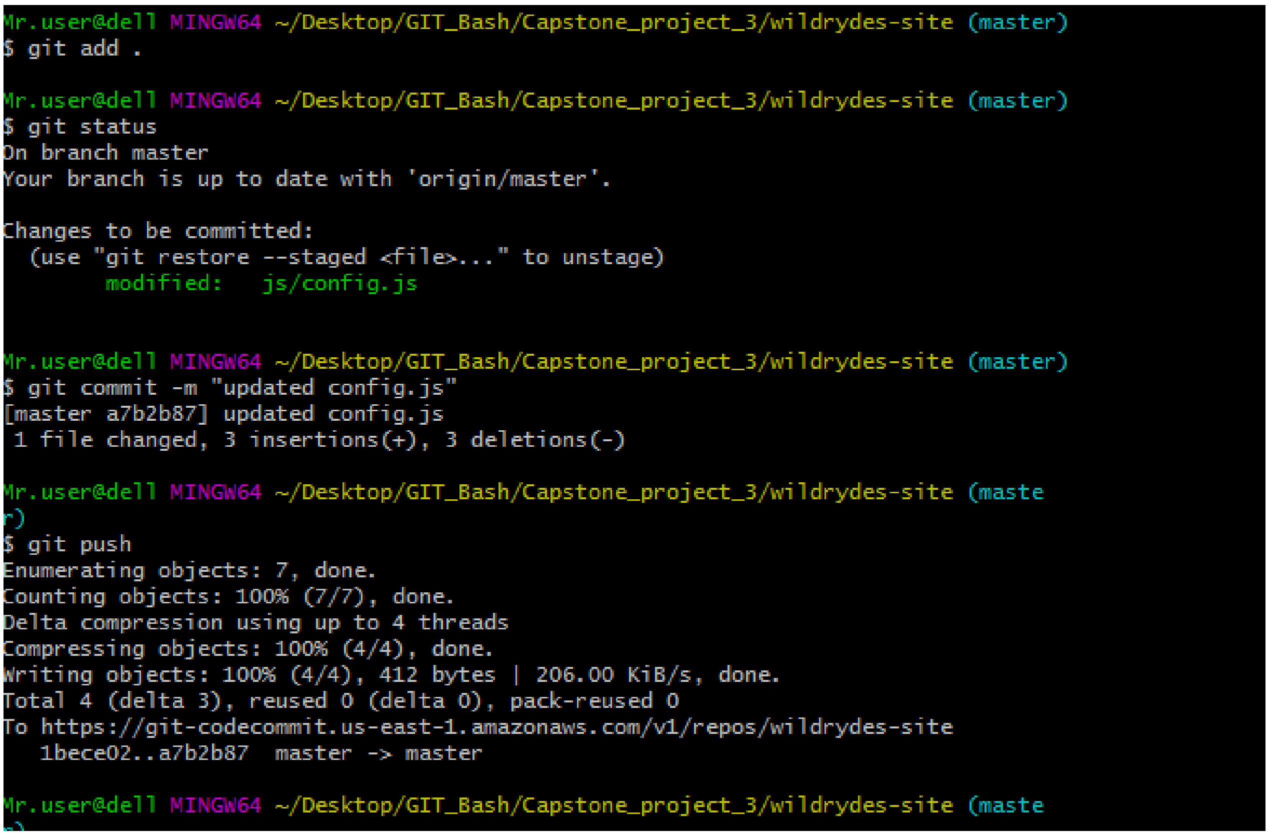
- Find the value for “userPoolId” on the Pool details page of the Amazon Cognito console.
- Find the value for “userPoolClientId” by selecting App clients from the left navigation bar. Use the value from the App client id field for the app created in the previous section.
- The value for region should be the AWS Region code where created user pool

These settings updated based on the current project settings.



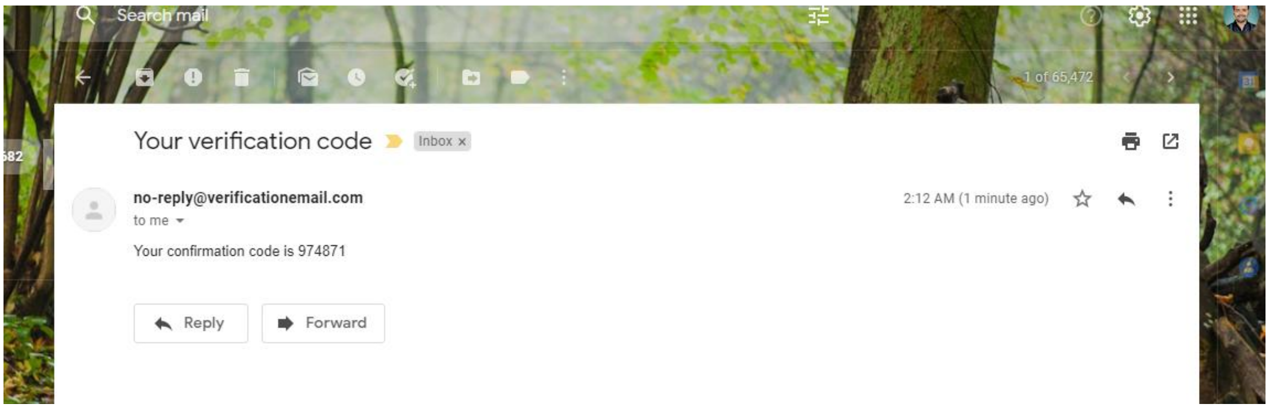
```
window._config = {
    cognito: {
        userPoolId: 'us-east-1_DKbnzAOH0', // e.g. us-east-2_uXboG5pAb
        userPoolClientId: '13qb5a95sqipmk5m862479rei2', // e.g. 25ddkmj4v6hfsfvruhpfi7n4hv
        region: '' // e.g. us-east-1
    },
    api: {
        invokeUrl: '' // e.g. https://rc7nyt4tql.execute-api.us-west-2.amazonaws.com/prod',
    }
};
```

Final step is to push this js file into CodeCommit repository.



```
Mr.user@dell MINGW64 ~/Desktop/GIT_Bash/Capstone_project_3/wildrydes-site (master)
$ git add .

Mr.user@dell MINGW64 ~/Desktop/GIT_Bash/Capstone_project_3/wildrydes-site (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   js/config.js


Mr.user@dell MINGW64 ~/Desktop/GIT_Bash/Capstone_project_3/wildrydes-site (master)
$ git commit -m "updated config.js"
[master a7b2b87] updated config.js
 1 file changed, 3 insertions(+), 3 deletions(-)

Mr.user@dell MINGW64 ~/Desktop/GIT_Bash/Capstone_project_3/wildrydes-site (maste
r)
$ git push
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 4 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 412 bytes | 206.00 KiB/s, done.
Total 4 (delta 3), reused 0 (delta 0), pack-reused 0
To https://git-codecommit.us-east-1.amazonaws.com/v1/repos/wildrydes-site
   1bece02..a7b2b87  master -> master

Mr.user@dell MINGW64 ~/Desktop/GIT_Bash/Capstone_project_3/wildrydes-site (maste
```

Opened Gmail and verified by registering as a new user.

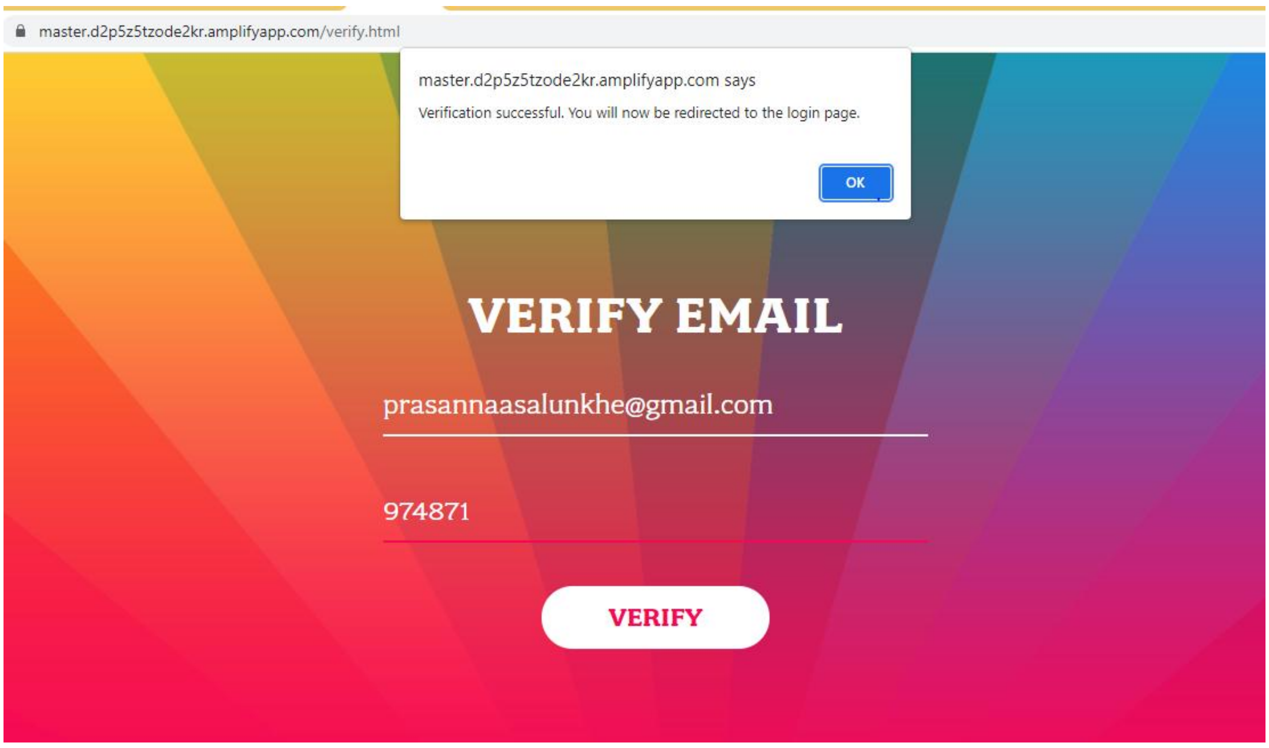


Your verification code

no-reply@verificationemail.com
to me

2:12 AM (1 minute ago)

Your confirmation code is 974871



master.d2p5z5tzode2kr.amplifyapp.com says

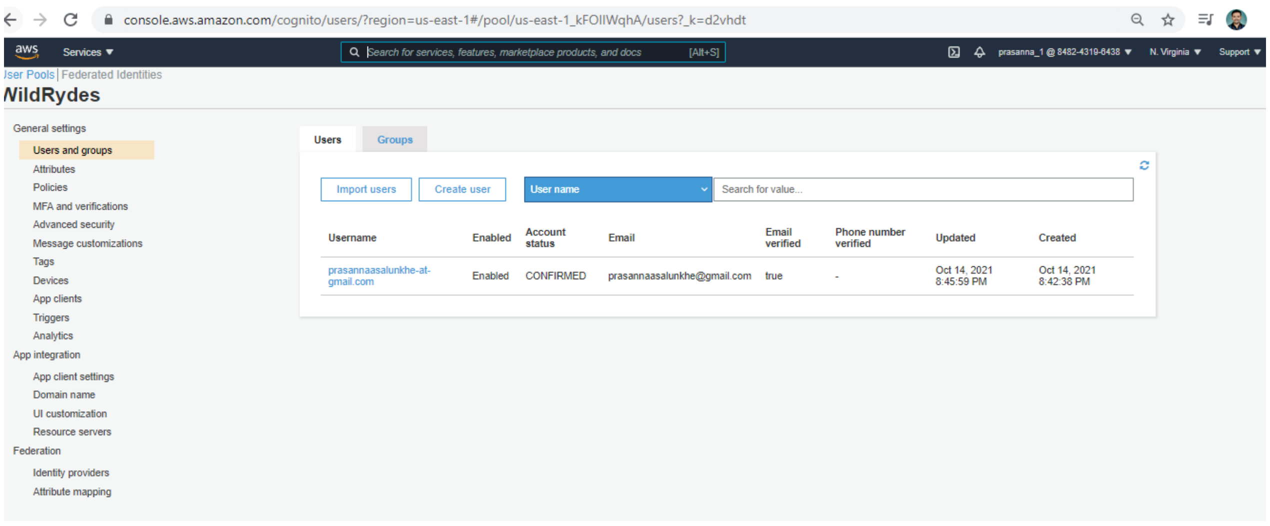
Verification successful. You will now be redirected to the login page.

VERIFY EMAIL

prasannaasalunkhe@gmail.com

974871

And finally new user is created.


console.aws.amazon.com/cognito/users/?region=us-east-1#/pool/us-east-1_kFOIlWqhA/users?_k=d2vhdt

User Pools | Federated Identities

WildRydes

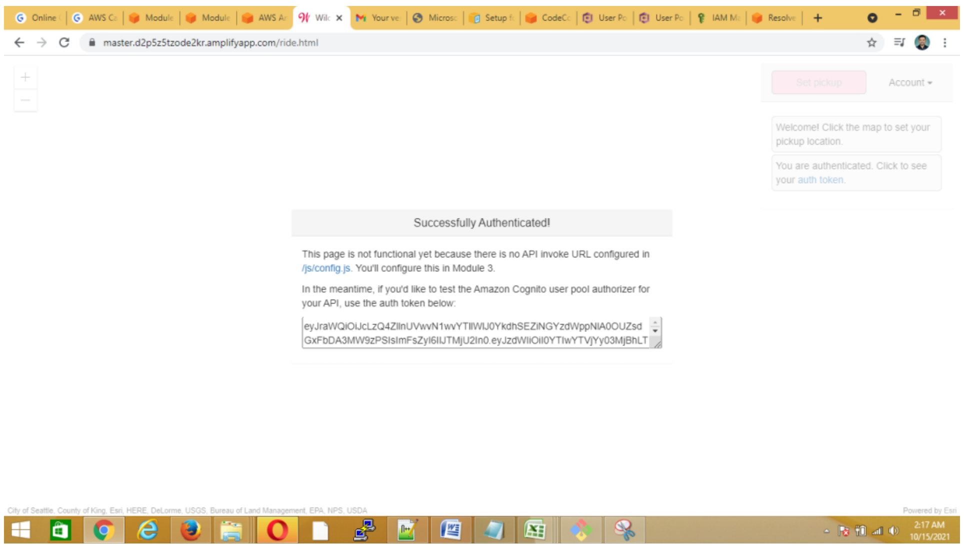
Username	Enabled	Account status	Email	Email verified	Phone number verified	Updated	Created
prasannaasalunkhe-at-gmail.com	Enabled	CONFIRMED	prasannaasalunkhe@gmail.com	true	-	Oct 14, 2021 8:45:59 PM	Oct 14, 2021 8:42:38 PM

And finally on the actual website, sign in is also success.

Validated the implementation by using steps below

- Visited /register.html under r website domain, or chosen the Giddy Up.
- Completed the registration form and chosen Let's Ryde.
- Confirmed new user using link came to r Gmail.
- Navigated to Cognito under Security, Identity & Compliance.
- Chosen Manage r User Pools. Chosen Confirm user to finalize the account creation process.
- Visited /signin.html and log in using the email address and password entered during the registration step.

Redirected to /ride.html. Pop-up came for Notification that the API is not configured. As shown below.


master.d2p5z5tzode2kr.amplifyapp.com/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:

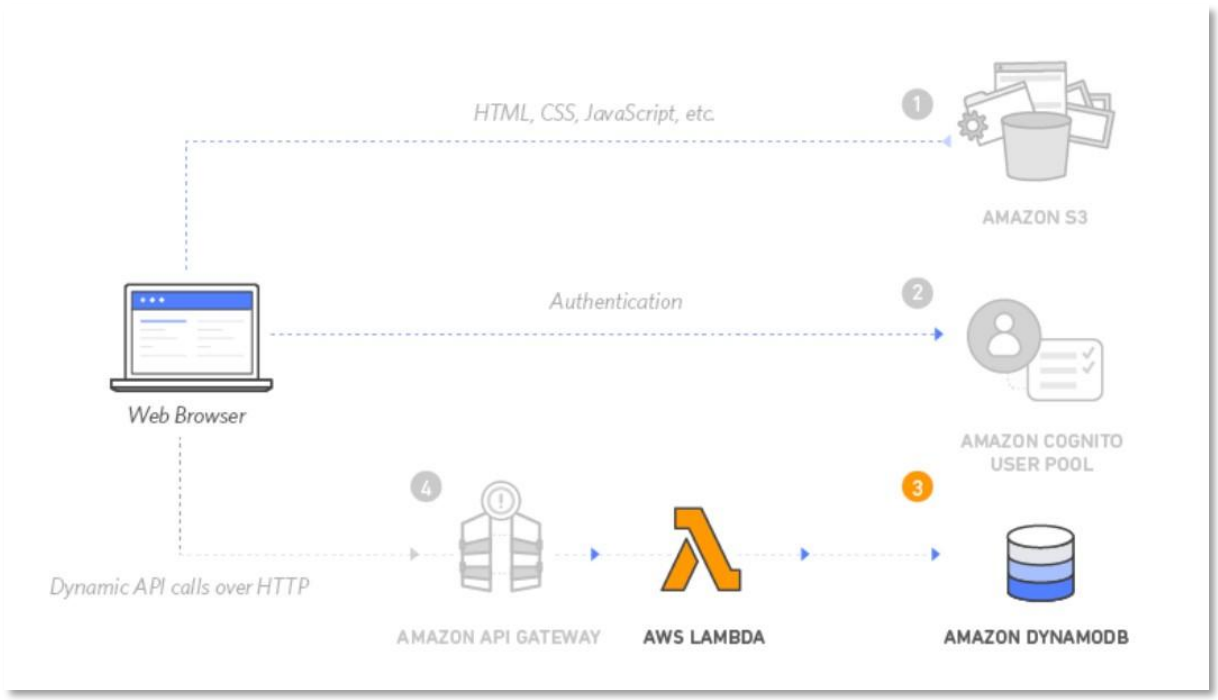
```
eyJraWQiOiJcLzQ4ZlInUVwvN1wvYTlIWlJ0YkdhSEZiNGYzdWppNlA0OUZsd
GxFbDA3MW9zPSIsImFsZyI6IlJTMjU2In0.eyJzdWIiOiI0YTIwYTVjYy03MjBhLT
```

Welcome! Click the map to set your pickup location.

You are authenticated. Click to see your auth token.

Module 3 – Build a server less backend

Architectural Overview:

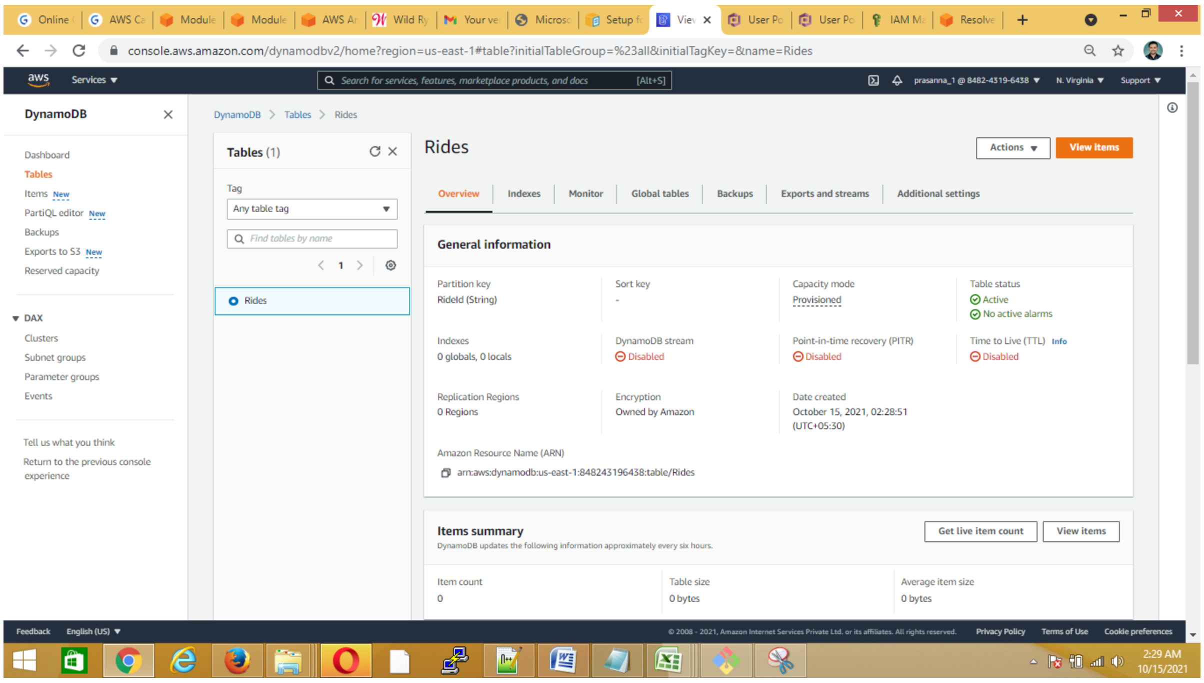


A Lambda function that will be invoked each time a user requests a unicorn will be implemented. The function will select a unicorn from the fleet, record the request in a DynamoDB table and then respond to the front-end application with details about the unicorn being dispatched.

The function is invoked from the browser using Amazon API Gateway.

Step 1 – Create an Amazon DynamoDB Table

- From the AWS Management Console, DynamoDB under Databases is selected.
- Create table is chosen
- "**Rides**" is entered for the Table name. This field is case sensitive.
- "**RideId**" is entered for the Partition key and String for the key type is selected. This field iscase sensitive.
- The Use default settings box is checked and Create button is clicked.
- Noted the ARN. We will use this in the next section.



Rides

General information

Partition key	Sort key	Capacity mode	Table status
RideId (String)	-	Provisioned	Active; No active alarms
Indexes	**DynamoDB stream**	**Point-in-time recovery (PITR)**	**Time to Live (TTL)**
0 globals, 0 locals	Disabled	Disabled	Disabled
Replication Regions	**Encryption**	**Date created**	
0 Regions	Owned by Amazon	October 15, 2021, 02:28:51 (UTC+05:30)	

Amazon Resource Name (ARN)
arn:aws:dynamodb:us-east-1:848243196438:table/Rides

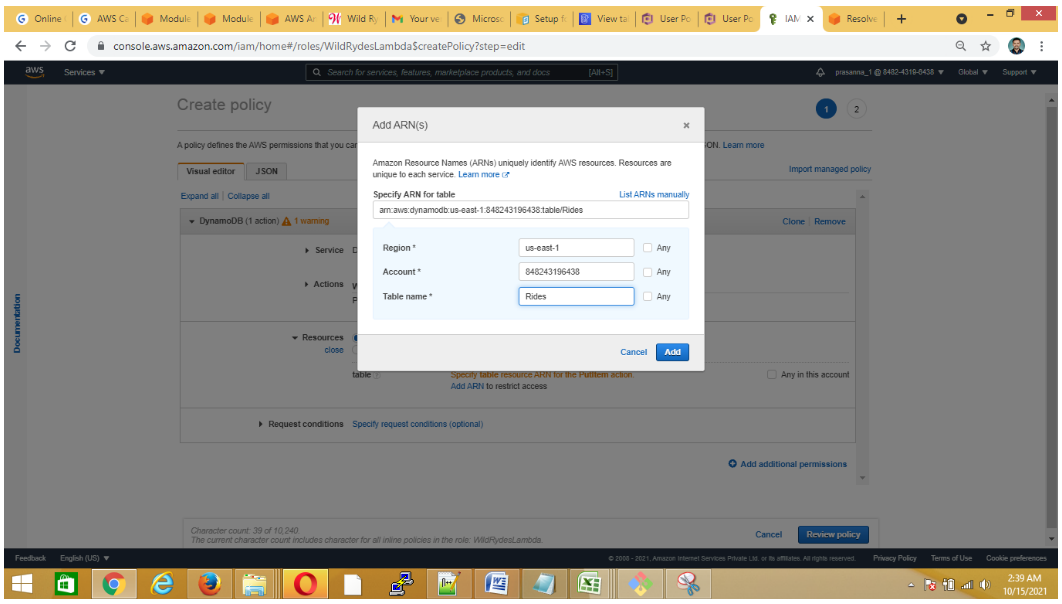
Items summary
DynamoDB updates the following information approximately every six hours.

Item count	Table size	Average item size
0	0 bytes	0 bytes

Step 2 – Create an IAM role for Lambda function

Every Lambda function has an IAM role associated with it .The Roles are selected from the left navigation bar in IAM services page, and then CreateNew Role is chosen.

- Lambda is selected for the role type from the AWS service group, then “Next: Permissions” button is selected.
- Then **“AWSLambdaBasicExecutionRole”** is entered in the Filter text box and selected.
- Enter “WildRydesLambda” as a Role Name and the role is created.
- **“WildRydesLambda”** is typed into the filter box on the Roles page and the role is chosen.
- Now on the Permissions tab, the inline policy link is chosen in the lower right corner to create a new inline policy.
- “Choose a service” is selected.
- “DynamoDB” is selected
- “Select actions” is chosen
- “PutItem” is entered into the search box labeled Filter actions and check the boxnext to PutItem when it appears.
- The Resources section is selected
- With the Specific option selected, the Add ARN link in the table section is chosen.



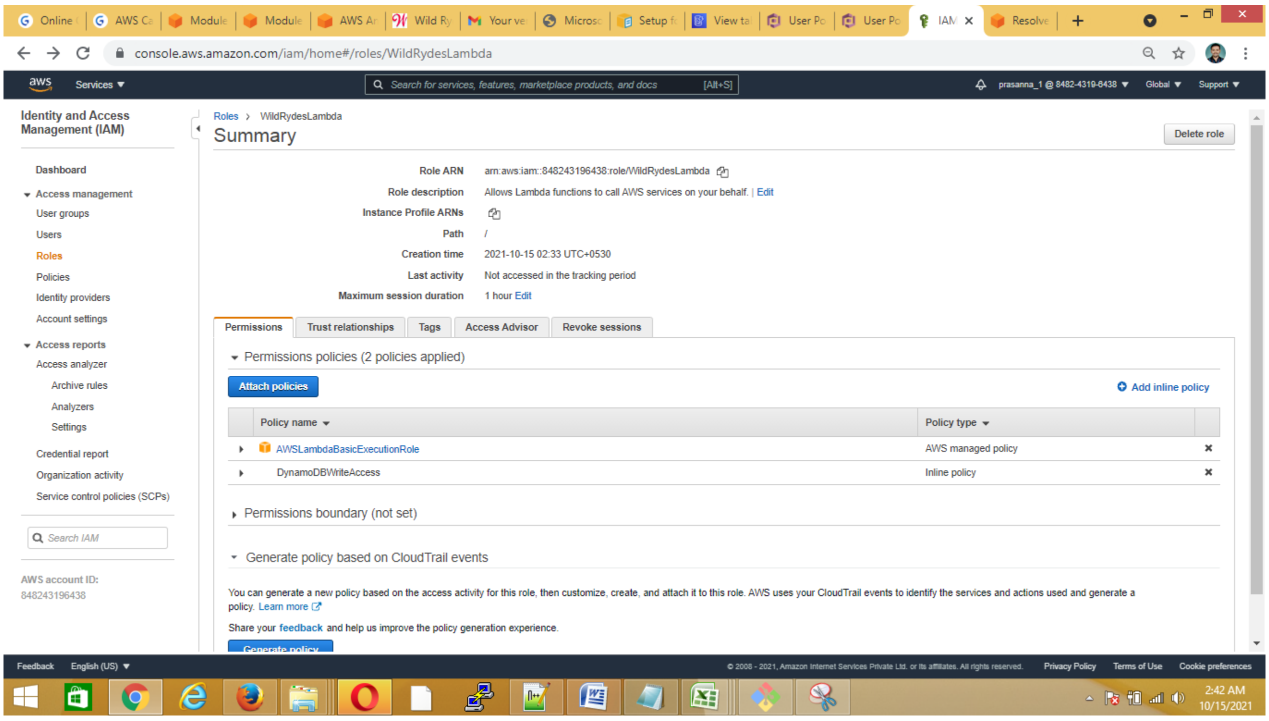
Add ARN(s)

Amazon Resource Names (ARNs) uniquely identify AWS resources. Resources are unique to each service. Learn more

Specify ARN for table: arn:aws:dynamodb:us-east-1:848243196438:table/Rides

Region *	us-east-1	Any
Account *	848243196438	Any
Table name *	Rides	Any

- Review Policy has been chosen
- **"DynamoDBWriteAccess"** is entered for the policy name and Create policy has been selected at teend.



Roles › WildRydesLambda

Summary

Role ARN	arn:aws:iam::848243196438:role/WildRydesLambda
Role description	Allows Lambda functions to call AWS services on your behalf. | Edit
Instance Profile ARNs	
Path	/
Creation time	2021-10-15 02:33 UTC+0530
Last activity	Not accessed in the tracking period
Maximum session duration	1 hour Edit

Permissions policies (2 policies applied)

Policy name	Policy type
AWSLambdaBasicExecutionRole	AWS managed policy
DynamoDBWriteAccess	Inline policy

Permissions boundary (not set)

Generate policy based on CloudTrail events

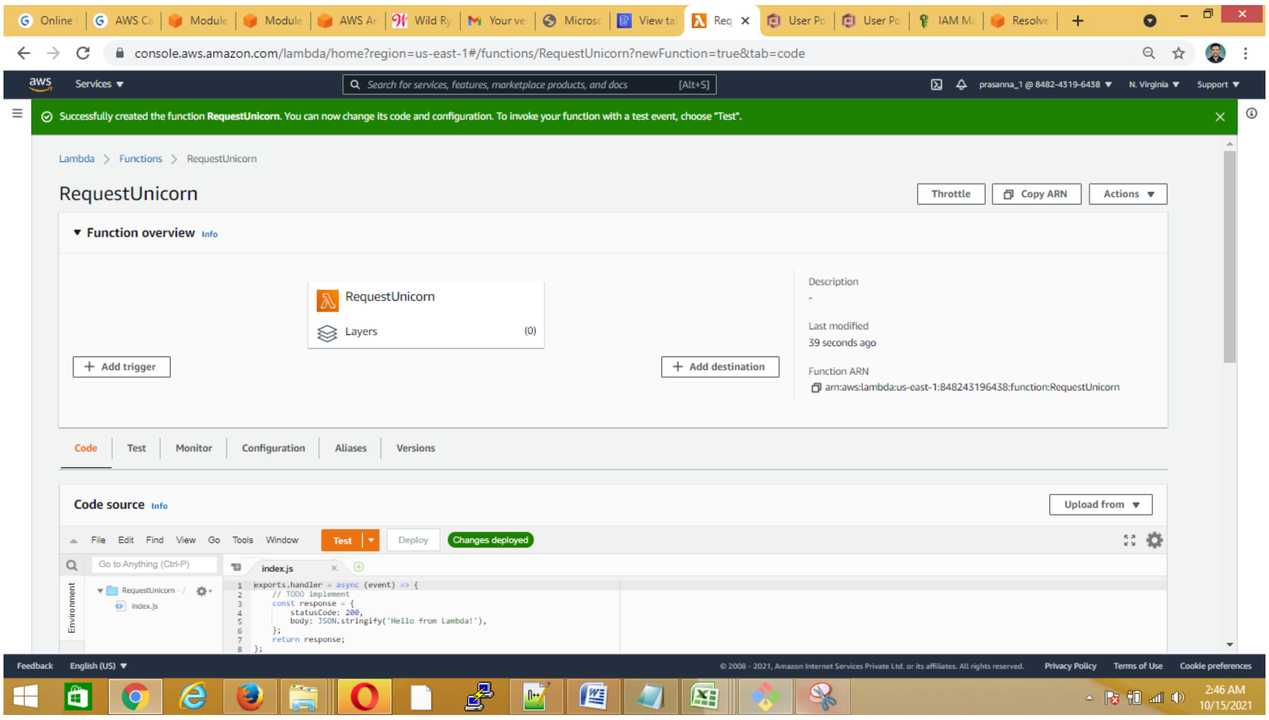
You can generate a new policy based on the access activity for this role, then customize, create, and attach it to this role. AWS uses your CloudTrail events to identify the services and actions used and generate a policy. Learn more

Share your feedback and help us improve the policy generation experience.

Step 3 – Create a Lambda Function for Handling Requests

Used the AWS Lambda console to create a new Lambda function called RequestUnicorn that will process the API requests. Use the provided requestUnicorn.js example implementation for function code. Uploaded zip into S3 and then used it by entering s-3 bucket URL in right hand side "Upload from".

- Navigate to Lambda services from console and click create a function
- The default Author from scratch card is selected.
- Function name is entered as "RequestUnicorn"
- Node.js 14.x for the Runtime is selected.
- **WildRydesLambda** role is selected.
- "Create function" is clicked to create the function.


console.aws.amazon.com/lambda/home?region=us-east-1#/functions/RequestUnicorn?newFunction=true&tab=code

Successfully created the function **RequestUnicorn**. You can now change its code and configuration. To invoke your function with a test event, choose "Test".

Lambda > Functions > RequestUnicorn

RequestUnicorn

Function overview

Description: -

Last modified: 39 seconds ago

Function ARN: arn:aws:lambda:us-east-1:848243196438:function:RequestUnicorn

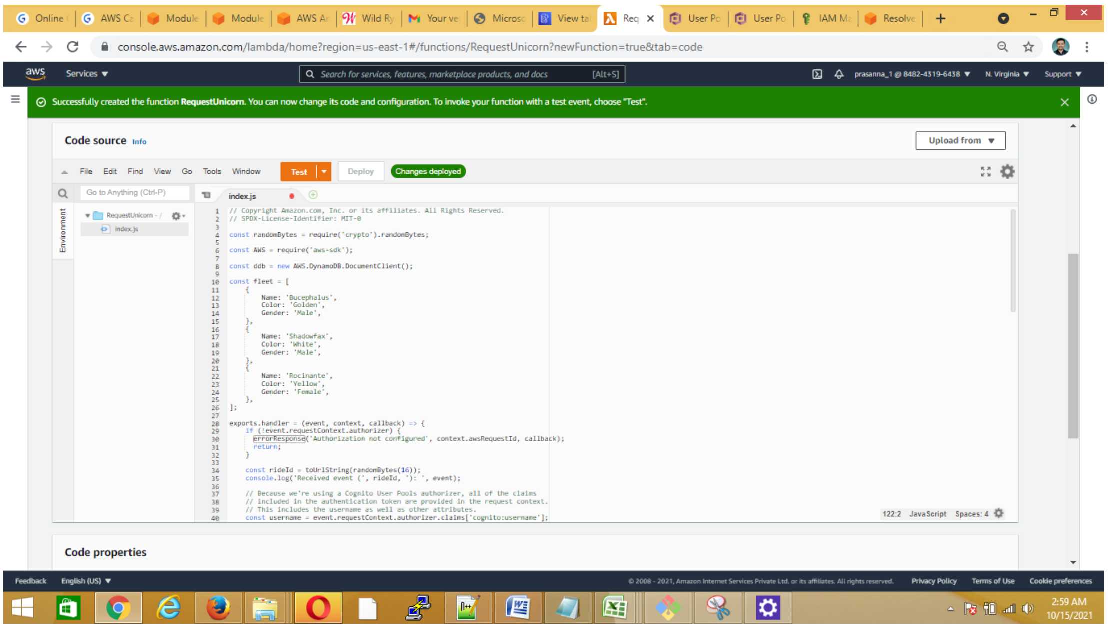
Code source

```
exports.handler = async (event) => {
    // TODO implement
    const response = {
        statusCode: 200,
        body: JSON.stringify('Hello from Lambda!'),
    };
    return response;
};
```

© 2008 - 2021, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.

Scroll down to the Function code section and replace the existing code inthe index.js code editor with the contents of requestUnicorn.js.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: MIT-0

const randomBytes = require('crypto').randomBytes;

const AWS = require('aws-sdk');

const ddb = new AWS.DynamoDB.DocumentClient();

const fleet = [
    {
        Name: 'Bucephalus',
        Color: 'Golden',
        Gender: 'Male',
    },
    {
        Name: 'Shadowfax',
        Color: 'White',
        Gender: 'Male',
    },
    {
        Name: 'Rocinante',
        Color: 'Yellow',
        Gender: 'Female',
    },
];

exports.handler = (event, context, callback) => {
    if (!event.requestContext.authorizer) {
      errorResponse('Authorization not configured', context.awsRequestId, callback);
      return;
    }

    const rideId = toUrlString(randomBytes(16));
    console.log('Received event (', rideId, '): ', event);

    // Because we're using a Cognito User Pools authorizer, all of the claims
    // included in the authentication token are provided in the request context.
    // This includes the username as well as other attributes.
    const username = event.requestContext.authorizer.claims['cognito:username'];

    // The body field of the event in a proxy integration is a raw string.
```


Successfully created the function **RequestUnicorn**. You can now change its code and configuration. To invoke your function with a test event, choose "Test".

Code source Info

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: MIT-0

const randomBytes = require('crypto').randomBytes;

const AWS = require('aws-sdk');

const ddb = new AWS.DynamoDB.DocumentClient();

const fleet = [
    {
        Name: 'Bucephalus',
        Color: 'Golden',
        Gender: 'Male',
    },
    {
        Name: 'Shadowfax',
        Color: 'White',
        Gender: 'Male',
    },
    {
        Name: 'Rocinante',
        Color: 'Yellow',
        Gender: 'Female',
    },
];

exports.handler = (event, context, callback) => {
    if (!event.requestContext.authorizer) {
      errorResponse('Authorization not configured', context.awsRequestId, callback);
      return;
    }

    const rideId = toUrlString(randomBytes(16));
    console.log('Received event (', rideId, '): ', event);

    // Because we're using a Cognito User Pools authorizer, all of the claims
    // included in the authentication token are provided in the request context.
    // This includes the username as well as other attributes.
    const username = event.requestContext.authorizer.claims['cognito:username'];
```

Code properties

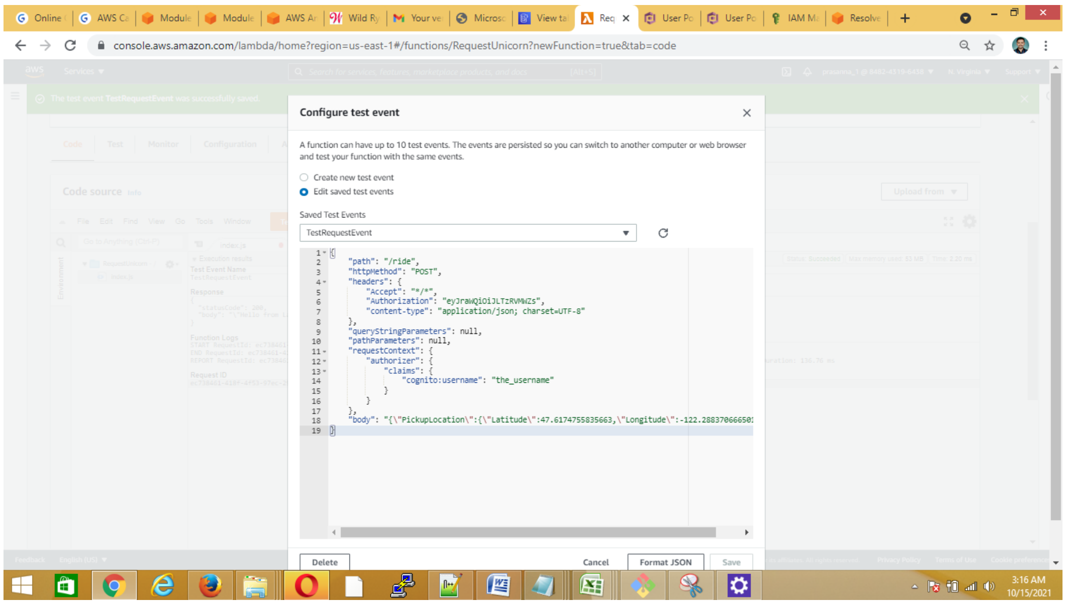
Clicked “save" in the file of the page.

Step 4 – Validate r Implementation

This is done by testing the function that built using the AWS Lambda console

Created and configured a test event as shown in the screenshot below:

- From the main edit screen for function, selected Configure test event from the Select a test event... dropdown.
- Keep Create new test event selected.
- Entered TestRequestEvent in the Event name field
- Copied and pasted the following test event into the editor


Configure test event

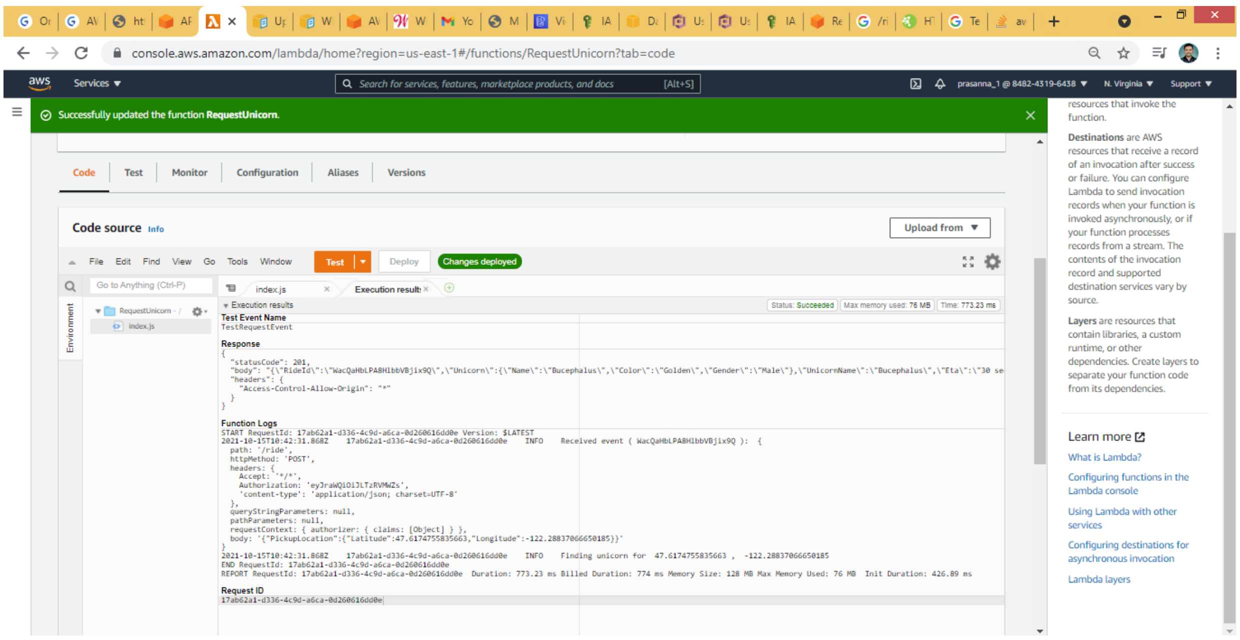
A function can have up to 10 test events. The events are persisted so you can switch to another computer or web browser and test your function with the same events.

Create new test event [✓ Edit saved test events]

Saved Test Events: 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\":-122.28837066650
}
```

Then on the function screen, clicked on Test to check the response status code as 201


console.aws.amazon.com/lambda/home?region=us-east-1#/functions/RequestUnicorn?tab=code
Services
Search for services, features, marketplace products, and docs [Alt+S]
N. Virginia
Support
Code
Test
Monitor
Configuration
Aliases
Versions
Code source Info
Upload from
File Edit Find View Go Tools Window
Test
Deploy
Changes deployed
Go to Anything (Ctrl-P)
Environment
RequestUnicorn
index.js
Execution results
Status: Succeeded
Max memory used: 76 MB
Time: 773.23 ms
Learn more
What is Lambda?
Configuring functions in the Lambda console
Using Lambda with other services
Configuring destinations for asynchronous invocation
Lambda layers

Successfully updated the function **RequestUnicorn**.

resources that invoke the function.

Destinations are AWS resources that receive a record of an invocation after success or failure. You can configure Lambda to send invocation records when your function is invoked asynchronously, or if your function processes records from a stream. The contents of the invocation record and supported destination services vary by source.

Layers are resources that contain libraries, a custom runtime, or other dependencies. Create layers to separate your function code from its dependencies.

Test Event Name

```
TestRequestEvent
```

Response

```
{
  "statusCode": 201,
  "body": "{\"RideId\":\"WacQaHbLPA8HlbbVBjix9Q\",\"Unicorn\":{\"Name\":\"Bucephalus\",\"Color\":\"Golden\",\"Gender\":\"Male\"},\"UnicornName\":\"Bucephalus\",\"Eta\":\"30 se
  "headers": {
    "Access-Control-Allow-Origin": "*"
  }
}
```

Function Logs

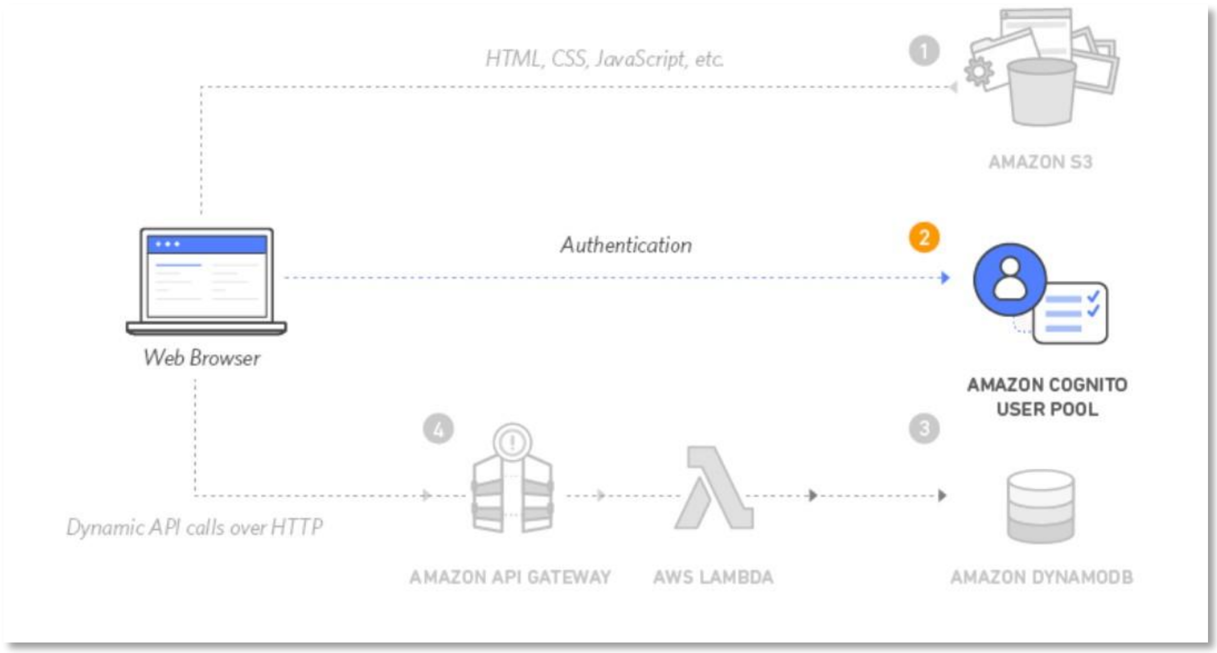
```
START RequestId: 17ab62a1-d336-4c9d-a6ca-0d260616dd0e Version: $LATEST
2021-10-15T10:42:31.868Z    17ab62a1-d336-4c9d-a6ca-0d260616dd0e    INFO    Received event ( WacQaHbLPA8HlbbVBjix9Q ):  {
  path: '/ride',
  httpMethod: 'POST',
  headers: {
    Accept: '*/*',
    Authorization: 'eyJraWQiOiJLTzRVMWZs',
    'content-type': 'application/json; charset=UTF-8'
  },
  queryStringParameters: null,
  pathParameters: null,
  requestContext: { authorizer: { claims: [Object] } },
  body: '{"PickupLocation":{"Latitude":47.6174755835663,"Longitude":-122.28837066650185}}'
}
2021-10-15T10:42:31.868Z    17ab62a1-d336-4c9d-a6ca-0d260616dd0e    INFO    Finding unicorn for  47.6174755835663 ,  -122.28837066650185
END RequestId: 17ab62a1-d336-4c9d-a6ca-0d260616dd0e
REPORT RequestId: 17ab62a1-d336-4c9d-a6ca-0d260616dd0e  Duration: 773.23 ms Billed Duration: 774 ms Memory Size: 128 MB Max Memory Used: 76 MB  Init Duration: 426.89 ms
```

Request ID

```
17ab62a1-d336-4c9d-a6ca-0d260616dd0e
```

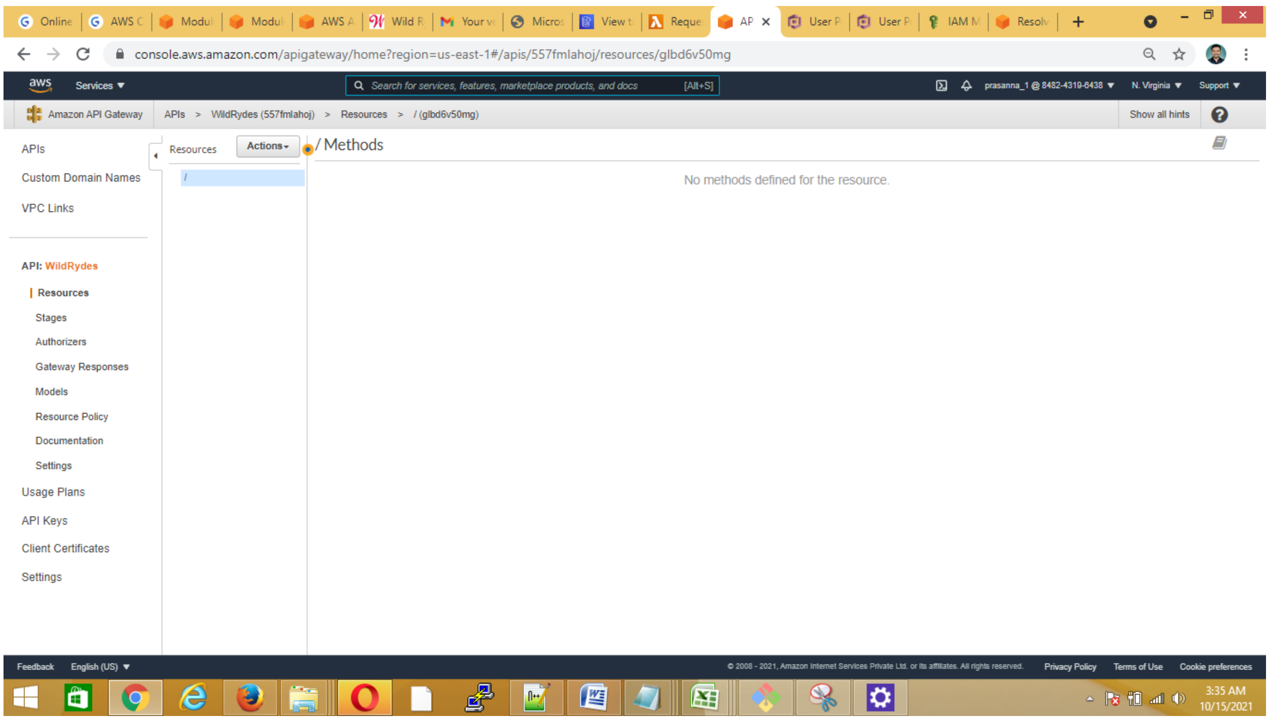
Module 4 – Deploy a RESTful API

Architectural Overview:



Step 1 – Create a new REST API

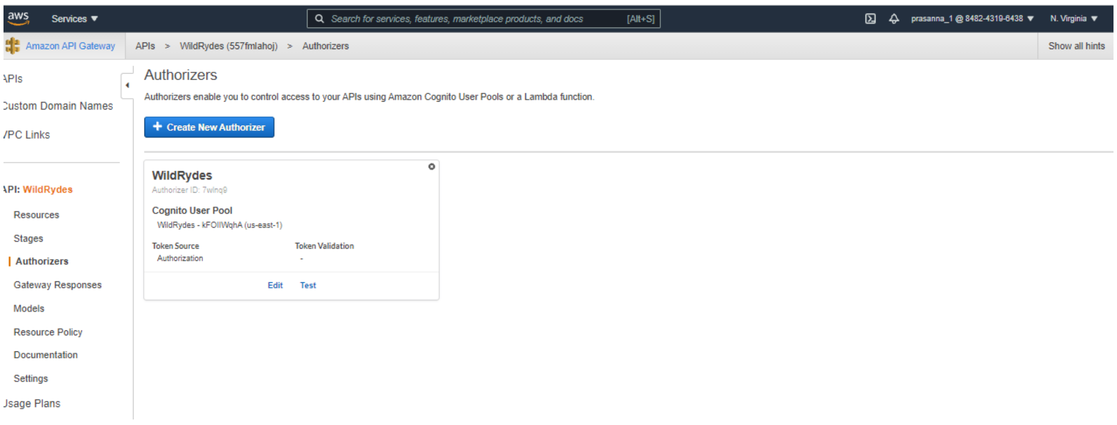
- Selected API Gateway under Application Services and Chosen Create API.
- Entered API Gateway Name as a “WildRydes”.
- Kept Edge optimized selected in the Endpoint Type.
- And clicked on create API



No methods defined for the resource.

Step 2 – Create a Cognito User Pools Authorizer

- A new authorizer of name WildRydes is created.
- Selected Cognito for the type.
- In the Region select the Region where created r Cognito user pool
- Type is Cognito and WildRydes is entered in Cognito User
  Pool input. Token Source is set as "Authorization".
- Chosen create.



Authorizers

Authorizers enable you to control access to your APIs using Amazon Cognito User Pools or a Lambda function.

WildRydes

Authorizer ID: 7wlnq9

Cognito User Pool

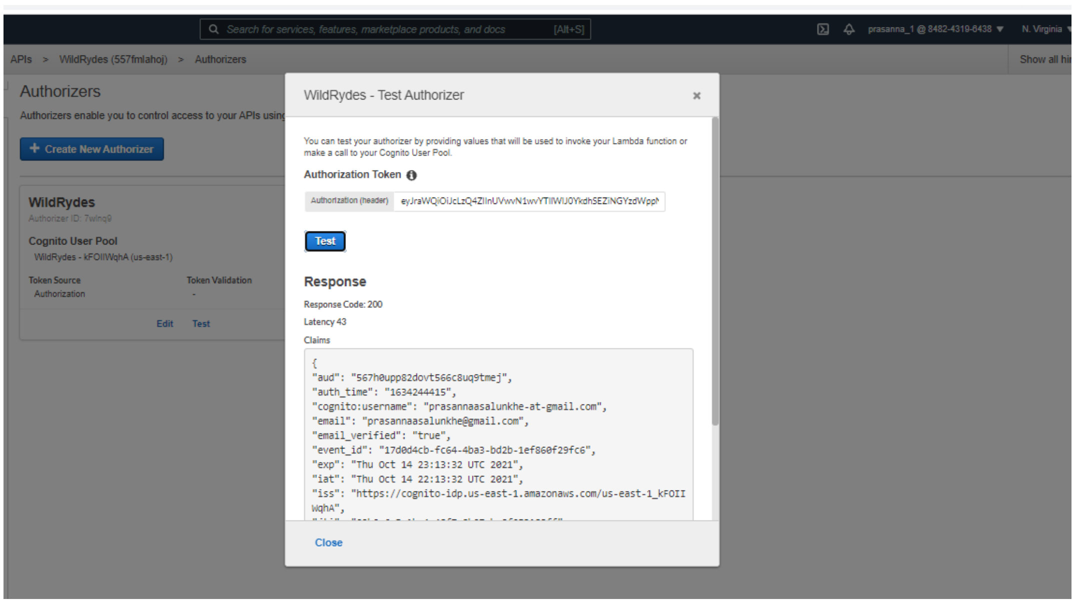
WildRydes - kFOIIWqhA (us-east-1)

Token Source: Authorization

Token Validation: -

Verified authorizer configuration as shown below

- Visited /ride.html under r website's domain.
- Copied the auth token from the notification on the /ride.html,
- Went back to creating the Authorizer
- Clicked on Test at the bottom of the card for the authorizer.
- Pasted the auth token into the Authorization Token field in the popup dialog.
- Clicked on test button and verified that the response code is 200.



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): eyJraWQiOiJcLzQ4ZlInUVwvN1wvYTllWlJ0YkdhSEZiNGYzdWppN

Response

Response Code: 200

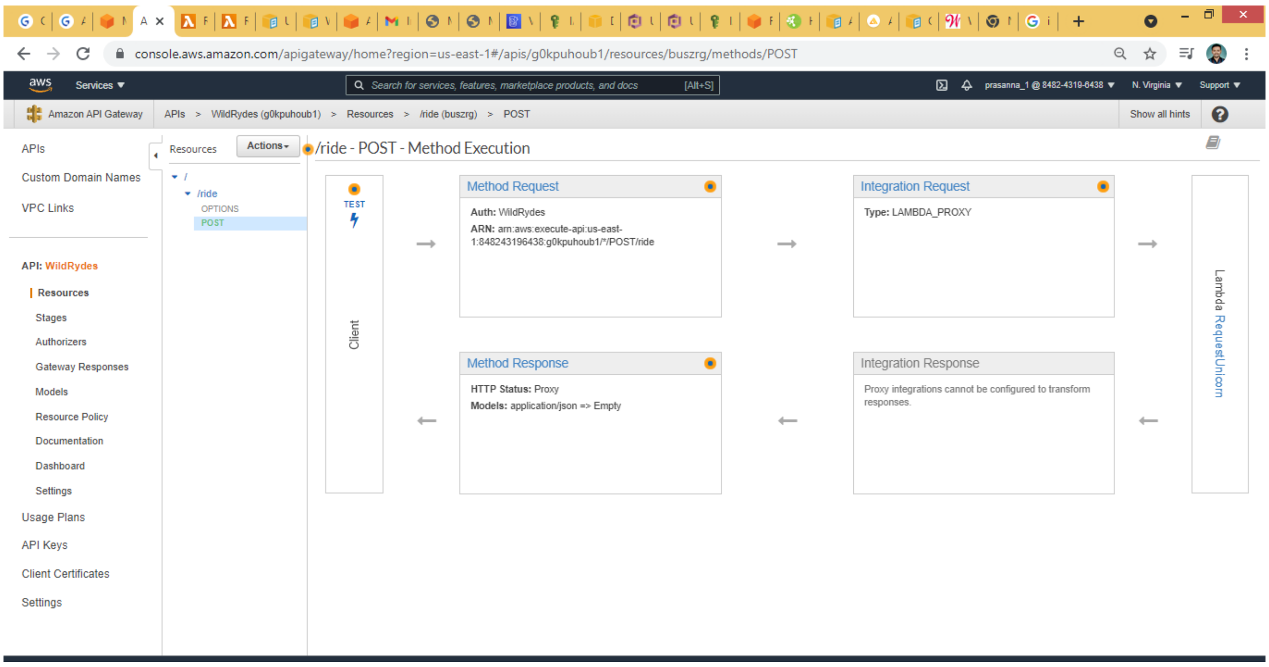
Latency 43

Claims

```
{
"aud": "567h0upp82dovt566c8uq9tmej",
"auth_time": "1634244415",
"cognito:username": "prasannaasalunkhe-at-gmail.com",
"email": "prasannaasalunkhe@gmail.com",
"email_verified": "true",
"event_id": "17d0d4cb-fc64-4ba3-bd2b-1ef860f29fc6",
"exp": "Thu Oct 14 23:13:32 UTC 2021",
"iat": "Thu Oct 14 22:13:32 UTC 2021",
"iss": "https://cognito-idp.us-east-1.amazonaws.com/us-east-1_kFOII
WqhA",
```

Step 3 – Create a new resource and method

- Scroll towards Resources under “WildRydes API.”
- Selected “Create Resource” from the Actions dropdown.
- Entered ride as the Resource Name.
- Selected Enable API Gateway CORS for the resource.
- Created Resource.
- Selected ride resource and from the “Action dropdown” selected “Create Method.”
- Selected POST from the new dropdown that appears then ticked the checkmark.
- Selected Lambda Function for the integration type and checked the box for Use Lambda Proxy integration and selected the Region.
- Entered the name of the function created “RequestUnicorn” for Lambda Function and chosen a save. When prompted to give Amazon API Gateway permission to invoke r function, chosen OK.
- Chosen on the Method Request card.
- Chosen the pencil icon next to Authorization.
- Selected the WildRydes Cognito user pool authorizer, and clicked the checkmark icon.



/ride - POST - Method Execution

Method Request
Auth: WildRydes
ARN: arn:aws:execute-api:us-east-1:848243196438:g0kpuhoub1/*/POST/ride

Integration Request
Type: LAMBDA_PROXY

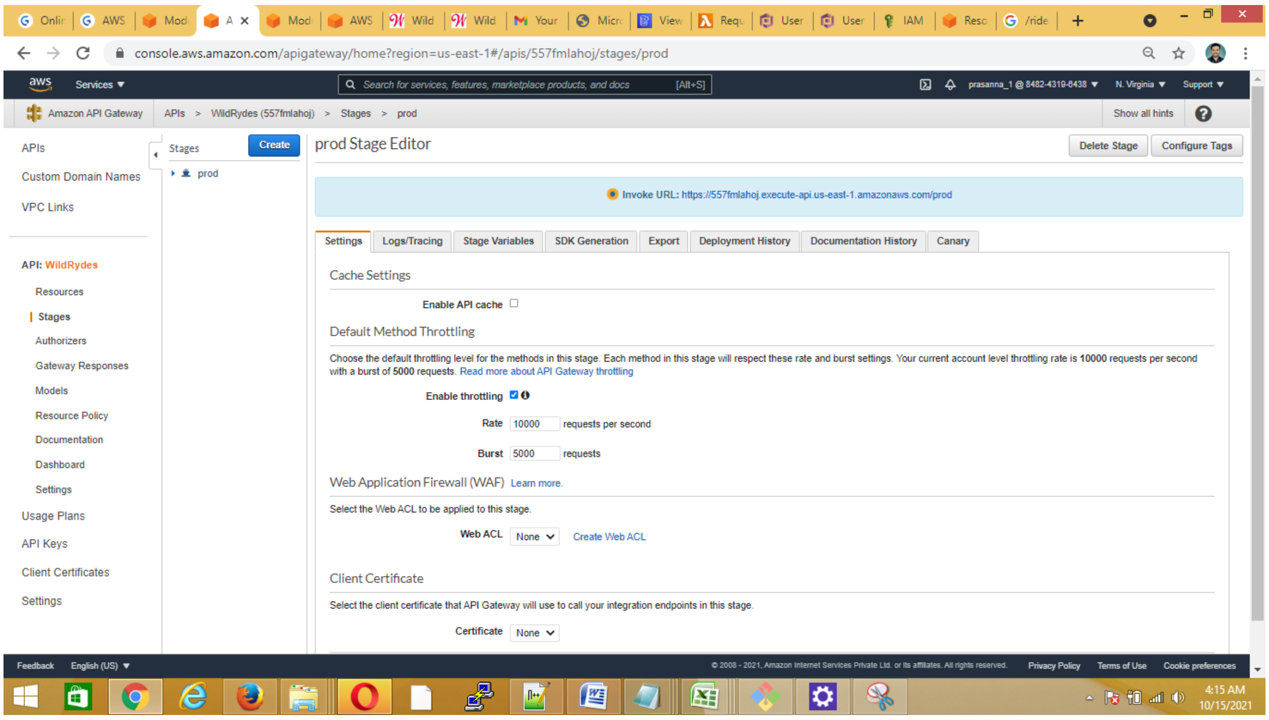
Method Response
HTTP Status: Proxy
Models: application/json => Empty

Integration Response
Proxy integrations cannot be configured to transform responses.

Step 4 – Deploy the API

In this step, a new “prod” deploy API is created. As shown below

- Selected “Deploy API” in the Actions drop-down list.
- Selected [New Stage] in the Deployment stage drop-down list.
- “Prod” name is given for the Stage Name.
- Chosen Deploy. And note the Invoke URL required to use it in the next section


console.aws.amazon.com/apigateway/home?region=us-east-1#/apis/557fmlahoj/stages/prod

Amazon API Gateway > APIs > WildRydes (557fmlahoj) > Stages > prod

prod Stage Editor

Invoke URL: https://557fmlahoj.execute-api.us-east-1.amazonaws.com/prod

Settings | Logs/Tracing | Stage Variables | SDK Generation | Export | Deployment History | Documentation History | Canary

Cache Settings

Enable API cache ☐

Default Method Throttling

Choose the default throttling level for the methods in this stage. Each method in this stage will respect these rate and burst settings. Your current account level throttling rate is **10000** requests per second with a burst of **5000** requests. Read more about API Gateway throttling

Enable throttling ✓

Rate 10000 requests per second

Burst 5000 requests

Web Application Firewall (WAF) Learn more.

Select the Web ACL to be applied to this stage.

Web ACL None

Client Certificate

Select the client certificate that API Gateway will use to call your integration endpoints in this stage.

Certificate None

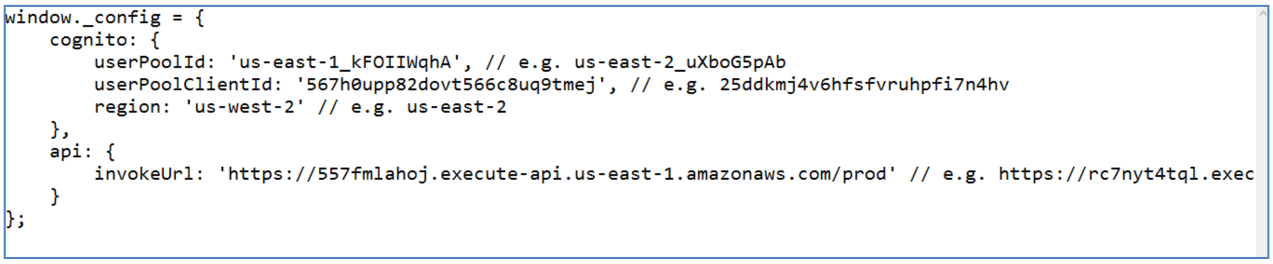
© 2008 - 2021, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.

Step 5 – Update the website config

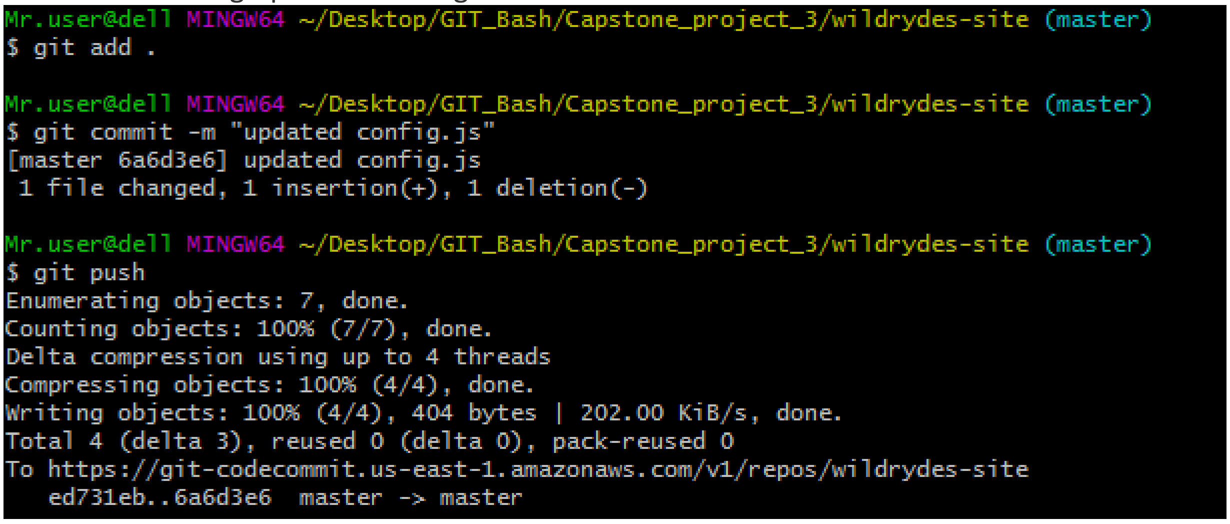
Opened the file pen the config.js file in a text editor.

And

In the js/config.js file, invokeURL is updated which we have created in last step.

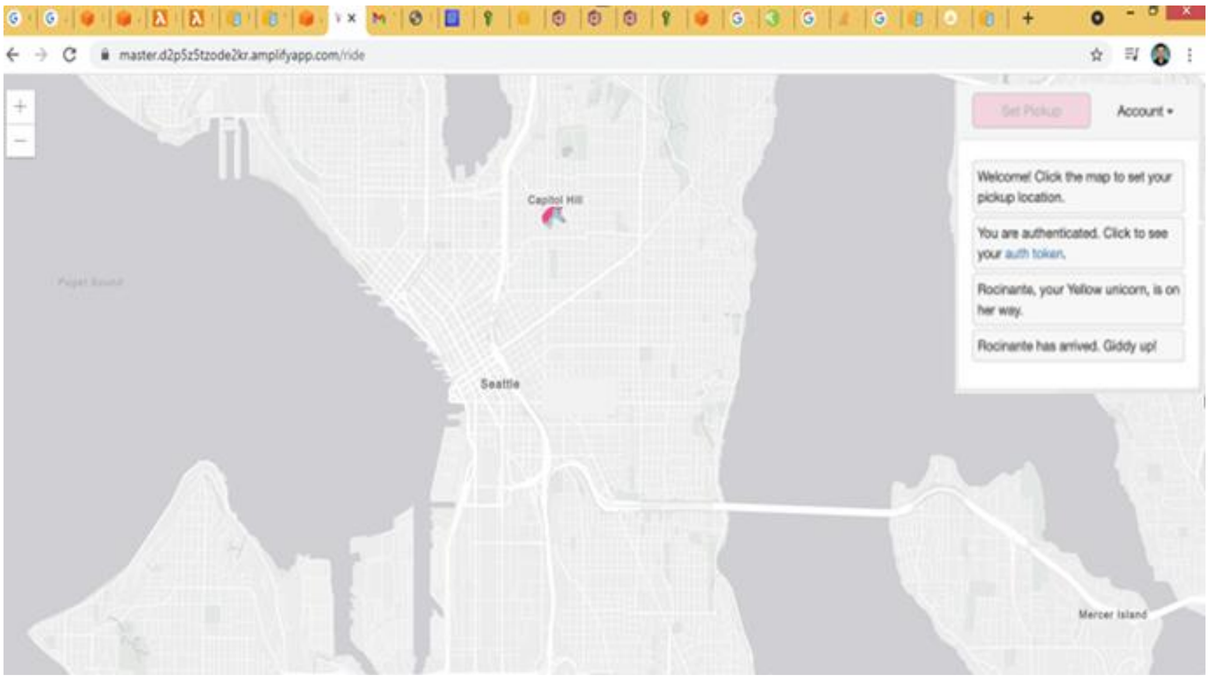

```
window._config = {
    cognito: {
        userPoolId: 'us-east-1_kFOIIWqhA', // e.g. us-east-2_uXboG5pAb
        userPoolClientId: '567h0upp82dovt566c8uq9tmej', // e.g. 25ddkmj4v6hfsfvruhpfi7n4hv
        region: 'us-west-2' // e.g. us-east-2
    },
    api: {
        invokeUrl: 'https://557fmlahoj.execute-api.us-east-1.amazonaws.com/prod' // e.g. https://rc7nyt4tql.exec
    }
};
```

And then done git push the changes to CodeCommit


```
Mr.user@dell MINGW64 ~/Desktop/GIT_Bash/Capstone_project_3/wildrydes-site (master)
$ git add .

Mr.user@dell MINGW64 ~/Desktop/GIT_Bash/Capstone_project_3/wildrydes-site (master)
$ git commit -m "updated config.js"
[master 6a6d3e6] updated config.js
 1 file changed, 1 insertion(+), 1 deletion(-)

Mr.user@dell MINGW64 ~/Desktop/GIT_Bash/Capstone_project_3/wildrydes-site (master)
$ git push
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 4 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 404 bytes | 202.00 KiB/s, done.
Total 4 (delta 3), reused 0 (delta 0), pack-reused 0
To https://git-codecommit.us-east-1.amazonaws.com/v1/repos/wildrydes-site
   ed731eb..6a6d3e6  master -> master
```

Step 6 – Validate the implementation

Visited /ride.html under website domain.
After the map has loaded, clicked anywhere on the map to set a pickup location.
Chosen Request Unicorn. A notification will be seen in the right sidebar that a unicorn is on its way and then see a unicorn icon fly to the pickup location.



Welcome! Click the map to set your pickup location.

You are authenticated. Click to see your auth token.

Rocinante, your Yellow unicorn, is on her way.

Rocinante has arrived. Giddy up!

Cost Analysis

Service		Upfront	Monthly	First 12 months total	currency	Configuration summary
Amazon cognito		0	50.75	609	$	Advanced security features (Enabled),Number of monthly active user (MAU)(1000)
AWS Lambda		0	0	0	$	
DynamoDB Provisioned capacity		180	26.39	496.68	$	Average item size(All attributes)(1KB),Write reserved capacity Term(1year),Read reserved capacity term(1year),Data storage size (1GB)
Amazon API Gateway	HTTP API	0	100	5400	$	HTTP API request units(million), Average size of each request(34KB), REST API request units(millions), Cache memory size(GB) None, Websocket message units (thousands), Average message size (32KB),Requests(100 per month)
	REST API		350			
Total			**527.14**	**5896.68**		

Lessons & Observations

Learnt the steps to create a server less web app application.

Learnt to host this web application on a front-end web server and connect it to a backend database.

Learnt to set up user authentication and will be able to collect and analyze user behavior.