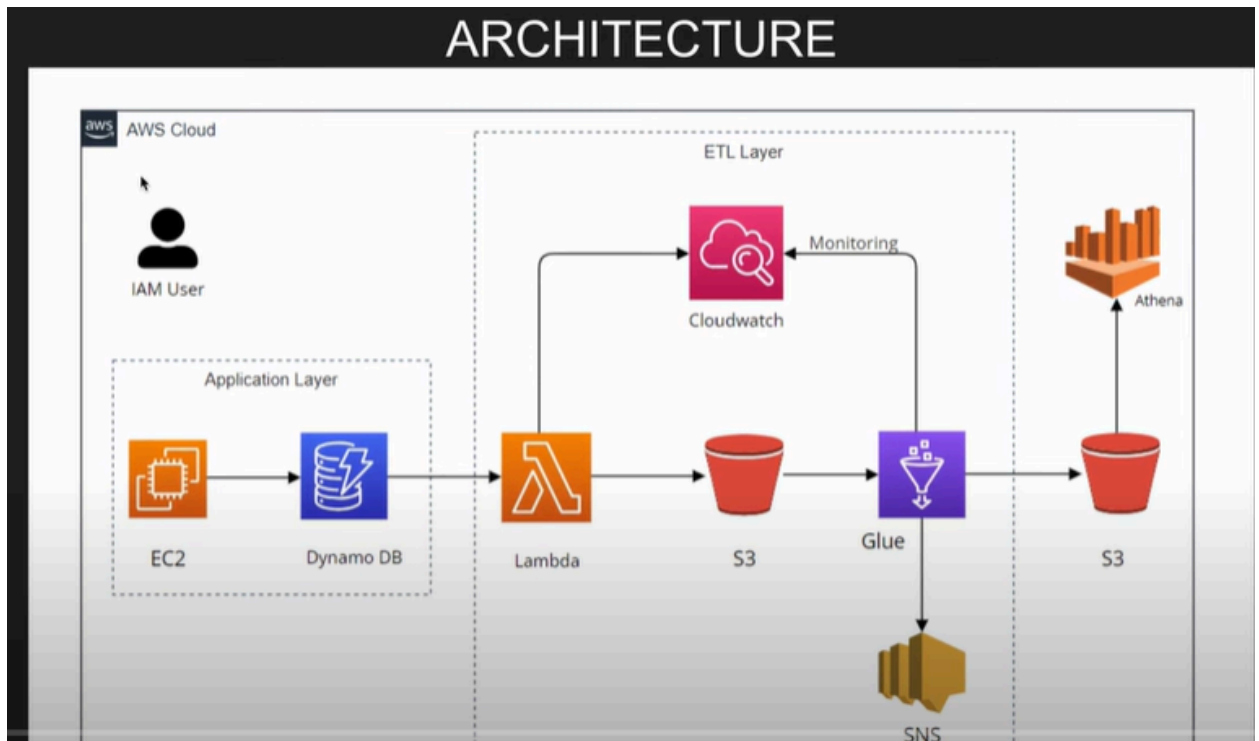


Architecture



- 1.
2. First here we are going to create an IAM user...using these user...we'll host our flask application
3. And all the data that is passed by the user..will be stored in Dynamo DB
4. Next we'll transfer our data to staging layer...using AWS lambda
5. Next we'll take help of AWS glue to transfer data from staging layer to datalake(AWS s3)
6. Once the data in the datalake...we can athena to query the data
7. We'll monitor our project using cloudwatchand whenever glue fails or get success ...we'll receive notification from AWS SNS
8. So this is our overall project

FORM

Reg No

Name

Class

Math

English

Science

Computer

Submit

student	marks
id integer	student_id integer
name varchar	subject varchar
class varchar	math integer
created_at timestamp	english integer
	science integer
	computer integer
	created_at timestamp

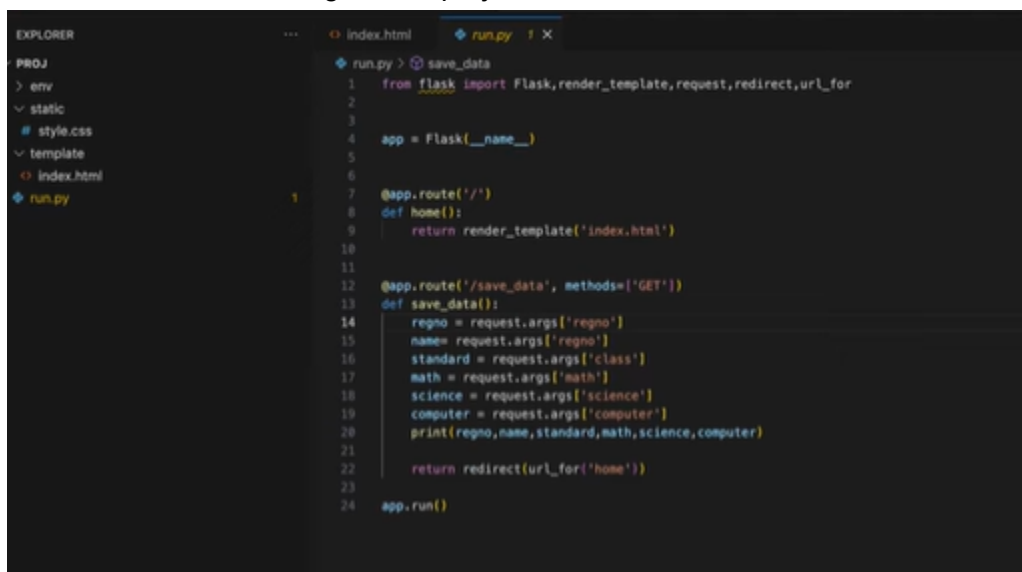
9. Here we'll be building flask application...and when we fill the form and submit it...the data will be stored in DynamoDB

student	marks
id integer	student_id integer
name varchar	subject varchar
class varchar	math integer
created_at timestamp	english integer
	science integer
	computer integer
	created_at timestamp

10. We have two tables here

Implementation of Project

1. Here first we'll be creating a flask project



```
run.py > save_data
1  from flask import Flask, render_template, request, redirect, url_for
2
3
4  app = Flask(__name__)
5
6
7  @app.route('/')
8  def home():
9      return render_template('index.html')
10
11
12  @app.route('/save_data', methods=['GET'])
13  def save_data():
14      regno = request.args['regno']
15      name = request.args['name']
16      standard = request.args['class']
17      math = request.args['math']
18      science = request.args['science']
19      computer = request.args['computer']
20      print(regno, name, standard, math, science, computer)
21
22      return redirect(url_for('home'))
23
24  app.run()
```

Creating IAM user



1. Here we'll be doing this
2. Always give only required permission to an IAM user
3. So to create IAM user..

4. When ever we have a large group of team...then we can make use of users group and assign the permission to that group
5. For this proj..we are only creating one user..

IAM > Users > Create user

Step 1
Specify user details

Step 2
Set permissions

Step 3
Review and create

Specify user details

User details

User name
project-iam

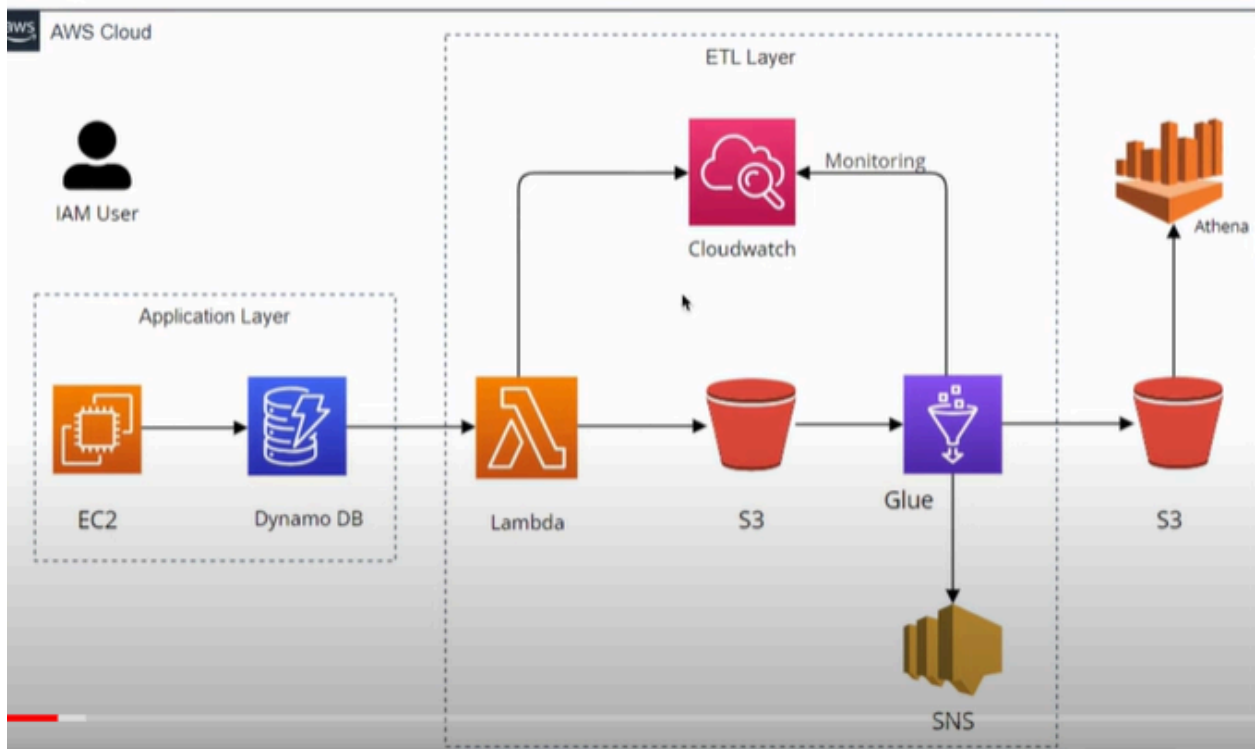
The user name can have up to 64 characters. Valid characters: A-Z, a-z, 0-9, and + , . @ _ - (hyphen)

☐ Provide user access to the AWS Management Console - optional
If you're providing console access to a person, it's a [best practice](#) to manage their access in IAM Identity Center.

Info If you are creating programmatic access through access keys or service-specific credentials for AWS CodeCommit or Amazon Keyspaces, you can generate them after you create this IAM user. [Learn more](#)

Cancel **Next**

6. By default an IAM user will get no permissions
7. We need to give all these permissions to the user



8. We have added all the required permissions for our user

9. We can also set max number of permission for an user using

▼ **Set permissions boundary - optional**

Set a permissions boundary to control the maximum permissions for this user. Use this advanced feature used to delegate permission management to others. [Learn more](#)

☐ Use a permissions boundary to control the maximum permissions
You can select one of the existing permissions policies to define the boundary.

10. Here we have attached all the permissions for our user

Permissions summary			
Name	Type	Used as	
AmazonAthenaFullAccess	AWS managed	Permissions policy	
AmazonCloudWatchEvidentlyFullAccess	AWS managed	Permissions policy	
AmazonDynamoDBFullAccess	AWS managed	Permissions policy	
AmazonEC2FullAccess	AWS managed	Permissions policy	
AmazonS3FullAccess	AWS managed	Permissions policy	
AmazonSNSFullAccess	AWS managed	Permissions policy	
AWSGlueConsoleFullAccess	AWS managed	Permissions policy	
AWSLambda_FullAccess	AWS managed	Permissions policy	
CloudWatchActionsEC2Access	AWS managed	Permissions policy	
IAMUserChangePassword	AWS managed	Permissions policy	

11. We have also created a tag

Key Value - optional

Q project X Q demo X Remove

Add new tag

Use "demo"

You can add up to 49 more tags.

and created this user

12. Now we'll login thru our IAM user

Hosting Flask Application in EC2

1. Here we'll create an ec2 instance with appropriate specifications

2. And we can connect to our ec2 instance using Ec2 Instance connect

Connect to instance [Info](#)

Connect to your instance i-036f4c4d38837ec9c (project-de) using any of these options

EC2 Instance Connect | Session Manager | SSH client | EC2 serial console

Instance ID
i-036f4c4d38837ec9c (project-de)

Connection Type

☒ **Connect using EC2 Instance Connect**
Connect using the EC2 Instance Connect browser-based client, with a public IPv4 address.

☐ **Connect using EC2 Instance Connect Endpoint**
Connect using the EC2 Instance Connect browser-based client, with a private IPv4 address and a VPC endpoint.

Public IP address
13.126.134.132

Username
Enter the username defined in the AMI used to launch the instance. If you didn't define a custom username, use the default username, ubuntu.

ubuntu

Note: In most cases, the default username, ubuntu, is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username.

Cancel **Connect**

3. Next we will update the package manager and install required things(Venv,etc) to run our flask application ...we install all the packages inside the env

```
mkdir project
cd project
python3 -m venv venv
source venv/bin/activate
```

Here, you create a project directory and set up a virtual environment within it. Activating the virtual environment isolates your project's dependencies, preventing conflicts with other Python projects on the same machine.

Step 3: Install Flask

```
pip install flask
```

This installs the Flask framework within your virtual environment, allowing you to develop web applications using Python.

Step 4: Create a Simple Flask API (Clone Github repo)

```
git clone <link>
```

You clone your Flask application's code from a GitHub repository. This step assumes you have your Flask application hosted on a Git repository.

```
python run.py
```

4. Next we'll clone our flask project into our ec2 instance
5. Now if we go to the ipv4 address of our ec2 instance..then we can our form displayed

Creating Dynamo DB and S3 Bucket

1.

CREATE DYNAMODB TABLE

- Log in as IAM User
- Create a table
- Partition and Sort Key
- Configure Dynamodb Stream

S3 BUCKET

- Log in as IAM User
- Create staging and dw layer

2. Here we'll be having two tables (data is coming from our form)

FORM

Reg No

Name

Class

Math

English

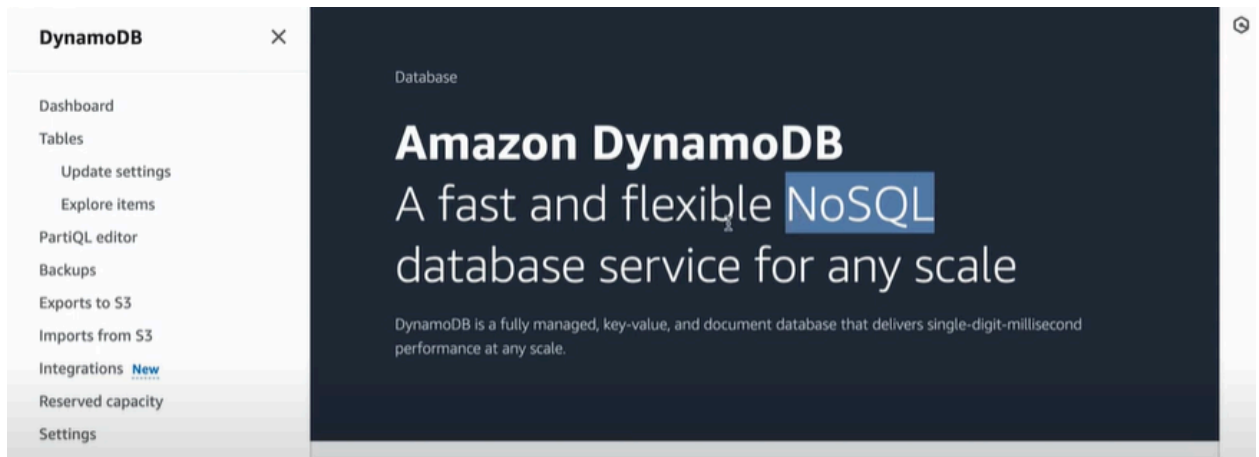
Science

Computer

Submit

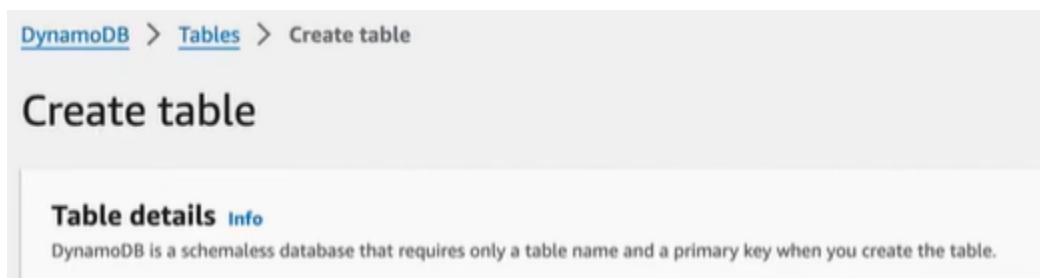
student		marks	
id	integer	student_id	integer
name	varchar	subject	varchar
class	varchar	math	integer
created_at	timestamp	english	integer
		science	integer
		computer	integer
		created_at	timestamp

3. Creating a dynamoDB



4. Here dynamoDB is NoSQL DB...so we dont need to give any predefined Schema..instead we'll be giving partition and key

5. Next we'll create a table called student



and define partition key(partition is used to distribute the data)..here id is our partition key

6. And we create this table

7. Lets create marks table now...here we have two primary keys

The screenshot shows the 'Create table' form in the Amazon DynamoDB console. It has three main sections: 1. 'Table name' with a text input containing 'marks' and a note: 'This will be used to identify your table. Between 3 and 255 characters, containing only letters, numbers, underscores (_), hyphens (-), and periods (.).'. 2. 'Partition key' with a text input containing 'student_id', a dropdown menu set to 'String', and a note: '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. 1 to 255 characters and case sensitive.' 3. 'Sort key - optional' with a text input containing 'subject', a dropdown menu set to 'String', and a note: '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. 1 to 255 characters and case sensitive.'

8. Default table settings are same for both tables

Default table settings		
These are the default settings for your new table. You can change some of these settings after creating the table.		
Setting	Value	Editable after creation
Table class	DynamoDB Standard	Yes
Capacity mode	Provisioned	Yes
Provisioned read capacity	5 RCU	Yes
Provisioned write capacity	5 WCU	Yes
Auto scaling	On	Yes
Local secondary indexes	-	No
Global secondary indexes	-	Yes
Encryption key management	Owned by Amazon DynamoDB	Yes
Deletion protection	Off	Yes

9. We define sample data into student table

[DynamoDB](#) > [Explore items: student](#) > Create item

Create 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](#)

Form JSON view

Attributes Add new attribute ▼

Attribute name	Value	Type	
id - Partition key	123	String	
name	Datewithdata	String	Remove
fathename	Deepak	String	Remove

Cancel Create item

10. Let's turn ON dynamoDB stream

[DynamoDB](#) > [Tables](#) > [student](#) > Turn on DynamoDB stream

Turn on DynamoDB stream

DynamoDB stream details

Capture item-level changes in your table, and push the changes to a DynamoDB stream. You then can access the change information through the DynamoDB Streams API.

View type
Choose which versions of the changed items you would like to push to the DynamoDB stream.

☐ **Key attributes only**
Only the key attributes of the changed item.

☐ **New image**
The entire item as it appears after it was changed.

☐ **Old image**
The entire item as it appears before it was changed.

☒ **New and old images**
Both the new and old images of the changed item.

Cancel Turn on stream

DynamoDB Streams act like a real-time log of changes made to your DynamoDB tables. Whenever you perform a write operation (PUT, UPDATE, or DELETE) on a table with DynamoDB Streams enabled, a corresponding stream record is created. This record captures details about the change, including:

- **Change type:** Indicates whether the item was inserted (INSERTED), updated (MODIFIED), or deleted (REMOVED).
- **Item keys:** The partition key and sort key (if used) that identify the modified item.
- **Old and new image (optional):** For updates, the stream record can contain both the old and new attribute values of the modified item. This is configurable.

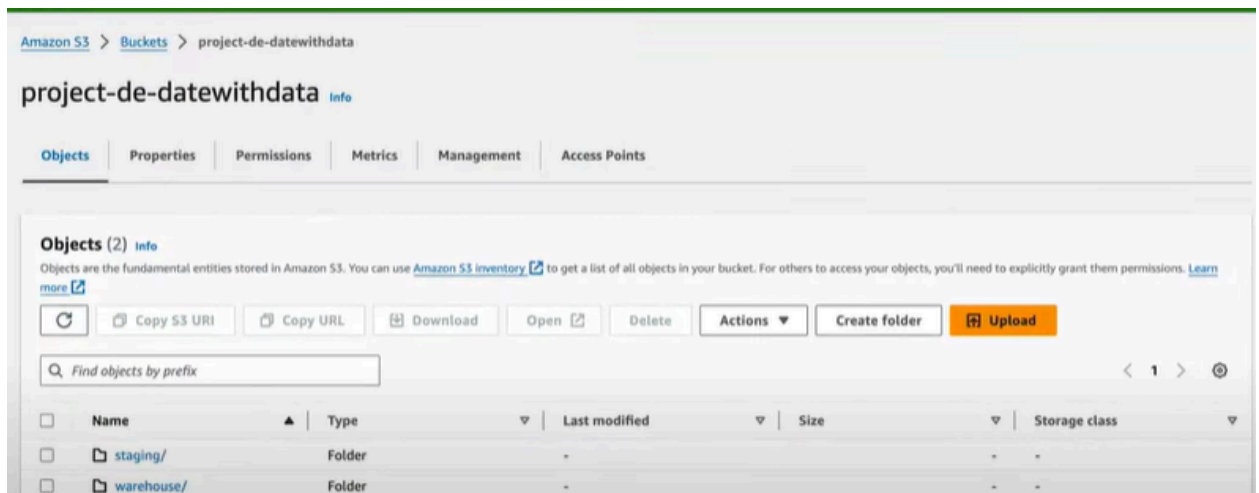
Benefits of DynamoDB Streams:

- **Near real-time updates:** Enables applications to react to data changes almost instantly.
- **Change Data Capture (CDC):** Provides a historical record of all modifications to your data.
- **Decoupled architecture:** Separates data storage from processing logic, allowing for scalable and flexible applications.

11. Here whenever a change to insert is done in DynamoDB..it should trigger the lambda

12. So after the lambda function is executed...it saves the data in the s3 bucket

13. First we'll be creating a S3 bucket..we'll create the bucket ..with the default configs..and we'll use these for both staging and warehouse



14. Here ..first our lambda sends the data to the staging ...and glue job will take the data from staging layer and insert it into warehouse

