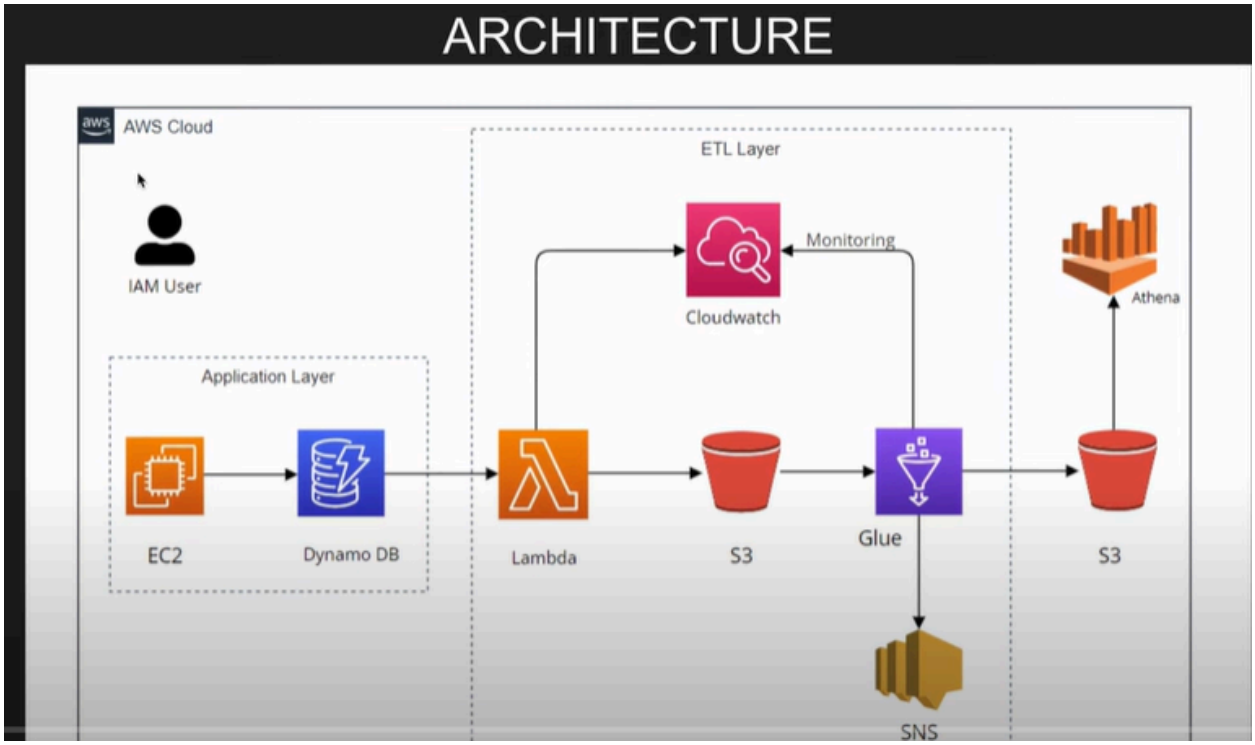
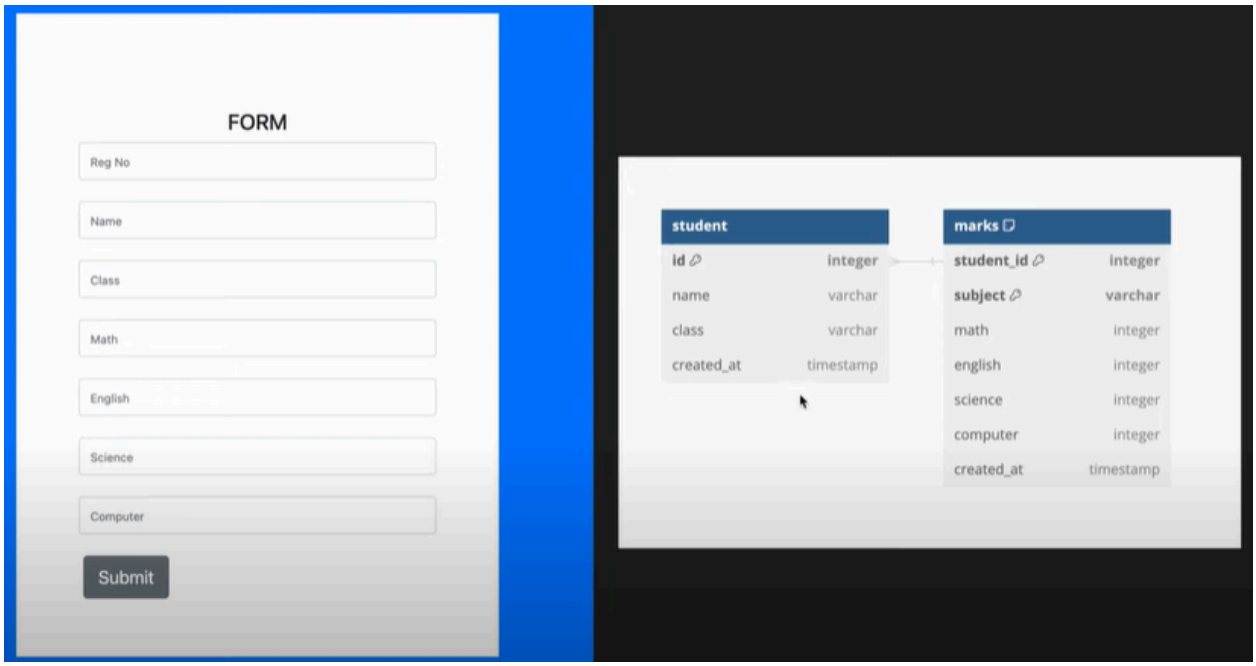
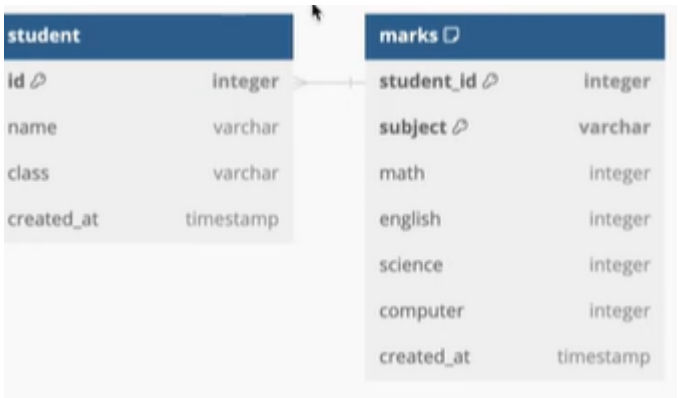
Architecture

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 cloudwatch ….and whenever glue fails or get success …we'll receive notification from AWS SNS
8. So this is our overall project

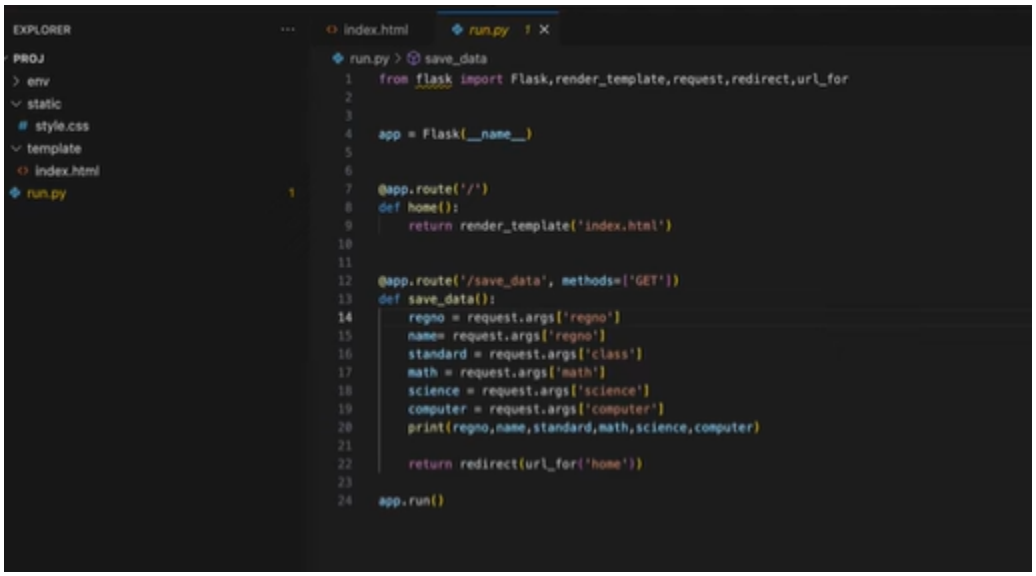
9. 

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

10. We have two tables here 

Implementation of Project

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

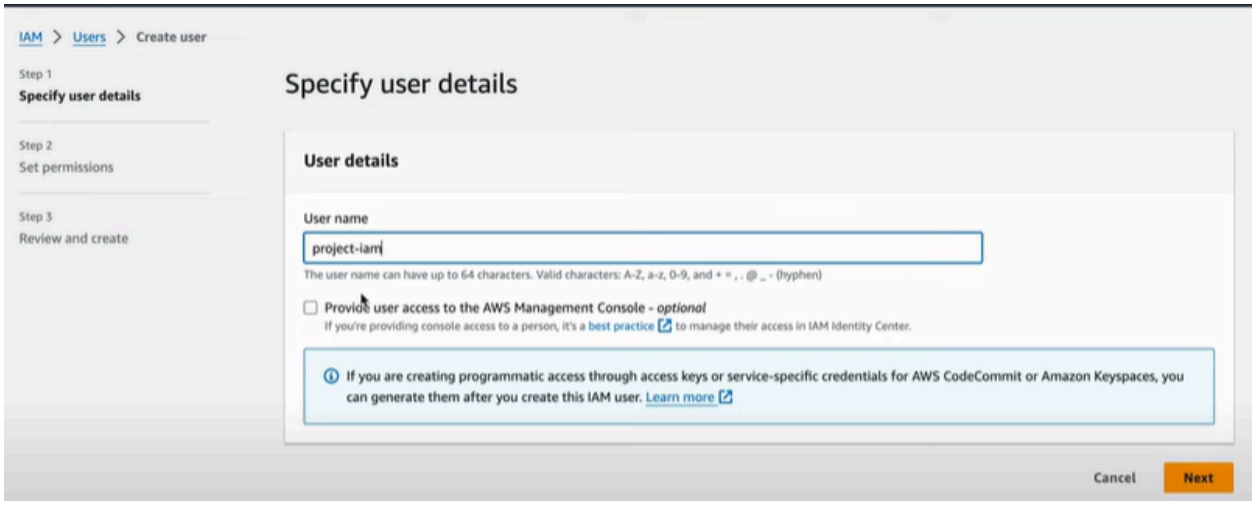


Creating IAM user

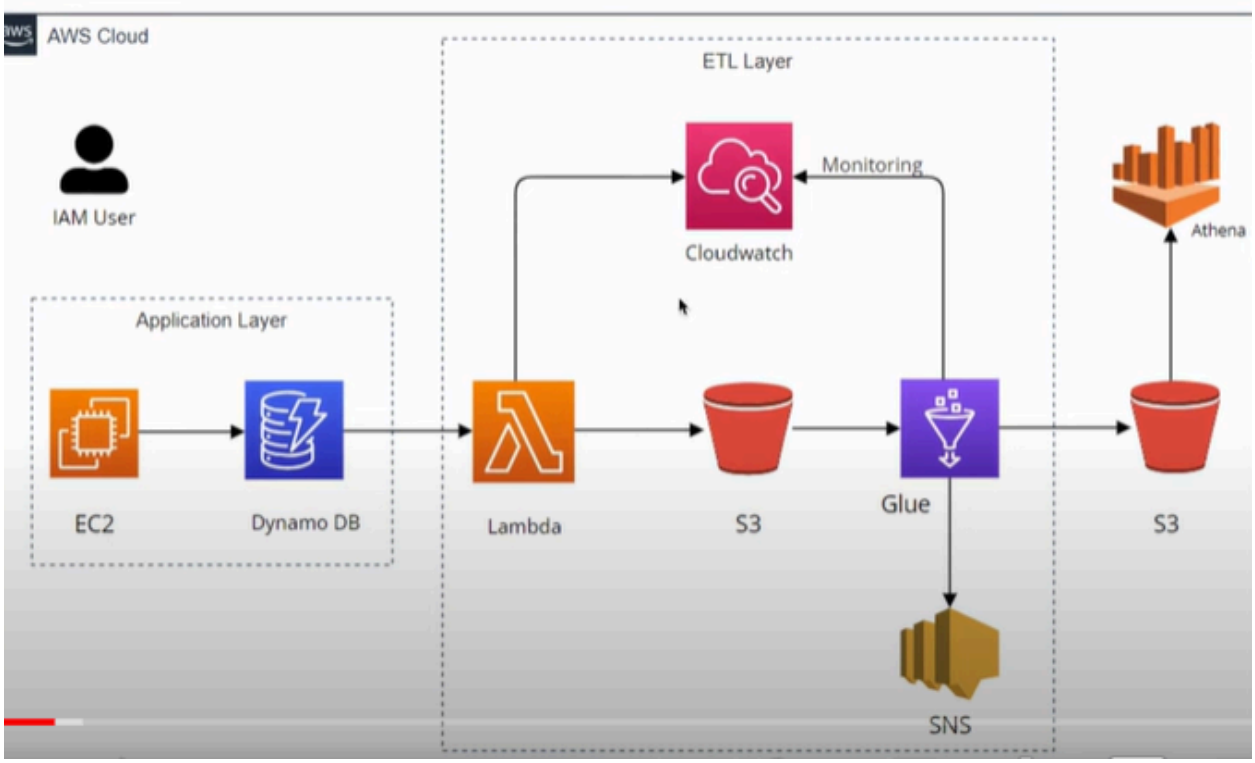
1. Here we'll be doing this 

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..

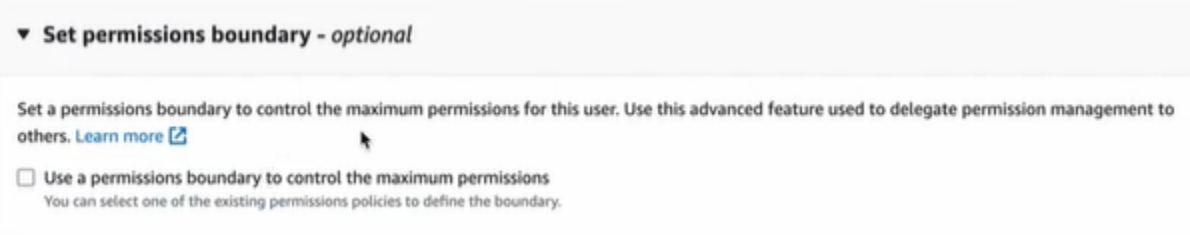


6. By default an IAM user will get no permissions
7. We need to give all these permissions to the user

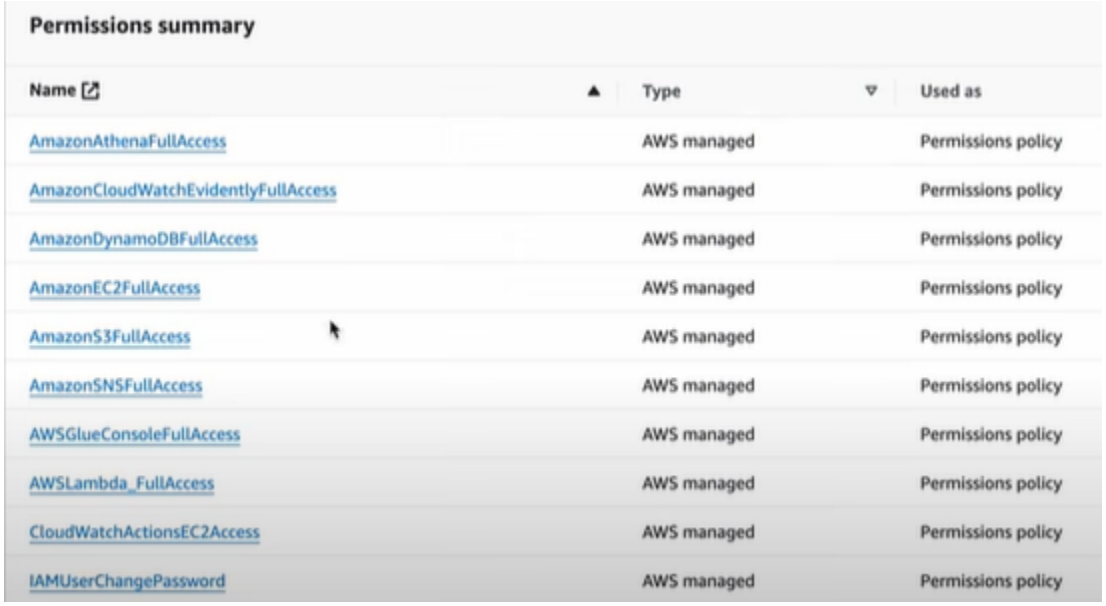


8. We have added all the required permissions for our user

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

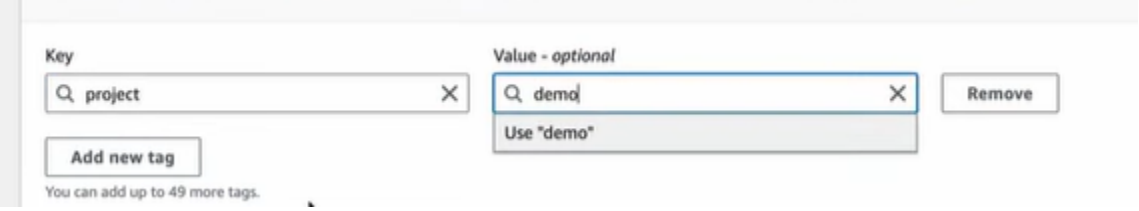


10. Here we have attached all the permissions for our user



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



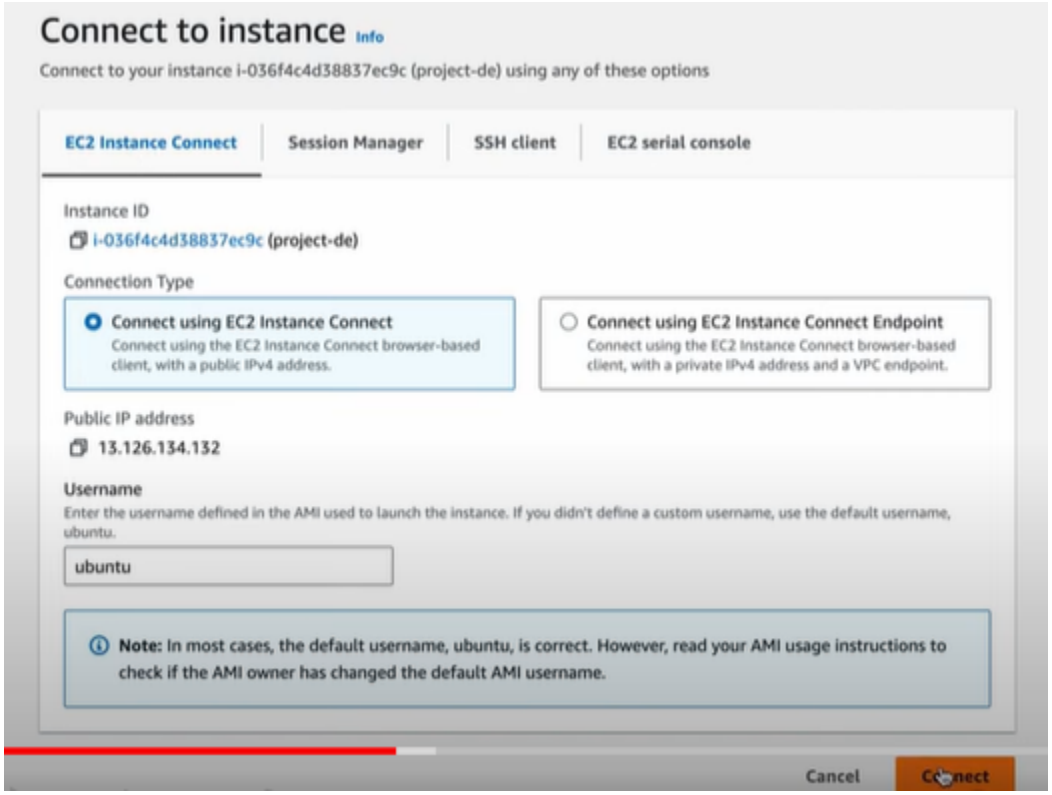
and created this user

12. Now we'll login thru our IAM user

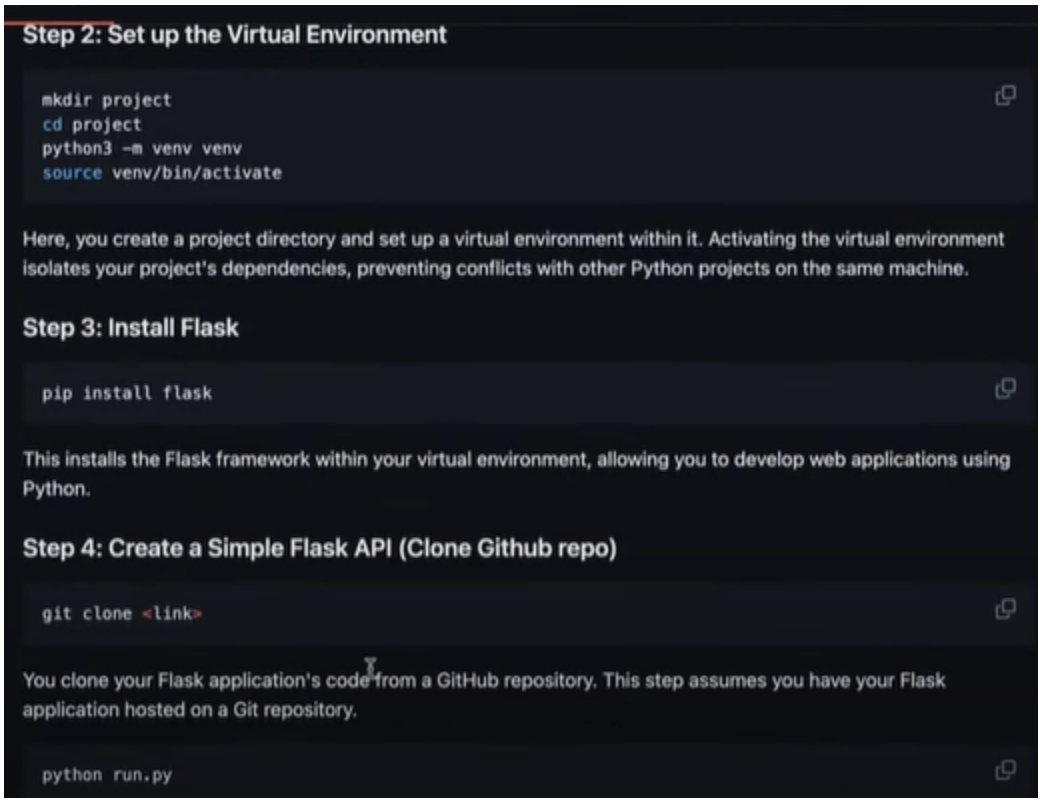
Hosting Flask Application in EC2

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

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



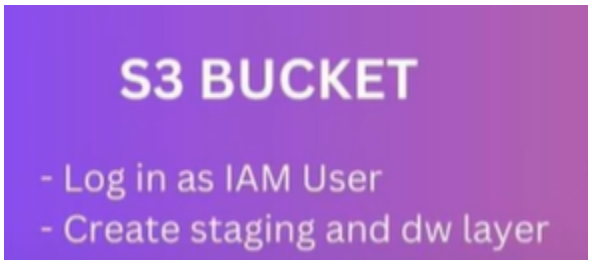
3. Next we will update the package manager and install required things(Venv,etc) to run our flask application …we install all the packages inside the env



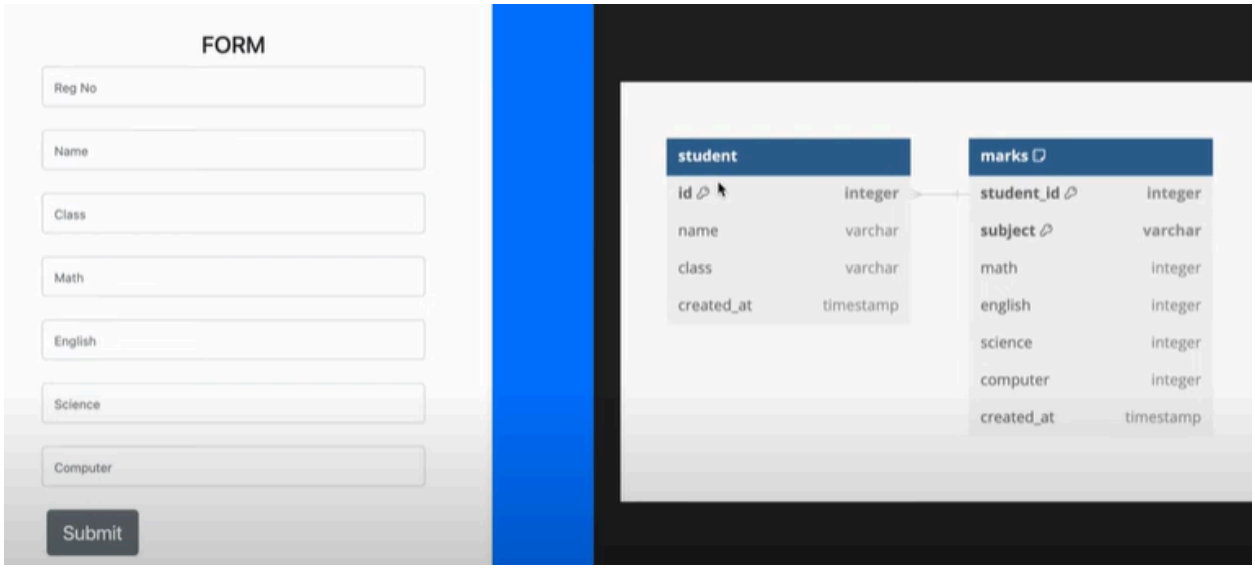
4. Next we'll clone our flask project into our ec2 instance
5. Now if we go to the ipv4 address of our ec2 instance..then we can our form displayed

Creating Dynamo DB and S3 Bucket

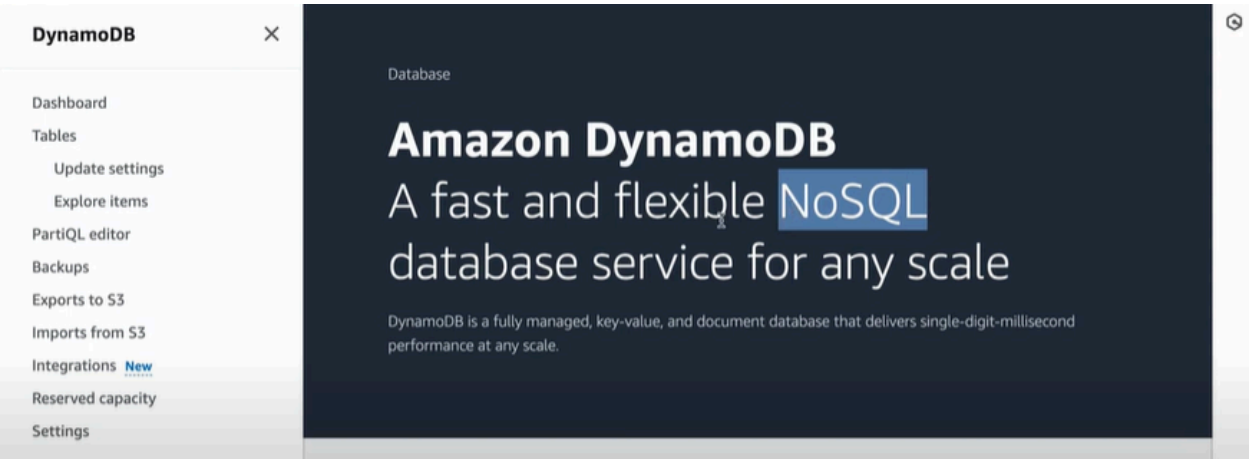
1. 



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



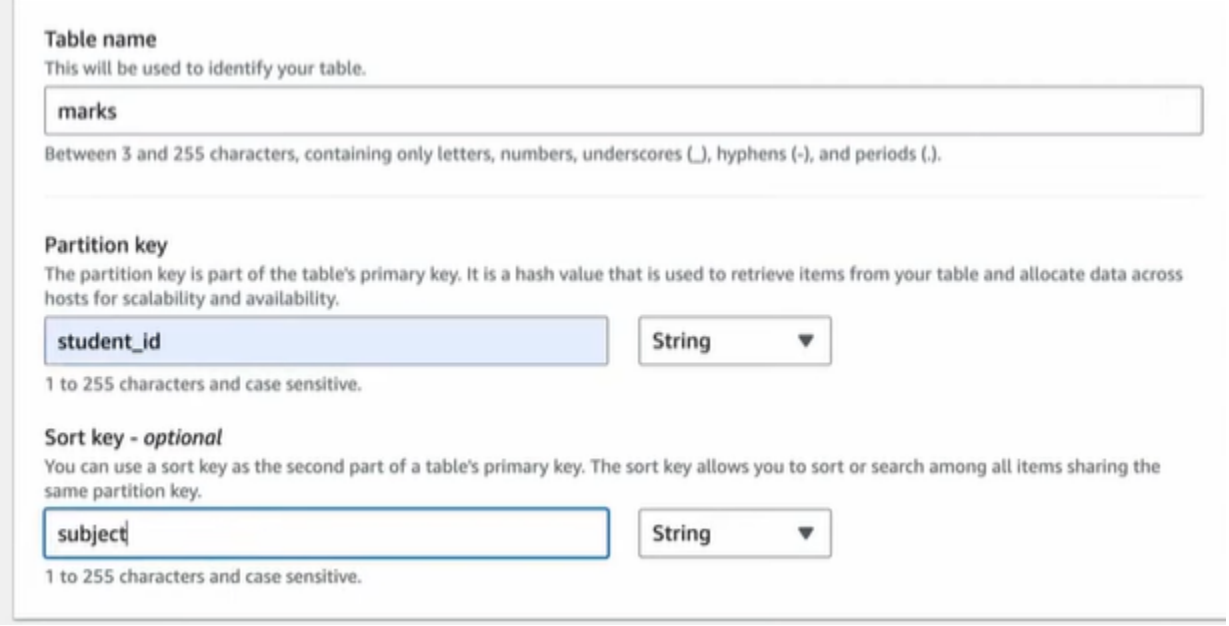
3. Creating a dynamoDB



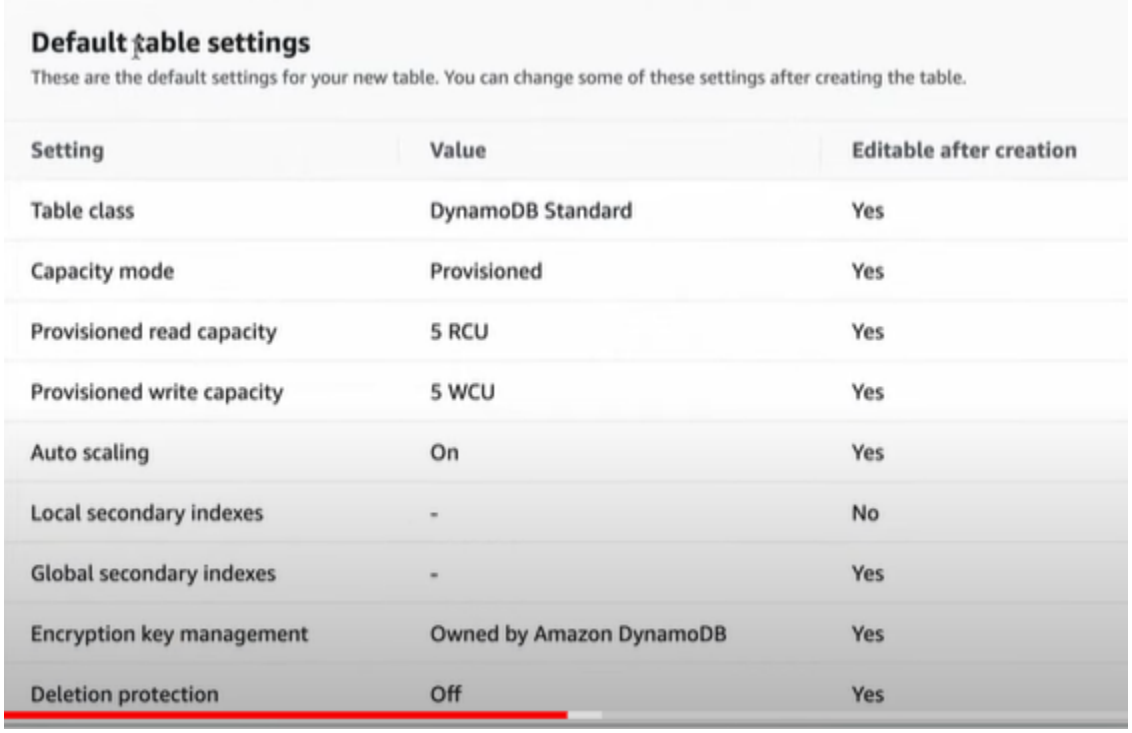
4. Here dynamoDB is NoSQL DB…so we dont need to give any predefined Schema..instead we'll be giving partition and key
5. Next we'll create a table called student

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



8. Default table settings are same for both tables

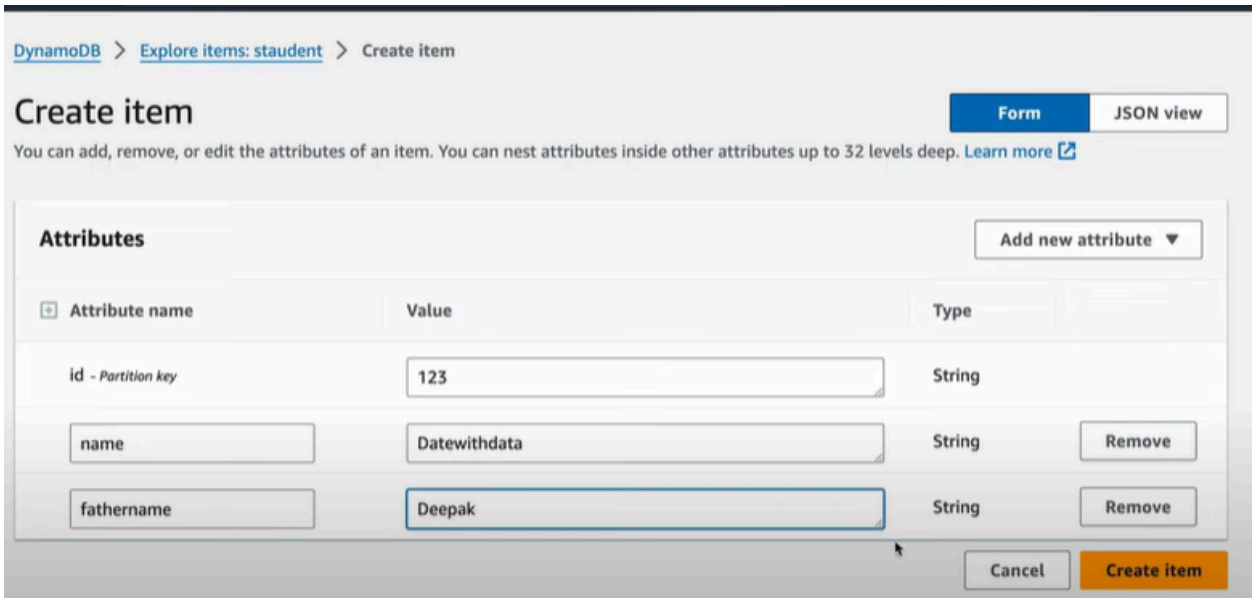


Default table settings

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: staudent > Create item

Create item

Form | JSON view

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

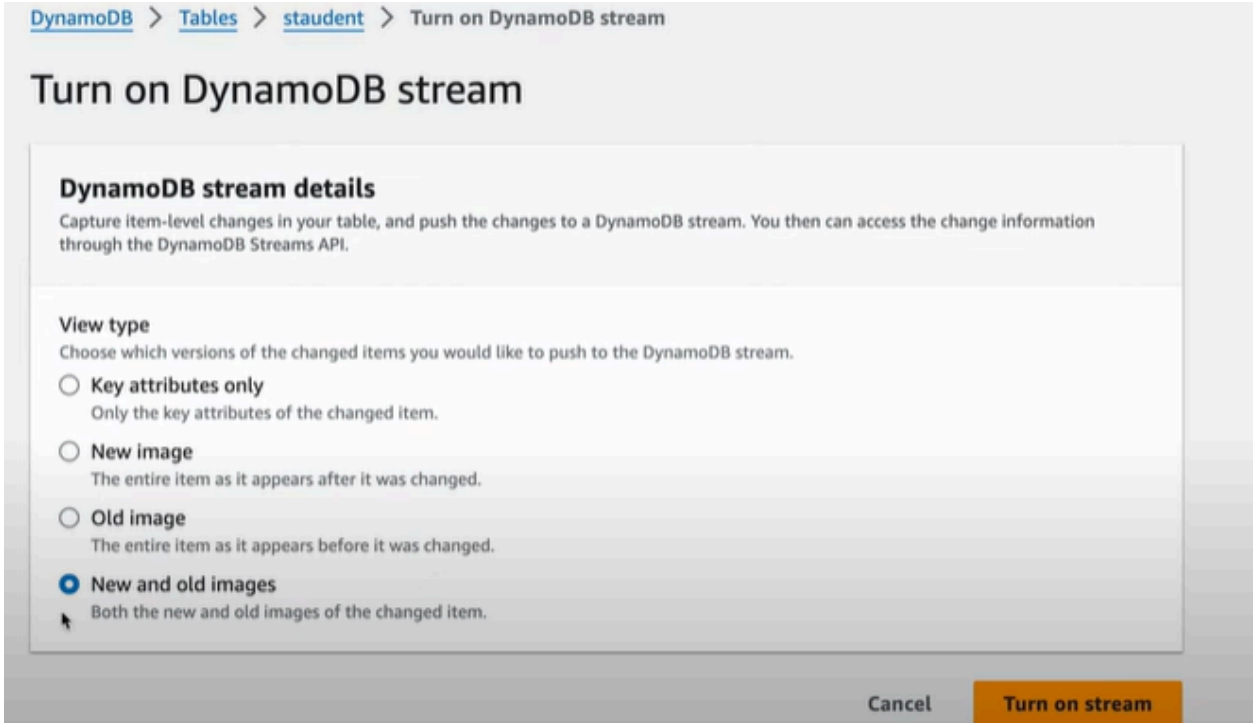
Attributes

Add new attribute

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

Cancel | Create item

10. Let's turnON dynamoDB 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:

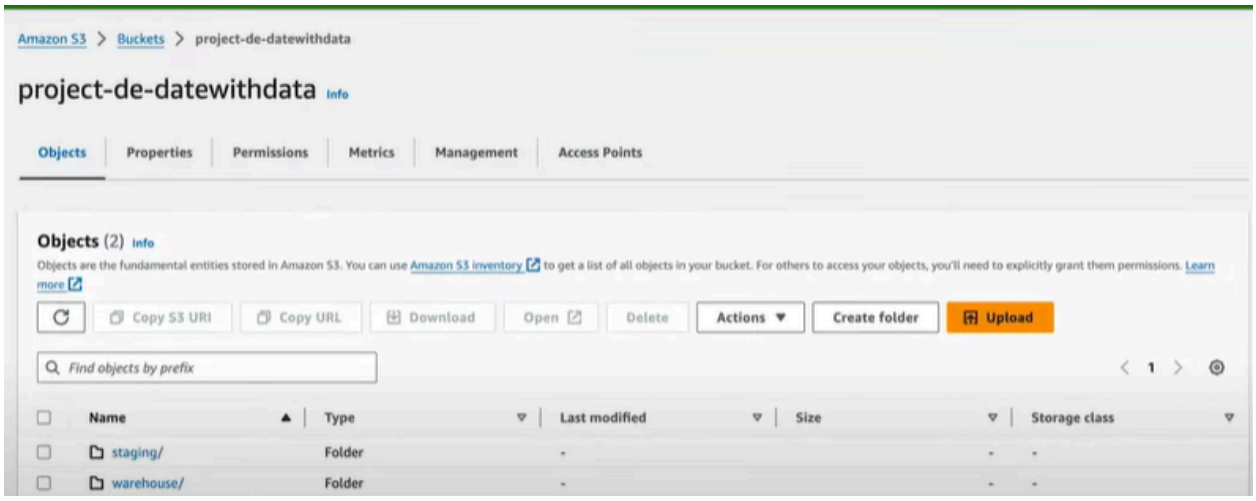
- **Change type:** Indicates whether the item was inserted (INSERTED), updated (MODIFIED), or deleted (REMOVED).
- **Item keys:** The partition key and sort key (if used) that identify the modified item.
- **Old and new image (optional):** For updates, the stream record can contain both the old and new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