AWS WEB APPLICATION WILDRYDES

ANUSHKA DHOLE

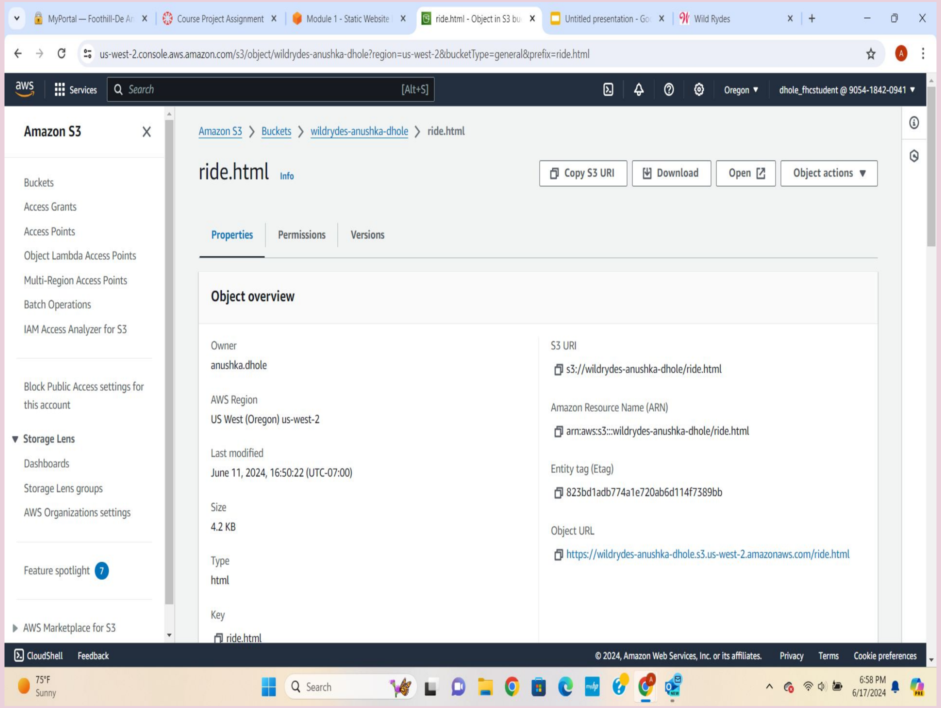
Cognito

Amazon
API Gateway

The URL for my Website Application - Links

https://wildrydes-anushka-dhole.s3.us-west-2.amazonaws.com/ride.htm

https://wildrydes-anushka-dhole.s3.us-west-2.amazonaws.com/signin.html


MyPortal — Foothill-De An
Course Project Assignment
Module 1 - Static Website
ride.html - Object in S3 bu
Untitled presentation - Go
Wild Rydes
us-west-2.console.aws.amazon.com/s3/object/wildrydes-anushka-dhole?region=us-west-2&bucketType=general&prefix=ride.html
Services
Search [Alt+S]
Oregon
dhole_fhcstudent @ 9054-1842-0941
Amazon S3
Buckets
Access Grants
Access Points
Object Lambda Access Points
Multi-Region Access Points
Batch Operations
IAM Access Analyzer for S3
Block Public Access settings for this account
Storage Lens
Dashboards
Storage Lens groups
AWS Organizations settings
Feature spotlight 7
AWS Marketplace for S3
Amazon S3 > Buckets > wildrydes-anushka-dhole > ride.html
Copy S3 URI
Download
Open
Object actions
Properties
Permissions
Versions
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences
75°F Sunny
Search
6:58 PM 6/17/2024

ride.html Info

Object overview

Owner
anushka.dhole

AWS Region
US West (Oregon) us-west-2

Last modified
June 11, 2024, 16:50:22 (UTC-07:00)

Size
4.2 KB

Type
html

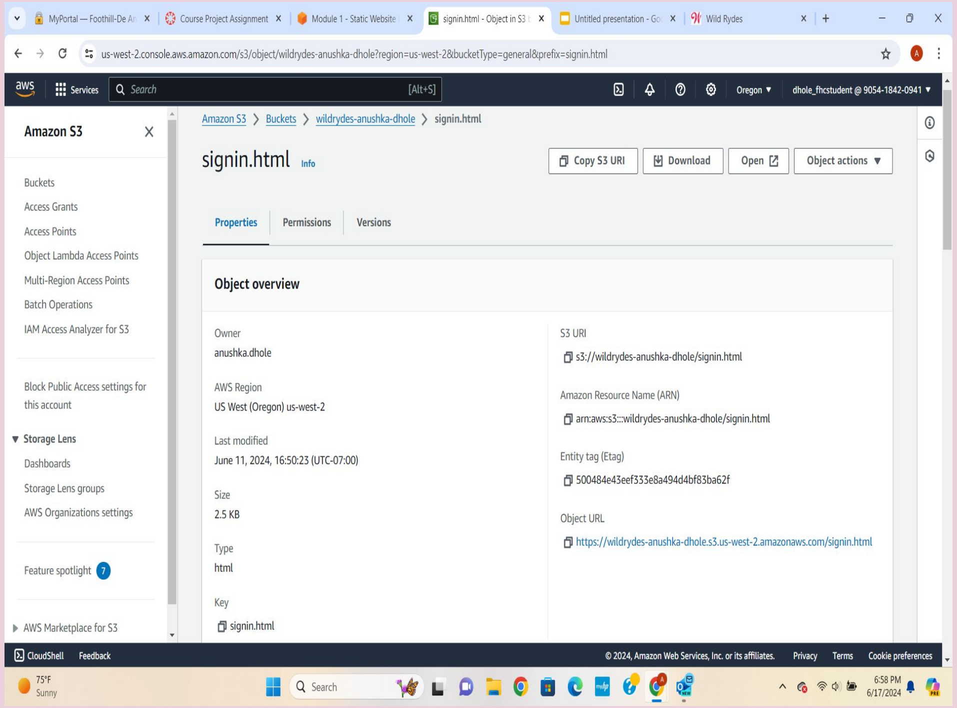
Key
ride.html

S3 URI
s3://wildrydes-anushka-dhole/ride.html

Amazon Resource Name (ARN)
arn:aws:s3:::wildrydes-anushka-dhole/ride.html

Entity tag (Etag)
823bd1adb774a1e720ab6d114f7389bb

Object URL
https://wildrydes-anushka-dhole.s3.us-west-2.amazonaws.com/ride.html


MyPortal — Foothill-De An
Course Project Assignment
Module 1 - Static Website
signin.html - Object in S3 b
Untitled presentation - Go
Wild Rydes
us-west-2.console.aws.amazon.com/s3/object/wildrydes-anushka-dhole?region=us-west-2&bucketType=general&prefix=signin.html
Services
Search [Alt+S]
Oregon
dhole_fhcstudent @ 9054-1842-0941
Amazon S3
Buckets
Access Grants
Access Points
Object Lambda Access Points
Multi-Region Access Points
Batch Operations
IAM Access Analyzer for S3
Block Public Access settings for this account
Storage Lens
Dashboards
Storage Lens groups
AWS Organizations settings
Feature spotlight 7
AWS Marketplace for S3
Amazon S3 > Buckets > wildrydes-anushka-dhole > signin.html
Copy S3 URI
Download
Open
Object actions
Properties
Permissions
Versions
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences
75°F Sunny
Search
6:58 PM 6/17/2024

signin.html Info

Object overview

Owner
anushka.dhole

AWS Region
US West (Oregon) us-west-2

Last modified
June 11, 2024, 16:50:23 (UTC-07:00)

Size
2.5 KB

Type
html

Key
signin.html

S3 URI
s3://wildrydes-anushka-dhole/signin.html

Amazon Resource Name (ARN)
arn:aws:s3:::wildrydes-anushka-dhole/signin.html

Entity tag (Etag)
500484e43eef333e8a494d4bf83ba62f

Object URL
https://wildrydes-anushka-dhole.s3.us-west-2.amazonaws.com/signin.html

Overview of the WildRydes Platform on AWS

The WildRydes platform is designed using various AWS services to create a scalable, serverless web application. The architecture utilizes AWS Lambda, Amazon API Gateway, Amazon DynamoDB, Amazon Cognito, and AWS S3.

1. **AWS S3 -** AWS S3 hosts the static web resources such as HTML, CSS, JavaScript, and image files. These resources are loaded in the user's browser when they access the application via the URL's. The static content is deployed to an S3 bucket, ensuring scalability and availability.

2. **Amazon Cognito -** Amazon Cognito manages user authentication and authorization. It secures the backend API by ensuring that only authenticated users can access specific endpoints. During sign-up or sign-in, Cognito handles the authentication flow, including sending verification emails and managing tokens. If a user fails to complete the registration, they will not gain access to the secure parts of the application, thus maintaining the integrity and security of the platform.

3. **Amazon DynamoDB -** Amazon DynamoDB acts as the database, storing and retrieving data required by the application. It interacts with the Lambda functions to store information such as user profiles and ride history. An example entry in DynamoDB includes:

- `RideId`: Unique identifier for each ride
- `User`: Username of the rider
- `Unicorn`: Details of the assigned unicorn
- `RequestTime`: Times of the ride request

Building Blocks, How Code Runs & Interactions

AWS Lambda - AWS Lambda functions handle the backend logic. These functions are triggered by requests from the API Gateway and perform operations such as processing ride requests and updating the DynamoDB table. For example, when a user requests a ride, the Lambda function processes the request, assigns a unicorn, stores the ride details in DynamoDB, and returns a response.

Amazon API Gateway - Acts as an interface between the frontend and the backend. Routes HTTP requests from the frontend to the appropriate Lambda functions. Provides endpoint security by integrating with Amazon Cognito for user authentication.

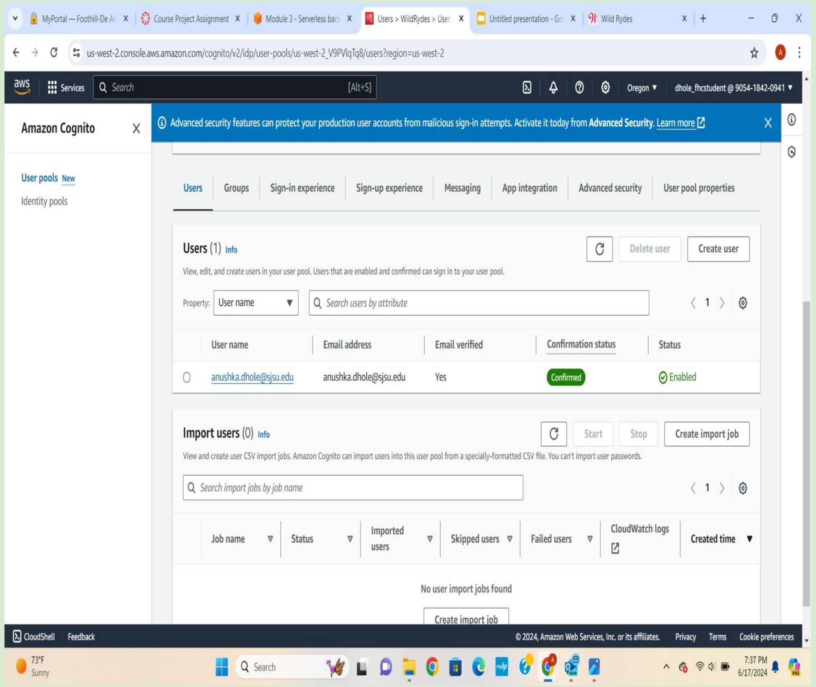
IAM Roles - Security and Permissions - IAM roles are assigned to Lambda functions to grant them the necessary permissions to interact with other AWS services such as DynamoDB and Cognito. This ensures that each function has the minimum necessary permissions, enhancing the security of the application.

The JavaScript code running in the browser interacts with AWS services in the following way:

1. **User Authentication**:
    - Users authenticate via Amazon Cognito, which manages user sign-up and sign-in.
    - Upon successful authentication, Cognito provides tokens used for subsequent API requests.
2. **Ride Request**:
    - The user requests a ride from the frontend, which sends a POST request to the API Gateway endpoint.
    - The request includes the user's pickup location and authentication token.
3. **Backend Processing**:
    - API Gateway invokes a Lambda function with the ride request.
    - The Lambda function verifies the authentication token using Cognito and processes the request.
    - The function selects a random unicorn from a predefined fleet and records the ride details in DynamoDB.
    - A response is sent back to the frontend with ride details including the unicorn's name, color, and estimated time of arrival (ETA).

Amazon Cognito, Account Screenshot, User Registration

- Manages user authentication and authorization.
- Ensures that only authenticated users can access the backend API.
- If a user does not successfully complete registration, they cannot access authenticated API endpoints.



Users (1)

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

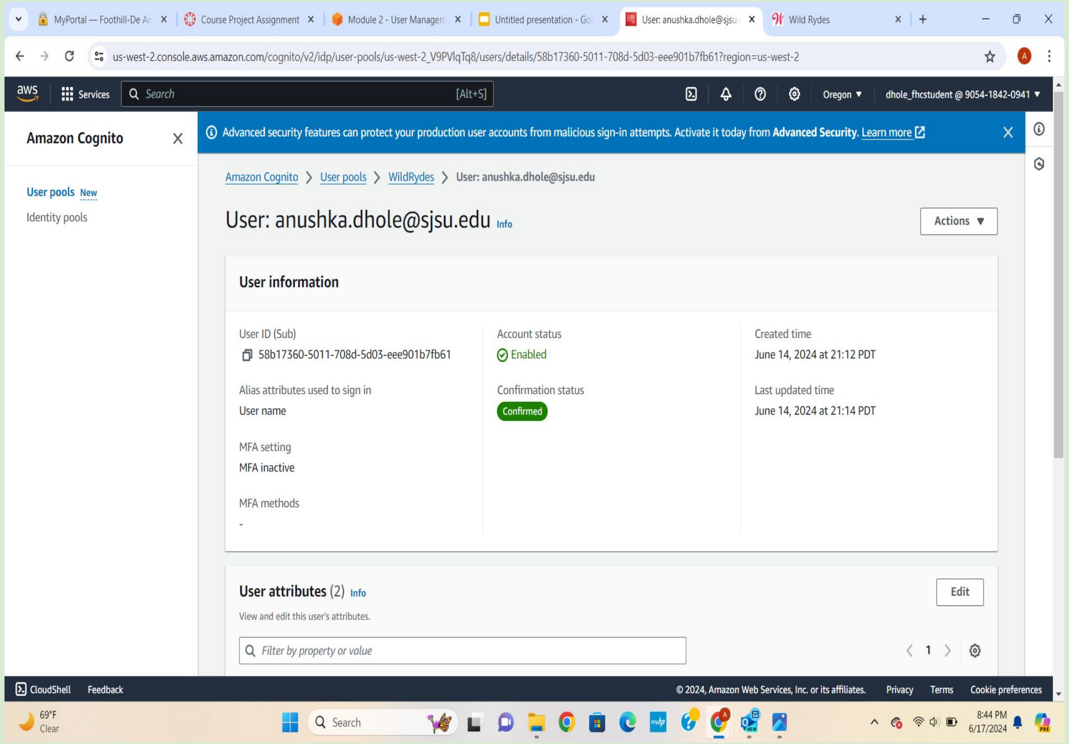
User name	Email address	Email verified	Confirmation status	Status
anushka.dhole@sjsu.edu	anushka.dhole@sjsu.edu	Yes	Confirmed	Enabled

Import users (0)

View and create user CSV import jobs. Amazon Cognito can import users into this user pool from a specially-formatted CSV file. You can't import user passwords.

Job name	Status	Imported users	Skipped users	Failed users	CloudWatch logs	Created time

No user import jobs found



User: anushka.dhole@sjsu.edu

User information

User ID (Sub): 58b17360-5011-708d-5d03-eee901b7fb61	Account status: Enabled	Created time: June 14, 2024 at 21:12 PDT
Alias attributes used to sign in: User name	Confirmation status: Confirmed	Last updated time: June 14, 2024 at 21:14 PDT
MFA setting: MFA inactive		
MFA methods: -		

User attributes (2)

View and edit this user's attributes.

Interaction with AWS Services

Interaction with AWS Services

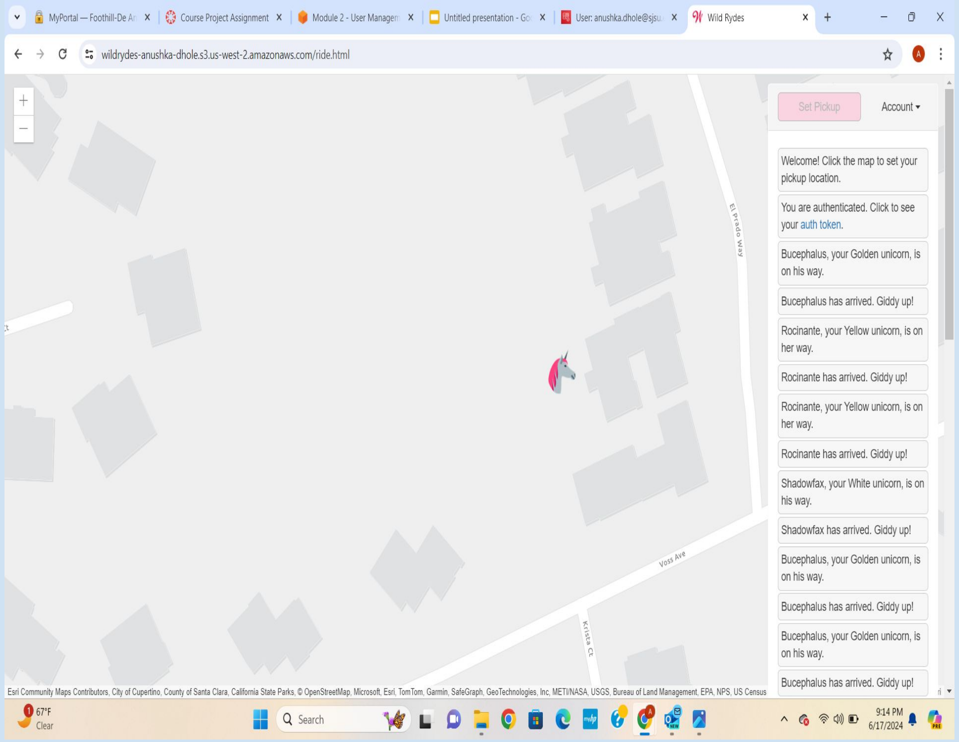
Amazon Cognito: Authenticates users and issues tokens for session management.

Amazon S3: Serves the static assets of the web application.

Amazon DynamoDB: Stores user and ride data.

AWS Lambda: Executes the business logic in response to API calls.

Amazon API Gateway: Routes HTTP requests from the frontend to the appropriate Lambda functions.


MyPortal — Foothill-De An
Course Project Assignment
Module 2 - User Managem
Untitled presentation - Go
User: anushka.dhole@sjsu
Wild Rydes
wildrydes-anushka-dhole.s3.us-west-2.amazonaws.com/ride.html
Set Pickup
Account
El Prado Way
Voss Ave
Krista Ct
67°F
Clear
Search
9:14 PM
6/17/2024

Welcome! Click the map to set your pickup location.

You are authenticated. Click to see your auth token.

Bucephalus, your Golden unicorn, is on his way.

Bucephalus has arrived. Giddy up!

Rocinante, your Yellow unicorn, is on her way.

Rocinante has arrived. Giddy up!

Rocinante, your Yellow unicorn, is on her way.

Rocinante has arrived. Giddy up!

Shadowfax, your White unicorn, is on his way.

Shadowfax has arrived. Giddy up!

Bucephalus, your Golden unicorn, is on his way.

Bucephalus has arrived. Giddy up!

Bucephalus, your Golden unicorn, is on his way.

Bucephalus has arrived. Giddy up!

Esri Community Maps Contributors, City of Cupertino, County of Santa Clara, California State Parks, © OpenStreetMap, Microsoft, Esri, TomTom, Garmin, SafeGraph, GeoTechnologies, Inc, METI/NASA, USGS, Bureau of Land Management, EPA, NPS, US Census

Amazon S3, Code Deployment, Code Function

Amazon S3: Amazon S3 (Simple Storage Service) is used to host the static web resources required by the WildRydes platform. This includes HTML, CSS, JavaScript, and image files. These resources are delivered to users' browsers when they access the web application, ensuring fast and reliable content delivery. Hosts the static files required by the web application. Ensures fast and reliable delivery of web content to users.

Developer Code Deployment

The **developer code deployed to Amazon S3 includes:**

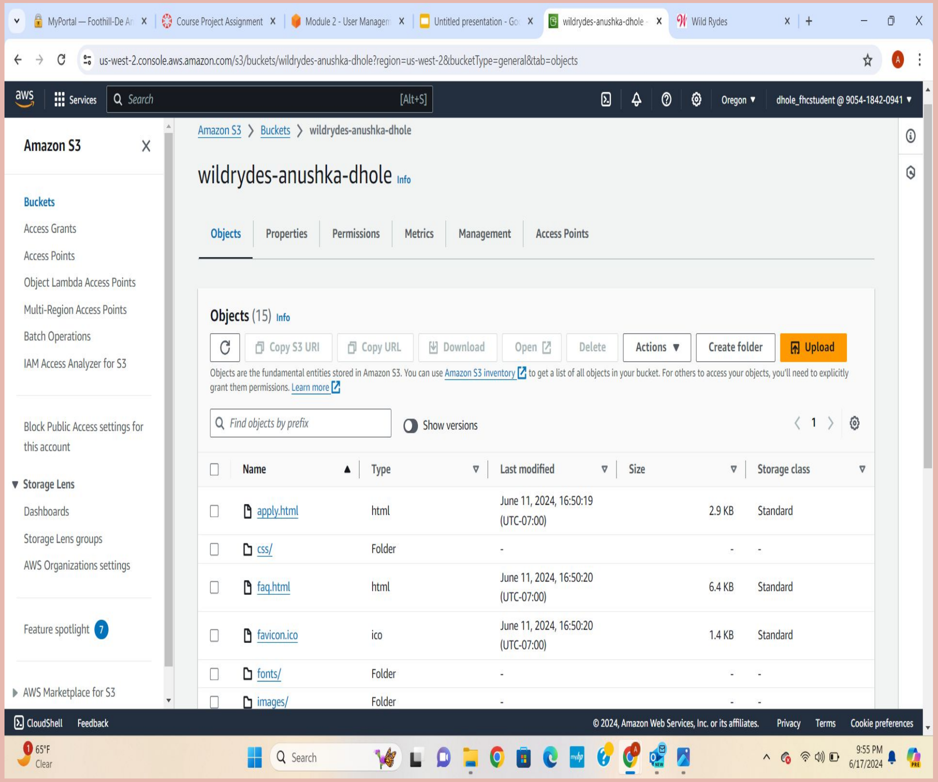
- **HTML files: Structure the web pages.**
- **CSS files: Style the web pages.**
- **JavaScript files: Provide interactivity and handle API calls**

  to the backend.

- **Images and other static assets: Enhance the visual aspects of the**
- **web application.**

These files are uploaded to an S3 bucket configured for static

website hosting, allowing users to access them via public URLs.



us-west-2.console.aws.amazon.com/s3/buckets/wildrydes-anushka-dhole?region=us-west-2&bucketType=general&tab=objects

Amazon S3 > Buckets > wildrydes-anushka-dhole

wildrydes-anushka-dhole Info

Objects (15) Info

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Last modified	Size	Storage class
apply.html	html	June 11, 2024, 16:50:19 (UTC-07:00)	2.9 KB	Standard
css/	Folder	-	-	-
faq.html	html	June 11, 2024, 16:50:20 (UTC-07:00)	6.4 KB	Standard
favicon.ico	ico	June 11, 2024, 16:50:20 (UTC-07:00)	1.4 KB	Standard
fonts/	Folder	-	-	-
images/	Folder	-	-	-

Code Functionality Amazon S3

Upload Files:

- Developers upload the HTML, CSS, JavaScript, and image files to an S3 bucket using the AWS Management Console, AWS CLI, or SDKs.
- Files are organized in a manner that mirrors the website structure

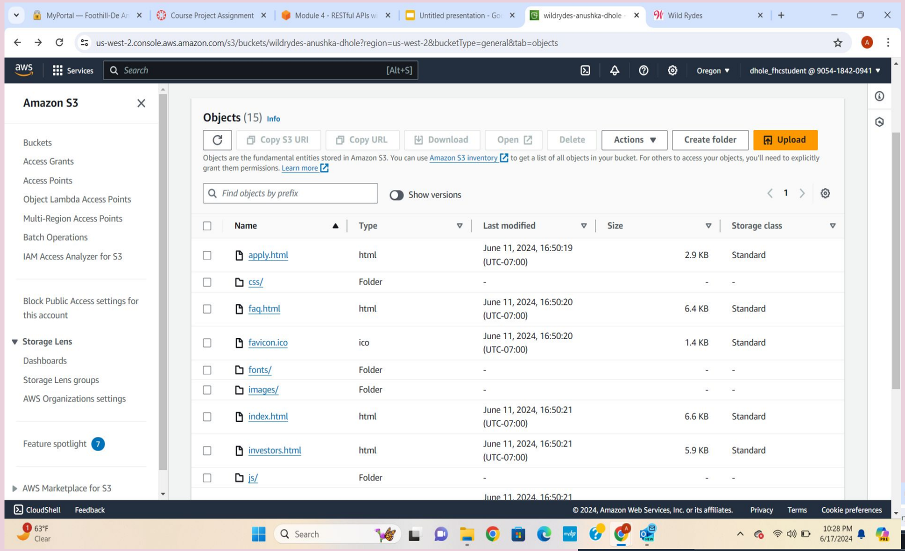
Set Permissions:

a. The S3 bucket is configured to allow public read access to the files, making them accessible to users.
b. Access control can be applied using S3 bucket policies and object ACLs.

Configure Static Website Hosting:

c. The S3 bucket is set up to host a static website, specifying the index document (e.g., `index.html`) and error document (e.g., `error.html`).

Screenshots of Amazon S3 Contents


us-west-2.console.aws.amazon.com/s3/buckets/wildrydes-anushka-dhole?region=us-west-2&bucketType=general&tab=objects
Amazon S3
Buckets
Access Grants
Access Points
Object Lambda Access Points
Multi-Region Access Points
Batch Operations
IAM Access Analyzer for S3
Block Public Access settings for this account
Storage Lens
Dashboards
Storage Lens groups
AWS Organizations settings
Feature spotlight 7
AWS Marketplace for S3
Copy S3 URI
Copy URL
Download
Open
Delete
Actions
Create folder
Upload
Find objects by prefix
Show versions

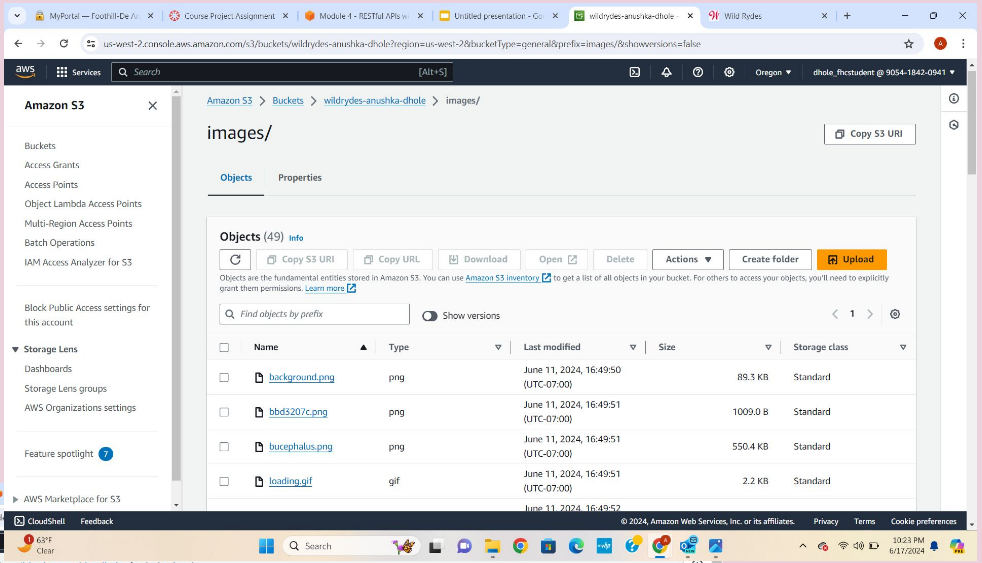
Objects (15) Info

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Last modified	Size	Storage class
apply.html	html	June 11, 2024, 16:50:19 (UTC-07:00)	2.9 KB	Standard
css/	Folder	-	-	-
faq.html	html	June 11, 2024, 16:50:20 (UTC-07:00)	6.4 KB	Standard
favicon.ico	ico	June 11, 2024, 16:50:20 (UTC-07:00)	1.4 KB	Standard
fonts/	Folder	-	-	-
images/	Folder	-	-	-
index.html	html	June 11, 2024, 16:50:21 (UTC-07:00)	6.6 KB	Standard
investors.html	html	June 11, 2024, 16:50:21 (UTC-07:00)	5.9 KB	Standard
js/	Folder	-	-	-

© 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

10:28 PM 6/17/2024


us-west-2.console.aws.amazon.com/s3/buckets/wildrydes-anushka-dhole?region=us-west-2&bucketType=general&prefix=images/&showversions=false
Amazon S3 > Buckets > wildrydes-anushka-dhole > images/
Copy S3 URI
Objects
Properties
Find objects by prefix
Show versions

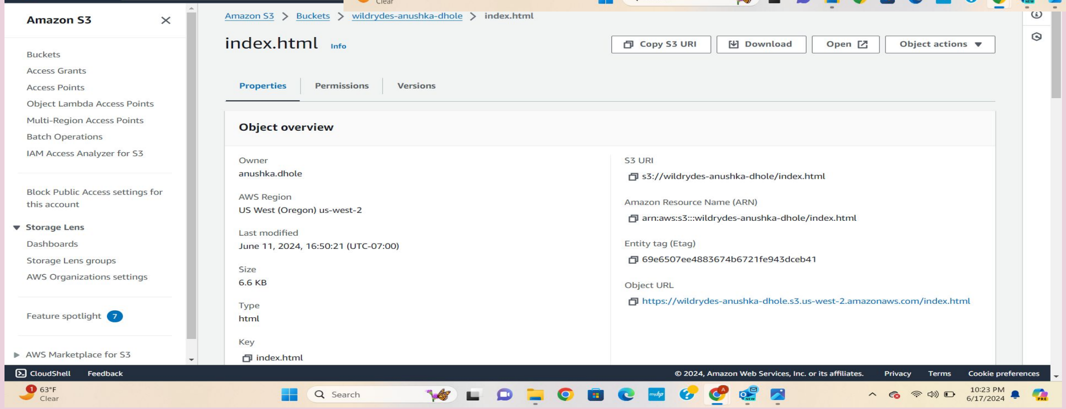
images/

Objects (49) Info

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Last modified	Size	Storage class
background.png	png	June 11, 2024, 16:49:50 (UTC-07:00)	89.3 KB	Standard
bbd3207c.png	png	June 11, 2024, 16:49:51 (UTC-07:00)	1009.0 B	Standard
bucephalus.png	png	June 11, 2024, 16:49:51 (UTC-07:00)	550.4 KB	Standard
loading.gif	gif	June 11, 2024, 16:49:51 (UTC-07:00)	2.2 KB	Standard

10:23 PM 6/17/2024



index.html Info

Object overview

Owner: anushka.dhole

AWS Region: US West (Oregon) us-west-2

Last modified: June 11, 2024, 16:50:21 (UTC-07:00)

Size: 6.6 KB

Type: html

Key: index.html

S3 URI: s3://wildrydes-anushka-dhole/index.html

Amazon Resource Name (ARN): arn:aws:s3:::wildrydes-anushka-dhole/index.html

Entity tag (Etag): 69e6507ee4883674b6721fe943dceb41

Object URL: https://wildrydes-anushka-dhole.s3.us-west-2.amazonaws.com/index.html

10:23 PM 6/17/2024

Amazon Lambda Role & Code Function

1. **Processing Ride Requests**:
    - When a user requests a unicorn ride through the web application, the request is routed through API Gateway to an AWS Lambda function.
    - The Lambda function executes the logic for handling the request, including user authentication and unicorn selection.
2. **Database Interactions**:
    - Lambda functions interact with Amazon DynamoDB to store and retrieve ride information.
    - The functions ensure that each ride request is logged with details such as the ride ID, user information, unicorn details, and timestamp.
3. **Integration with Other Services**:
    - Lambda integrates with Amazon Cognito for user authentication and authorization.
    - It uses AWS SDKs to interact with various AWS services, ensuring seamless operation of the backend logic.

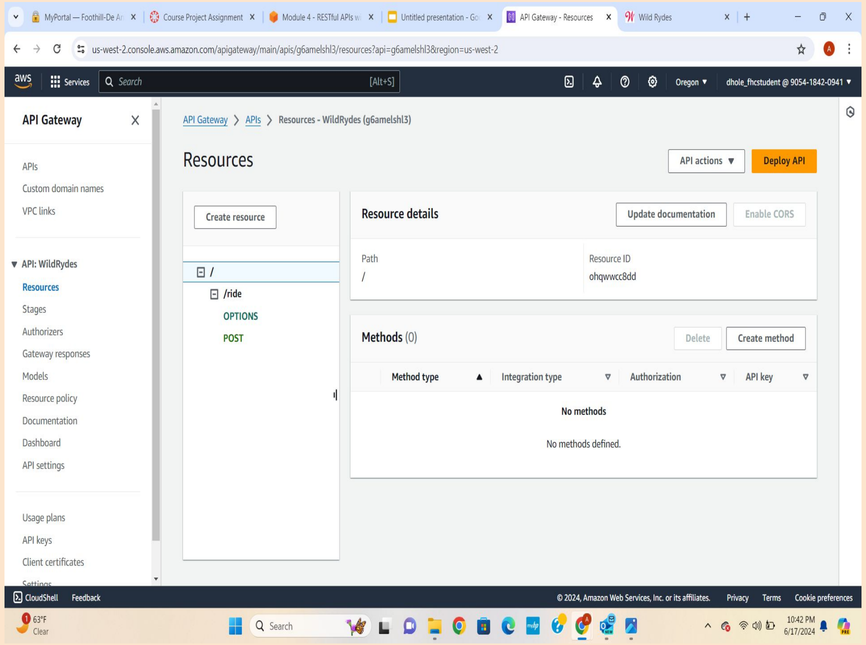
The developer code deployed in the Lambda functions performs several key tasks:

1. **Authorization Check**:
    - Ensures that requests are authorized by checking for valid authentication tokens provided by Amazon Cognito.
2. **Ride Request Handling**:
    - Generates a unique ride ID for each request.
    - Parses the incoming request to extract user and pickup location details.
    - Selects a unicorn from a predefined fleet based on the provided location.
3. **Recording the Ride**:
    - Stores the ride details in the DynamoDB table to maintain a record of all ride requests.

API GATEWAY Role

Amazon API Gateway:

- Provides a secure interface for frontend applications to communicate with the backend.
- Handles request routing, authentication, and response formatting.
- Routes HTTP requests from the frontend to the appropriate Lambda functions.
- Acts as the mediator between the client-side JavaScript and the serverless backend.



us-west-2.console.aws.amazon.com/apigateway/main/apis/g6amelshl3/resources?api=g6amelshl3&region=us-west-2

API Gateway > APIs > Resources - WildRydes (g6amelshl3)

Resources

Resource details

Path	Resource ID
/	ohqwwcc8dd

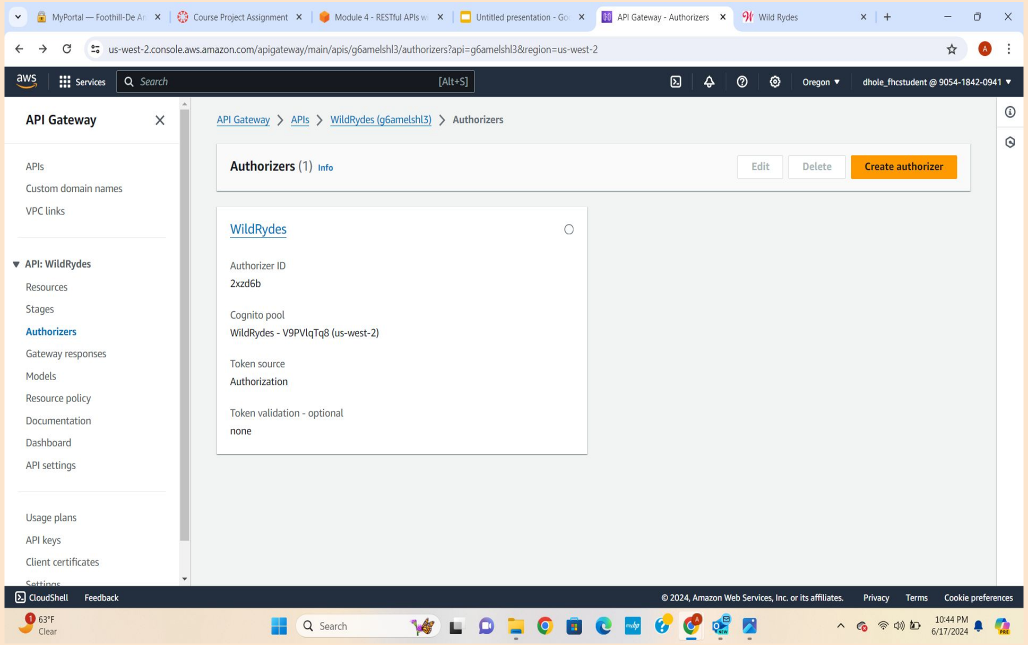
- /
  - /ride
    - OPTIONS
    - POST

Methods (0)

Method type	Integration type	Authorization	API key

No methods

No methods defined.



us-west-2.console.aws.amazon.com/apigateway/main/apis/g6amelshl3/authorizers?api=g6amelshl3&region=us-west-2

API Gateway > APIs > WildRydes (g6amelshl3) > Authorizers

Authorizers (1) Info

WildRydes

Authorizer ID
2xzd6b

Cognito pool
WildRydes - V9PVlqTq8 (us-west-2)

Token source
Authorization

Token validation - optional
none

DynamoDB Table Entries

The Amazon DynamoDB table used in the WildRydes platform stores ride details. Each entry in the table represents a ride requested by a user. The key attributes stored in the table include:

1. **RideId**: A unique identifier for each ride.
2. **User**: The username of the rider.
3. **Unicorn**: Details of the assigned unicorn (name, color, and gender).
4. **RequestTime**: The timestamp when the ride request was made

RideId	User	Unicorn	RequestTime
SvLnijIAtg6inAFUBRT+Fg==	john_doe	{"Name": "Rocinante", "Color": "Yellow", "Gender": "Female"}	2023-06-15T12:34:56Z
ASDbnkjASF_!lkadsfj==	jane_smith	{"Name": "Bucephalus", "Color": "Golden", "Gender": "Male"}	2023-06-15T13:45:12Z
qwe123xZcvn_dsfkj==-	alice_jones	{"Name": "Shadowfax", "Color": "White", "Gender": "Male"}	2023-06-16T09:22:33Z

Use of DynamoDB Entries

Tracking Ride Requests:

- **Each ride request is uniquely identified and tracked using the `RideId`.**
- **Details such as the user who requested the ride and the assigned unicorn are stored for reference and analytics.**

User Ride History:

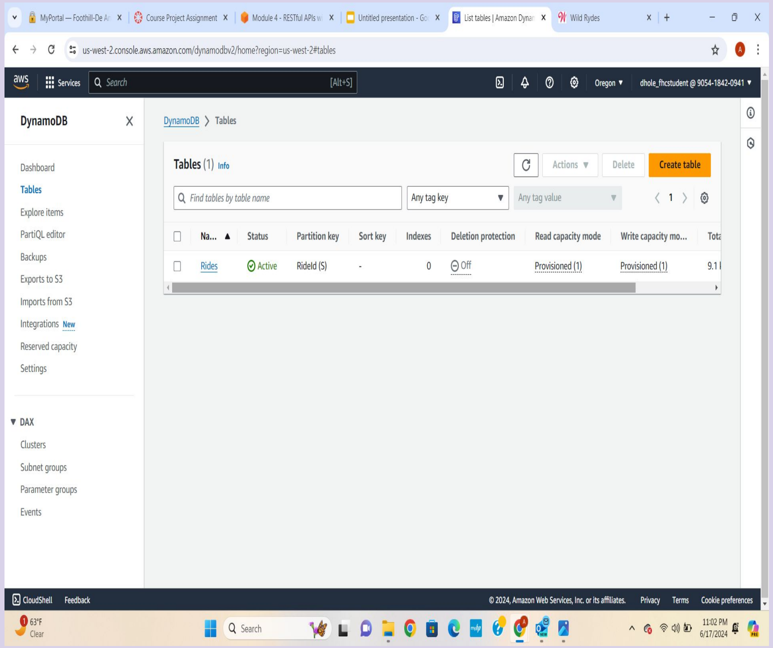
- **Users can view their ride history by querying the DynamoDB table using their username (`User` attribute).**
- **This provides a personalized experience where users can see their past rides and details.**

Analytics and Reporting:

- **The data can be used to generate reports on ride requests, user activity, and unicorn assignment.**
- **This helps in understanding user behavior and preferences, which can inform future business decisions.**

Real-Time Updates and Notifications:

- **The `RequestTime` attribute helps in calculating the Estimated Time of Arrival (ETA) for rides.**
- **Real-time notifications can be sent to users about their ride status based on this data.**



us-west-2.console.aws.amazon.com/dynamodbv2/home?region=us-west-2#tables

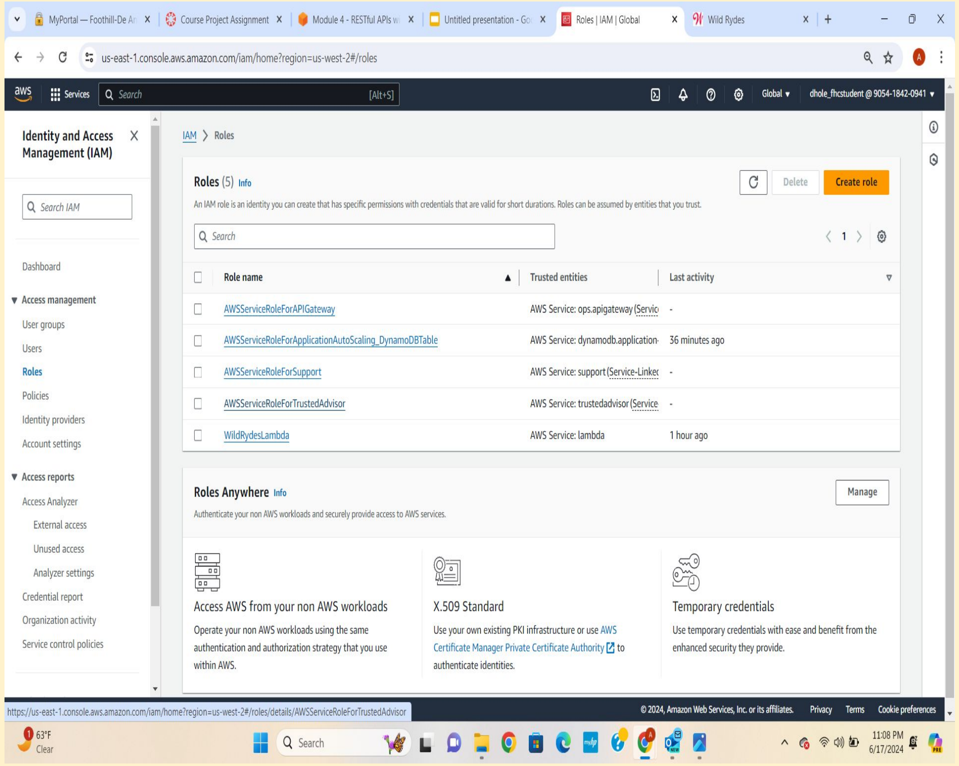
DynamoDB > Tables

Tables (1) Info

Na... ▲	Status	Partition key	Sort key	Indexes	Deletion protection	Read capacity mode	Write capacity mo...	Tota
Rides	✓ Active	RideId (S)	-	0	Off	Provisioned (1)	Provisioned (1)	9.1

IAM Roles for Lambda Function

IAM Roles - Security and Permissions - IAM roles are assigned to Lambda functions to grant them the necessary permissions to interact with other AWS services such as DynamoDB and Cognito. This ensures that each function has the minimum necessary permissions, enhancing the security of the application.



us-east-1.console.aws.amazon.com/iam/home?region=us-west-2#/roles

IAM > Roles

Roles (5) Info

An IAM role is an identity you can create that has specific permissions with credentials that are valid for short durations. Roles can be assumed by entities that you trust.

Role name	Trusted entities	Last activity
AWSServiceRoleForAPIGateway	AWS Service: ops.apigateway (Servic	-
AWSServiceRoleForApplicationAutoScaling_DynamoDBTable	AWS Service: dynamodb.application-	36 minutes ago
AWSServiceRoleForSupport	AWS Service: support (Service-Linke	-
AWSServiceRoleForTrustedAdvisor	AWS Service: trustedadvisor (Service	-
WildRydesLambda	AWS Service: lambda	1 hour ago

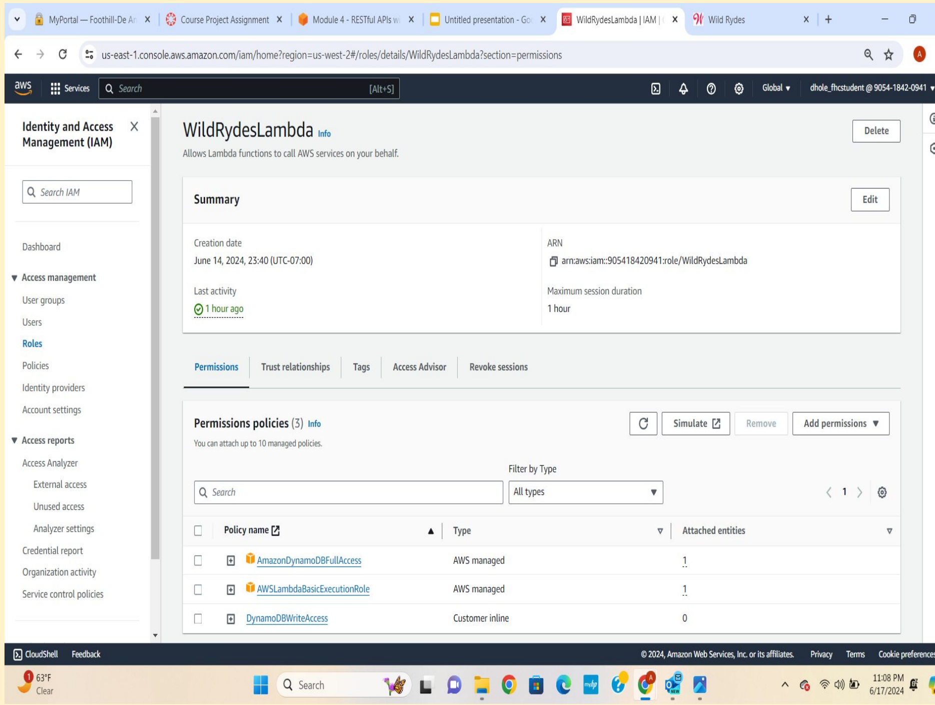
Roles Anywhere Info

Authenticate your non AWS workloads and securely provide access to AWS services.

Access AWS from your non AWS workloads — Operate your non AWS workloads using the same authentication and authorization strategy that you use within AWS.

X.509 Standard — Use your own existing PKI infrastructure or use AWS Certificate Manager Private Certificate Authority to authenticate identities.

Temporary credentials — Use temporary credentials with ease and benefit from the enhanced security they provide.



us-east-1.console.aws.amazon.com/iam/home?region=us-west-2#/roles/details/WildRydesLambda?section=permissions

WildRydesLambda Info

Allows Lambda functions to call AWS services on your behalf.

Summary

Creation date	June 14, 2024, 23:40 (UTC-07:00)
ARN	arn:aws:iam::905418420941:role/WildRydesLambda
Last activity	1 hour ago
Maximum session duration	1 hour

Permissions policies (3) Info

You can attach up to 10 managed policies.

Policy name	Type	Attached entities
AmazonDynamoDBFullAccess	AWS managed	1
AWSLambdaBasicExecutionRole	AWS managed	1
DynamoDBWriteAccess	Customer inline	0

Deployment and Design Choices

1. **Bucket Configuration**:
   - **Choice**: Enabled versioning for the S3 bucket.
   - **Reason**: Versioning allows for easier rollback to previous versions in case of accidental overwrites or deletions, improving data safety.
2. **Static Website Hosting**:
   - **Choice**: Configured the bucket to host a static website with a custom error document.
   - **Reason**: This allows users to have a better user experience with friendly error pages instead of generic S3 errors, improving user experience.
3. **Permissions**:
   - **Choice**: Implemented fine-grained access controls using bucket policies and object ACLs.
   - **Reason**: Ensuring that only the necessary permissions are granted enhances security, preventing unauthorized access while allowing public read access to the static files.

Amazon Cognito for User Management

1. **User Pool Configuration**:
   - **Choice**: Enabled multi-factor authentication (MFA) and email verification for user sign-ups.
   - **Reason**: Enhances security by adding additional layers of authentication, reducing the risk of unauthorized access.
2. **Custom Attributes**:
   - **Choice**: Added custom attributes to the user pool for storing additional user information.
   - **Reason**: Allows for more detailed user profiles and better personalization of services.

1.

Deployment choices

Amazon API Gateway for RESTful API

1. **Endpoint Type**:
   - **Choice**: Chose Regional endpoints for the API Gateway.
   - **Reason**: Regional endpoints reduce latency by ensuring that the API requests are served from the nearest AWS region, improving performance for users in that region.

Amazon DynamoDB for Persistence

1. **Table Design**:
   - **Choice**: Designed the DynamoDB table with a composite primary key (partition key and sort key).
   - **Reason**: Improves query efficiency by allowing more complex queries and better-organized data storage.

Testing and Validation

Unit Testing:

- Validated Lambda functions using sample events to ensure correct processing.
- Tested API Gateway endpoints for proper routing and response formatting.

Integration Testing:

- Ensured end-to-end functionality from user sign-in to ride request handling.

Amazon Cognito User Management

Test: User Registration and Authentication

- **Objective**: Verify that users can register and authenticate using Amazon Cognito.
- **Procedure**:
  - Register a new user using the signup form.
  - Verify email confirmation.
  - Login using the registered credentials.
- **Validation**: Ensure that the user can successfully register, confirm their email, and log in.

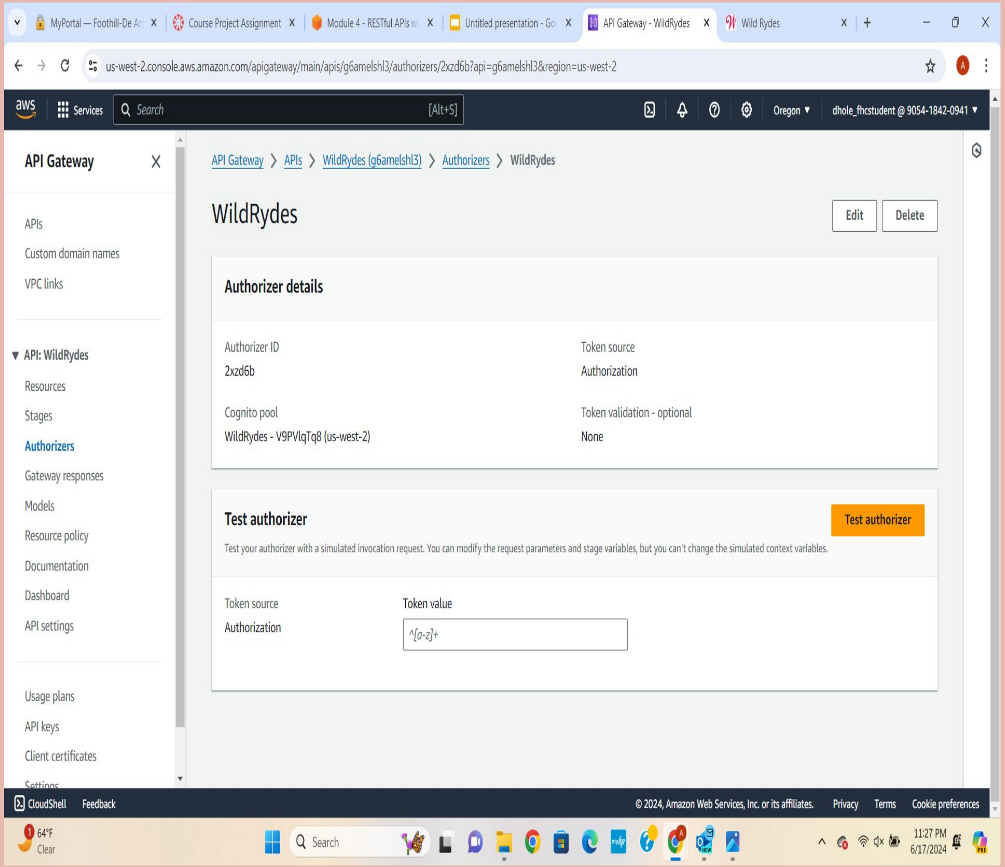
S3 Test & Token Authorization Test

Test: Accessibility and Content Delivery

- **Objective**: Verify that the static web resources (HTML, CSS, JavaScript, and images) are accessible via the configured S3 bucket URL.
- **Procedure**:
  - Navigate to the S3 bucket URL in a web browser.
  - Check the loading of various resources (e.g., signin.html and ride.html).
- **Validation**: Ensure that all static resources load correctly without errors.

Token Authorization Test Passes

Test Screenshots



us-west-2.console.aws.amazon.com/apigateway/main/apis/g6amelshl3/authorizers/2xzd6b?api=g6amelshl3&region=us-west-2

API Gateway > APIs > WildRydes (g6amelshl3) > Authorizers > WildRydes

WildRydes

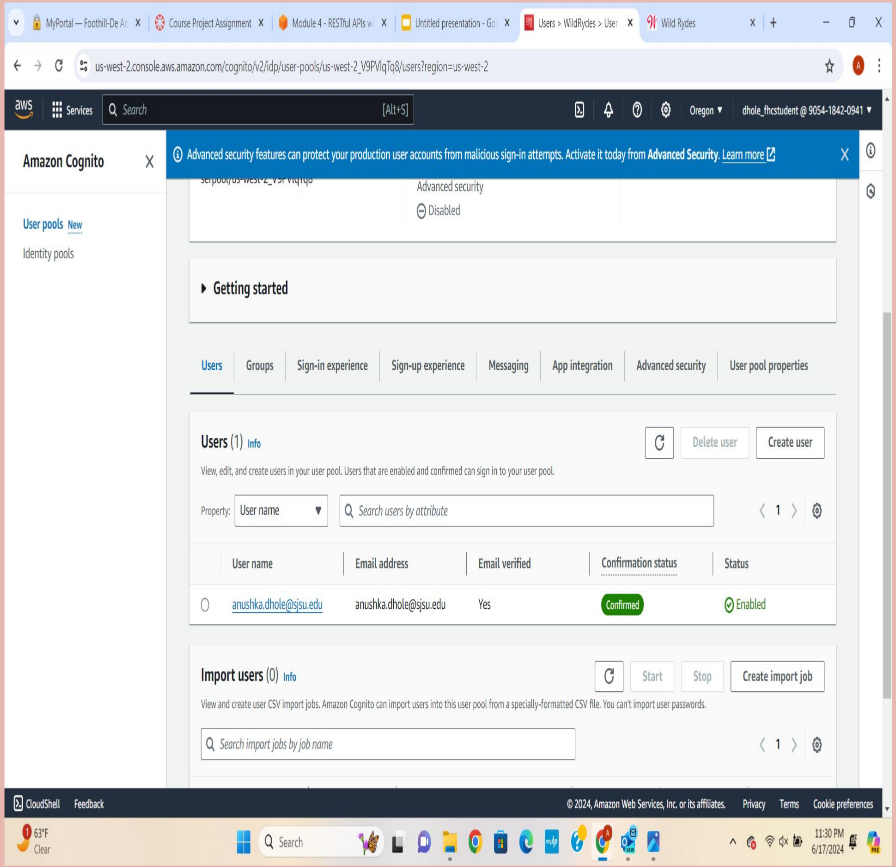
Authorizer details

Authorizer ID	Token source
2xzd6b	Authorization
Cognito pool	Token validation - optional
WildRydes - V9PVlqTq8 (us-west-2)	None

Test authorizer

Test your authorizer with a simulated invocation request. You can modify the request parameters and stage variables, but you can't change the simulated context variables.

Token source	Token value
Authorization	^[a-z]+



us-west-2.console.aws.amazon.com/cognito/v2/idp/user-pools/us-west-2_V9PVlqTq8/users?region=us-west-2

Advanced security features can protect your production user accounts from malicious sign-in attempts. Activate it today from Advanced Security. Learn more

Advanced security: Disabled

Users (1)

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

User name	Email address	Email verified	Confirmation status	Status
anushka.dhole@sjsu.edu	anushka.dhole@sjsu.edu	Yes	Confirmed	Enabled

Import users (0)

View and create user CSV import jobs. Amazon Cognito can import users into this user pool from a specially-formatted CSV file. You can't import user passwords.

Cost Analysis & Assumptions

Using the AWS Pricing Calculator:

- **Amazon S3**: Charges for storage and data transfer.
- **Amazon Cognito**: Charges for MAUs (Monthly Active Users).
- **AWS Lambda**: Charges based on the number of requests and execution time.
- **API Gateway**: Charges for API calls and data transfer.
- **DynamoDB**: Charges for storage, read/write throughput, and data transfer.

Cost Analysis Table Assumptions

Service	Monthly Usage	Estimated Cost USD
Amazon S3	50GB Storage	$1.25
Cognito	10M MAUS	$25000
Lambda	100M Requests	$10000
API Gateway	100M Requests	$35000
DynamoDB	1TB Storage	$256.80

Total Estimated Monthly Cost: $71,256.80