BUILDING A RIDE SHARING APP USING AWS

AWS SERVICES TO BE USED

1. Codecommit
2. Amplify
3. Cognito
4. Lambda
5. IAM
6. API Gateway
7. DynamoDB

What we need

* All the code is provided by AWS and we need a way to get it into a source control system
* We need to be able to handle permissions for the code
* A place to host the website and also make updates
* A way for users to register and login
* We need some ride sharing functionality
* Somewhere to store and return the ride results
* We need a way to invoke the ride sharing functionality from the browser

The top 2 are basically getting our code set up and were going to use AWS’s source control system which is CodeCommit(kind of like github for aws). This is where we will store html,javascript, and css files etc for the site

You have to stay consistent with the region that you are using

A screenshot of a computer

Description automatically generated

For getting the html,css, and javascript, we will not be typing anything from scratch as aws has provided the codes for us. So we will be taking that code and creating a codecommit repo and copy the code into our repo where we can work with it. To do that we need to follow the following steps;

1. Create an empty repository in code commit

A screenshot of a computer

Description automatically generated

1. Now we need to work on permissions via IAM so we can access codecommit. Add a policy toyour IAM user so you can access codecommit…always make sure you are not using your root account as that is not inline with the aws best practices. Its best to create an iam user and attach admin permissions.

A screenshot of a computer

Description automatically generated

You go to the iam user

And then you navigate to the permissions tab and click on ad permissions

A screenshot of a computer

Description automatically generated

Then you attach policies directly

A screenshot of a computer

Description automatically generated

You then search for codecommitpoweruser

A screenshot of a computer

Description automatically generated

Add permissions

A screenshot of a computer

Description automatically generated

This will give the codecommit access that we need

1. Create git credentials for the IAM User to allow HTTPS connections to codecommit. Codecommit uses git technology and hencethere is an additional set of credentials we need to get everything working

Go to security credentials

A screenshot of a computer

Description automatically generated

Scroll down to the https git credentials for aws code commit

A screenshot of a computer

Description automatically generatedclick on generate credentials

The generated credentials will allowus make https calls to codecommit. Download the credentials cos we will use them later

1. Clone the repository(create an empty folder for future code)

We’ll go to the repository that we created

A screenshot of a computer

Description automatically generated

We will then get the url that will be used to clone the repo . we will go with the https version

A screenshot of a computer

Description automatically generatedwhen you click on ‘clone https’ it copies it and then we need to open a cloud shell( a command line interface that runs in the browser) at the top of the browser

So first you open the cloud shell with islocated to the right of the search bar.

You type in git clone and then paste the https url of the repository and hit enter

You then type in the username you generated for codecommit as well as the password

A screenshot of a computer

Description automatically generated

However it takes us to an directory in cloud shell that hosts the repository so we will have to change it to our created repository by using ‘cd’

Cloning the repository created an empty folder and this is where our code files will end up

A black screen with white text

Description automatically generated

1. Copy the project code from the s3 bucket and commit it to the new repo

So were going to download the files from an s3 bucket. We use ls to check the files and see that they are in the folder but not in the cloudcommit repo. To push them into the cloud commit repo we type ‘git add.’ And then ‘git commit’.

Its then going to ask for your email and username. Take note that the username required is the iam username and not the generated one.

You then type ‘git push’

You will be asked for another username but this time you put in the generated code commit username

After pushing the code to the repository

You go back to the wildrydes repository and you seethe files loaded in there

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

The files were in an s3 bucket and were retrieved into our repo where we can work with .

1. We are going to use amplify to create a website

You open aws amplify, click on host web app. Go to code commit and hit continue and select the wildrydes repository from the list of options and hit next

Click on allow aws amplify to automatically deploy files hosted in yout project root firectory. This means when theres an update or change in the code file it automatically deploys. After doing that you hit next.

This will take you to the review page where you just save and deploy

This is a serverless service where we did not need to provide and maintain an ec2 instance

You can click the link to check out the site in a new tab

We are going to try and see if the continuous deployment is working. We do that by going to codecommit and navigating to the index.html file, clicking edit. You make whatever change you want and then fill in in the tabs which specify your name and email and then click on commit changes.

It then triggers amplify which starts the build process. When its done, you go back to the site and hit refresh and see that the changes hav been applied

1. Now we need to work on a way to login and register as those fuctionalities haven’t been worked on. We will be using AWS Cognito(used to do authentication)

So we go over to aws cognito and then click on create user pool

You click on user name from the sign in options tab and go to next

At the security requirements select cognito defaults for the passwork policy and no mfa to keep things simple and then go to next

Well leave all the deafaults on the next page and click on next

On the next page which is the message delivery page we will click on send email with cognito,leave all the defaults and then hit next

On the next page which is the integrate your app page we will give a name to our user pool and then give an app client name as wildrydeswebapp. We will leave the other defaults. Then hit next

Review and create the user pool.the user pool has been created

Now we will click into the user pool. We will need to copy somethings-user pool ID.

We then click on app integration and copy the client ID

Now we need to hook up cognito with our code by updating the config file in our application code to point to the dedicated user pool

So we go back to codecommit and open our repo and then navigate to the js file and then the config.js file

Click on edit . you paste the user pool ID,user pool client ID, and the region and then commit the changes which will trigger amplify again

Now you go over to the wild rydes site and click on giddyup to registerfill in the details and cognito will send an email with the verification code.once you put in the details you will be routed to the login page where you wil need to login again

1. Now we need to work on the ride sharing functionality

We will use a lambda function and a dynamodb database. The user will request a ride that will invoke the lambda fuction and the request will be stored I the dynamo table

We will set up a dynamodb table first

Go to dynamodb and navigate to tables where you create a new table

You name the table Rides and give the partition key as RideId and eave everything as default and hit create table

Click into the created table, go to general information and additional info and copy the aws arn

We will need to create a role for our lambda function to be able to write into our table so we would go to IAM

Click on create new role

The trusted entity type will be aws service and the service will be lambda

Now we look for a policy to attach to the role

We select the aws basic lambda execution role and hit next

We give the name of the role as wildrydeslambda and leave the defaults of everything else and create role

Now we will open up the role and add some additional permissions. Go to add permissions and hit create inline policy

You chose the service as dynamodb and then you select putitem and then add the arn

You give the policy a name-dynamodbwriteaccess and create policy

Now you go to aws lambda and create a new lambda function by clicking create function

We will author from scratch. The name of the function will be requestunicorn

For the runtime we use Node.js 16x

For the execution role we will select ‘use an existing role’ which is the role we created in IAM and then hit create function

We then scroll down tot the code source and replace the original lambda code with the code provided by aws

Make sure to deploy the changes

Now we have to configure a test event

Copy our test function and paste it in the json file and hit save

Now we need to run the test and see that rh test succeeded

We then check the dynamodb table to see if everything was recorded properly

In the rides table you click on explore table items

When we scroll down we see the one item for the test

1. We need to find a way to invoke the ride sharing functionality

We will use API Gateway for this

We go to api gateway and then hit create new api

Select rest api

Give the api a name ‘wildrydes’

Select edge optimized and then create api

Now we need to create an authorizer to authenticate calls api gateway uses that are returned by cognito

After creating the authorizer, click into it to test if things are working

Paste in the authorization token from the wildrydes page and hit test

We get a feedback of 200 meaning everything is working fine

Now we go into our api and then create resource which is how we will hook it up to the lambda function.

We make sure to select cors when creating the resource and then hit create resource

After creating the resource , we need to create a method. We click create method from inside the ride resource

The method type will be a post type to integrate with the lambda function, select lambda as the integration type and then select the unicorns lambda function and then create method

After creating the post method, select the method request card and then click on edit

We will authorize using the cognito user pool authorization

After its all said and done, we depLoy the API

A pop up tab will show where you select stage s new stage and name it ‘dev’ and deploy

Copy the invoke url and save it somewhere

Now we go back to codecommit inside the config.js file and update the involve url with the one we cpied from our api and commit the change

now to check if everything is working properly we go to our rides site and refresh

after requesting your unicorn you can go check the dynamodb table to see if it was recorded.everything is working as it should