ASSIGNMENT 07:

Name: Atharv Satish Nikam

Task 1:

1. **EC2:**
   a. Purpose:
   AWS's core and flexible solution, Amazon Elastic Compute Cloud (EC2), offers cloud-based virtual servers via server virtualization. Windows, Linux, and Mac OS are just a few of the operating systems that users may deploy EC2 instances with. Different types of EC2 instances are available, enabling customers to select configurations with CPU, memory, storage, and networking capabilities that meet their individual requirements.
   b. Key features:
   Launching instances in Virtual Private Clouds (VPCs) and allocating subnets for network access control are two of EC2's primary functionalities. Three different types of IP addresses are linked to instances: Elastic IP for static public addresses, Private IP for internal communication, and Public IP for internet access. Users have the flexibility to choose instance types based on their use cases, with AWS managing the underlying physical hardware and virtualization layer.
   c. Benefits:
   EC2 is categorized as Infrastructure-as-a-Service (IaaS), where AWS takes care of physical hardware and virtualization, while users manage the operating system and installed applications. Instances in public subnets are accessible from the internet, while private subnets leverage NAT gateways for outbound internet connectivity. EC2 provides flexibility, scalability, and the ability to tailor resources to specific requirements, making it a crucial component for businesses seeking efficient and customizable cloud computing solutions.
2. **S3:**
   a. Purpose:
   AWS's primary and extremely scalable storage solution, Amazon Simple Storage solution (S3), allows users to store and retrieve any volume of data from any location on the internet. Its main goal is to offer a reliable, adaptable, and safe storage option for a range of uses and applications.
   b. Key features:
      i. Because S3 is scalable, customers can store almost infinite quantities of data. It ensures great availability and durability by automatically replicating data across several sites.
      ii. Users can effectively manage and organize their data with S3. Multiple versions of an item can be kept as it supports versioning. This is a useful feature for auditing, recovery, and data security.
      iii. S3 provides several storage classes in order to optimize expenses according to patterns of data access. Among these classes are Glacier for

        long-term archival, Intelligent-Tiering for automated cost reductions, Standard for regularly accessible data, and others.

   c. Benefits:
      i. S3 ensures high durability and availability of stored data, making it a reliable choice for critical business applications.
      ii. With various storage classes and pricing options, S3 allows users to optimize costs based on their specific requirements, promoting cost-effectiveness.
      iii. S3 is designed for simplicity, offering a straightforward interface and easy integration with other AWS services and third-party tools.

3. **RDS:**
   a. Purpose:
   AWS's managed relational database solution, Amazon Relational Database solution (RDS), makes it easier to scale, manage, and deploy relational databases. Many database engines, such as Amazon Aurora, MySQL, MariaDB, Oracle, Microsoft SQL Server, and PostgreSQL, are supported by RDS.
   b. Key features:
      i. By handling standard database maintenance responsibilities like backups, patches, and upgrades, RDS frees customers to concentrate on developing applications rather than handling administrative work.
      ii. Because RDS supports numerous database engines, it can accommodate a broad range of application requirements and preferences, hence offering flexibility.
      iii. By upgrading to bigger instance types, users may vertically expand their database instances, improving performance to handle higher workloads, reads, and writes.
   c. Benefits:
      i. RDS simplifies database administration tasks, automating routine maintenance and allowing developers to focus on building applications.
      ii. RDS supports both vertical and horizontal scaling, providing flexibility to adapt to changing workloads and ensuring optimal database performance.
      iii. RDS improves database availability and reliability with features like Multi-AZ deployments, reducing downtime and providing automated failover in the event of infrastructure problems.

4. **CloudFormation:**
   a. Purpose:
   Within the Infrastructure as Code (IaC) space, Amazon CloudFormation is a powerful tool offered by AWS that enables customers to automate infrastructure deployment and maintenance using code. CloudFormation allows users to design and configure AWS resources, including subnets and Virtual Private Clouds (VPCs), using a template file written in either YAML or JSON.
   b. Key features:
      i. The AWS architecture that is desired may be defined through code using CloudFormation templates. These templates allow users to define resources and associated settings.

    ii. A stack in CloudFormation represents the entire environment described by a template. CloudFormation handles the creation, updating, and deletion of stacks, ensuring consistency and reproducibility.
    iii. Change sets are provided by CloudFormation, which gives an overview of suggested changes to the stack. Before implementing modifications, users may evaluate them, guaranteeing control and the chance to make changes.

c. Benefits:
    i. By automating infrastructure deployment and administration, it lowers the need for manual intervention and guarantees consistency between environments.
    ii. The users can efficiently deploy and manage complex infrastructure setups, saving time and resources compared to manual processes.
    iii. Version control systems provide the ability to store CloudFormation templates, which facilitates team collaboration and versioning while improving auditability and traceability.

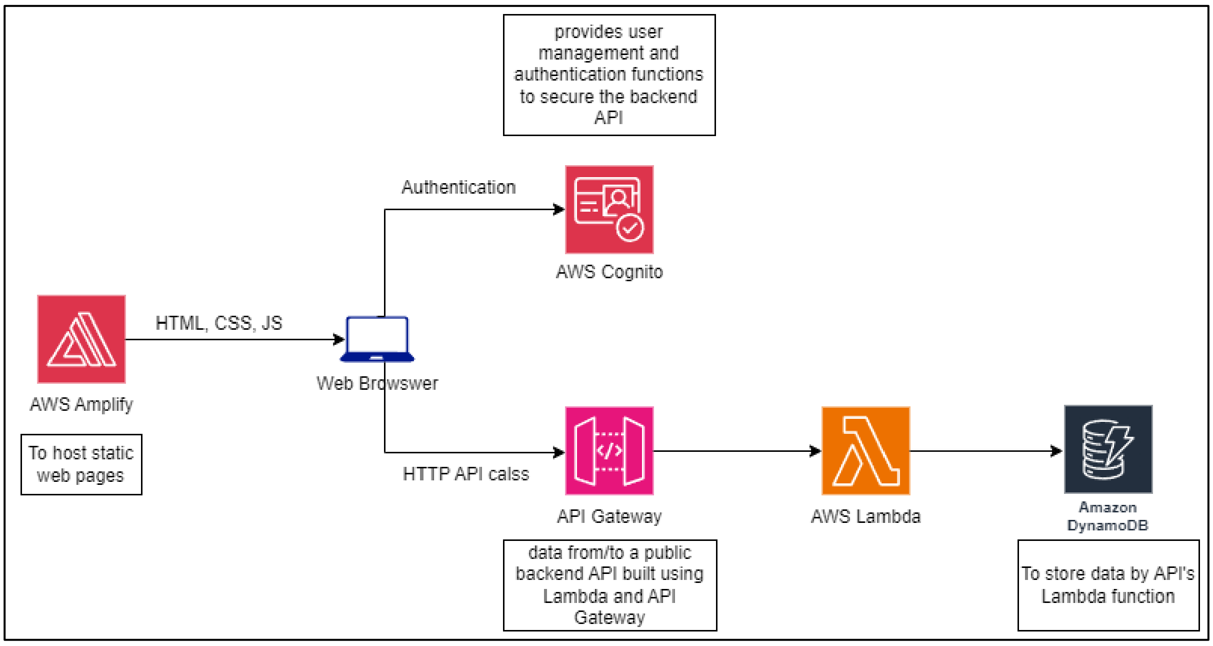
Task 2:

- **Scenario:** Local Services Marketplace Web Application
- **Project case:**
  Consider creating a web application for a marketplace for local services where people can provide and request a range of services from gardening to plumbing to electrical repair. The platform's goal is to easily match customers with knowledgeable service providers so that meeting local service demands is simple.
- **AWS Services used:**
  - **Compute Resources (Scalability): Amazon EC2 Auto Scaling-**
    Install the web application on Amazon EC2 instances that are set up in an Auto Scaling group to manage heavy demand. With auto scaling, the program is guaranteed to scale dynamically according to demand, adding or deleting instances as required. This offers scalability to effectively manage different user activity levels.
  - **Relational Database (User Data Storage): Amazon RDS-**
    Utilizing Amazon RDS to store user data, service requests, and other relational data. The relational database solution offered by the RDS database engine (e.g., MySQL, PostgreSQL) is controlled and scalable. This guarantees that user data is safely saved and that the program may access it with ease.
  - **File Storage: Amazon S3 (Simple Storage Service)-**
    Images, documents, and any other attachments pertaining to service requests are stored and retrieved using Amazon S3. It provides highly durable and available object storage that is expandable. Multimedia content related to service requests may be handled effectively with the help of integration with Amazon S3.
  - **Infrastructure as Code (IaaC): AWS CloudFormation-**
    We utilize AWS CloudFormation or AWS CDK to manage and provision the complete infrastructure as code. With this method, we can create and launch the application architecture in a version-controlled and repeatable way, utilizing

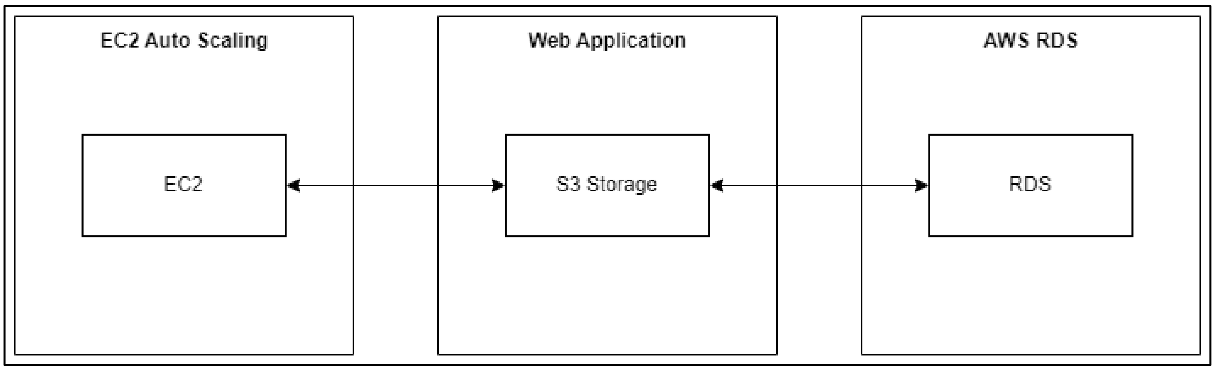
EC2 instances, RDS databases, and S3 buckets. Any modifications to the infrastructure may be monitored and uniformly applied in various settings.

Task 3:

- Architecture:


provides user management and authentication functions to secure the backend API
Authentication
AWS Cognito
HTML, CSS, JS
AWS Amplify
Web Browswer
To host static web pages
HTTP API calss
API Gateway
AWS Lambda
Amazon DynamoDB
data from/to a public backend API built using Lambda and API Gateway
To store data by API's Lambda function

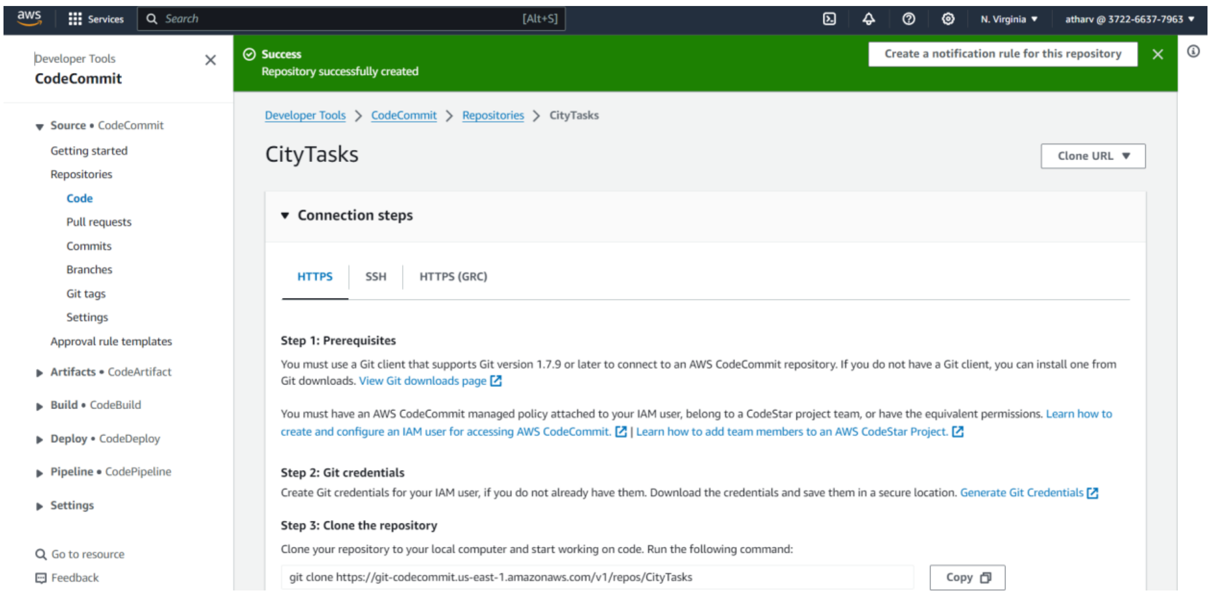
- Components:
  - Compute Resources (EC2 Auto Scaling):
    - EC2 instances to host the web application.
    - Configure Auto Scaling groups to automatically adjust the number of instances based on traffic.
  - Relational Database (Amazon RDS):
    - RDS with a suitable database engine (e.g., MySQL or PostgreSQL) for storing user data, service requests, and relational information.
    - Multi-AZ deployment for high availability and reliability.
  - File Storage (Amazon S3):
    - Amazon S3 to store and retrieve files related to service requests (e.g., images, documents).
    - Versioning and access control to manage file storage efficiently.
  - Infrastructure as Code (AWS CloudFormation or AWS CDK):
    - CloudFormation to define and provision the infrastructure.
    - Templates to create and configure EC2 instances, RDS databases, S3 buckets, and other resources.



<u>Task 4 to Task 9:</u>

- Reference: https://aws.amazon.com/getting-started/hands-on/build-serverless-web-app-lambda-apigateway-s3-dynamodb-cognito/
- As inexperienced in web app development, I have heavily referred the HTML/CSS scripts from the above linked tutorial.
- For easiness, and practice, initially I deployed the "UnicornRide" app mentioned in the tutorial. After that, I deployed my own web pages for my own "CityTasker" web app by following the same procedures.
- During the process, sometimes, I have reused previously deployed services [table/lambda] in my own application. Thus, the names of tables/lambda functions may seem unrelevant to the CityTasker application [for. E.g.- CabTransport (created for unicorn app)]. However, I have done it mainly to save time, and to reduce the costs during the AWS usage.

Step: Creating repository and populating it with the web pages code



Success
Repository successfully created

CityTasks

Connection steps

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 AWS CodeStar Project.

Step 2: Git credentials

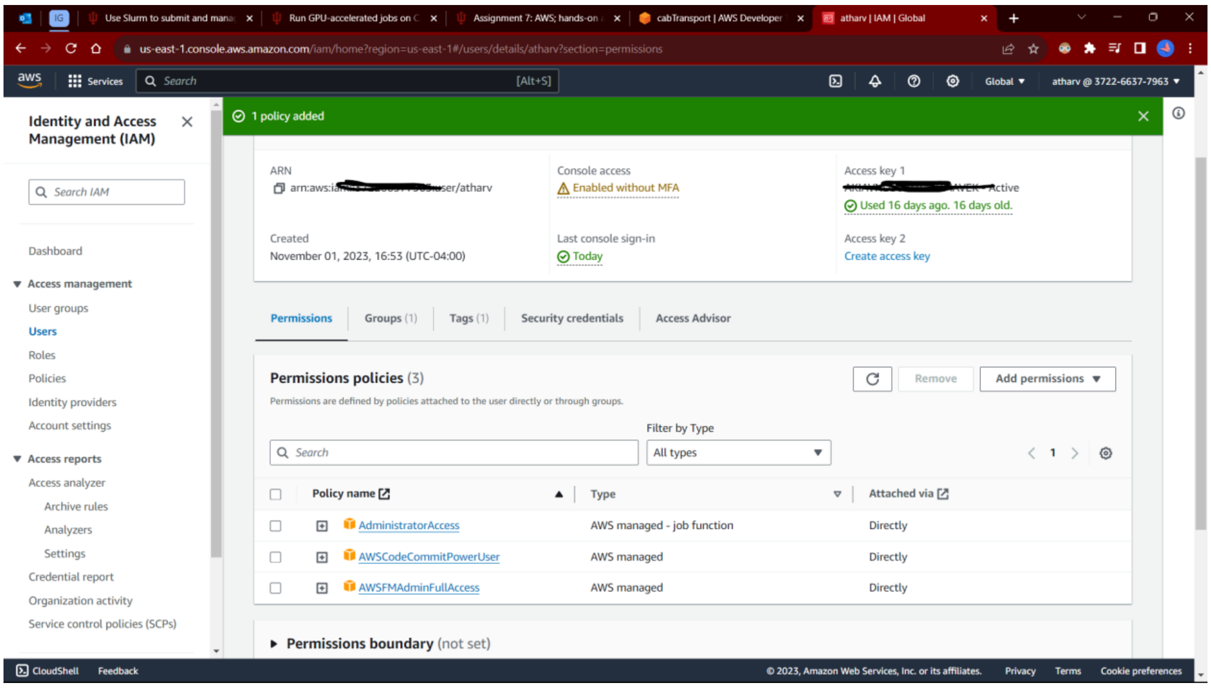
Create Git credentials for your IAM user, if you do not already have them. Download the credentials and save them in a secure location. Generate Git Credentials

Step 3: Clone the repository

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/CityTasks
```

Step: managing the access rights using IAM



us-east-1.console.aws.amazon.com/iam/home?region=us-east-1#/users/details/atharv?section=permissions

Identity and Access Management (IAM)

1 policy added

ARN
arn:aws:[redacted]ser/atharv

Console access
Enabled without MFA

Access key 1
[redacted] Active
Used 16 days ago. 16 days old.

Created
November 01, 2023, 16:53 (UTC-04:00)

Last console sign-in
Today

Access key 2
Create access key

Permissions | Groups (1) | Tags (1) | Security credentials | Access Advisor

Permissions policies (3)

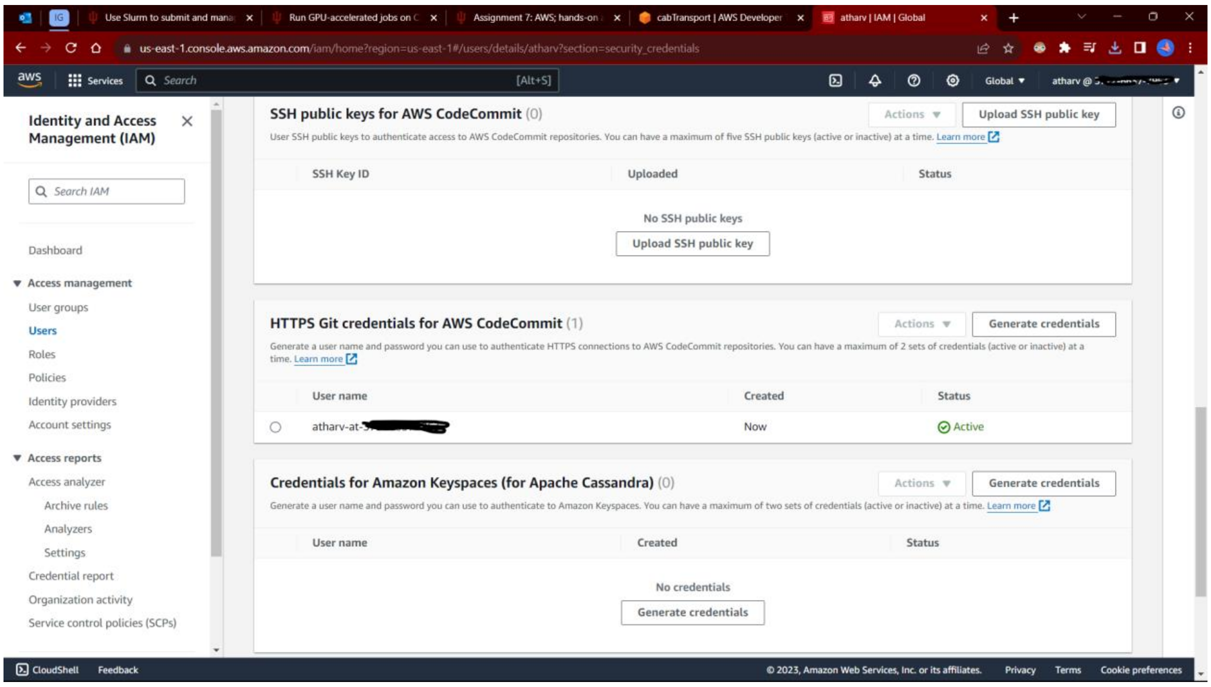
Permissions are defined by policies attached to the user directly or through groups.

Filter by Type: All types

Policy name	Type	Attached via
AdministratorAccess	AWS managed - job function	Directly
AWSCodeCommitPowerUser	AWS managed	Directly
AWSFMAdminFullAccess	AWS managed	Directly

Permissions boundary (not set)

© 2023, Amazon Web Services, Inc. or its affiliates.



us-east-1.console.aws.amazon.com/iam/home?region=us-east-1#/users/details/atharv?section=security_credentials

SSH public keys for AWS CodeCommit (0)

User SSH public keys to authenticate access to AWS CodeCommit repositories. You can have a maximum of five SSH public keys (active or inactive) at a time. Learn more

SSH Key ID	Uploaded	Status

No SSH public keys

HTTPS Git credentials for AWS CodeCommit (1)

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

User name	Created	Status
atharv-at-[redacted]	Now	Active

Credentials for Amazon Keyspaces (for Apache Cassandra) (0)

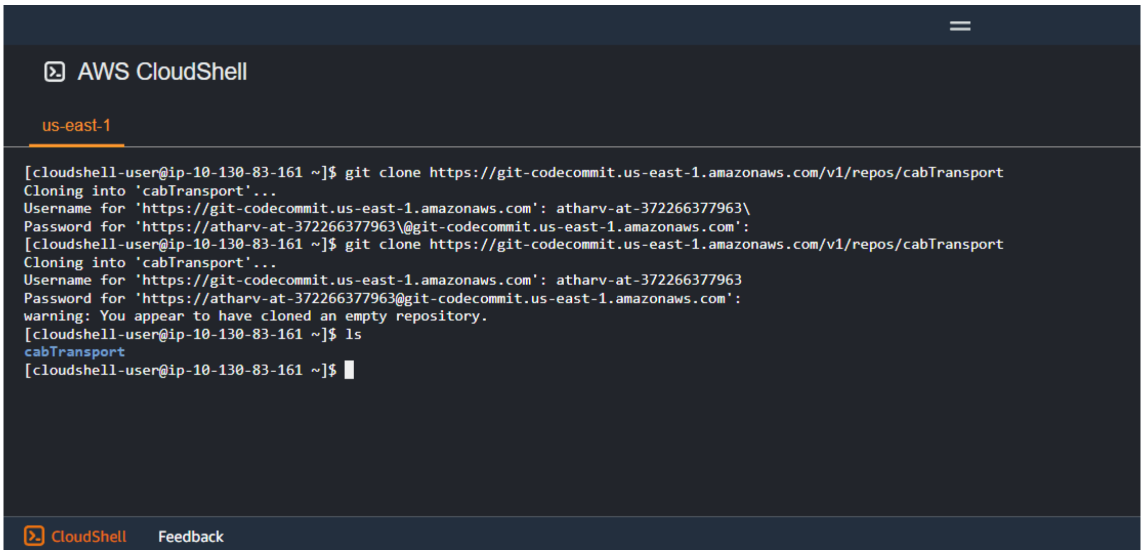
Generate a user name and password you can use to authenticate to Amazon Keyspaces. You can have a maximum of two sets of credentials (active or inactive) at a time. Learn more

User name	Created	Status

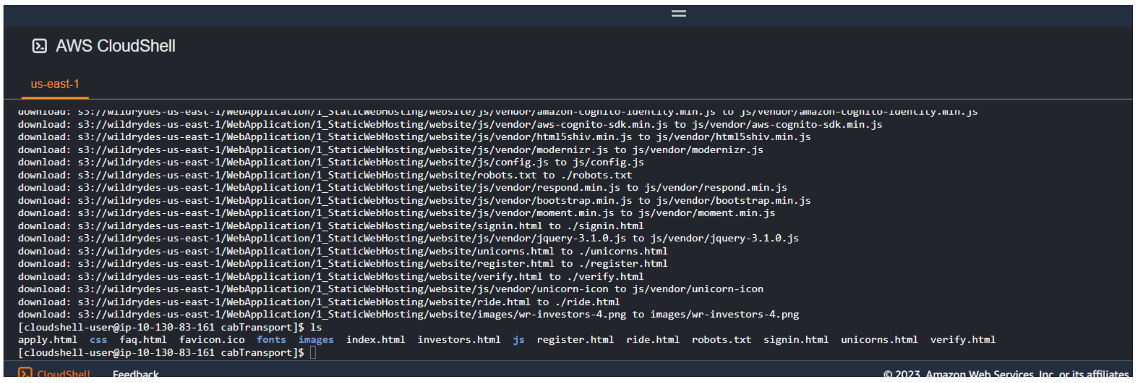
No credentials

© 2023, Amazon Web Services, Inc. or its affiliates.

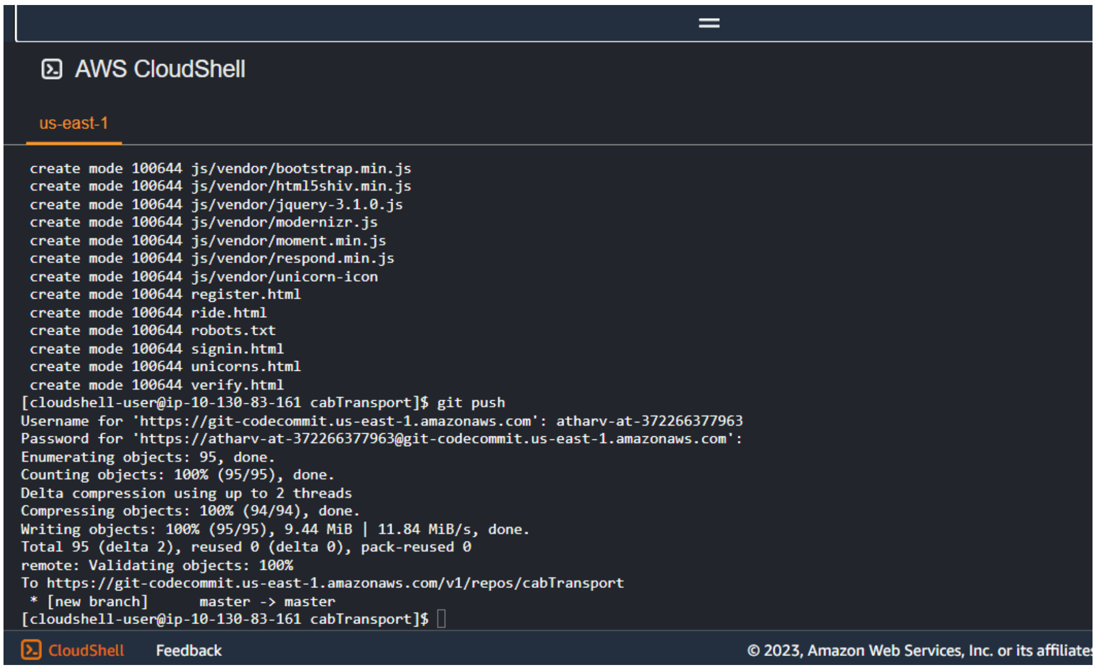
Step: Using CLI to configure and populate repository



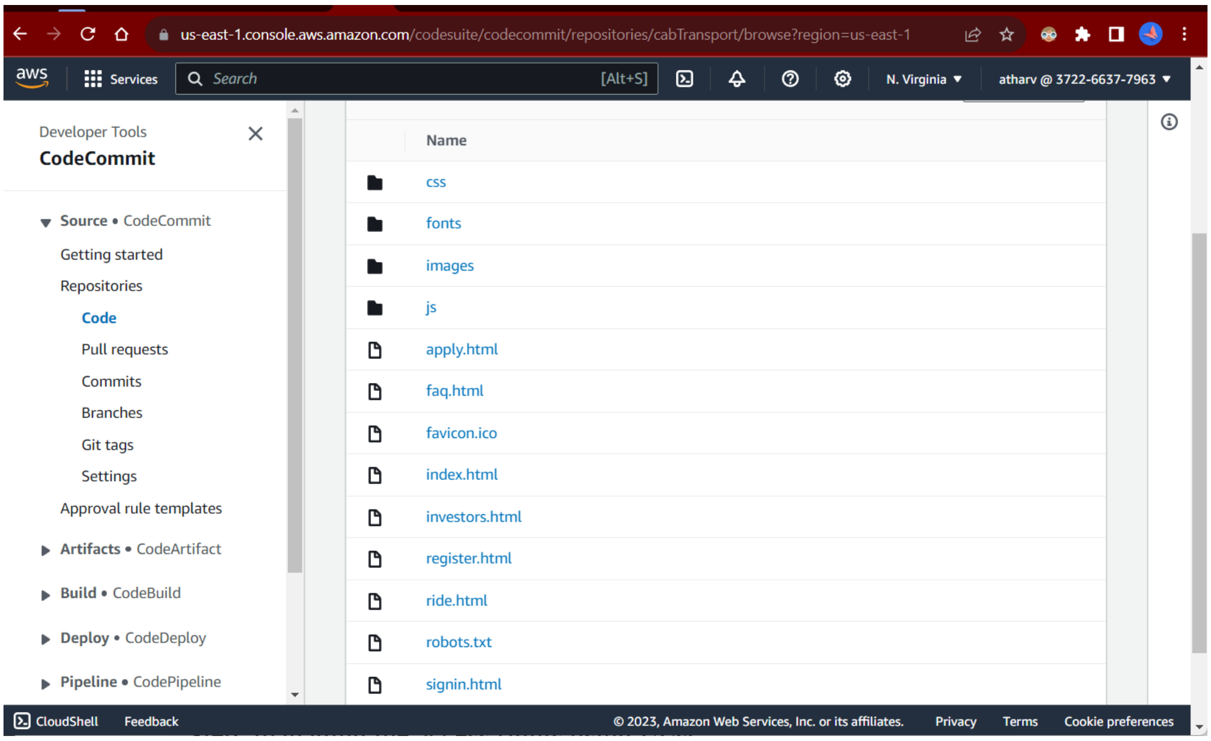
```
[cloudshell-user@ip-10-130-83-161 ~]$ git clone https://git-codecommit.us-east-1.amazonaws.com/v1/repos/cabTransport
Cloning into 'cabTransport'...
Username for 'https://git-codecommit.us-east-1.amazonaws.com': atharv-at-372266377963\
Password for 'https://atharv-at-372266377963\@git-codecommit.us-east-1.amazonaws.com':
[cloudshell-user@ip-10-130-83-161 ~]$ git clone https://git-codecommit.us-east-1.amazonaws.com/v1/repos/cabTransport
Cloning into 'cabTransport'...
Username for 'https://git-codecommit.us-east-1.amazonaws.com': atharv-at-372266377963
Password for 'https://atharv-at-372266377963@git-codecommit.us-east-1.amazonaws.com':
warning: You appear to have cloned an empty repository.
[cloudshell-user@ip-10-130-83-161 ~]$ ls
cabTransport
[cloudshell-user@ip-10-130-83-161 ~]$
```



```
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/html5shiv.min.js to js/vendor/html5shiv.min.js
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/config.js to js/config.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/js/vendor/respond.min.js to js/vendor/respond.min.js
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/js/vendor/moment.min.js to js/vendor/moment.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/jquery-3.1.0.js to js/vendor/jquery-3.1.0.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/register.html to ./register.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/unicorn-icon to js/vendor/unicorn-icon
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/images/wr-investors-4.png to images/wr-investors-4.png
[cloudshell-user@ip-10-130-83-161 cabTransport]$ ls
apply.html  css  faq.html  favicon.ico  fonts  images  index.html  investors.html  js  register.html  ride.html  robots.txt  signin.html  unicorns.html  verify.html
[cloudshell-user@ip-10-130-83-161 cabTransport]$
```

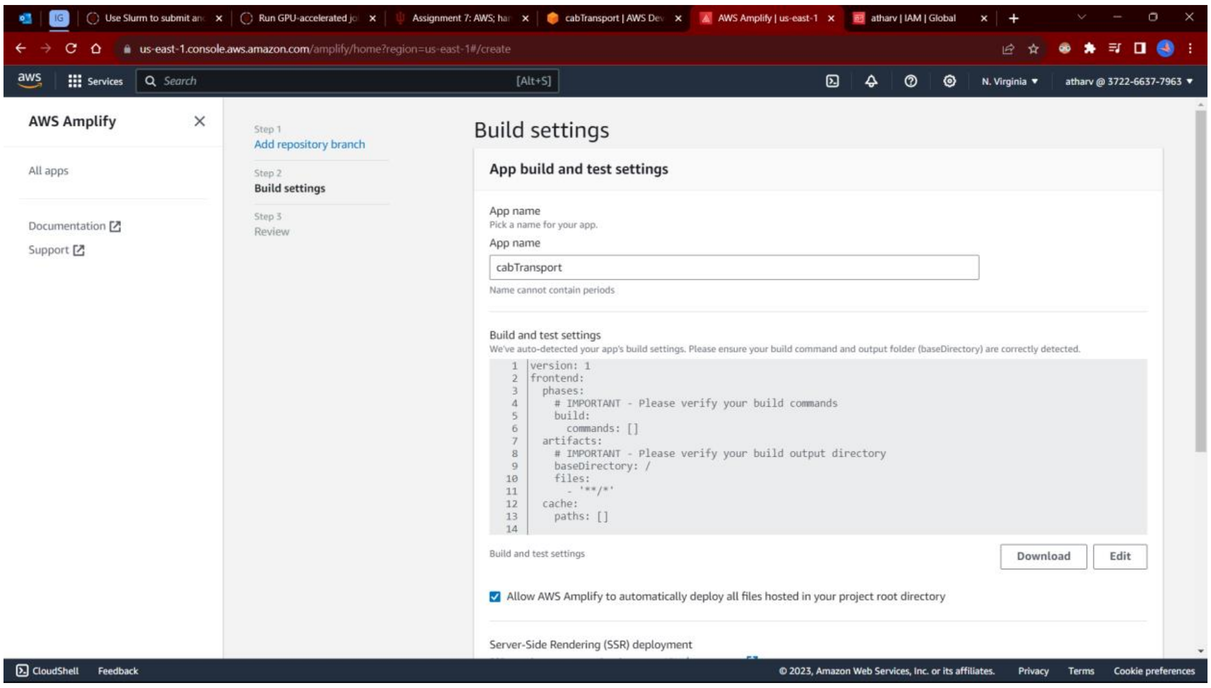


```
 create mode 100644 js/vendor/bootstrap.min.js
 create mode 100644 js/vendor/html5shiv.min.js
 create mode 100644 js/vendor/jquery-3.1.0.js
 create mode 100644 js/vendor/modernizr.js
 create mode 100644 js/vendor/moment.min.js
 create mode 100644 js/vendor/respond.min.js
 create mode 100644 js/vendor/unicorn-icon
 create mode 100644 register.html
 create mode 100644 ride.html
 create mode 100644 robots.txt
 create mode 100644 signin.html
 create mode 100644 unicorns.html
 create mode 100644 verify.html
[cloudshell-user@ip-10-130-83-161 cabTransport]$ git push
Username for 'https://git-codecommit.us-east-1.amazonaws.com': atharv-at-372266377963
Password for 'https://atharv-at-372266377963@git-codecommit.us-east-1.amazonaws.com':
Enumerating objects: 95, done.
Counting objects: 100% (95/95), done.
Delta compression using up to 2 threads
Compressing objects: 100% (94/94), done.
Writing objects: 100% (95/95), 9.44 MiB | 11.84 MiB/s, done.
Total 95 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Validating objects: 100%
To https://git-codecommit.us-east-1.amazonaws.com/v1/repos/cabTransport
 * [new branch]      master -> master
[cloudshell-user@ip-10-130-83-161 cabTransport]$
```


us-east-1.console.aws.amazon.com/codesuite/codecommit/repositories/cabTransport/browse?region=us-east-1

Name
css
fonts
images
js
apply.html
faq.html
favicon.ico
index.html
investors.html
register.html
ride.html
robots.txt
signin.html

Step: Amplify Hosting to Host


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

Build settings

App build and test settings

App name
Pick a name for your app.
App name
cabTransport
Name cannot contain periods

Build and test settings
We've auto-detected your app's build settings. Please ensure your build command and output folder (baseDirectory) are correctly detected.

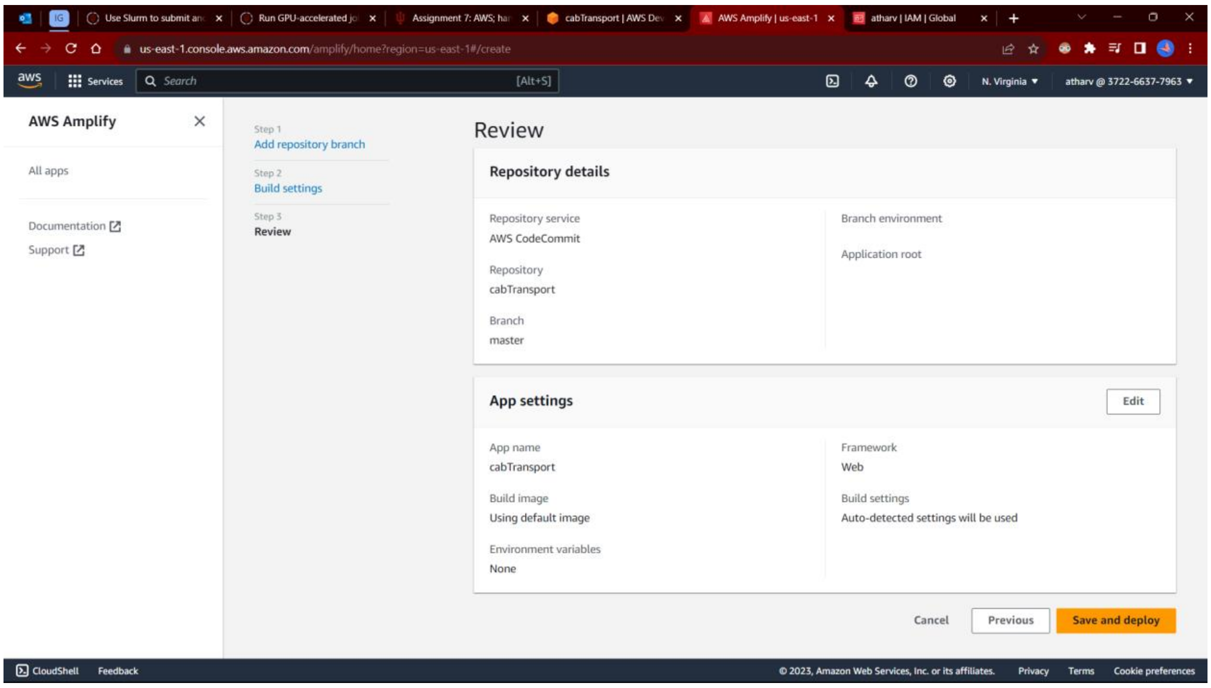
```
version: 1
frontend:
  phases:
    # IMPORTANT - Please verify your build commands
    build:
      commands: []
  artifacts:
    # IMPORTANT - Please verify your build output directory
    baseDirectory: /
    files:
      - '**/*'
  cache:
    paths: []

```

Build and test settings

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

Server-Side Rendering (SSR) deployment


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

AWS Amplify

Step 1 Add repository branch
Step 2 Build settings
Step 3 Review

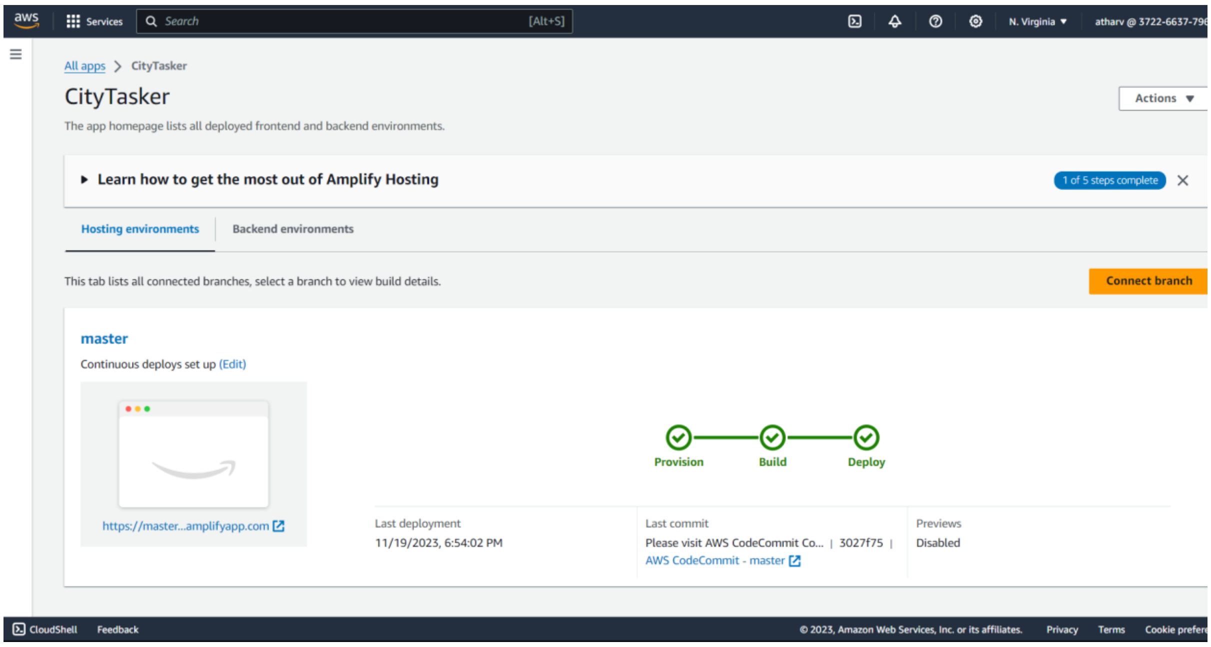
Review

Repository details

Repository service	Branch environment
AWS CodeCommit	
Repository	Application root
cabTransport	
Branch	
master	

App settings

App name	Framework
cabTransport	Web
Build image	Build settings
Using default image	Auto-detected settings will be used
Environment variables	
None	


All apps > CityTasker

CityTasker

The app homepage lists all deployed frontend and backend environments.

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

Hosting environments | Backend environments

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

master

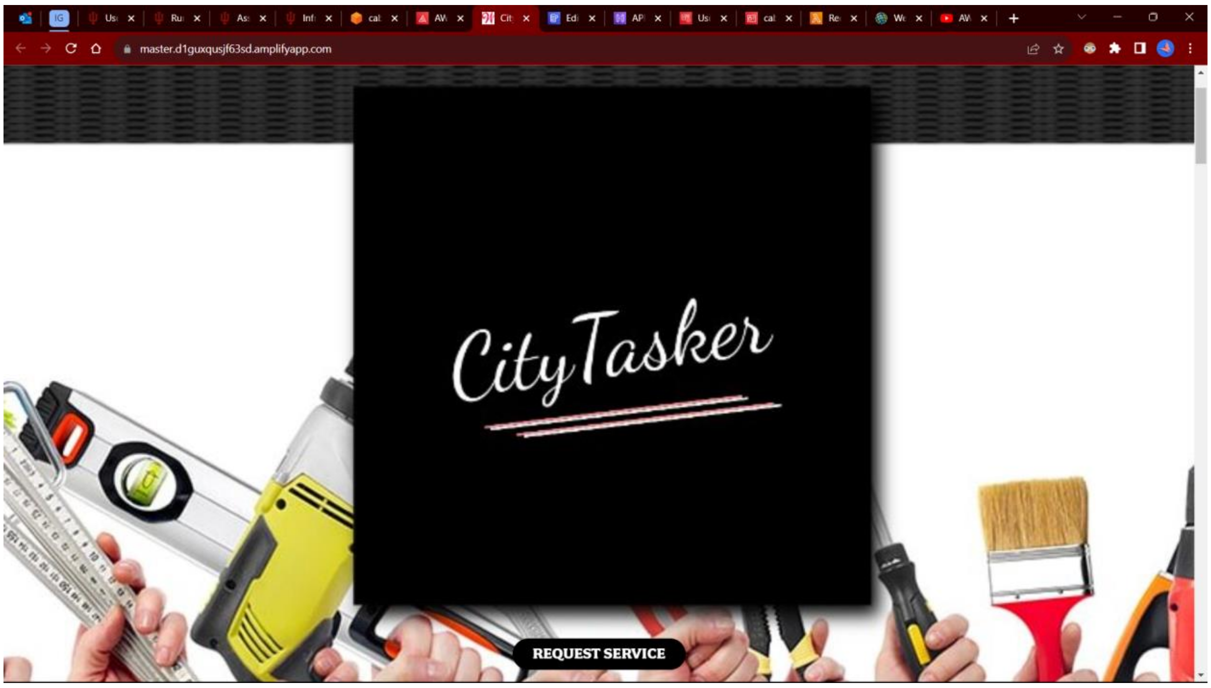
Continuous deploys set up (Edit)

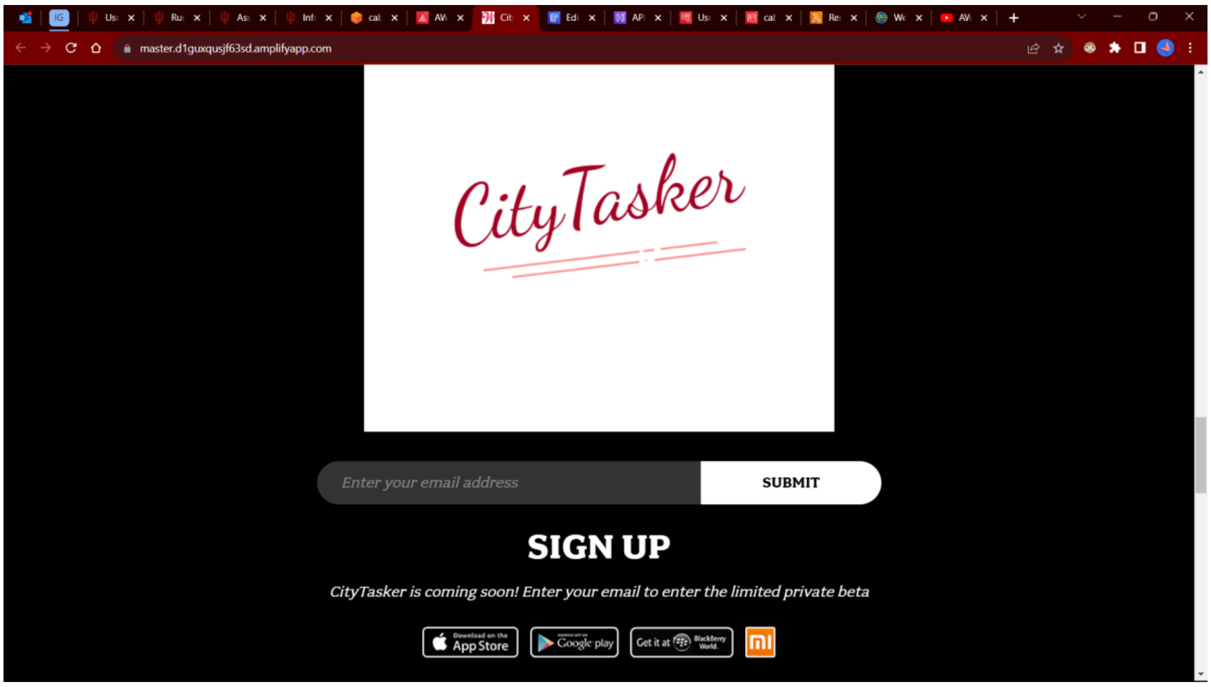
https://master...amplifyapp.com

Provision — Build — Deploy

Last deployment	Last commit	Previews
11/19/2023, 6:54:02 PM	Please visit AWS CodeCommit Co... | 3027f75 | AWS CodeCommit - master	Disabled

Step: Web App Running:

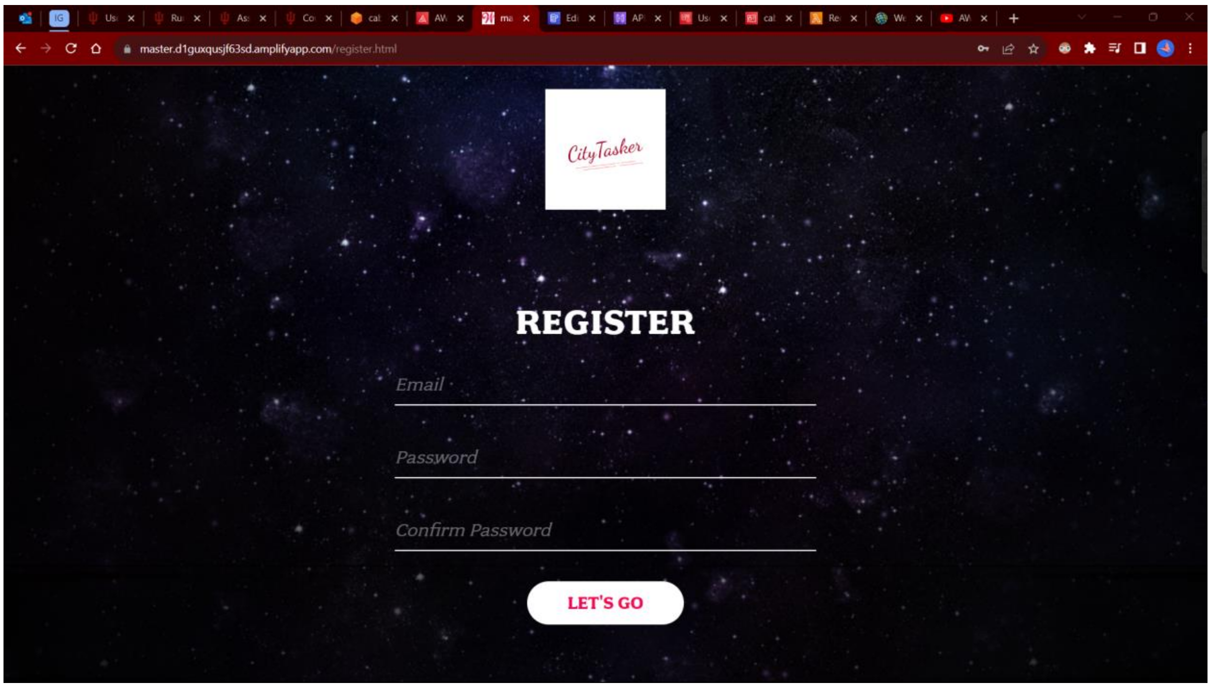



master.d1guxqusjf63sd.amplifyapp.com
CityTasker
Enter your email address
SUBMIT

SIGN UP

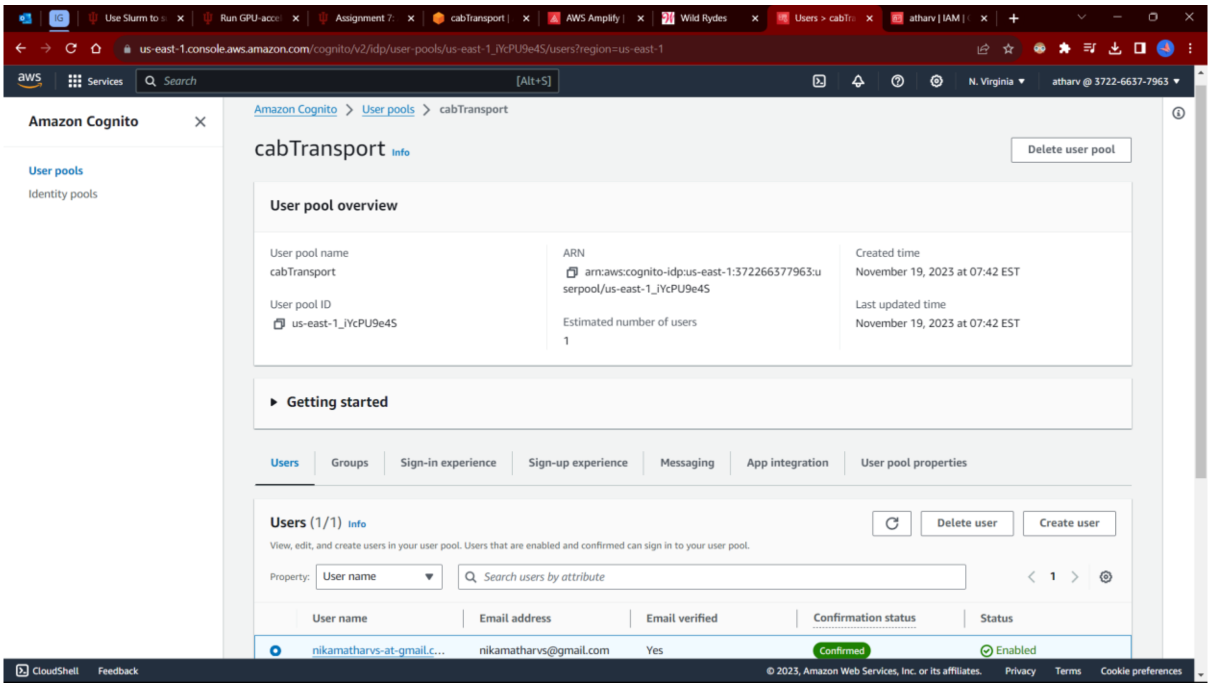
CityTasker is coming soon! Enter your email to enter the limited private beta

App Store
Google play
Get it at BlackBerry World



Step: Cognito and User Pool Set up for sign-in and register

[screenshots for unicorn app, used same for cityTasker app later]



cabTransport Info

User pool overview

User pool name
cabTransport

User pool ID
us-east-1_iYcPU9e4S

ARN
arn:aws:cognito-idp:us-east-1:372266377963:userpool/us-east-1_iYcPU9e4S

Estimated number of users
1

Created time
November 19, 2023 at 07:42 EST

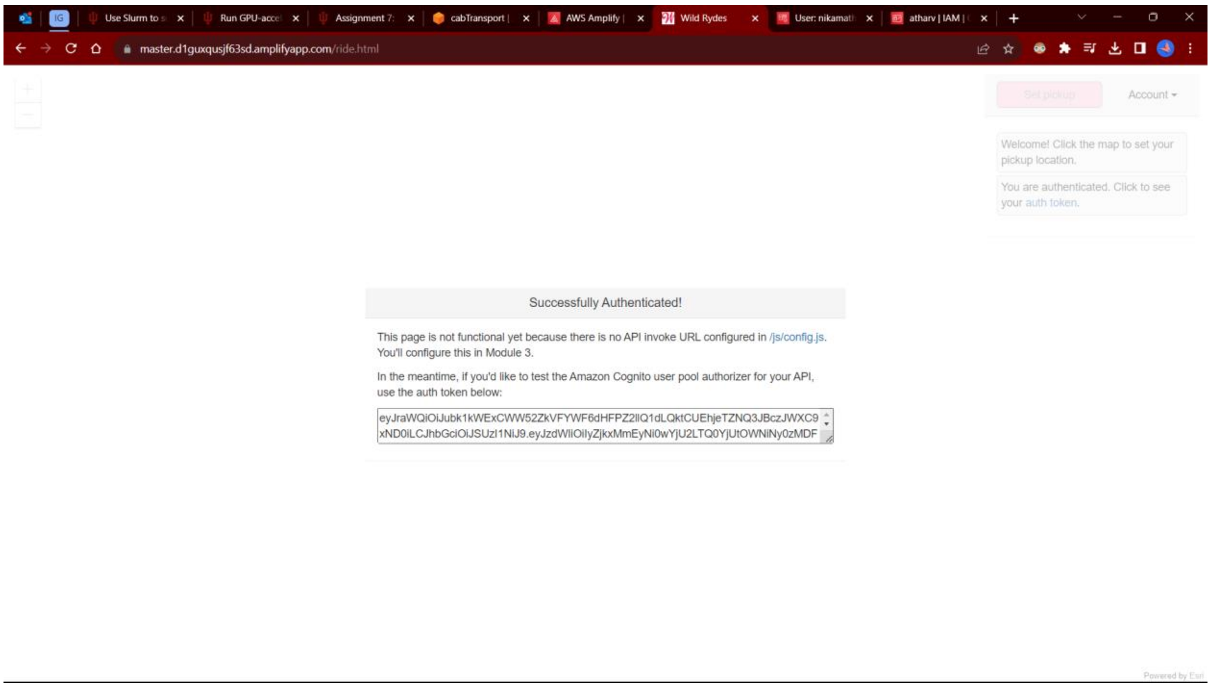
Last updated time
November 19, 2023 at 07:42 EST

Users (1/1) Info

View, edit, and create users in your user pool. Users that are enabled and confirmed can sign in to your user pool.

Property: User name

User name	Email address	Email verified	Confirmation status	Status
nikamatharvs-at-gmail.c...	nikamatharvs@gmail.com	Yes	Confirmed	Enabled



master.d1guxqusjf63sd.amplifyapp.com/ride.html

Welcome! Click the map to set your pickup location.

You are authenticated. Click to see your auth token.

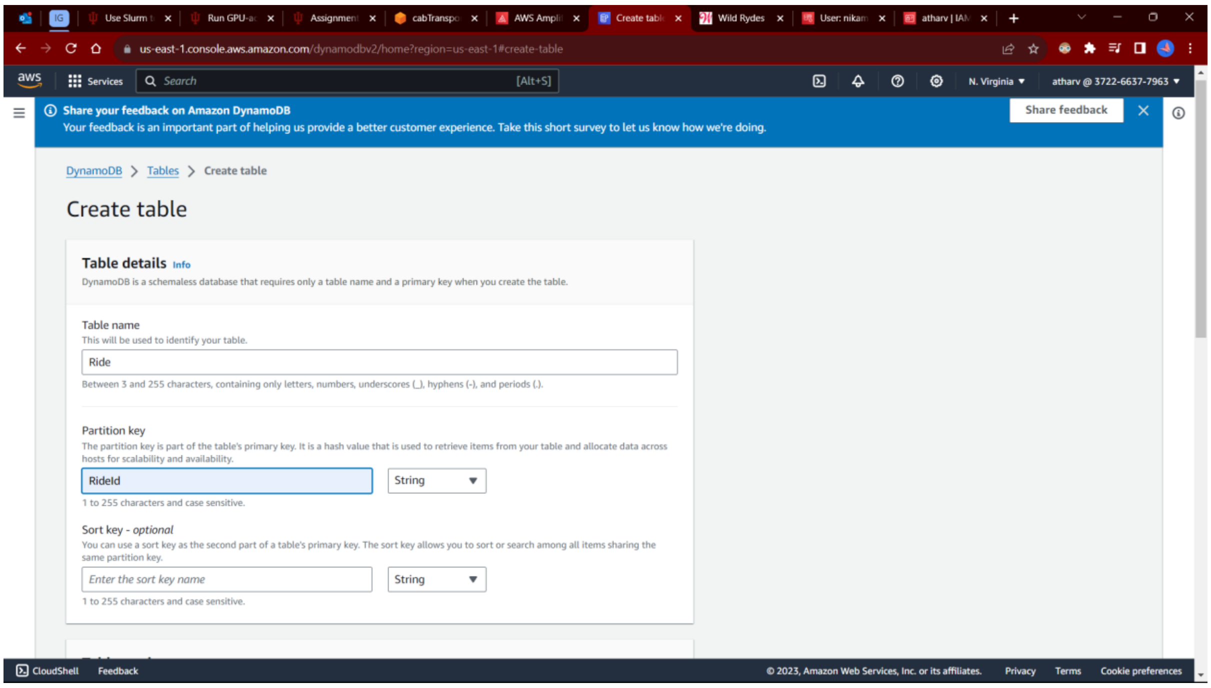
Successfully Authenticated!

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:

```
eyJraWQiOiJubk1kWExCWW52ZkVFYWF6dHFPZ2lIQ1dLQktCUEhjeTZNQ3JBczJWXC9
xND0iLCJhbGciOiJSUzI1NiJ9.eyJzdWIiOiIyZjkxMmEyNi0wYjU2LTQ0YjUtOWNiNy0zMDF
```

Step: Serverless Backend: DynamoDB Table creation-



us-east-1.console.aws.amazon.com/dynamodbv2/home?region=us-east-1#create-table

Share your feedback on Amazon DynamoDB
Your feedback is an important part of helping us provide a better customer experience. Take this short survey to let us know how we're doing.

DynamoDB > Tables > Create table

Create table

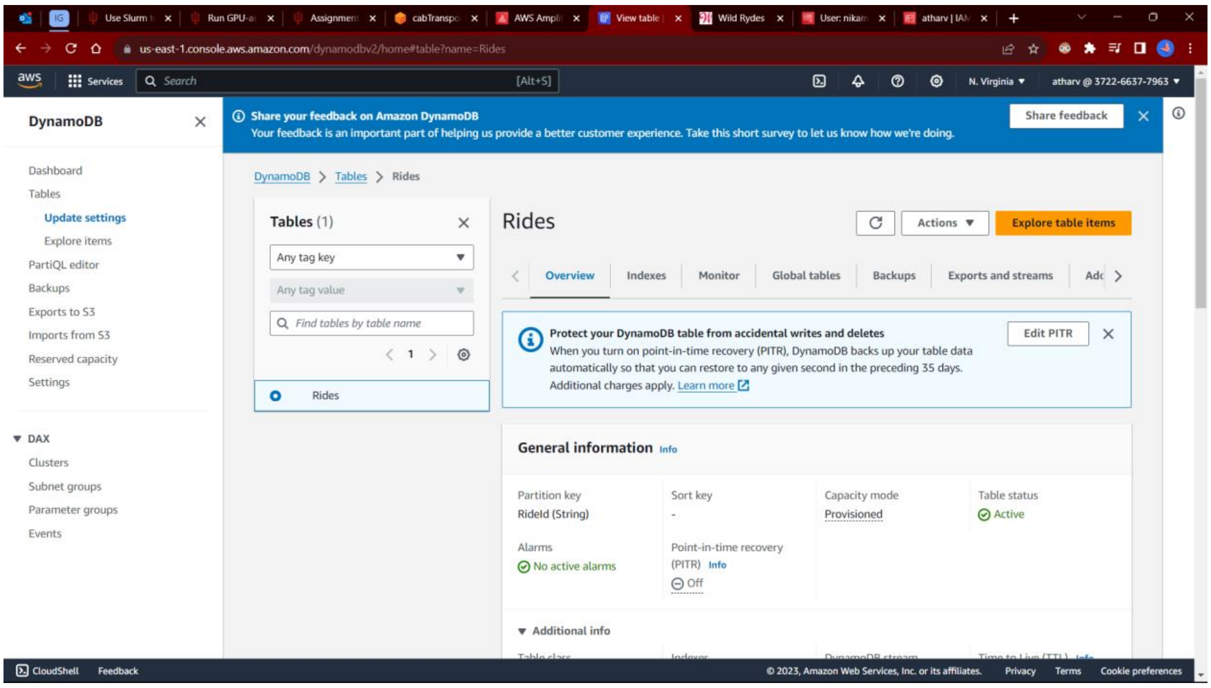
Table details Info
DynamoDB is a schemaless database that requires only a table name and a primary key when you create the table.

Table name
This will be used to identify your table.
Ride
Between 3 and 255 characters, containing only letters, numbers, underscores (_), hyphens (-), and periods (.).

Partition key
The partition key is part of the table's primary key. It is a hash value that is used to retrieve items from your table and allocate data across hosts for scalability and availability.
RideId | String
1 to 255 characters and case sensitive.

Sort key - *optional*
You can use a sort key as the second part of a table's primary key. The sort key allows you to sort or search among all items sharing the same partition key.
Enter the sort key name | String
1 to 255 characters and case sensitive.

© 2023, Amazon Web Services, Inc. or its affiliates.



us-east-1.console.aws.amazon.com/dynamodbv2/home#table?name=Rides

DynamoDB

Share your feedback on Amazon DynamoDB
Your feedback is an important part of helping us provide a better customer experience. Take this short survey to let us know how we're doing.

DynamoDB > Tables > Rides

Tables (1)

Rides

Rides

Protect your DynamoDB table from accidental writes and deletes
When you turn on point-in-time recovery (PITR), DynamoDB backs up your table data automatically so that you can restore to any given second in the preceding 35 days. Additional charges apply. Learn more

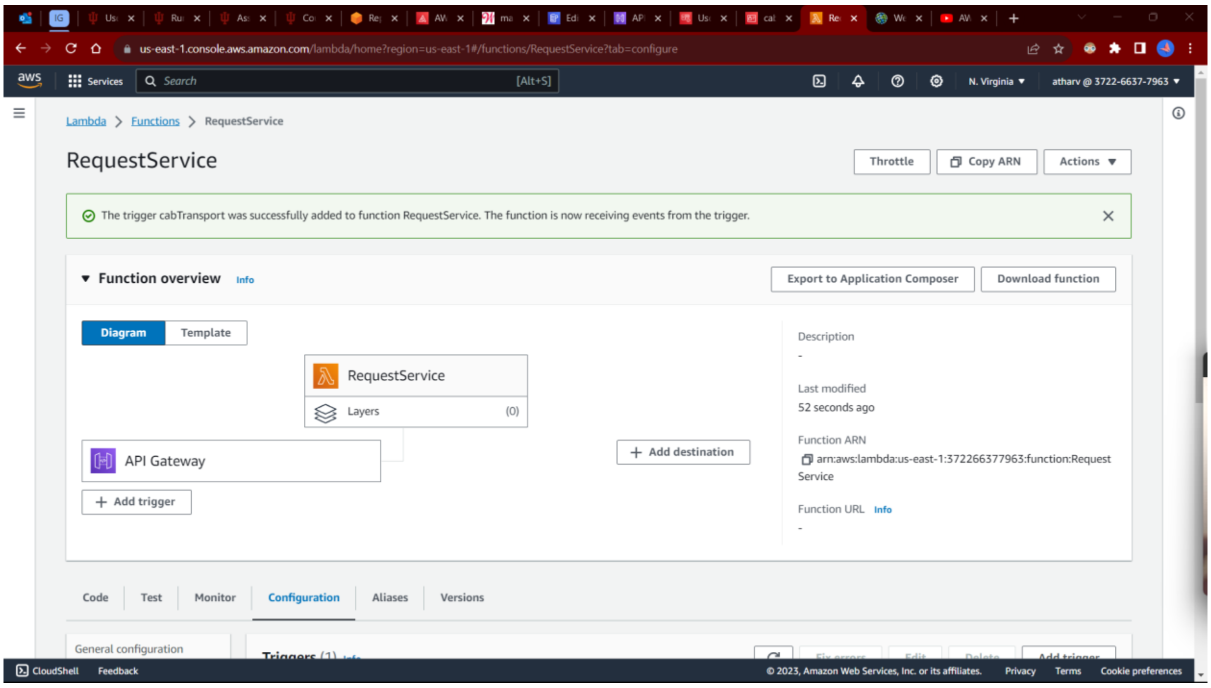
General information

Partition key	Sort key	Capacity mode	Table status
RideId (String)	-	Provisioned	Active

Alarms	Point-in-time recovery (PITR)
No active alarms	Off

Additional info

Step: Lambda function creation and set up for requests handling-



us-east-1.console.aws.amazon.com/lambda/home?region=us-east-1#/functions/RequestService?tab=configure

Lambda > Functions > RequestService

RequestService

The trigger cabTransport was successfully added to function RequestService. The function is now receiving events from the trigger.

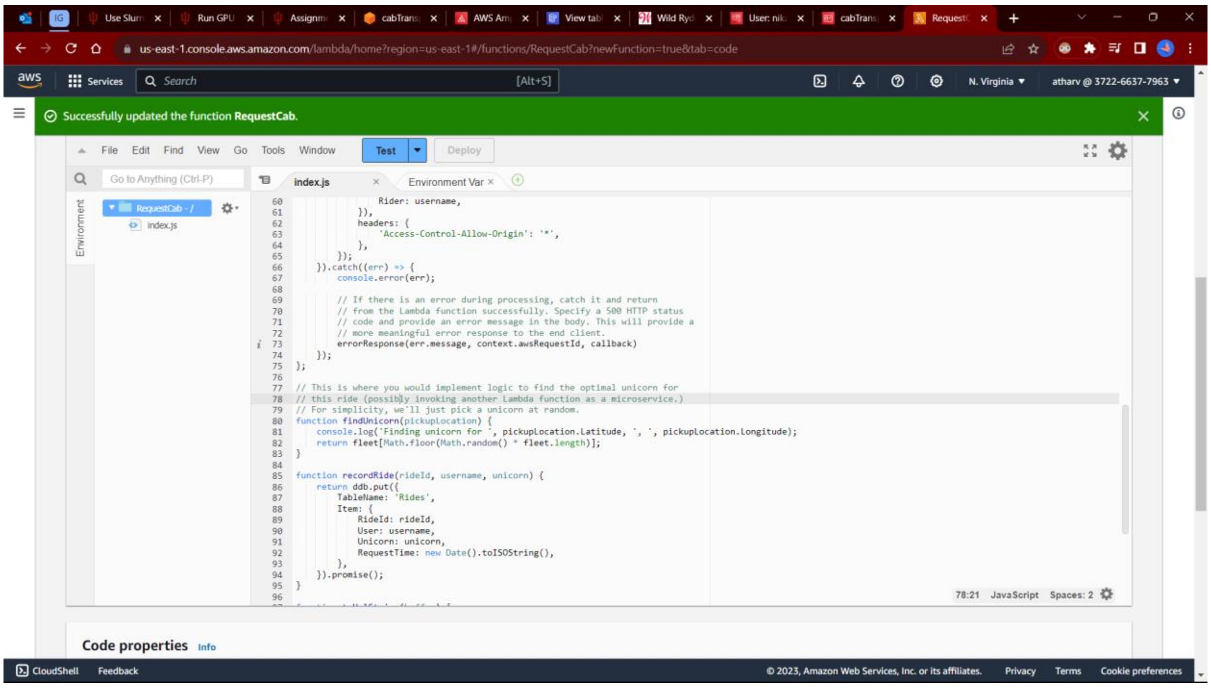
Function overview

Description
-

Last modified
52 seconds ago

Function ARN
arn:aws:lambda:us-east-1:372266377963:function:RequestService

Function URL
-

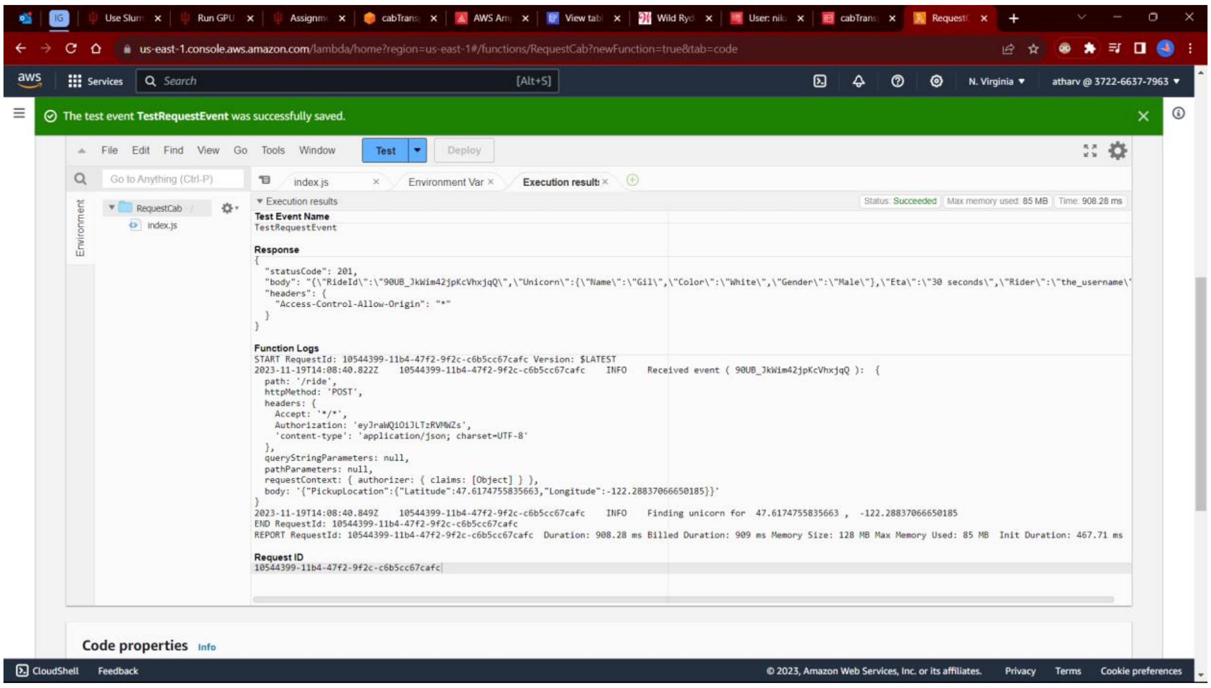

Successfully updated the function **RequestCab**.

```
                Rider: username,
            }),
            headers: {
                'Access-Control-Allow-Origin': '*',
            },
        });
    }).catch((err) => {
        console.error(err);

        // If there is an error during processing, catch it and return
        // from the Lambda function successfully. Specify a 500 HTTP status
        // code and provide an error message in the body. This will provide a
        // more meaningful error response to the end client.
        errorResponse(err.message, context.awsRequestId, callback)
    });
};

// This is where you would implement logic to find the optimal unicorn for
// this ride (possibly invoking another Lambda function as a microservice.)
// For simplicity, we'll just pick a unicorn at random.
function findUnicorn(pickupLocation) {
    console.log('Finding unicorn for ', pickupLocation.Latitude, ', ', pickupLocation.Longitude);
    return fleet[Math.floor(Math.random() * fleet.length)];
}

function recordRide(rideId, username, unicorn) {
    return ddb.put({
        TableName: 'Rides',
        Item: {
            RideId: rideId,
            User: username,
            Unicorn: unicorn,
            RequestTime: new Date().toISOString(),
        },
    }).promise();
}
```


The test event **TestRequestEvent** was successfully saved.

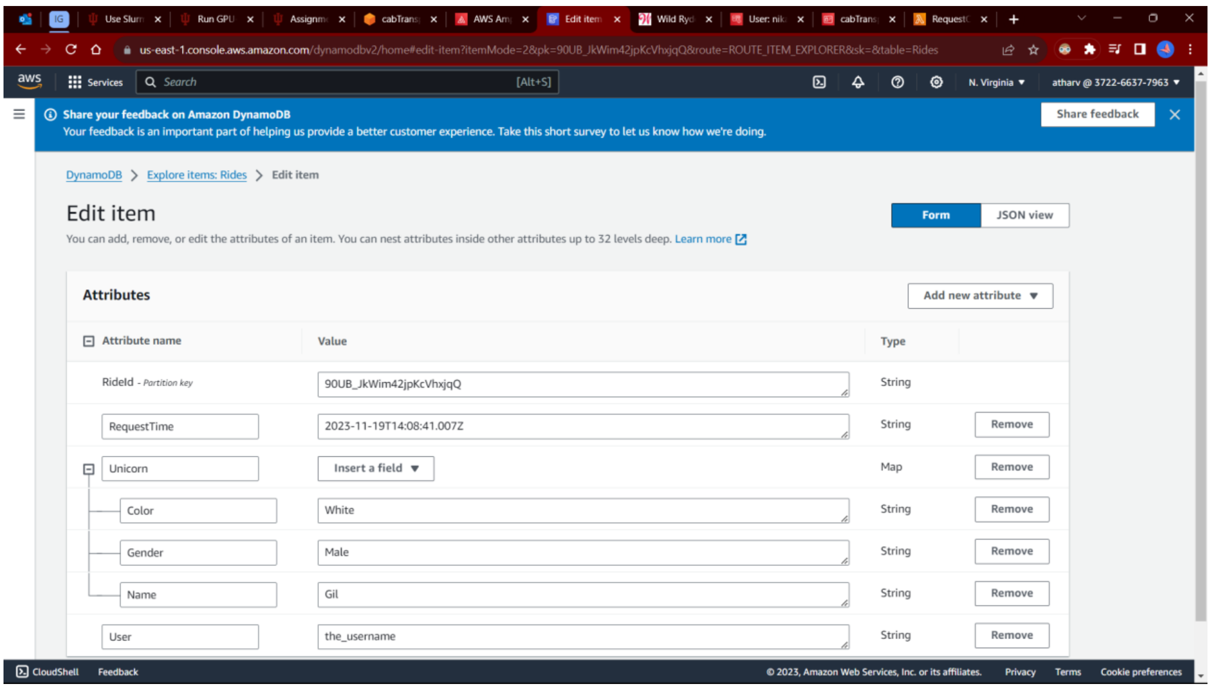
Execution results — Status: Succeeded | Max memory used: 85 MB | Time: 908.28 ms

Test Event Name
TestRequestEvent

Response
```
{
  "statusCode": 201,
  "body": "{\"RideId\":\"90UB_JkWim42jpKcVhxjqQ\",\"Unicorn\":{\"Name\":\"Gil\",\"Color\":\"White\",\"Gender\":\"Male\"},\"Eta\":\"30 seconds\",\"Rider\":\"the_username\"
  "headers": {
    "Access-Control-Allow-Origin": "*"
  }
}
```

Function Logs
```
START RequestId: 10544399-11b4-47f2-9f2c-c6b5cc67cafc Version: $LATEST
2023-11-19T14:08:40.822Z    10544399-11b4-47f2-9f2c-c6b5cc67cafc    INFO    Received event ( 90UB_JkWim42jpKcVhxjqQ ):  {
  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}}'
}
2023-11-19T14:08:40.849Z    10544399-11b4-47f2-9f2c-c6b5cc67cafc    INFO    Finding unicorn for  47.6174755835663 ,  -122.28837066650185
END RequestId: 10544399-11b4-47f2-9f2c-c6b5cc67cafc
REPORT RequestId: 10544399-11b4-47f2-9f2c-c6b5cc67cafc  Duration: 908.28 ms Billed Duration: 909 ms Memory Size: 128 MB Max Memory Used: 85 MB  Init Duration: 467.71 ms
```

Request ID
10544399-11b4-47f2-9f2c-c6b5cc67cafc



us-east-1.console.aws.amazon.com/dynamodbv2/home#edit-item?itemMode=2&pk=90UB_JkWim42jpKcVhxjqQ&route=ROUTE_ITEM_EXPLORER&sk=&table=Rides

Share your feedback on Amazon DynamoDB
Your feedback is an important part of helping us provide a better customer experience. Take this short survey to let us know how we're doing.

DynamoDB > Explore items: Rides > Edit item

Edit item

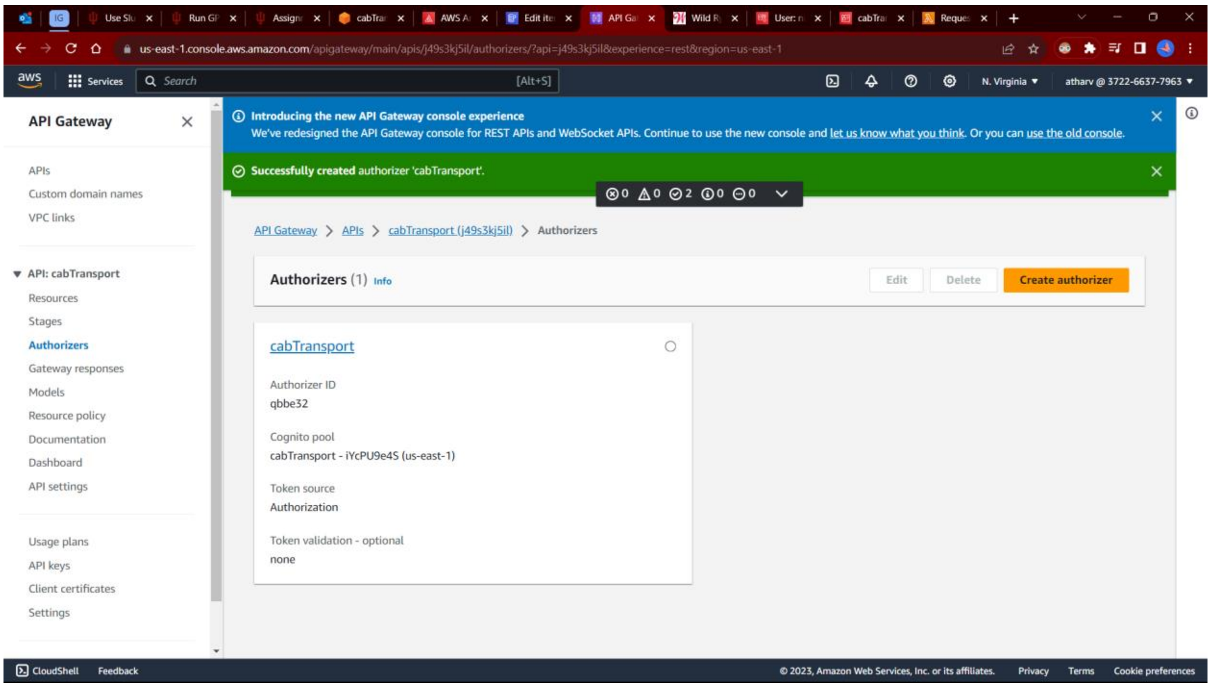
You can add, remove, or edit the attributes of an item. You can nest attributes inside other attributes up to 32 levels deep. Learn more

Attributes

Attribute name	Value	Type
RideId - *Partition key*	90UB_JkWim42jpKcVhxjqQ	String
RequestTime	2023-11-19T14:08:41.007Z	String
Unicorn	Insert a field	Map
Color	White	String
Gender	Male	String
Name	Gil	String
User	the_username	String

© 2023, Amazon Web Services, Inc. or its affiliates.

Step: API Gateway setup for requests, and Testing-



us-east-1.console.aws.amazon.com/apigateway/main/apis/j49s3kj5il/authorizers/?api=j49s3kj5il&experience=rest&region=us-east-1

Introducing the new API Gateway console experience
We've redesigned the API Gateway console for REST APIs and WebSocket APIs. Continue to use the new console and let us know what you think. Or you can use the old console.

Successfully created authorizer 'cabTransport'.

API Gateway > APIs > cabTransport (j49s3kj5il) > Authorizers

Authorizers (1)

cabTransport

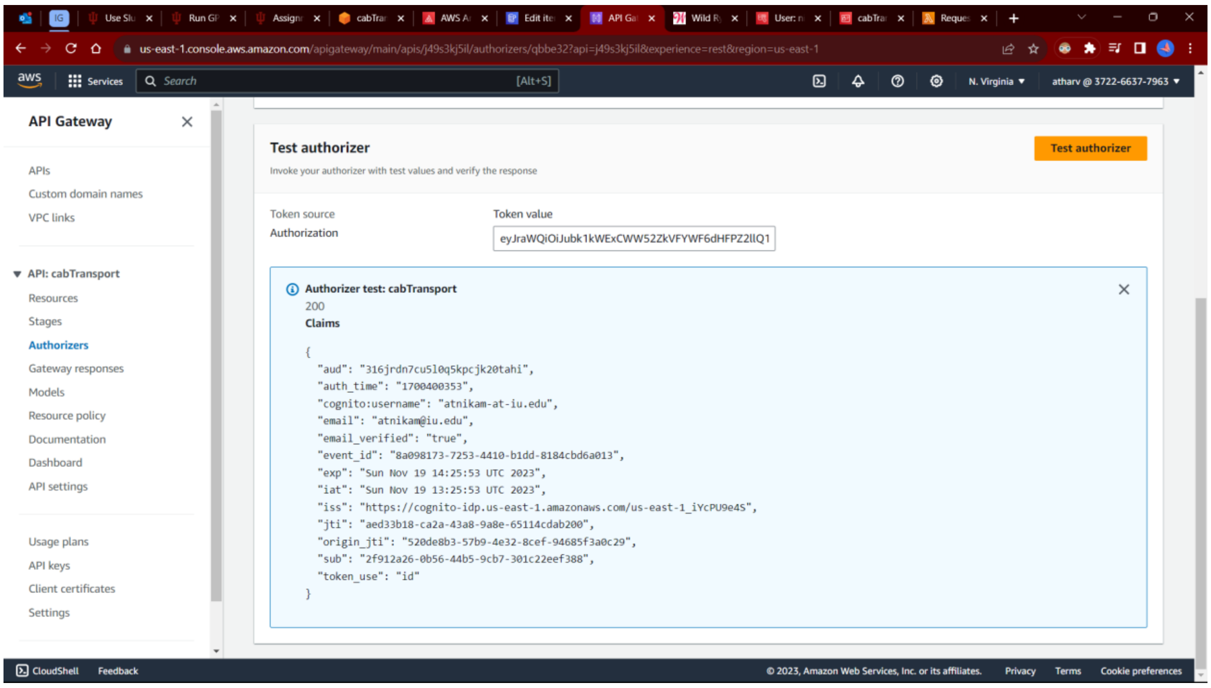
Authorizer ID
qbbe32

Cognito pool
cabTransport - iYcPU9e4S (us-east-1)

Token source
Authorization

Token validation - optional
none

© 2023, Amazon Web Services, Inc. or its affiliates.


us-east-1.console.aws.amazon.com/apigateway/main/apis/j49s3kj5il/authorizers/qbbe32?api=j49s3kj5il&experience=rest&region=us-east-1

Test authorizer

Invoke your authorizer with test values and verify the response

Token source: Authorization

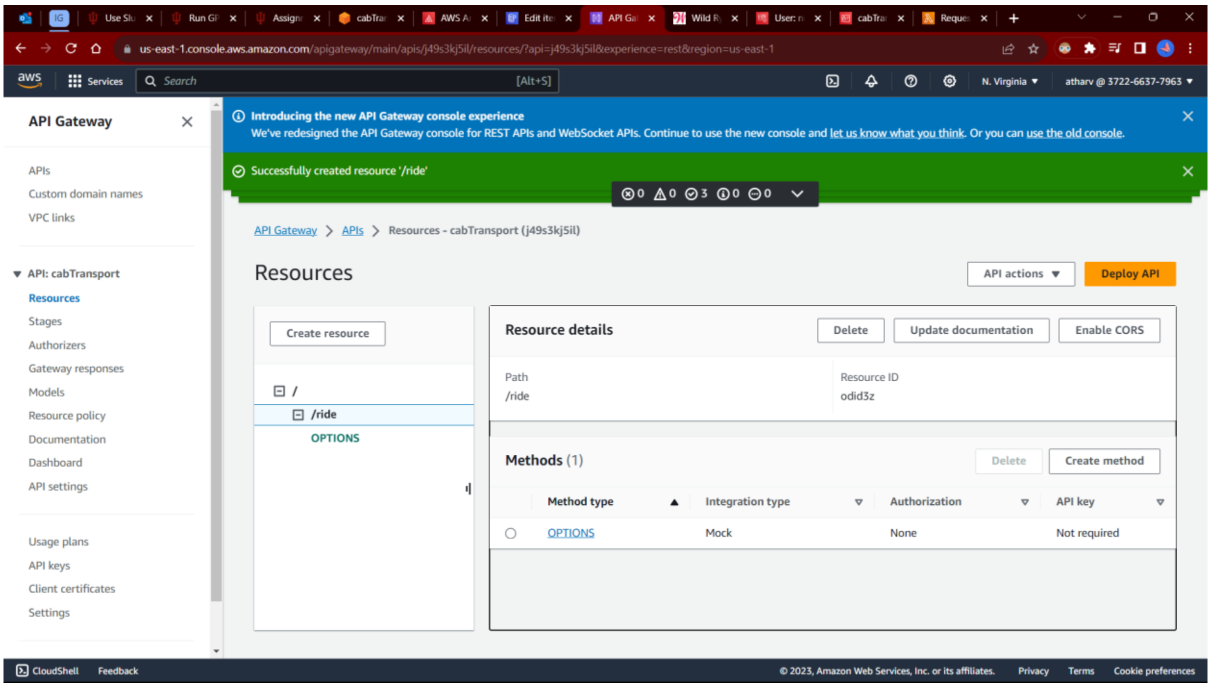
Token value: eyJraWQiOiJubk1kWExCWW52ZkVFYWF6dHFPZ2llQ1

Authorizer test: cabTransport

200

Claims

```
{
  "aud": "316jrdn7cu5l0q5kpcjk20tahi",
  "auth_time": "1700400353",
  "cognito:username": "atnikam-at-iu.edu",
  "email": "atnikam@iu.edu",
  "email_verified": "true",
  "event_id": "8a098173-7253-4410-b1dd-8184cbd6a013",
  "exp": "Sun Nov 19 14:25:53 UTC 2023",
  "iat": "Sun Nov 19 13:25:53 UTC 2023",
  "iss": "https://cognito-idp.us-east-1.amazonaws.com/us-east-1_iYcPU9e4S",
  "jti": "aed33b18-ca2a-43a8-9a8e-65114cdab200",
  "origin_jti": "520de8b3-57b9-4e32-8cef-94685f3a0c29",
  "sub": "2f912a26-0b56-44b5-9cb7-301c22eef388",
  "token_use": "id"
}
```


us-east-1.console.aws.amazon.com/apigateway/main/apis/j49s3kj5il/resources?api=j49s3kj5il&experience=rest&region=us-east-1

Introducing the new API Gateway console experience
We've redesigned the API Gateway console for REST APIs and WebSocket APIs. Continue to use the new console and let us know what you think. Or you can use the old console.

Successfully created resource '/ride'

API Gateway > APIs > Resources - cabTransport (j49s3kj5il)

Resources

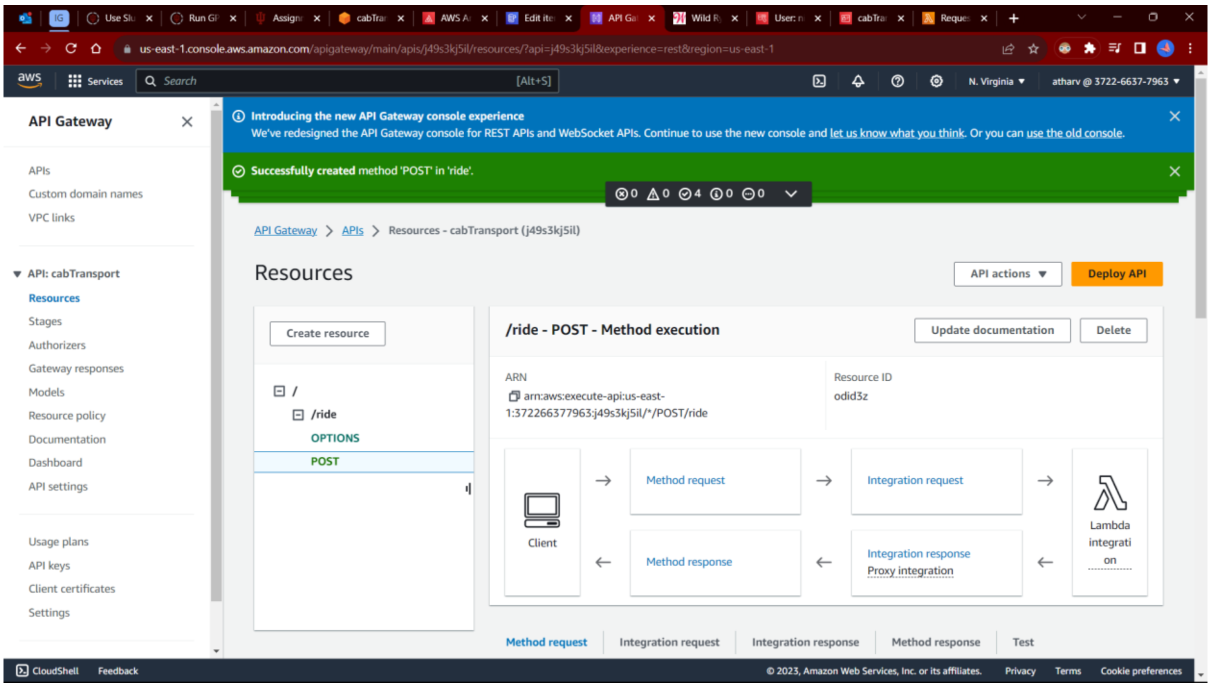
/
/ride
OPTIONS

Resource details

Path	Resource ID
/ride	odid3z

Methods (1)

Method type	Integration type	Authorization	API key
OPTIONS	Mock	None	Not required



us-east-1.console.aws.amazon.com/apigateway/main/apis/j49s3kj5il/resources/?api=j49s3kj5il&experience=rest&region=us-east-1

Introducing the new API Gateway console experience
We've redesigned the API Gateway console for REST APIs and WebSocket APIs. Continue to use the new console and let us know what you think. Or you can use the old console.

Successfully created method 'POST' in 'ride'.

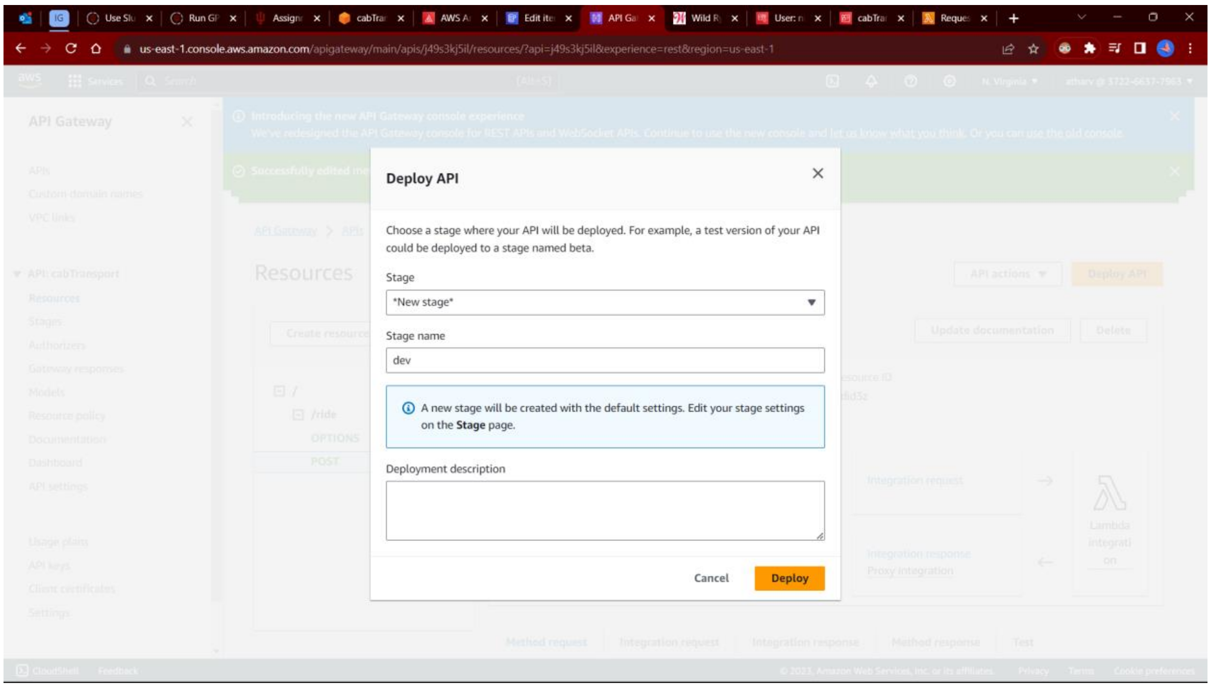
API Gateway > APIs > Resources - cabTransport (j49s3kj5il)

Resources

/ride - POST - Method execution

ARN
arn:aws:execute-api:us-east-1:372266377963:j49s3kj5il/*/POST/ride

Resource ID
odid3z



us-east-1.console.aws.amazon.com/apigateway/main/apis/j49s3kj5il/resources/?api=j49s3kj5il&experience=rest&region=us-east-1

Deploy API

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.

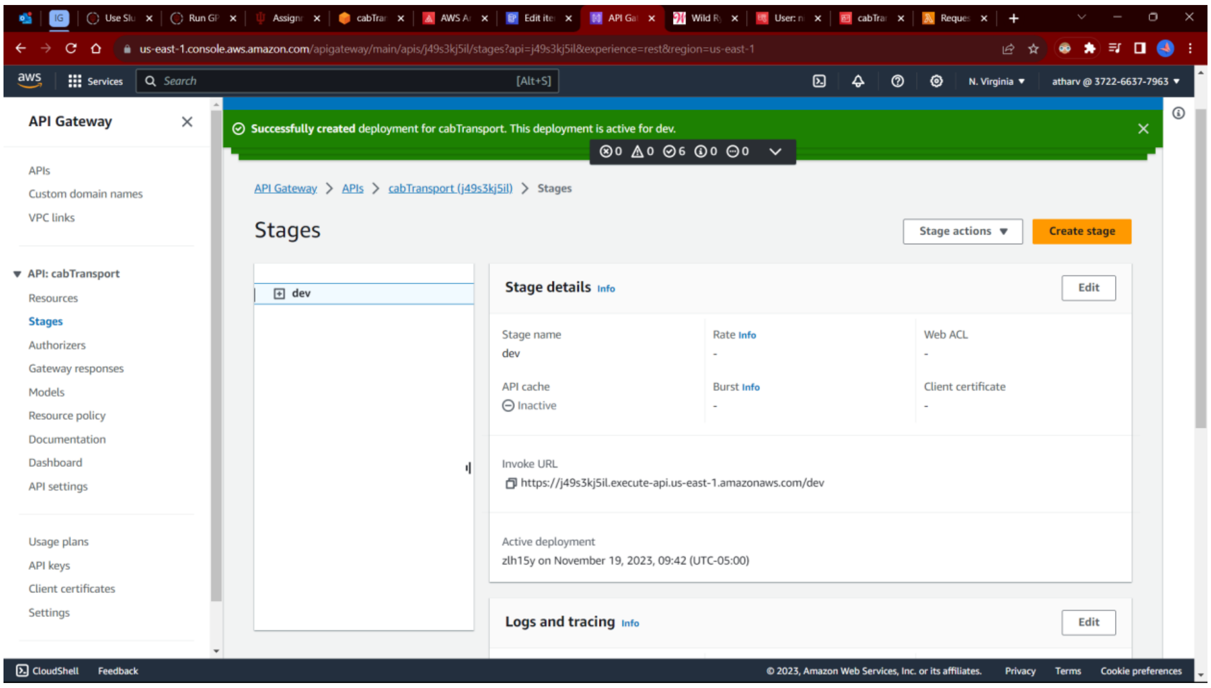
Stage
New stage ▾

Stage name
dev

A new stage will be created with the default settings. Edit your stage settings on the **Stage** page.

Deployment description

Step: Setting stages



us-east-1.console.aws.amazon.com/apigateway/main/apis/j49s3kj5il/stages?api=j49s3kj5il&experience=rest&region=us-east-1

Successfully created deployment for cabTransport. This deployment is active for dev.

API Gateway > APIs > cabTransport (j49s3kj5il) > Stages

Stages

Stage details Info

Stage name
dev

Rate Info
-

Web ACL
-

API cache
Inactive

Burst Info
-

Client certificate
-

Invoke URL
https://j49s3kj5il.execute-api.us-east-1.amazonaws.com/dev

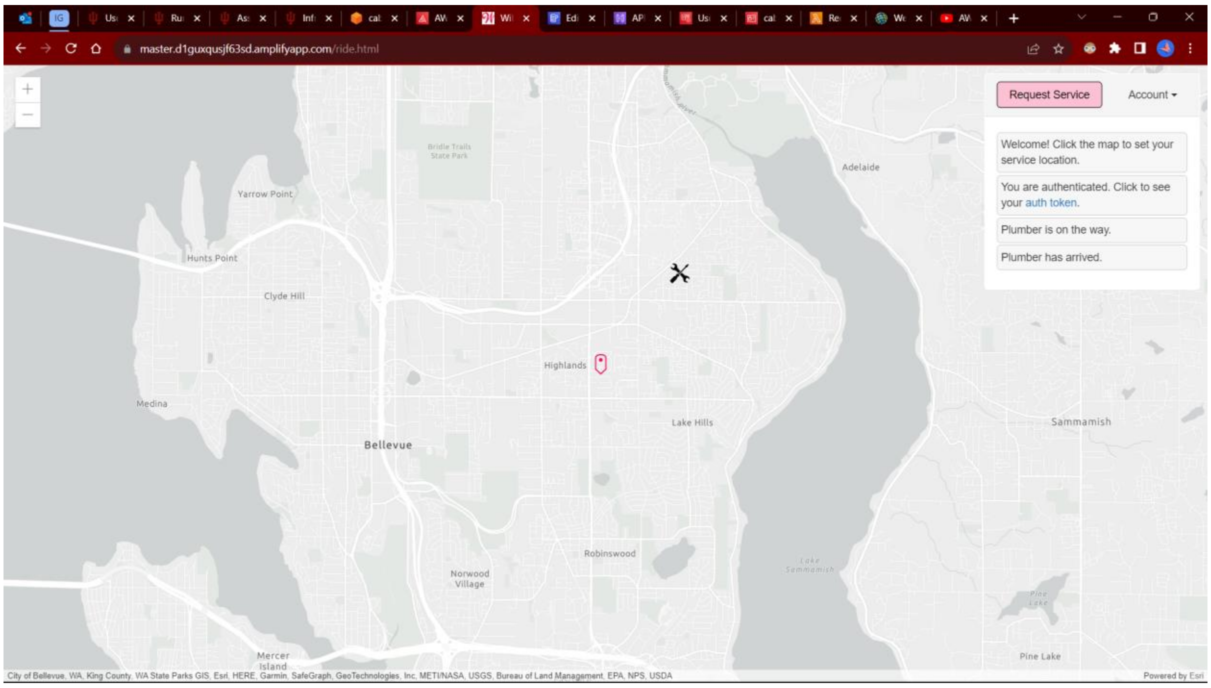
Active deployment
zlh15y on November 19, 2023, 09:42 (UTC-05:00)

Logs and tracing Info

Step: Testing the web application for different loads-

Here, tried creating many users, and also sending multiple service requests. App worked well during this process.

CityTasks app:



master.d1guxqusjf63sd.amplifyapp.com/ride.html

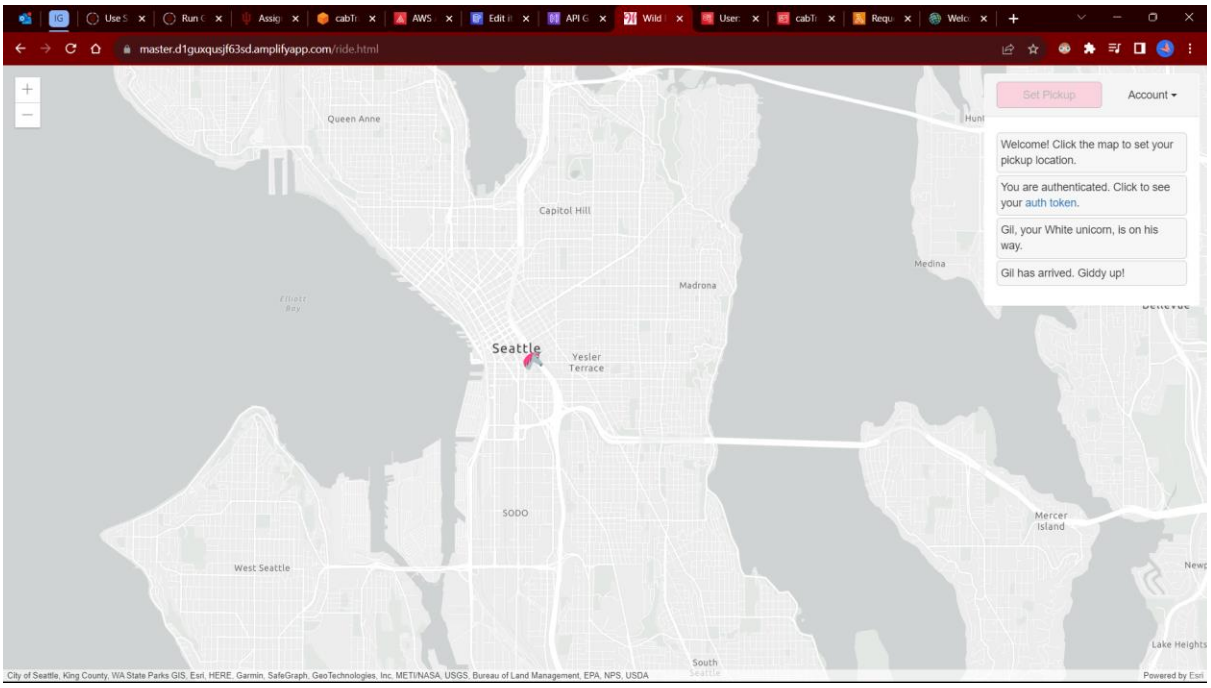
Welcome! Click the map to set your service location.

You are authenticated. Click to see your auth token.

Plumber is on the way.

Plumber has arrived.

Unicorn (cabTransport) app:



Welcome! Click the map to set your pickup location.

You are authenticated. Click to see your auth token.

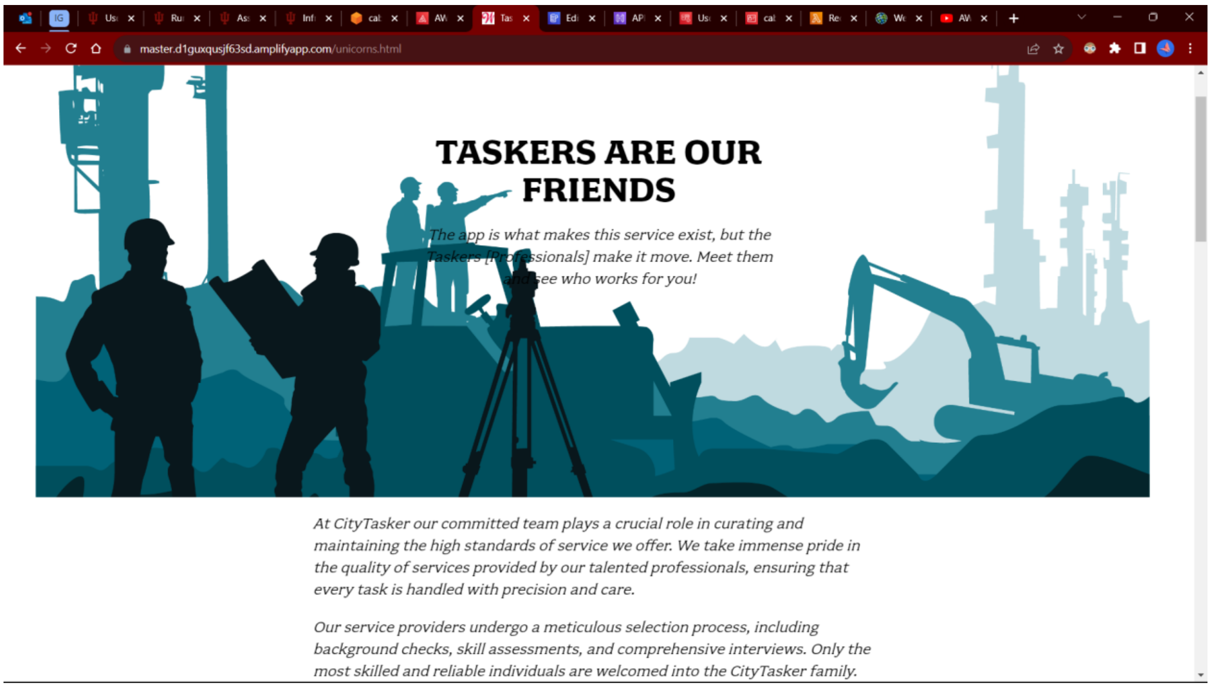
Gil, your White unicorn, is on his way.

Gil has arrived. Giddy up!

City of Seattle, King County, WA State Parks GIS, Esri, HERE, Garmin, SafeGraph, GeoTechnologies, Inc, METI/NASA, USGS, Bureau of Land Management, EPA, NPS, USDA

Powered by Esri

Other pages:

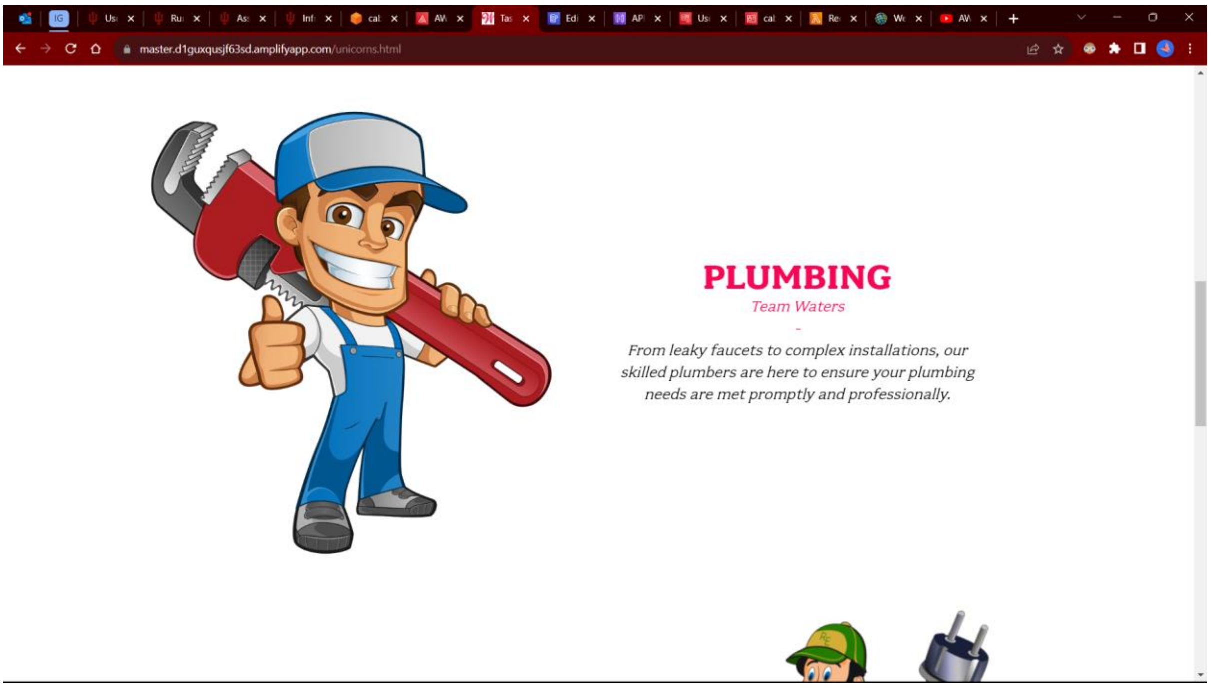


TASKERS ARE OUR FRIENDS

The app is what makes this service exist, but the Taskers [professionals] make it move. Meet them and see who works for you!

At CityTasker our committed team plays a crucial role in curating and maintaining the high standards of service we offer. We take immense pride in the quality of services provided by our talented professionals, ensuring that every task is handled with precision and care.

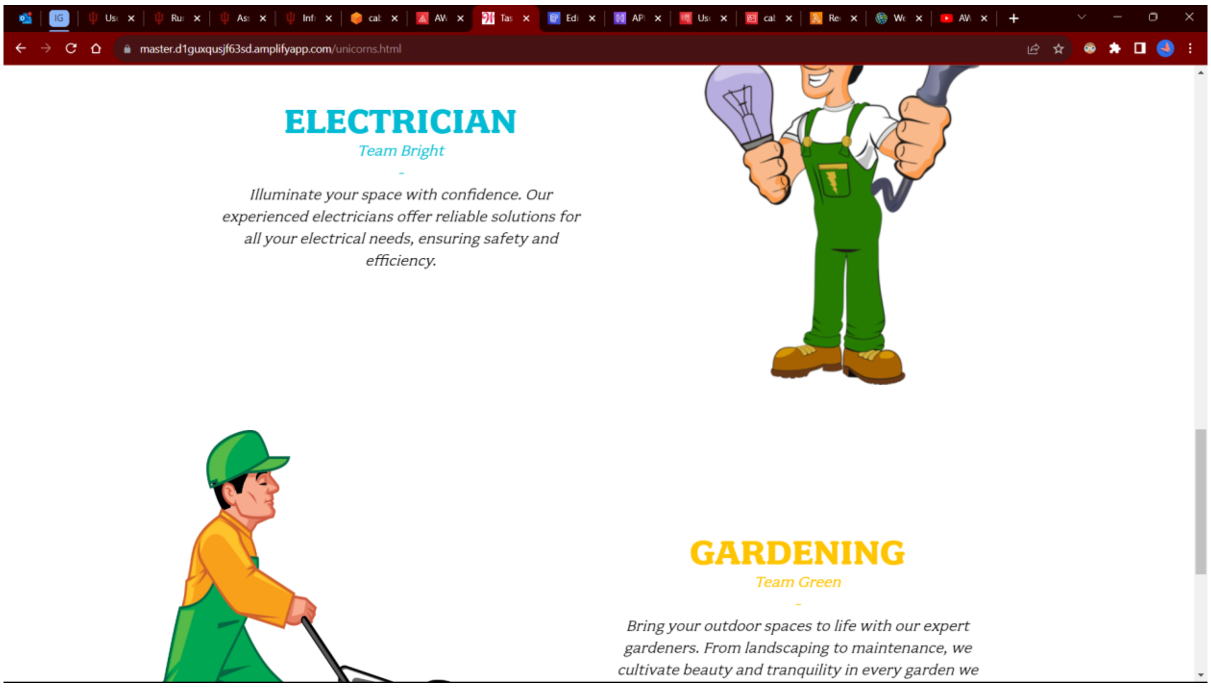
Our service providers undergo a meticulous selection process, including background checks, skill assessments, and comprehensive interviews. Only the most skilled and reliable individuals are welcomed into the CityTasker family.


master.d1guxqusjf63sd.amplifyapp.com/unicorns.html

PLUMBING

Team Waters

From leaky faucets to complex installations, our skilled plumbers are here to ensure your plumbing needs are met promptly and professionally.


master.d1guxqusjf63sd.amplifyapp.com/unicorns.html

ELECTRICIAN

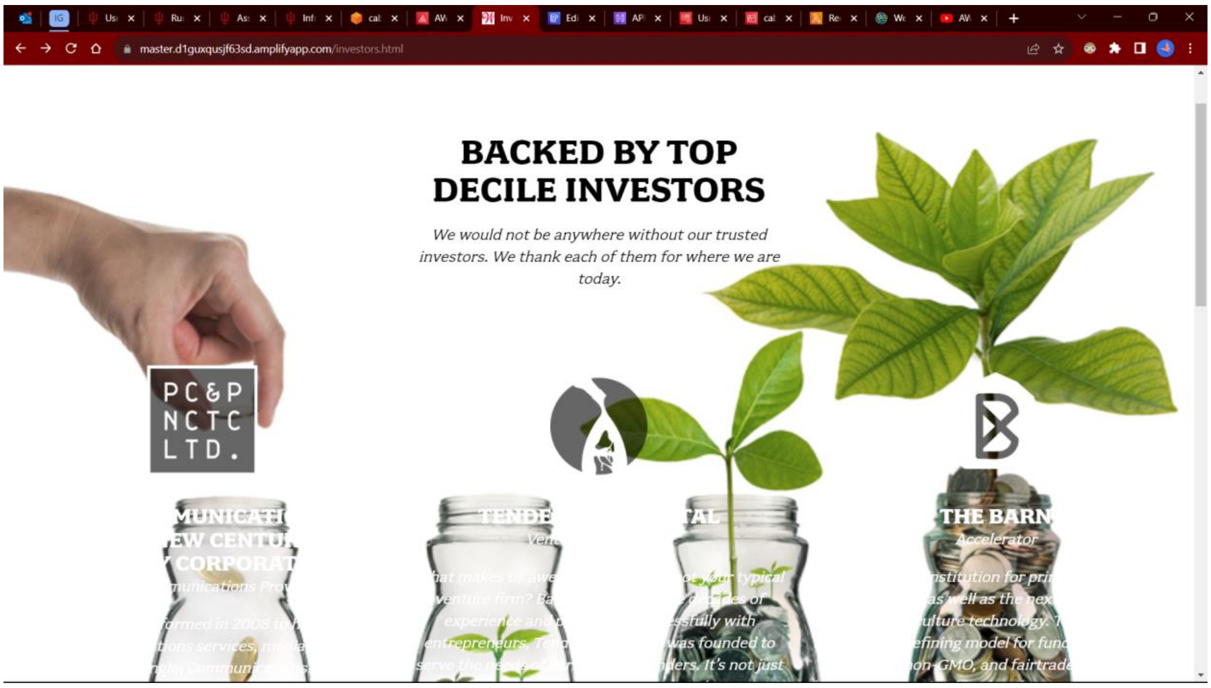
Team Bright

Illuminate your space with confidence. Our experienced electricians offer reliable solutions for all your electrical needs, ensuring safety and efficiency.

GARDENING

Team Green

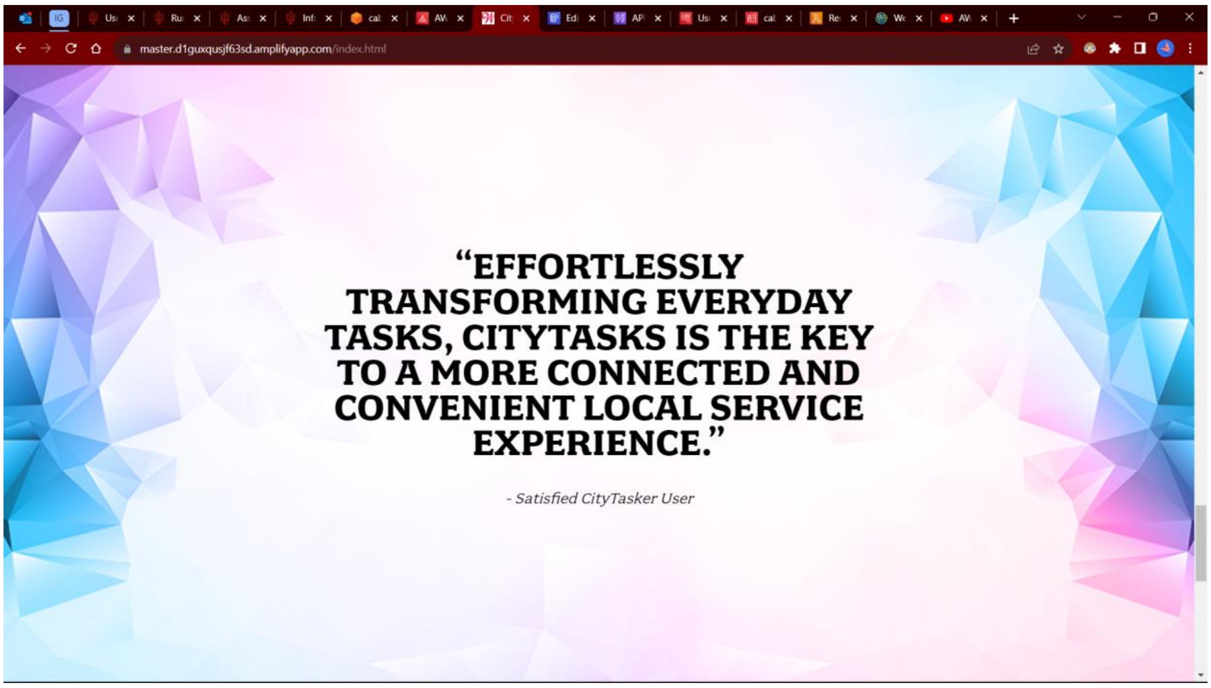
Bring your outdoor spaces to life with our expert gardeners. From landscaping to maintenance, we cultivate beauty and tranquility in every garden we



master.d1guxqusjf63sd.amplifyapp.com/investors.html

BACKED BY TOP DECILE INVESTORS

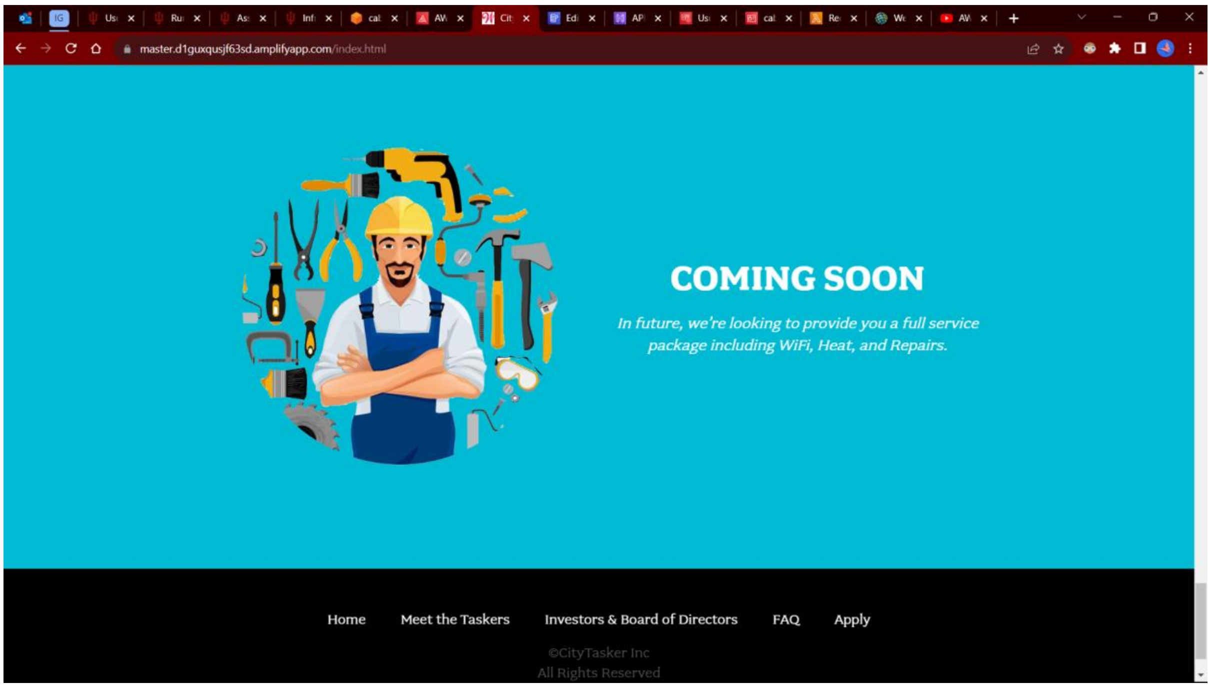
We would not be anywhere without our trusted investors. We thank each of them for where we are today.



master.d1guxqusjf63sd.amplifyapp.com/index.html

“EFFORTLESSLY TRANSFORMING EVERYDAY TASKS, CITYTASKS IS THE KEY TO A MORE CONNECTED AND CONVENIENT LOCAL SERVICE EXPERIENCE.”

- Satisfied CityTasker User


master.d1guxqusjf63sd.amplifyapp.com/index.html
COMING SOON
In future, we're looking to provide you a full service package including WiFi, Heat, and Repairs.
Home
Meet the Taskers
Investors & Board of Directors
FAQ
Apply
©CityTasker Inc
All Rights Reserved

- **Benefits and advantages of using AWS Compute, Storage, Database, and Infrastructure Management Services:**
  - **Compute Resources (Amazon EC2 Auto Scaling):**
    - With auto scaling, the number of EC2 instances may be dynamically adjusted by the web application based on demand, providing peak performance during periods of high traffic and cost savings during periods of low demand.
    - In order to optimize expenses and guarantee effective use, EC2 instances may be configured with the appropriate computing resources required for the application.
    - Application deployment may be streamlined by leveraging AWS services to provision and deploy EC2 instances with ease.
  - **Relational Database:**
    - Regular database maintenance, including patching, backups, and scalability, is handled by AWS. The development team's administrative workload is lessened as a result.
    - By replicating the database across different availability zones, multi-AZ deployment guarantees fault tolerance and high availability. This improves the application's dependability.
  - **File Storage:**
    - S3 provides robust and highly scalable object storage, guaranteeing the platform can manage the increasing volumes of data linked to service requests.
    - Cost-effective storage is possible with pay-as-you-go pricing since there are no upfront obligations. This is particularly helpful for a platform whose storage requirements could fluctuate.
  - **AWS CloudFormation:**

        - The complete infrastructure may be created consistently and reproducibly with Infrastructure as Code. This guarantees consistency across the environments used for development, testing, and production.
        - Version control for infrastructure code allows teams to monitor changes, work together efficiently, and revert to earlier iterations as needed.

<u>Task 10:</u>

- I followed AWS tutorials and documentation while implementing the deployment. Thus, did not experience many errors or problems.
- While embedding the map API into the application, earlier I ended up configuring it incorrectly. But after few rechecks, I could solve it.
- I followed one of the AWS tutorial code for creating the web page scripts and highly referred their code, but different helper packages made it easier.