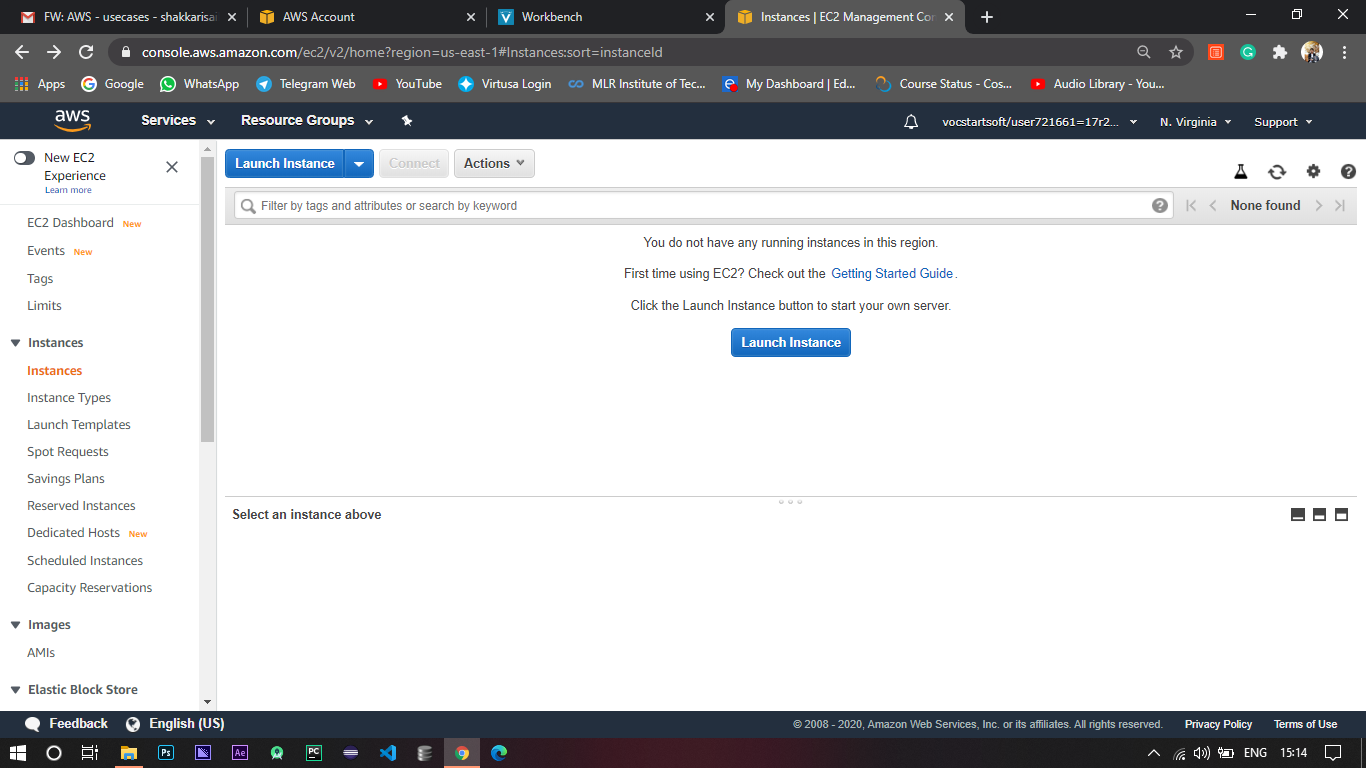
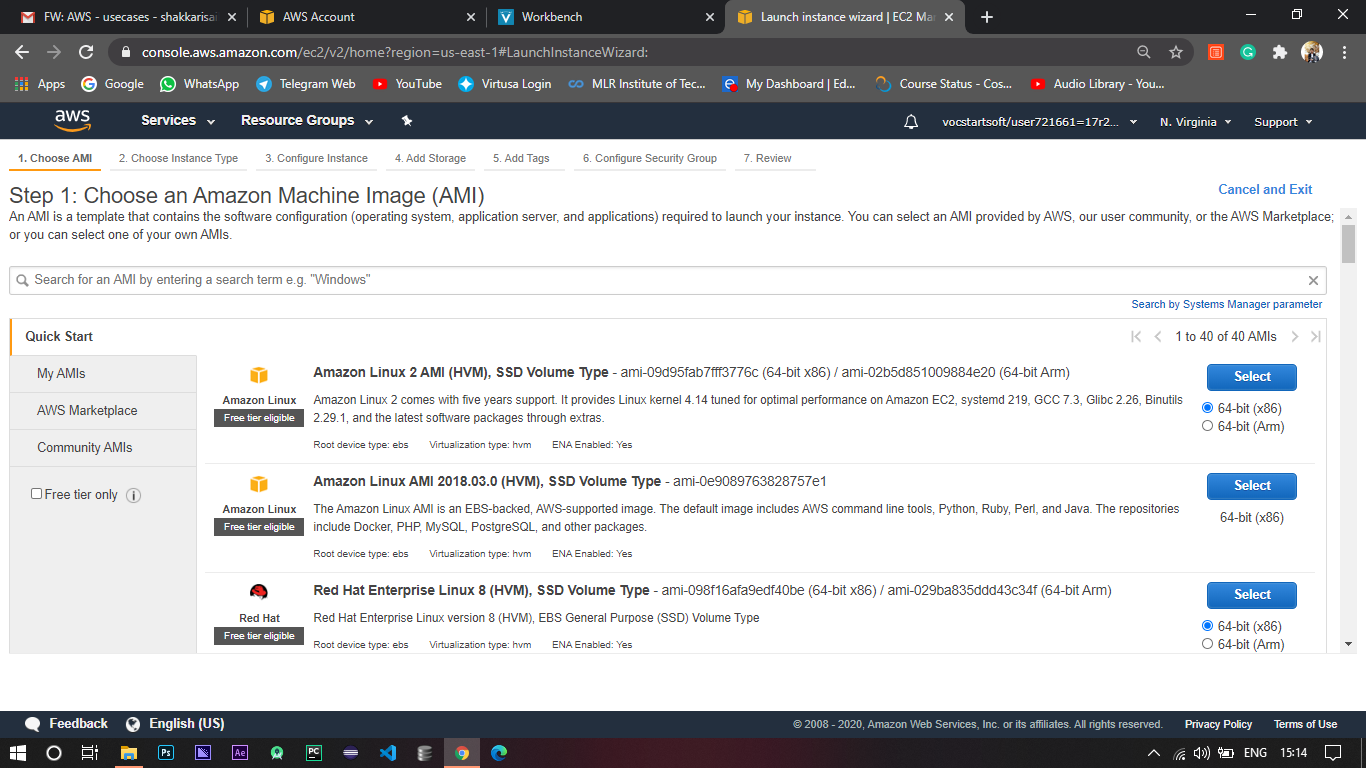
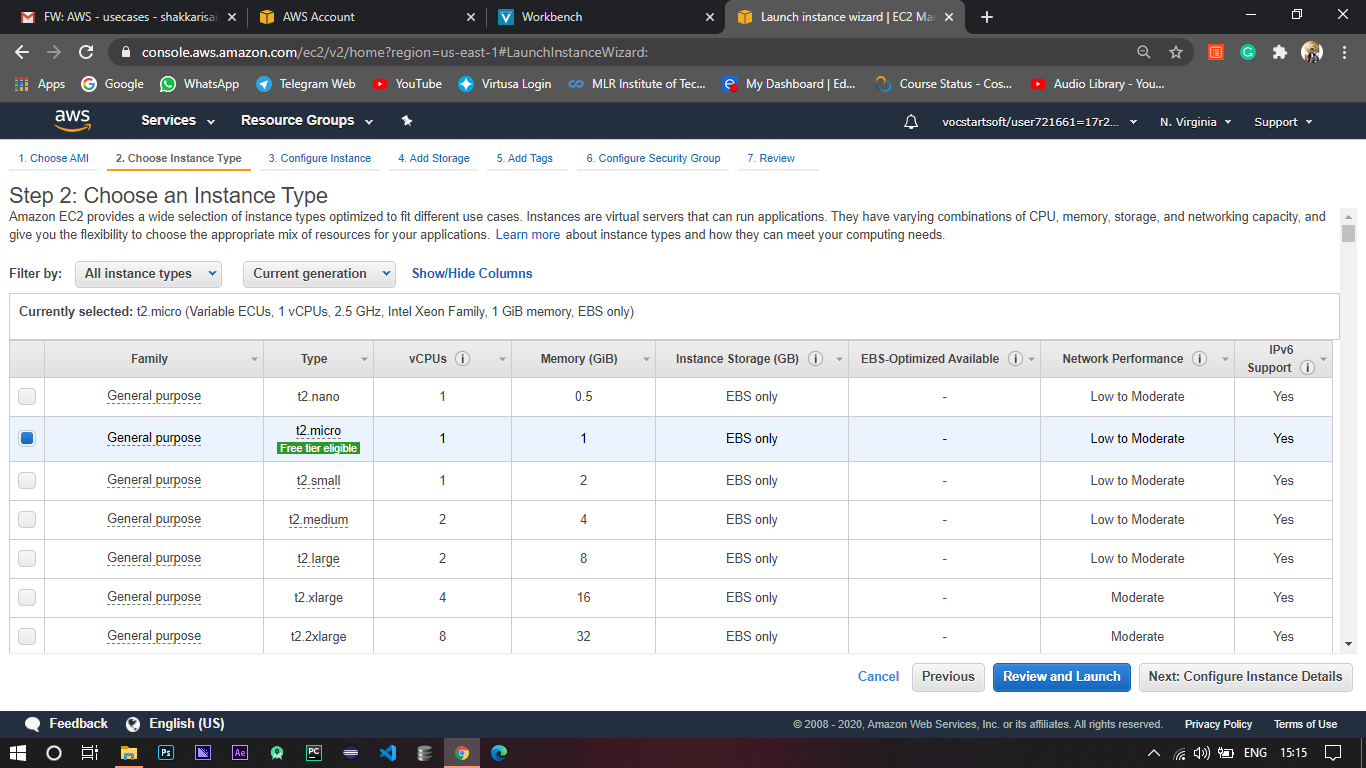
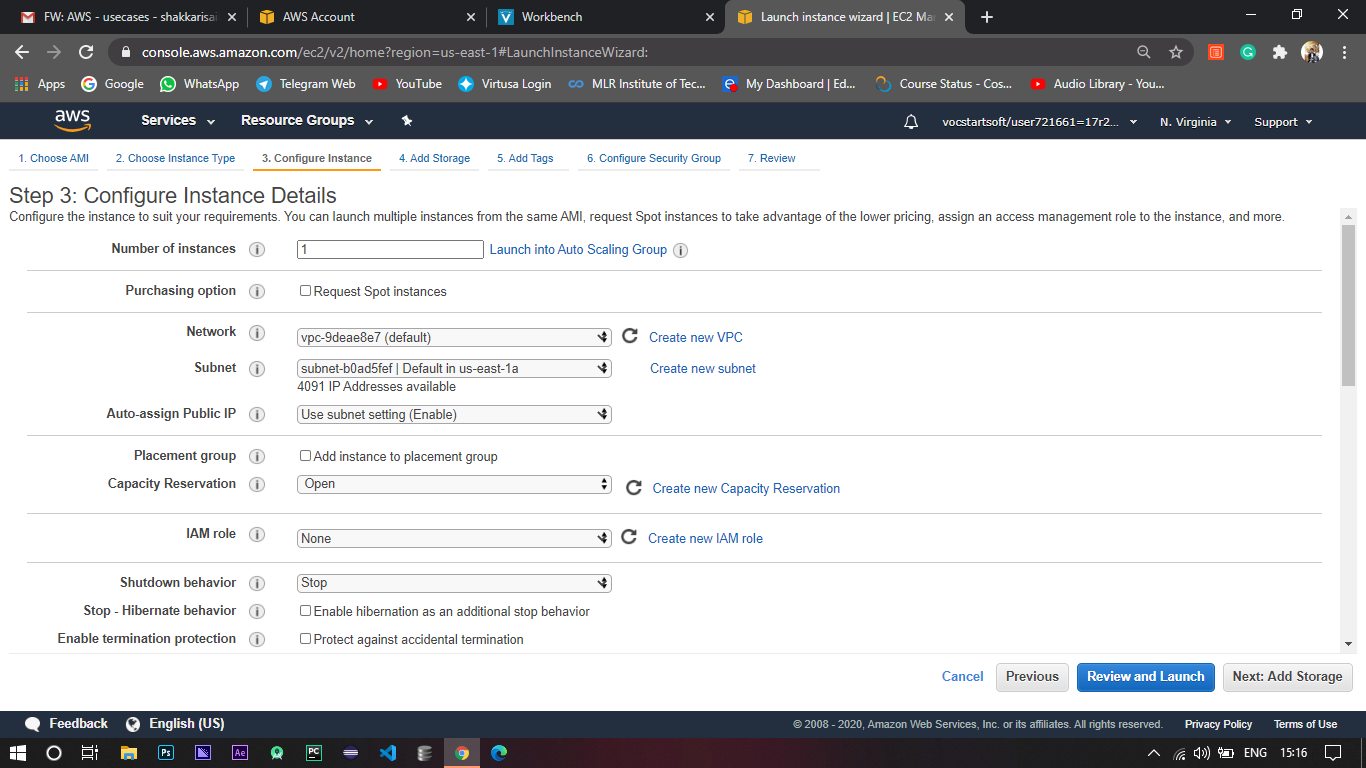
# 1. Launching 3 EC2 instances in 3 different subnets under single VPC.

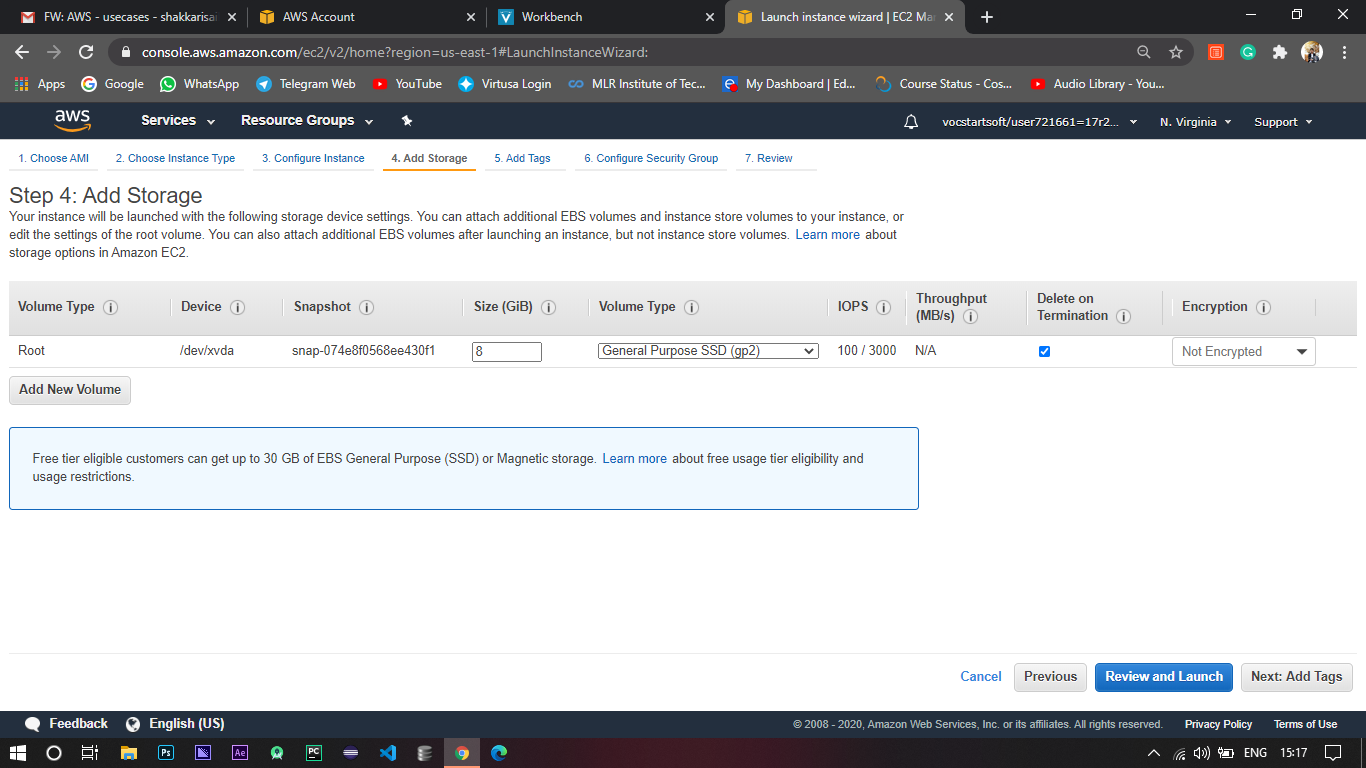
1. Log into the AWS Console.
2. Select the Region in which you want to launch the Instance.
3. Go to EC2 which is under Compute section in AWS Services.
4. If you have any previous launched instances you can click on Running Instances Button to see them.
5. If you don’t have any instances running, the interface will be somewhat like the above one.
6. To launch an Instance click on Launch Instance button.
7. Then you need to select an AMI (Amazon Machine Image) which is more like an image of a machine which has some software preinstalled (mostly OS and such).
8. For now I’m selecting Linux machine which comes under Free Tier.
9. Based on your compute requirement’s select the Instance Type.



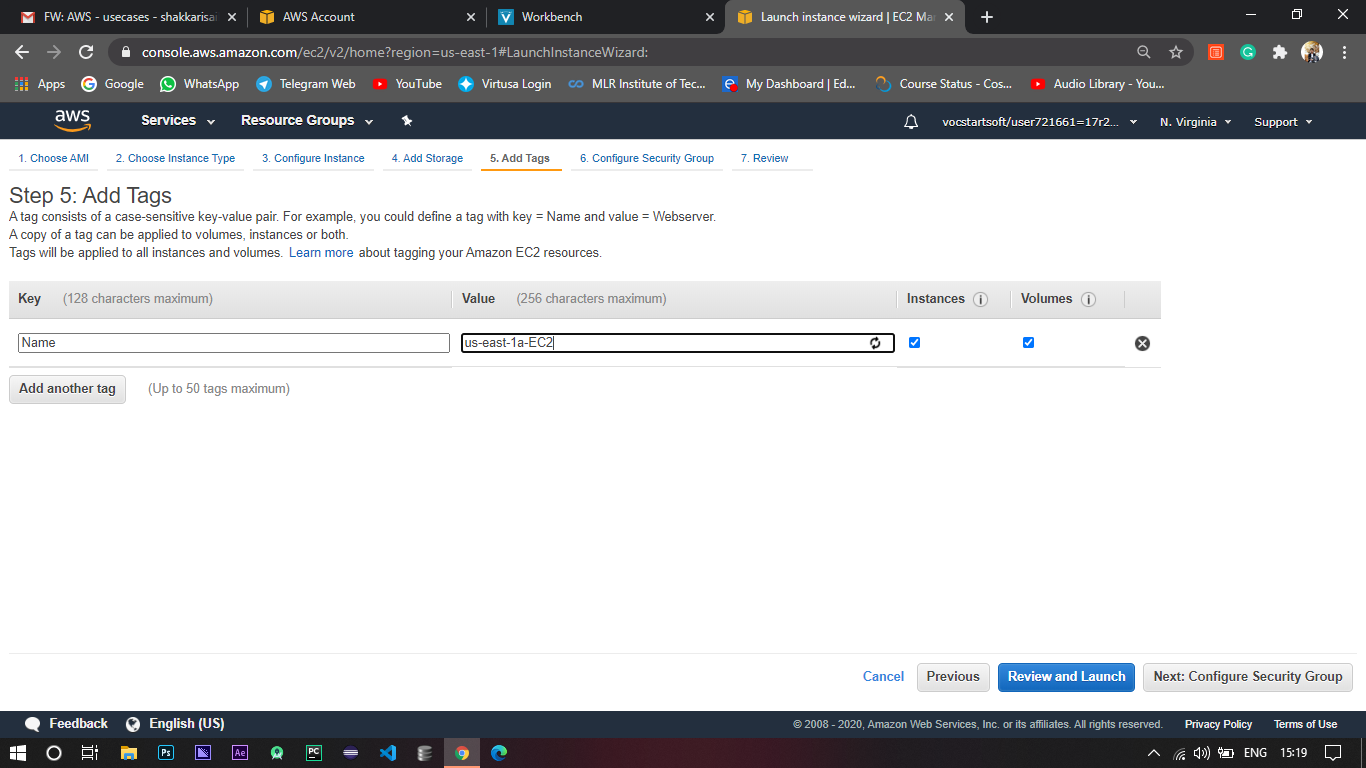
1. For now I’m going with t2 micro which is again a Free Tier eligible.
2. Next you need to configure the Instance which is the main part of this use case.
3. For now I’m using the default VPC and launching the instances in us east 1a, 1b, 1c.



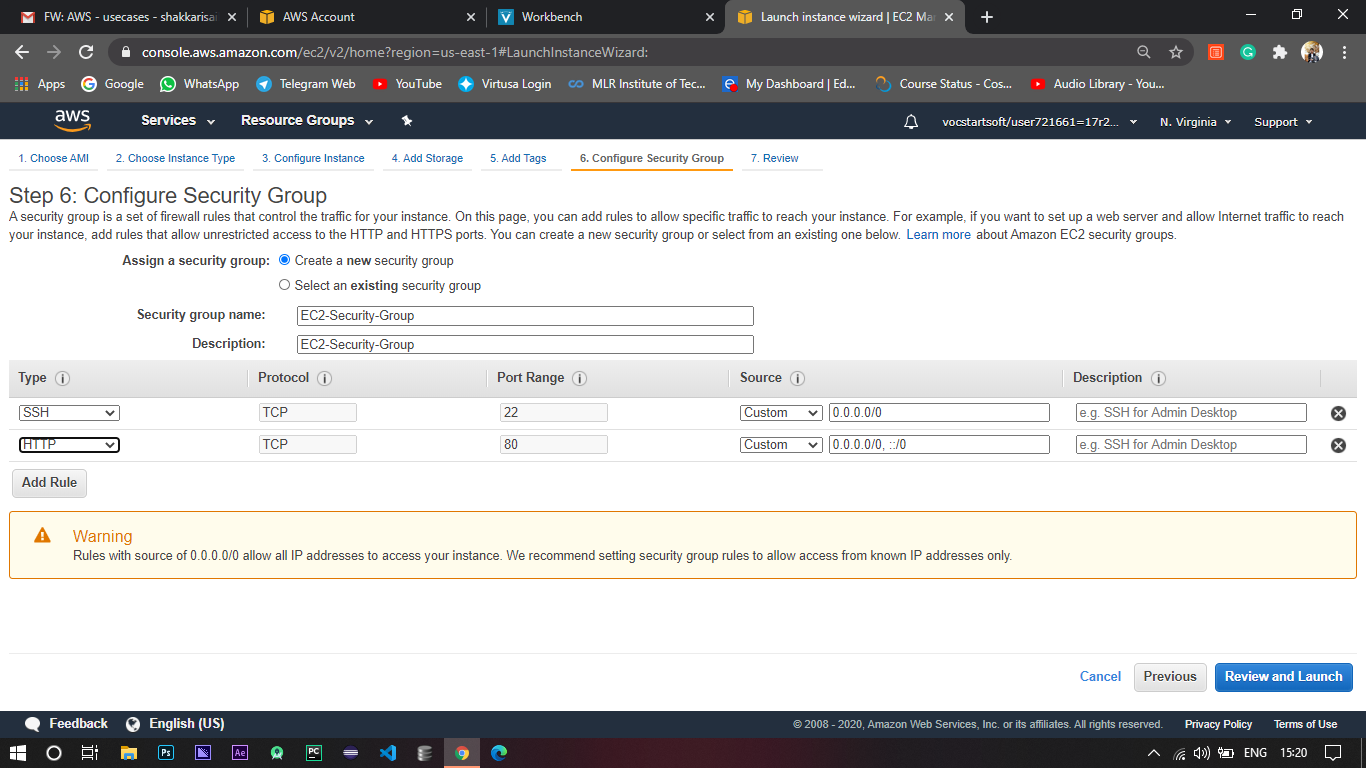
1. Once done with Configuration, we need to select the storage.
2. By default you will be given a root volume of 8gb which can be increased and the no. of volumes per instance can also be increased.



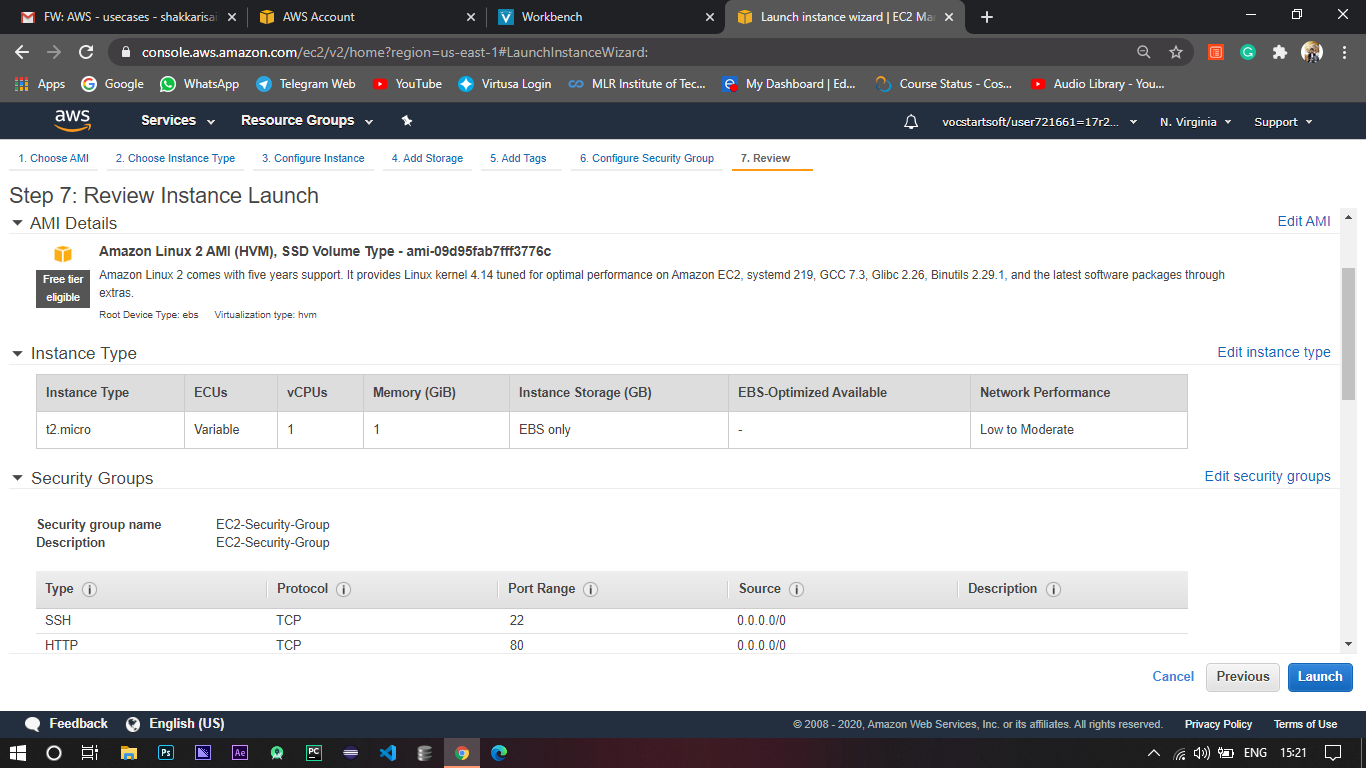
1. Once done with that, you can give any tags to filter the instance later.



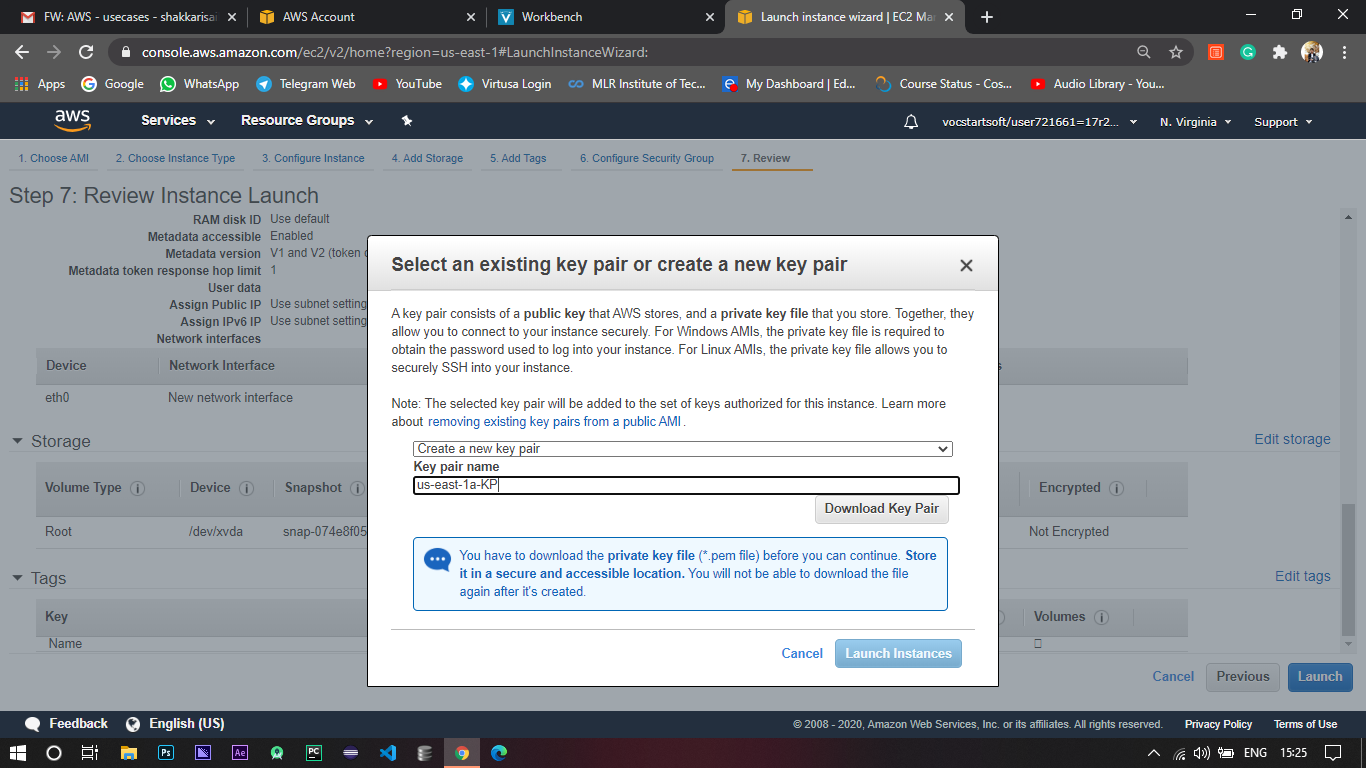
1. After that you need to configure the Security group which acts like a fire wall for your Instance Traffic.
2. Security Group has both inbound and outbound rules, if an inbound rule is applied it automatically applies to the outbound rule.
3. Security Group can only allow traffic (It can’t deny).



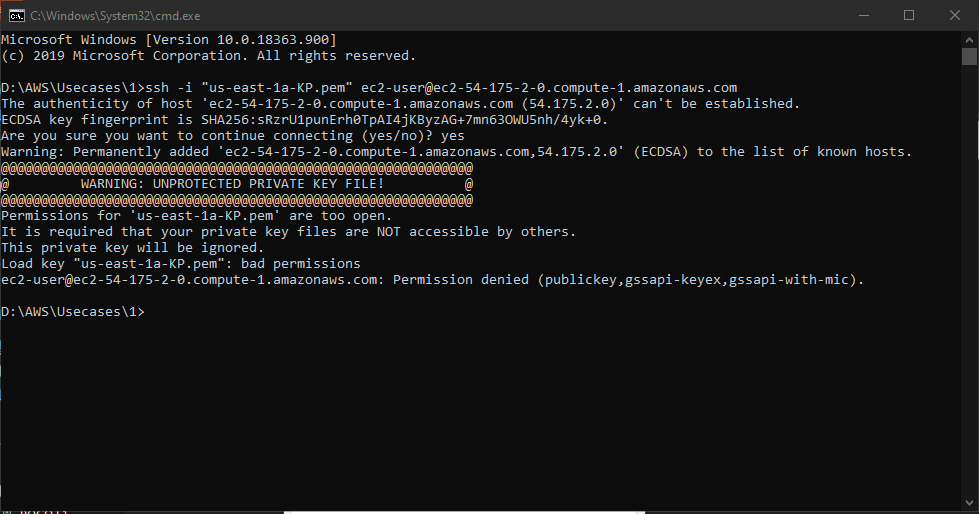
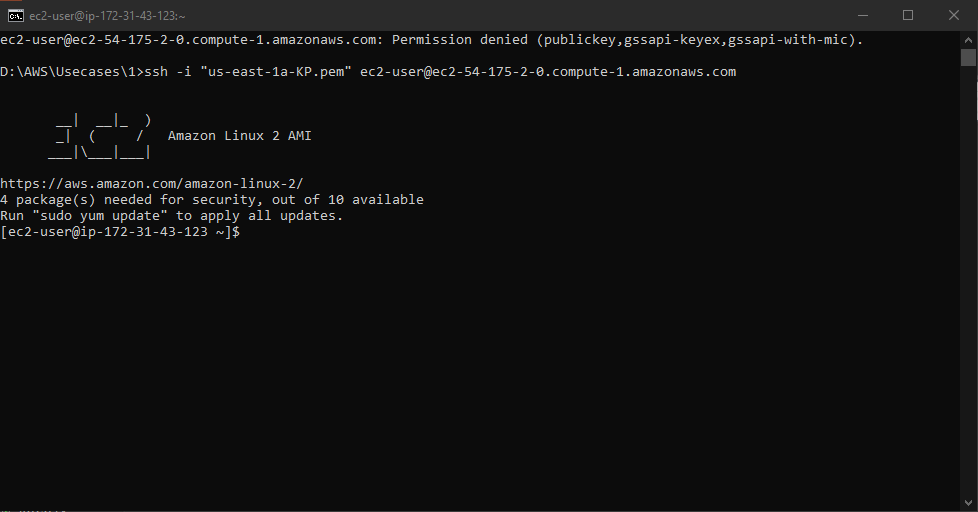
1. Finally you can review the configuration you have done so far.



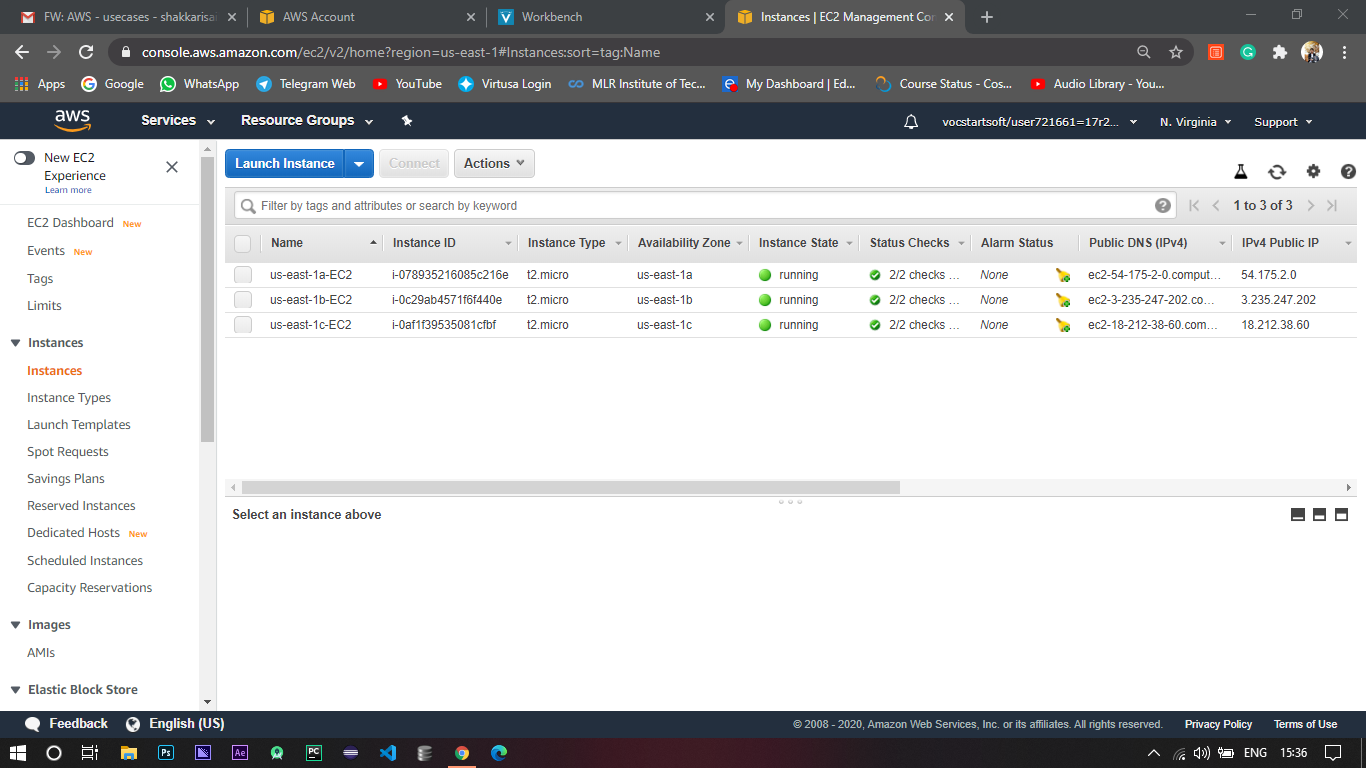
1. Once you click Launch you’ll get to create a key pair which consists of a public key and a private key, Amazon will have the public key and the private key will be given to you which will help you to access the Instance.



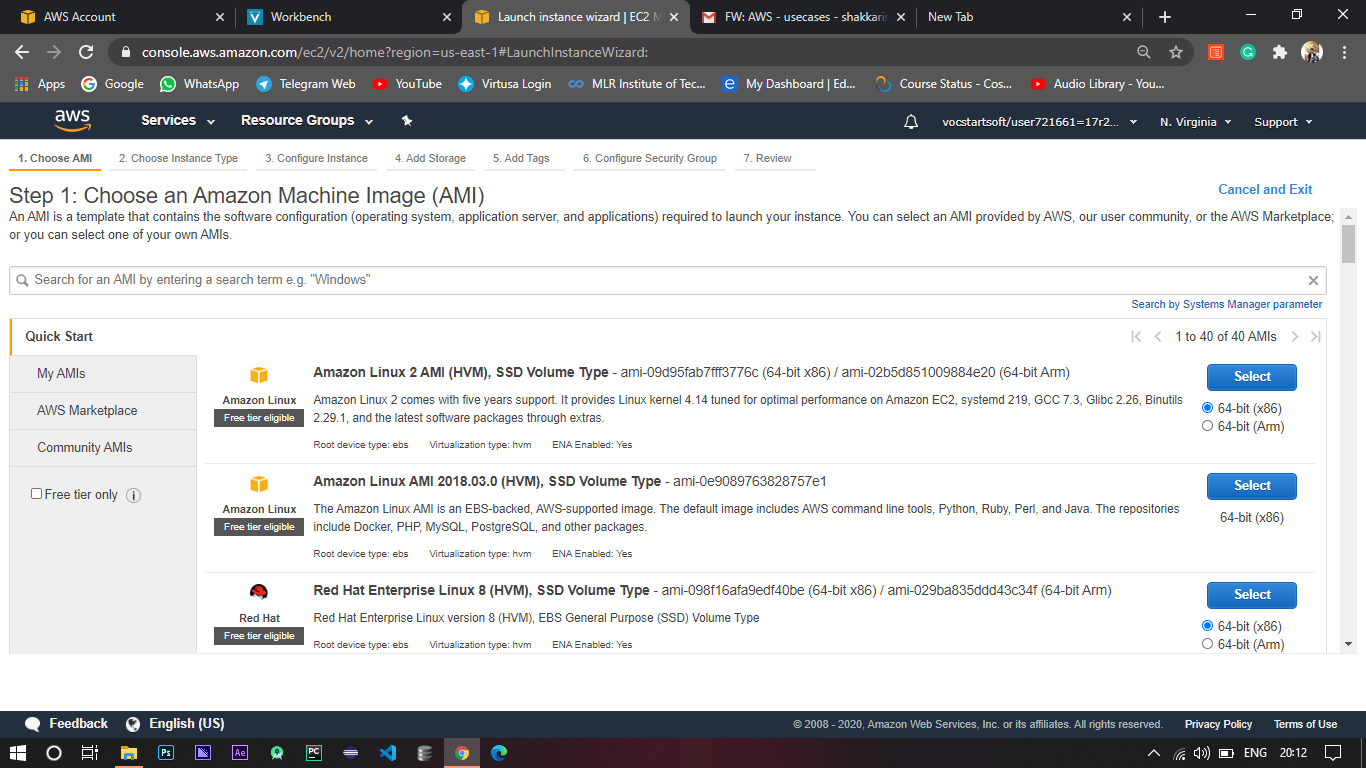
1. Always remember to change the permissions of the private key or else you’ll be getting an error like this.
2. Upon Successful connection you’ll be getting an interface like below.

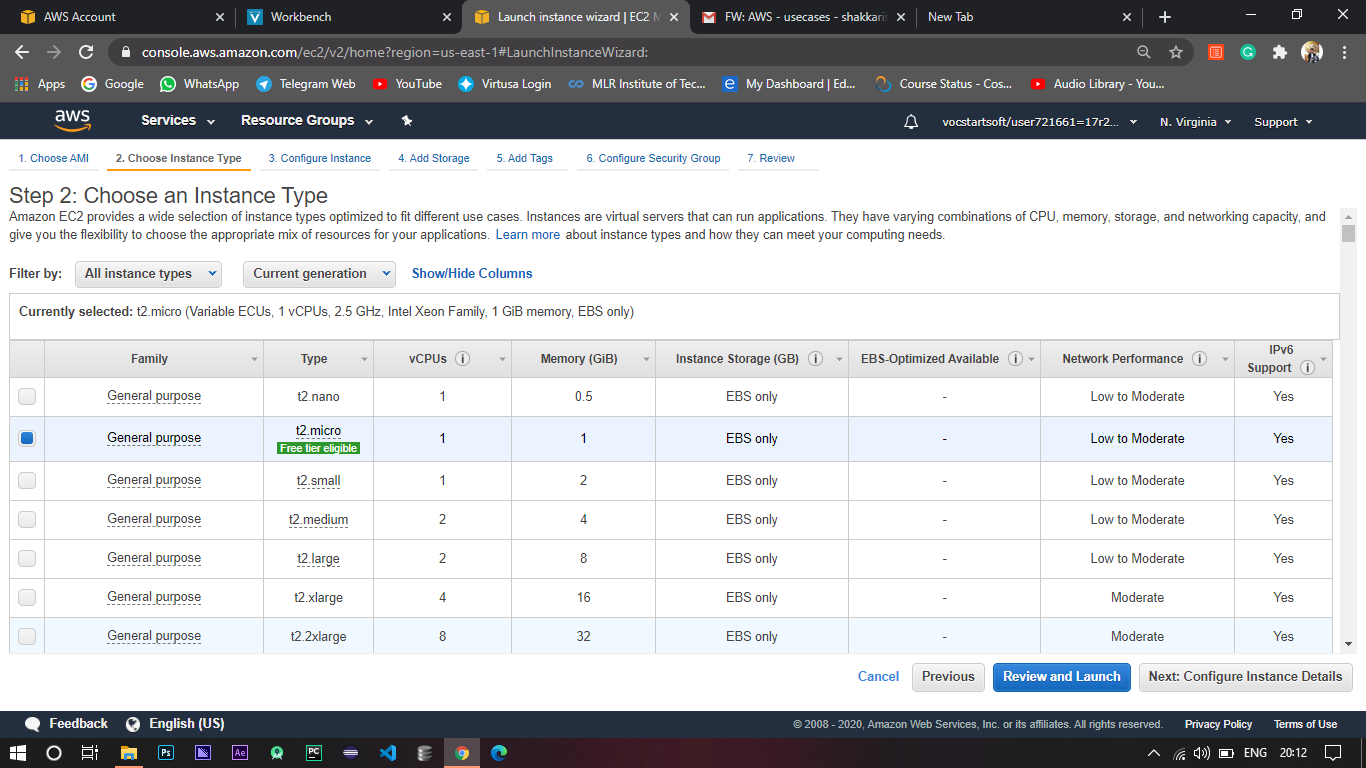


1. After creating all the instances you can see the instances in your dashboard like below.

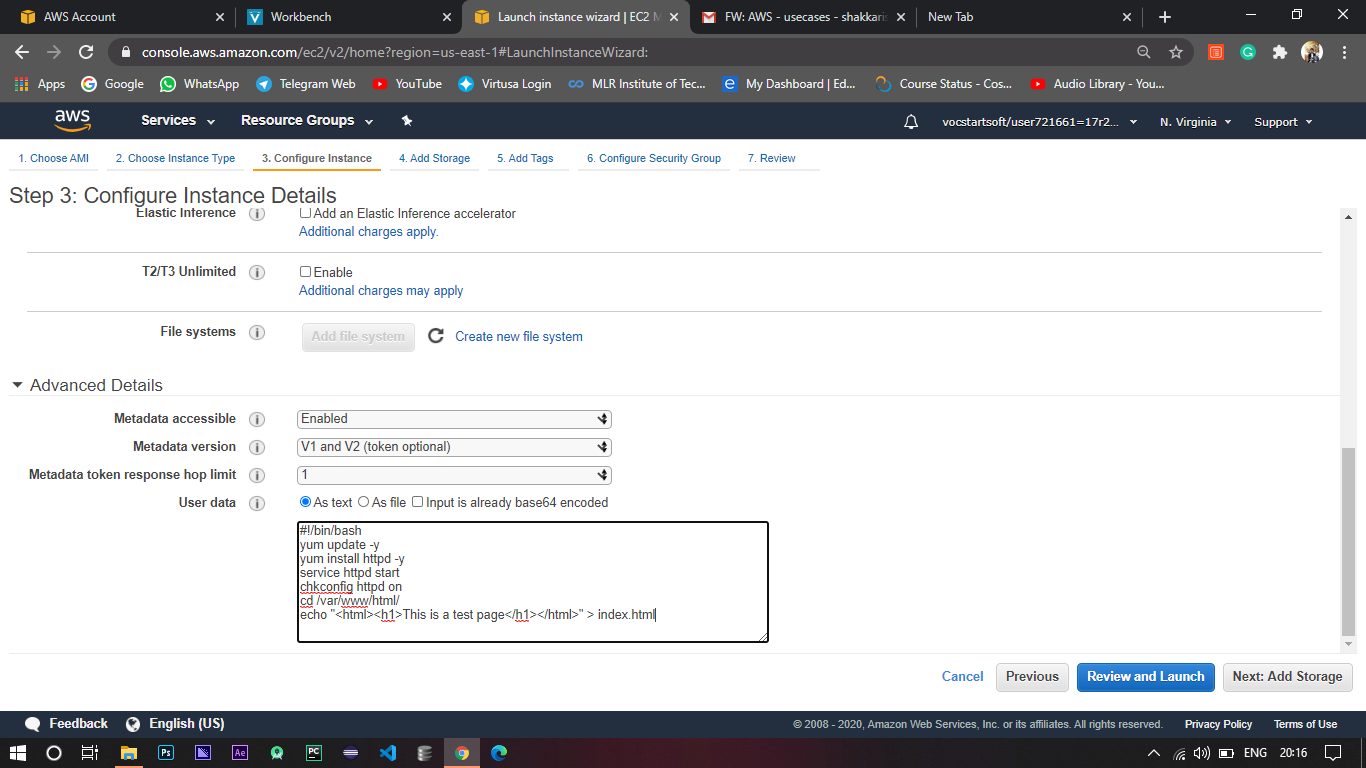


# 2. Bring EC2 instance using the user data and Installing Apache.

1. Log in to AWS Console.
2. Select the Region in which you want to launch the Instance.
3. Go to EC2 which is under Compute section in AWS Services.
4. Click on Launch Instance, and then you need to select an AMI (Amazon Machine Image) which is more like an image of a machine which has some software preinstalled (mostly OS and such).
5. For now I’m selecting Linux machine which comes under Free Tier.
6. Based on your compute requirement’s select the Instance Type.
7. For now I’m going with t2 micro which is again a Free Tier eligible.

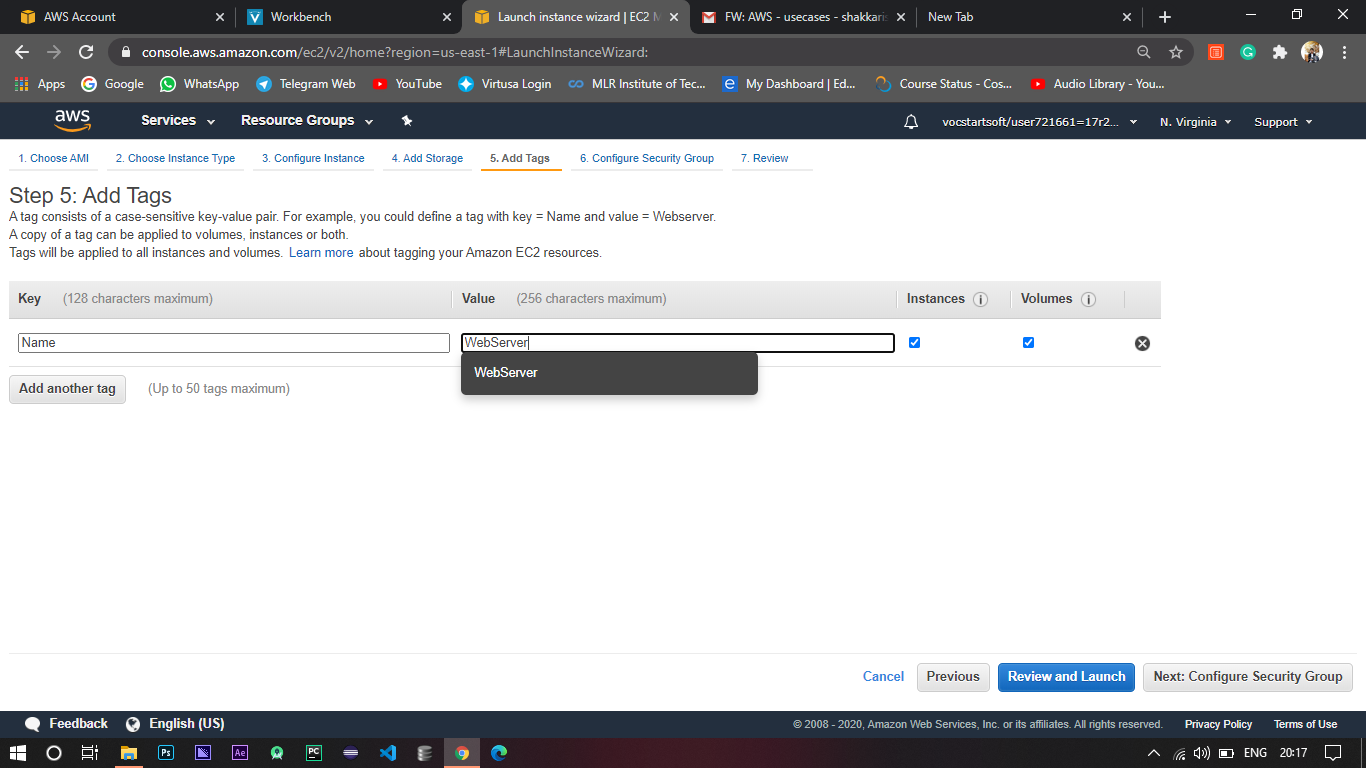


1. Next you need to configure the Instance which is the main part of this use case.
2. For now I’m using the default VPC and launching the instances in us east 1a.

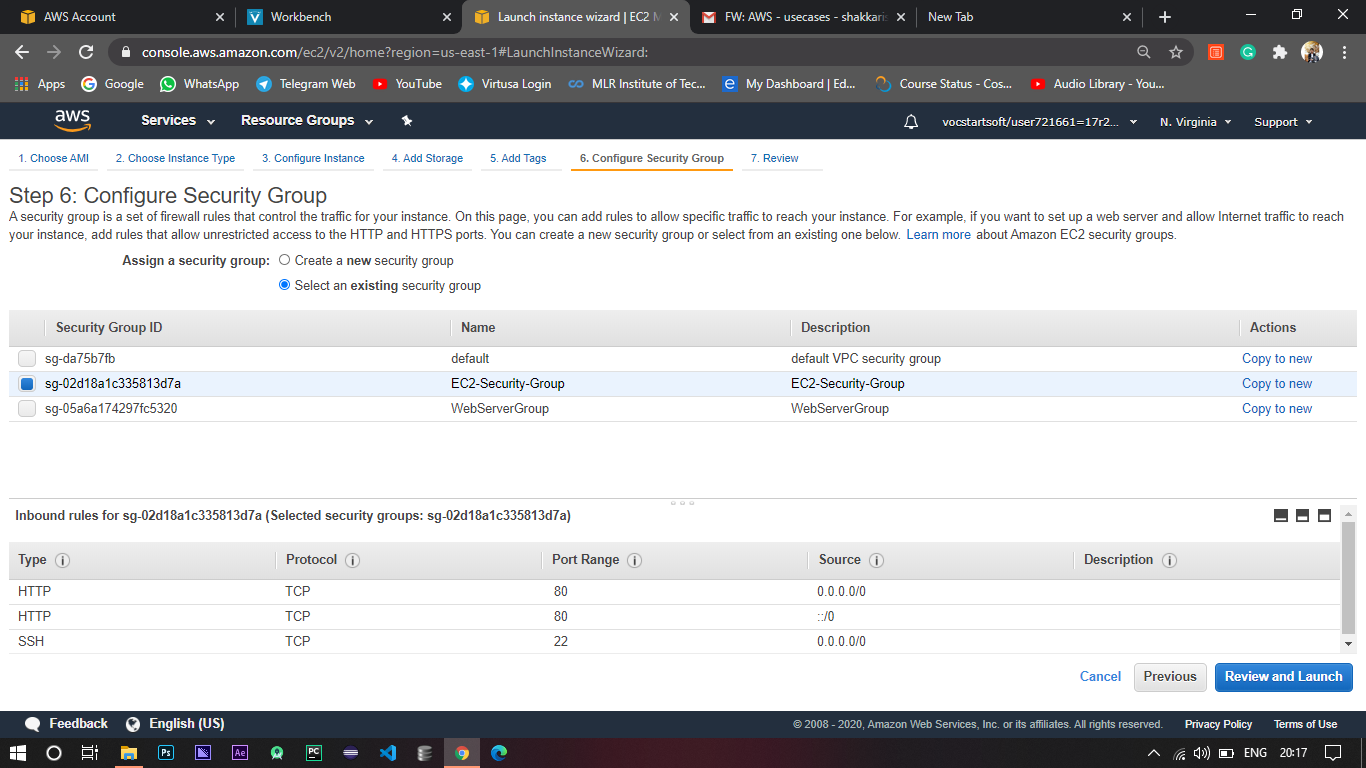


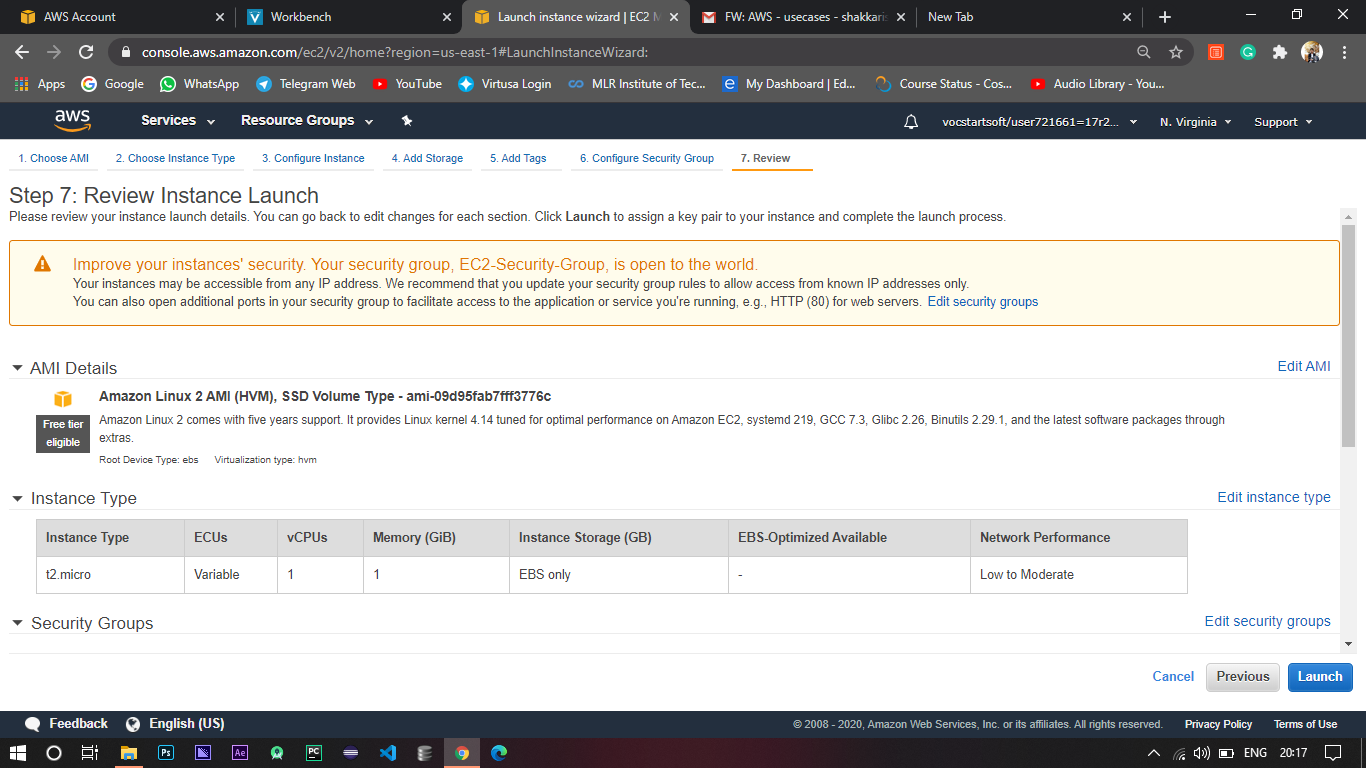
1. Coming to the user data.

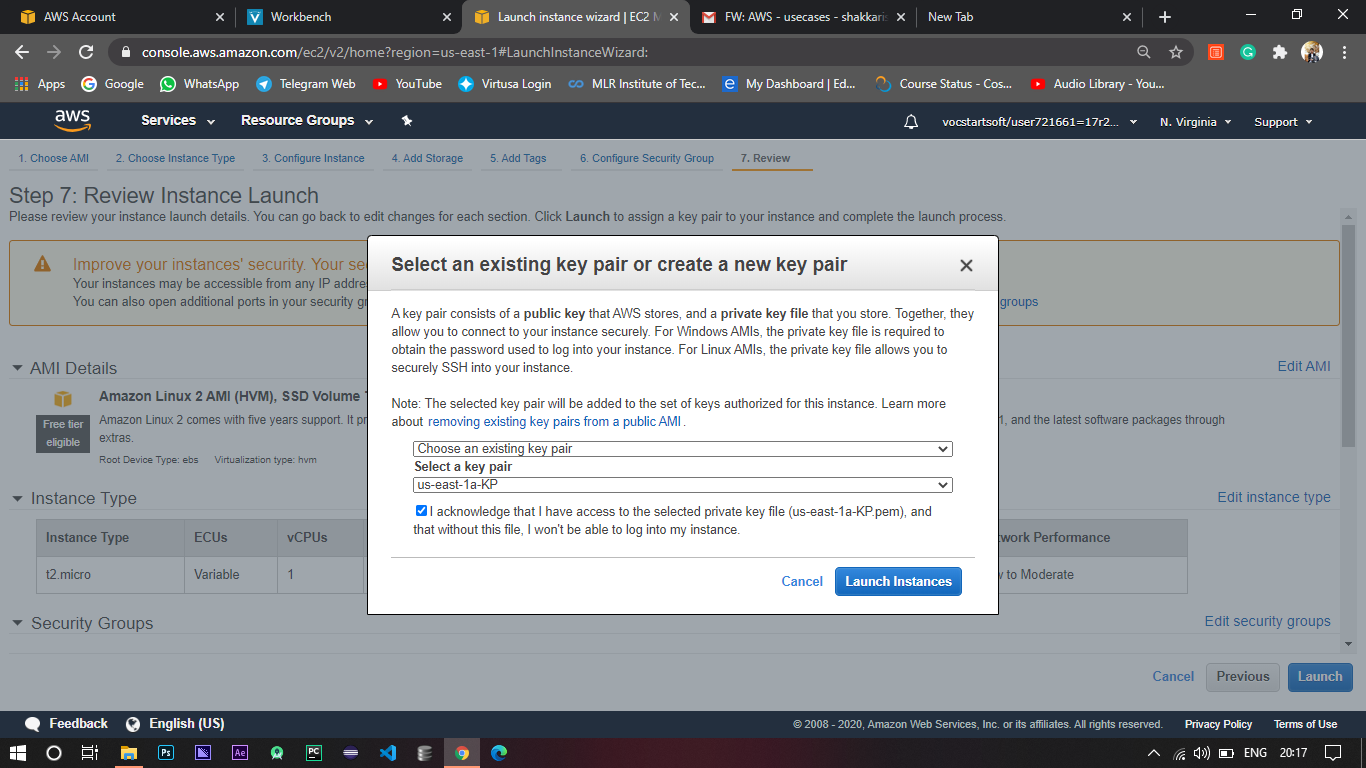
* First I have updated the system software using the ‘yum update –y’ command.
* Then I installed the Apache using the ‘yum install httpd –y’ command.
* ‘service httpd start’ command is used to start the server.
* The command ‘chkconfig httpd on’ is used to start the Apache service even after restarting the Instance.
* Then I’m changing the directory to the html directory and creating an index.html file to check the working of the Instance.

1. Add storage to the Instance.
2. Add tags to the Instance for further filtration.

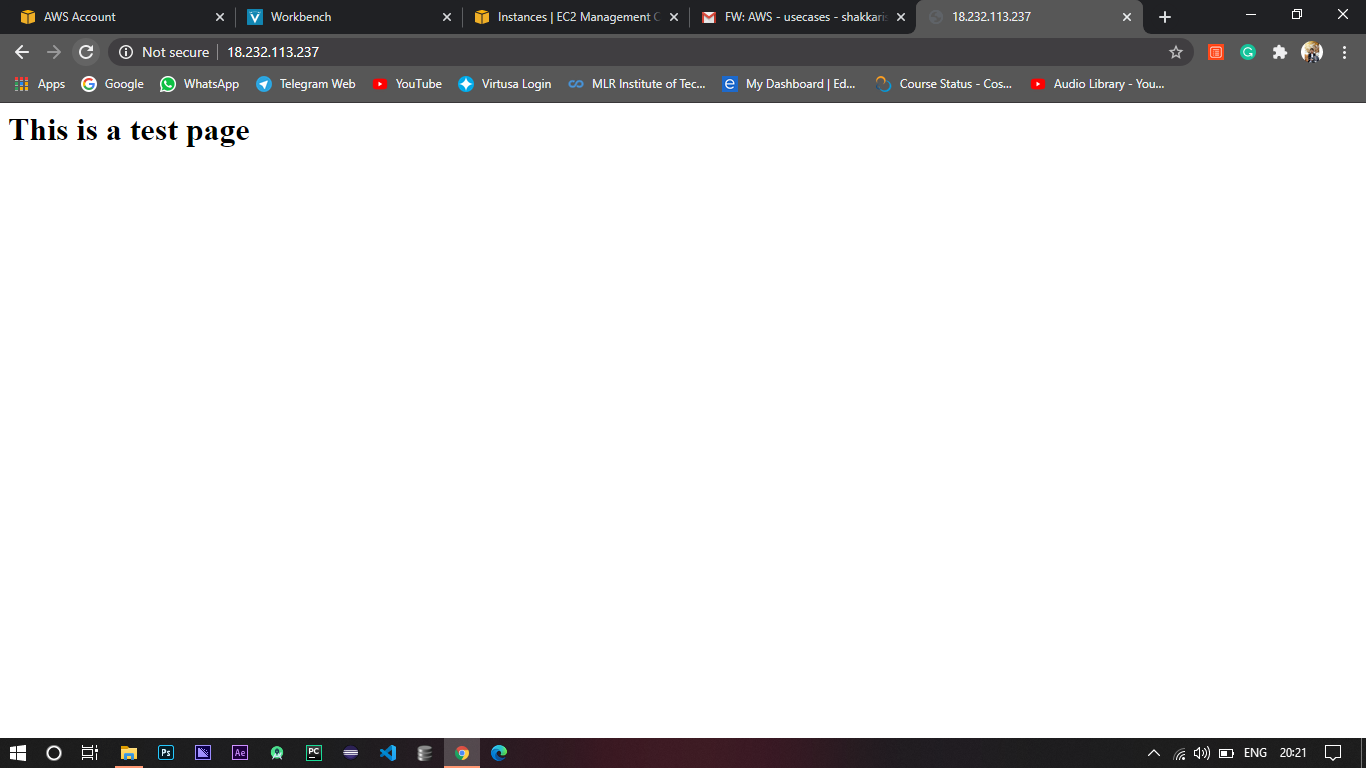
1. After that you need to configure the Security group which acts like a fire wall for your Instance Traffic.



1. Finally you can review the configuration you have done so far.
2. Select an existing / create a new key pair.

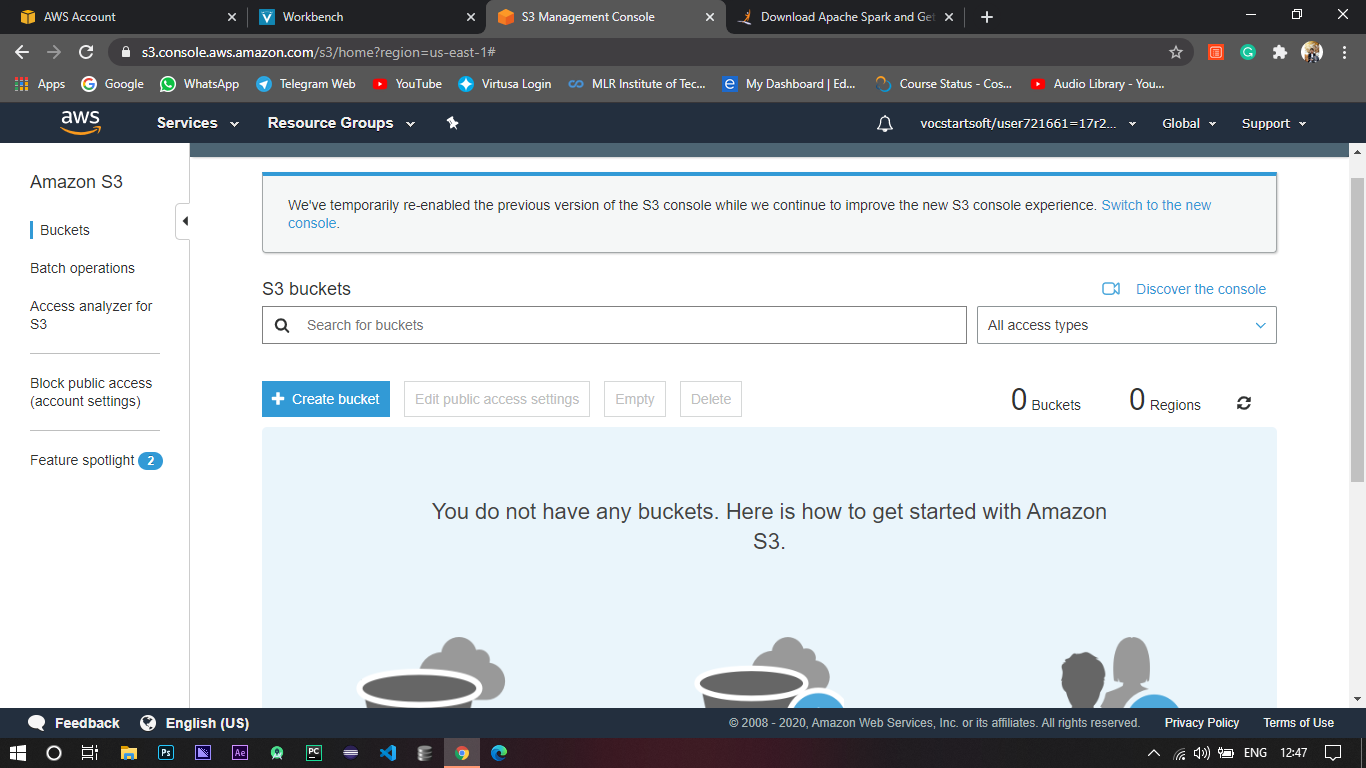


1. After the Instance launch you can access the public IP assigned to the Instance to see the content you have hosted in the html directory.

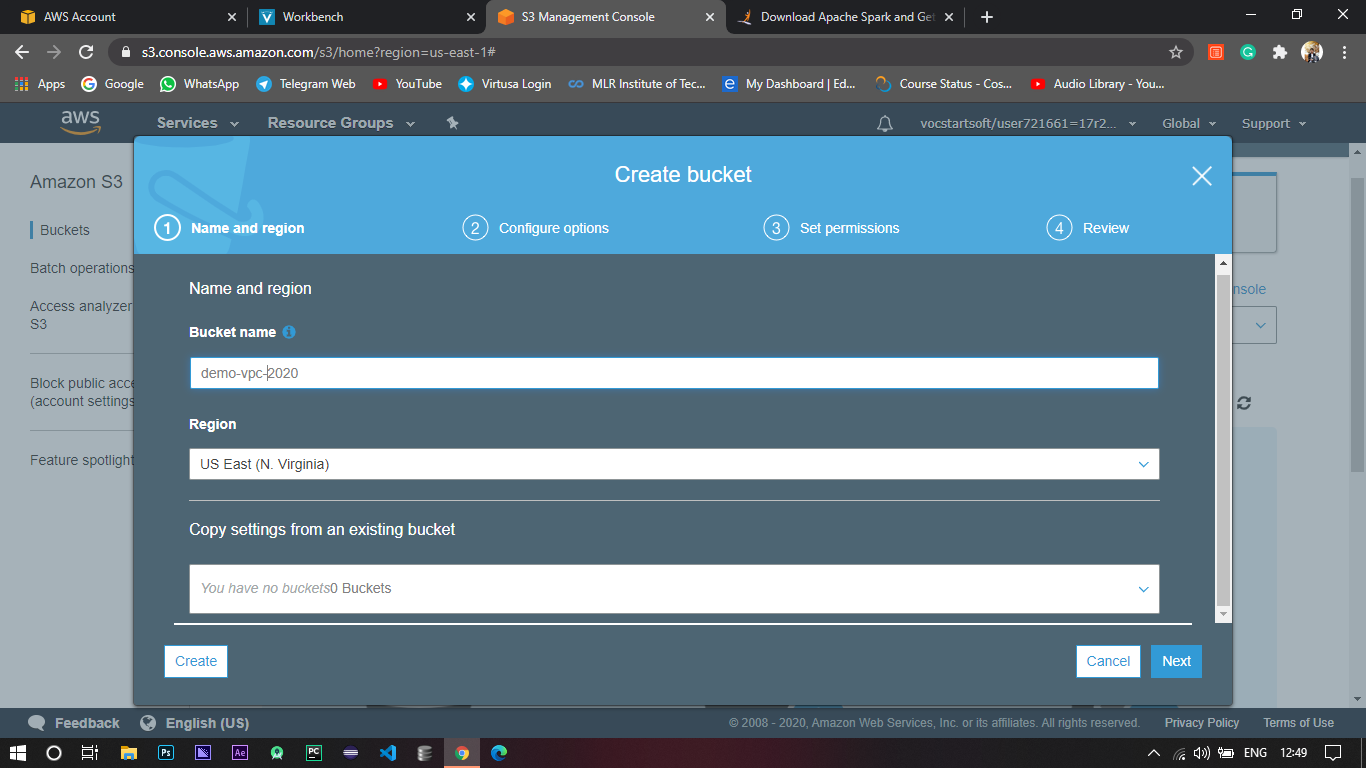


# 3. Create S3 bucket with Encryption and configure with FA/IAA/Archival mechanism.

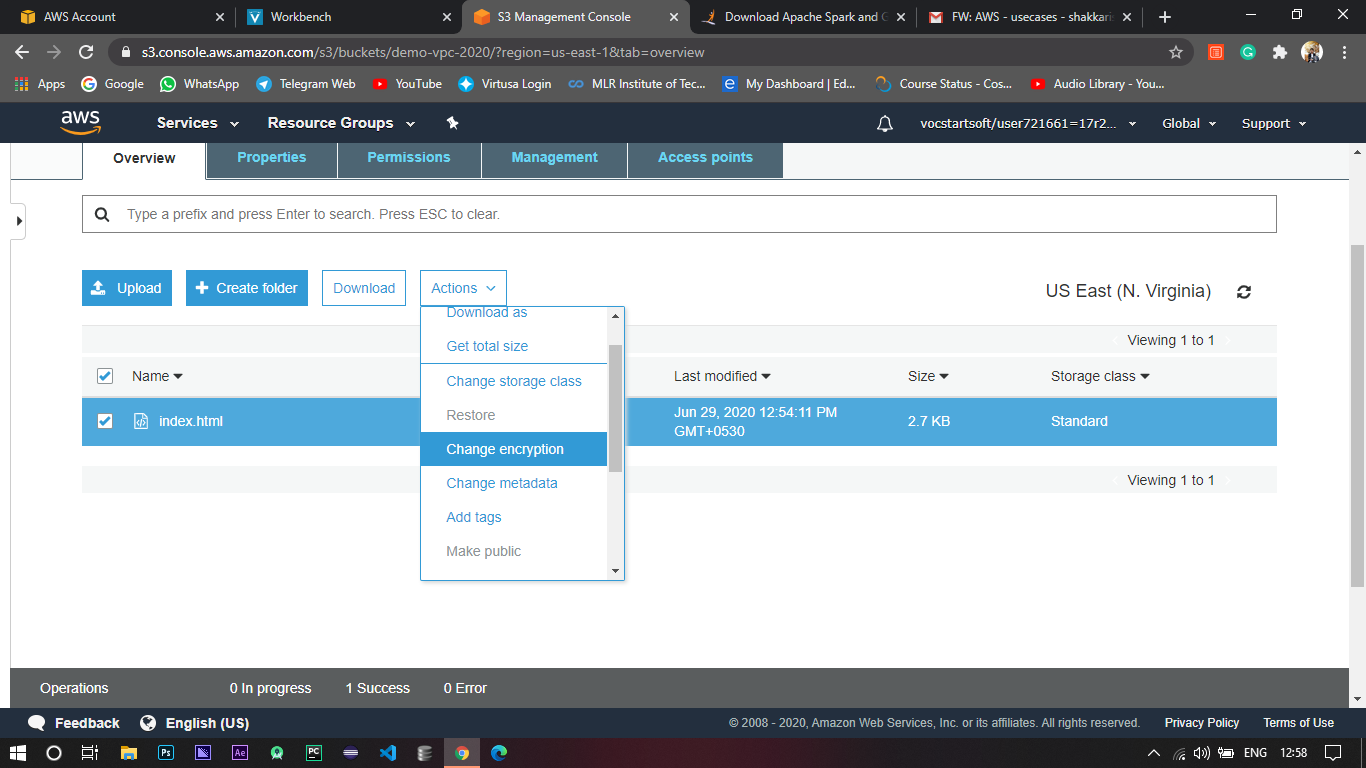
1. Log in to AWS Console.
2. Go to S3 which is under Storage in AWS services.
3. S3 buckets are universal namespaced i.e. the name of the bucket must be unique throughout the world and also one can see the region is set to global whenever you are in S3 dashboard.



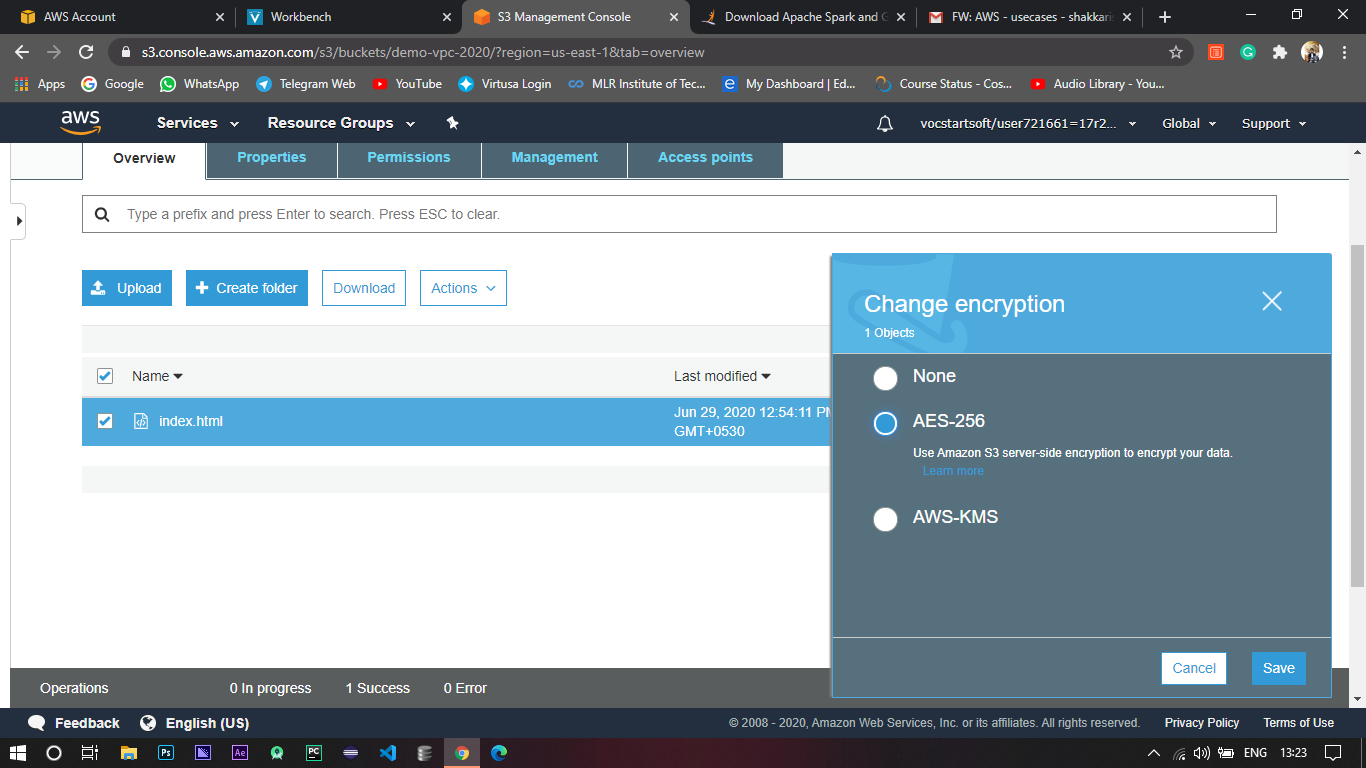
1. To create a bucket click on Create bucket button.
2. Then give the bucket a unique name and click Create.
3. Or else you can configure according to your need by turning on logging (where all the activities of the bucket are logged) or versioning (where all the old versions are stored).
4. Then you can give public access to the bucket where everyone can access the objects in that bucket.
5. And finally review and click create to create the bucket.



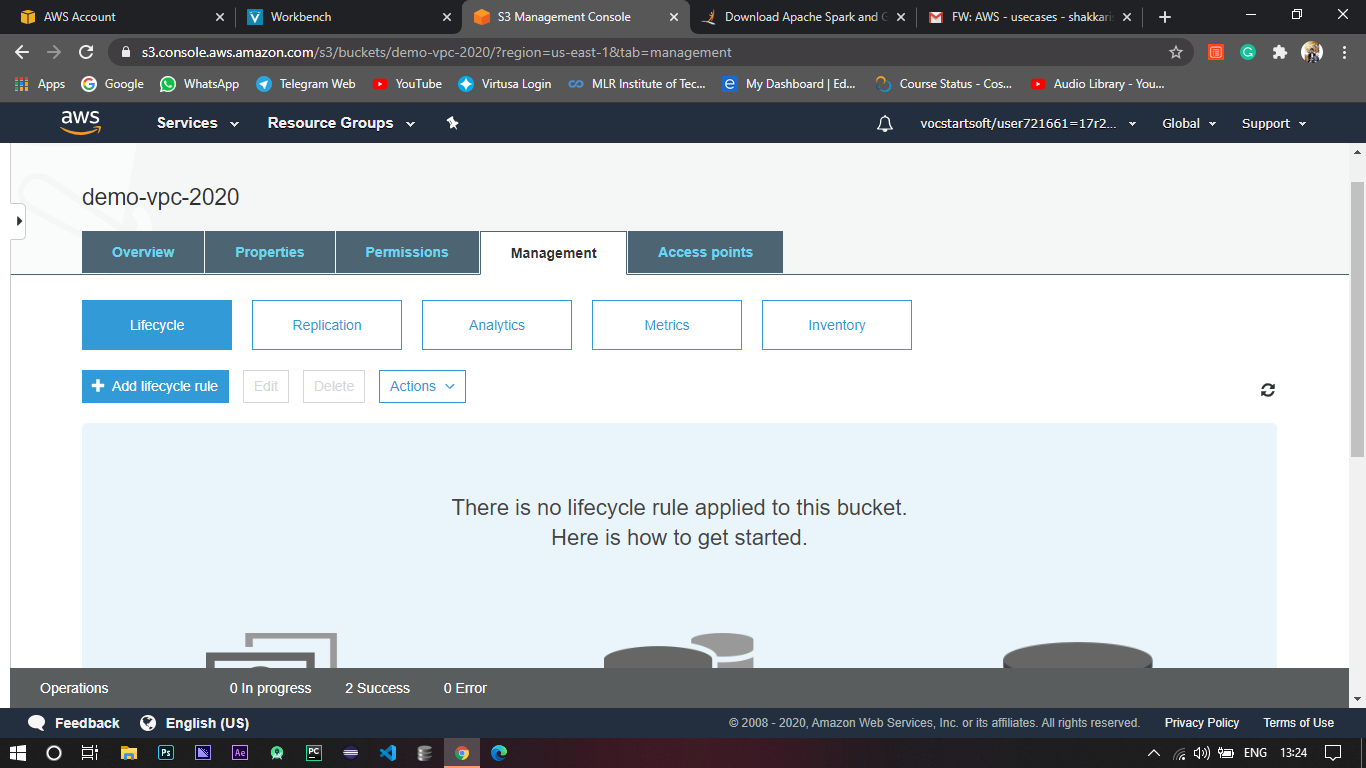
1. Upload any random file and to encrypt it select the file -> Actions -> Change encryption.



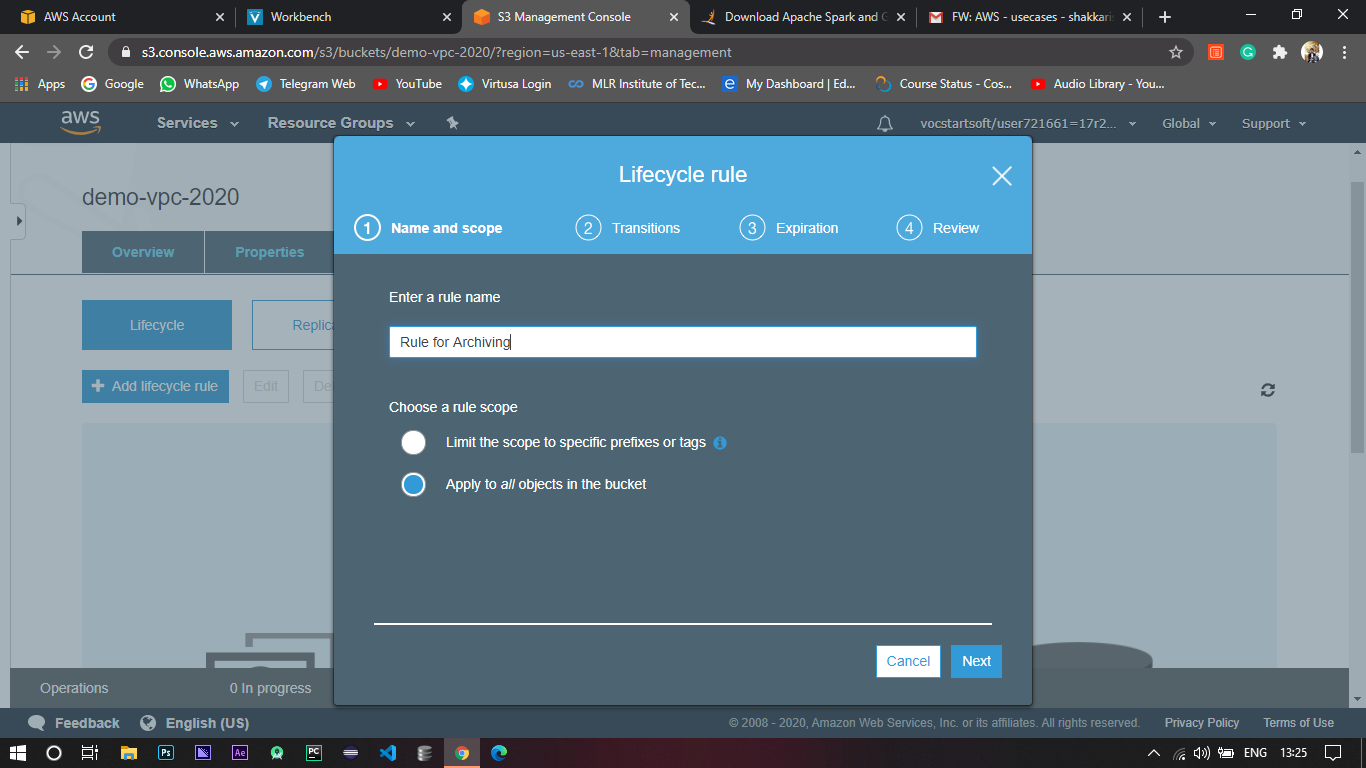
1. Select the mode of encryption. For now I’m using AES - 256(Advanced Encryption Standard) which is totally handled by Amazon.



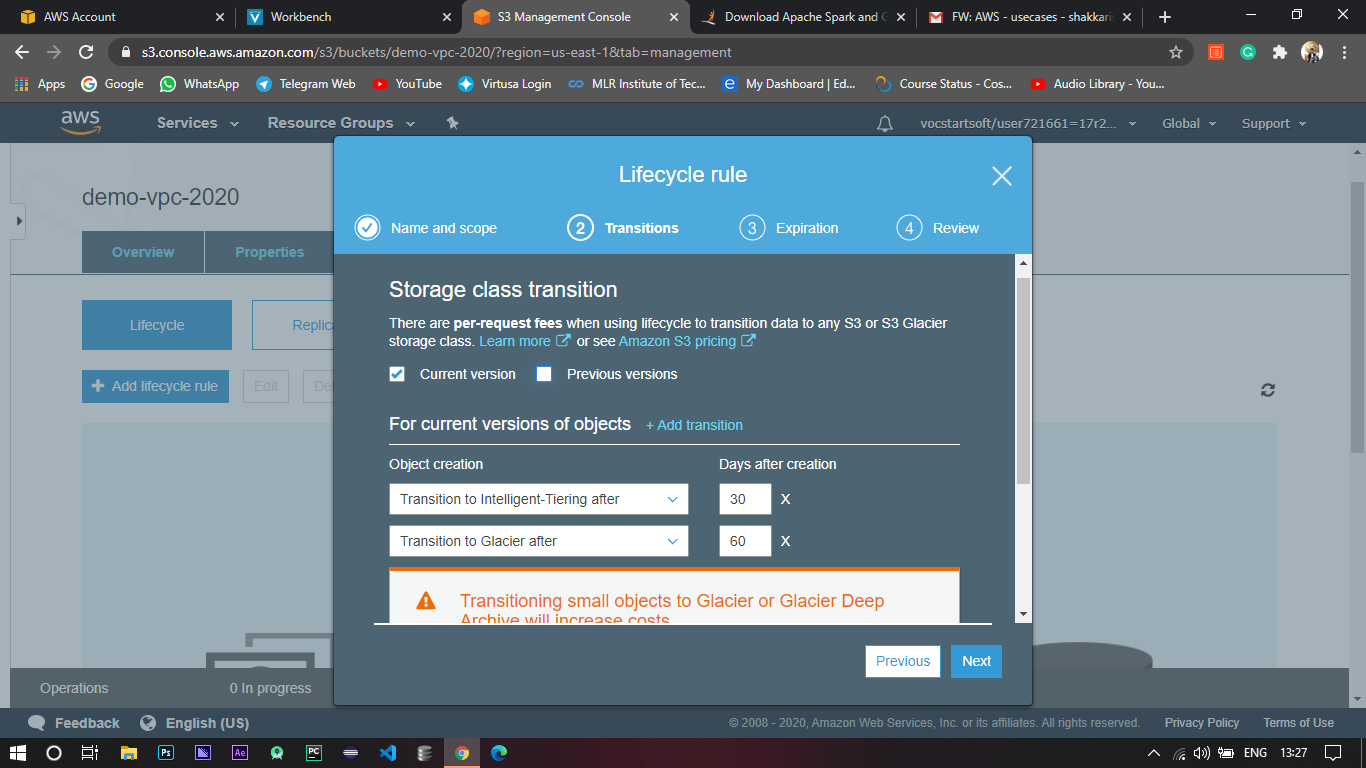
1. Then to achieve archival mechanism, go to Management -> Lifecycle.
2. Click on Add lifecycle rule.



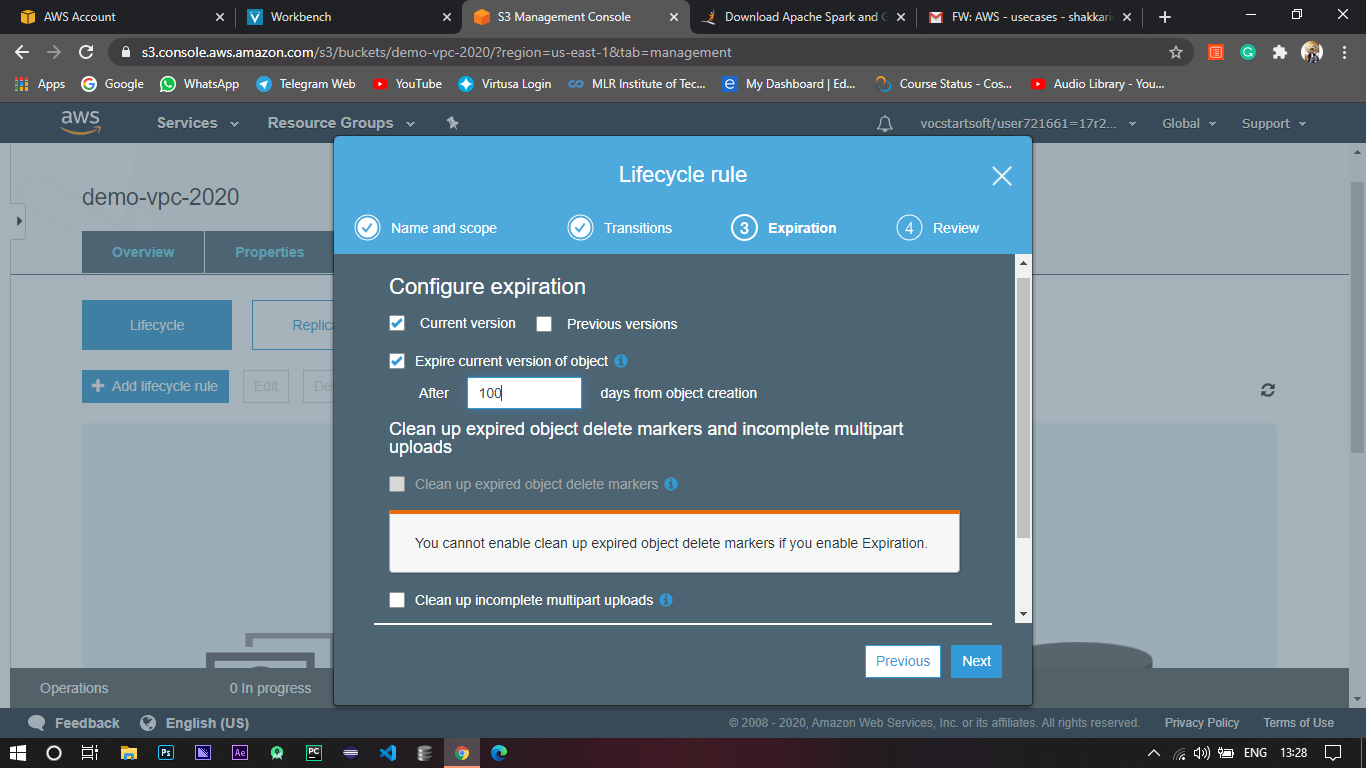
1. Give it a name and select the scope of the lifecycle rule.



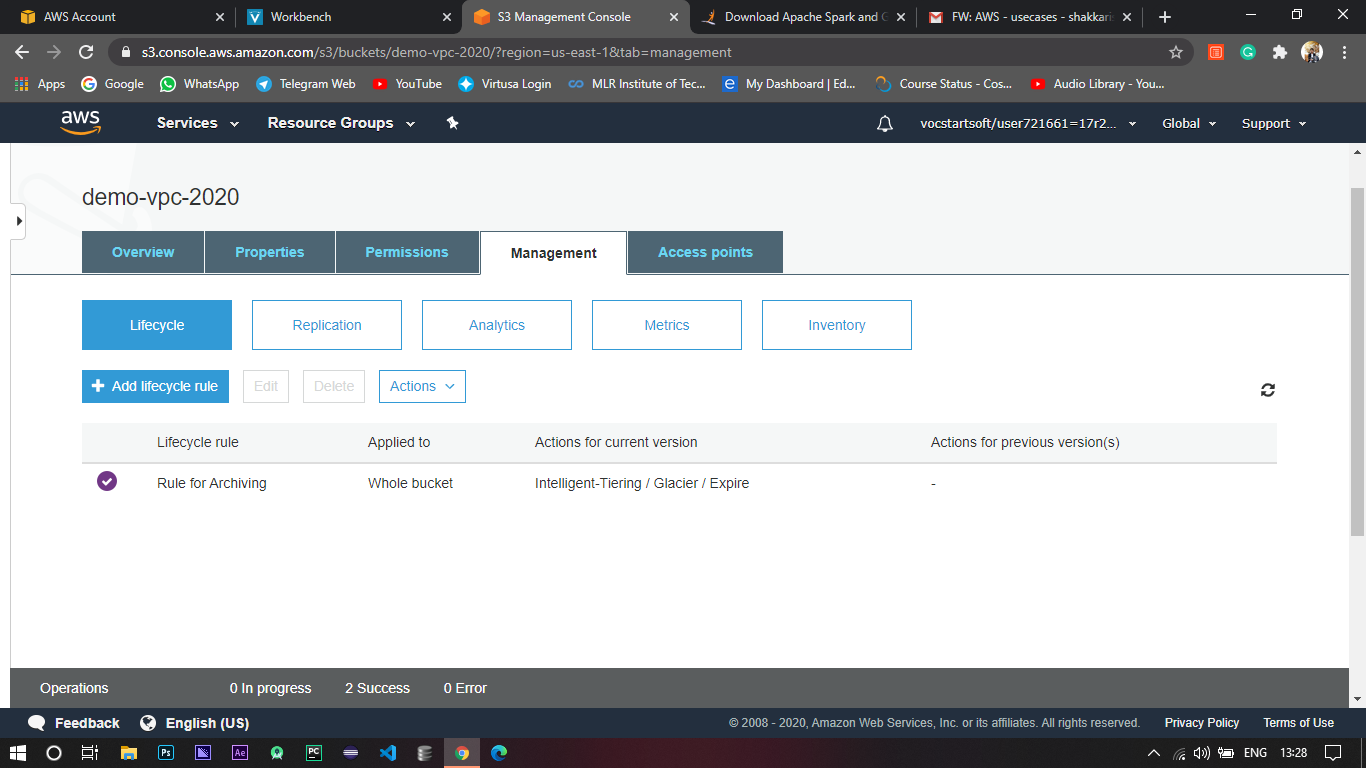
1. Here is the main part, Lifecycle rules can work along with versioning so you can apply transitions for both the current and previous versions.



1. For now I just applying it for the current version and giving two transitions that after 30 days of creation the object should transform from S3 standard to Intelligent Tiering and after 60 days it is transformed to Glacier.
2. Lastly set the expiry of the objects and click review.



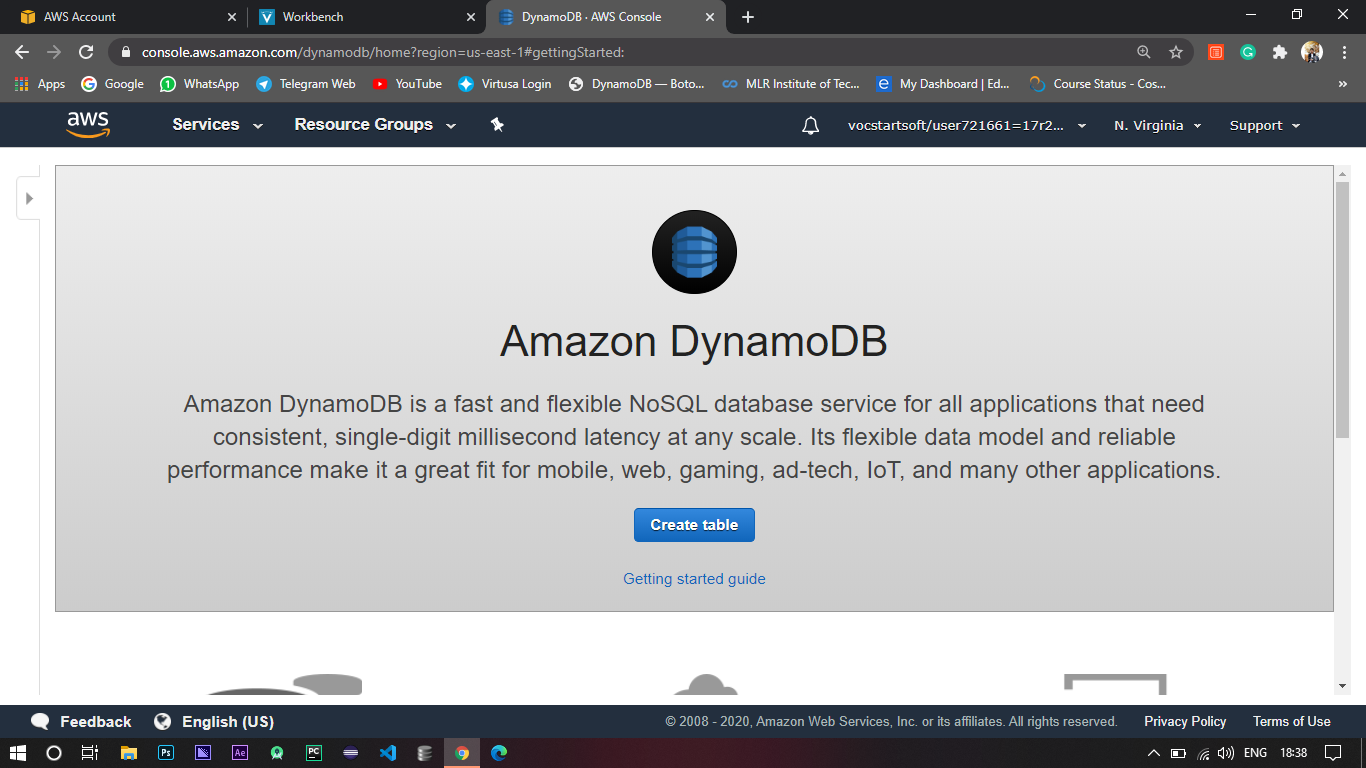
1. Review and click create.

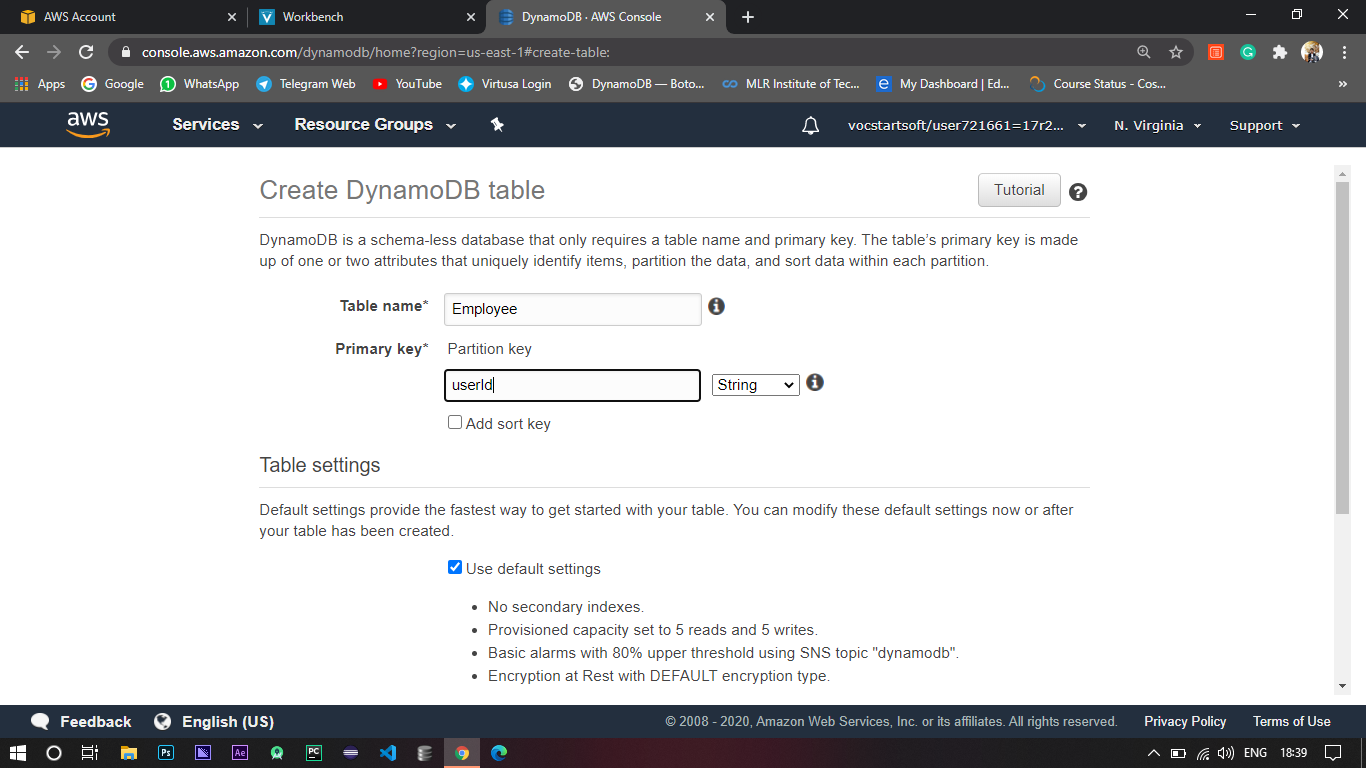


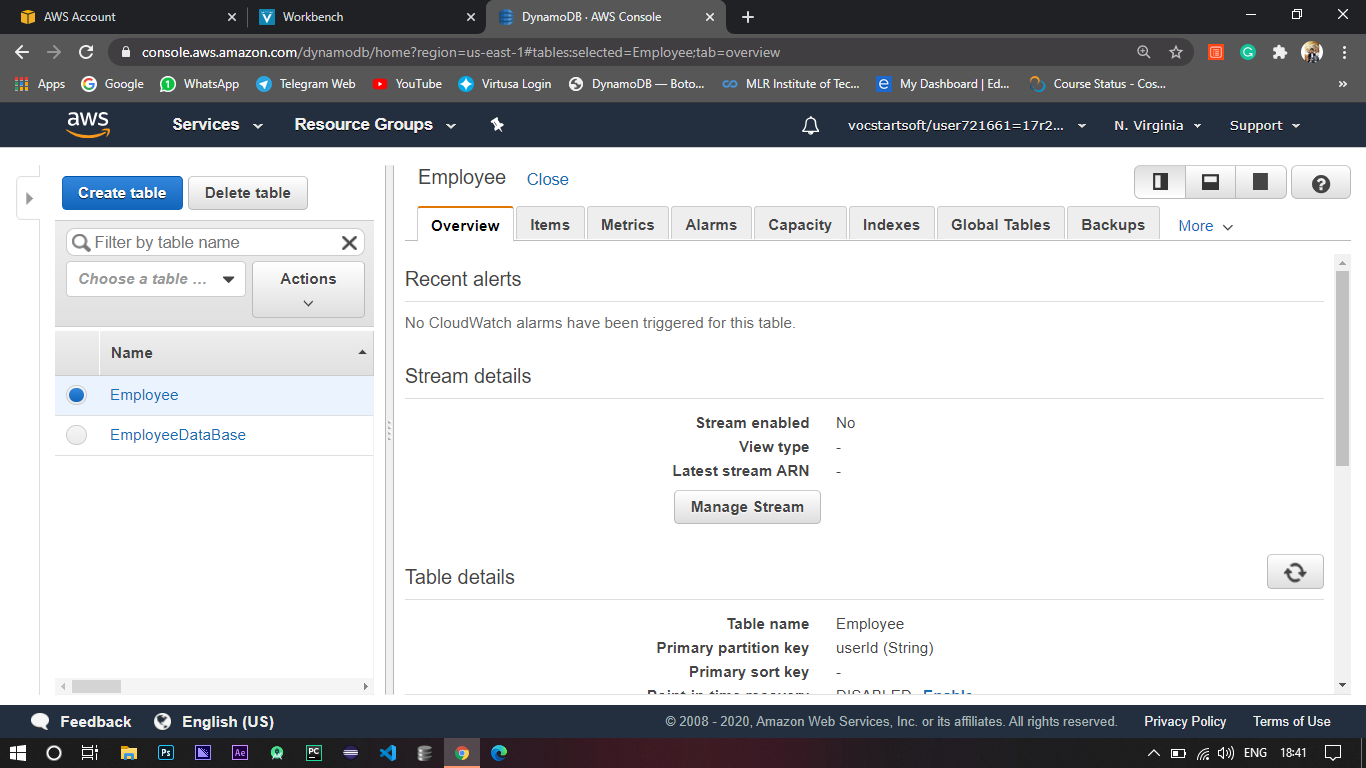
1. This is how it looks when everything went right.

# 4. Create lambda functions with S3 call, RDS call ( Ex : Whenever a new file is placed in specific folder any RDS instance like Dynamo DB function should triggered and update the table.

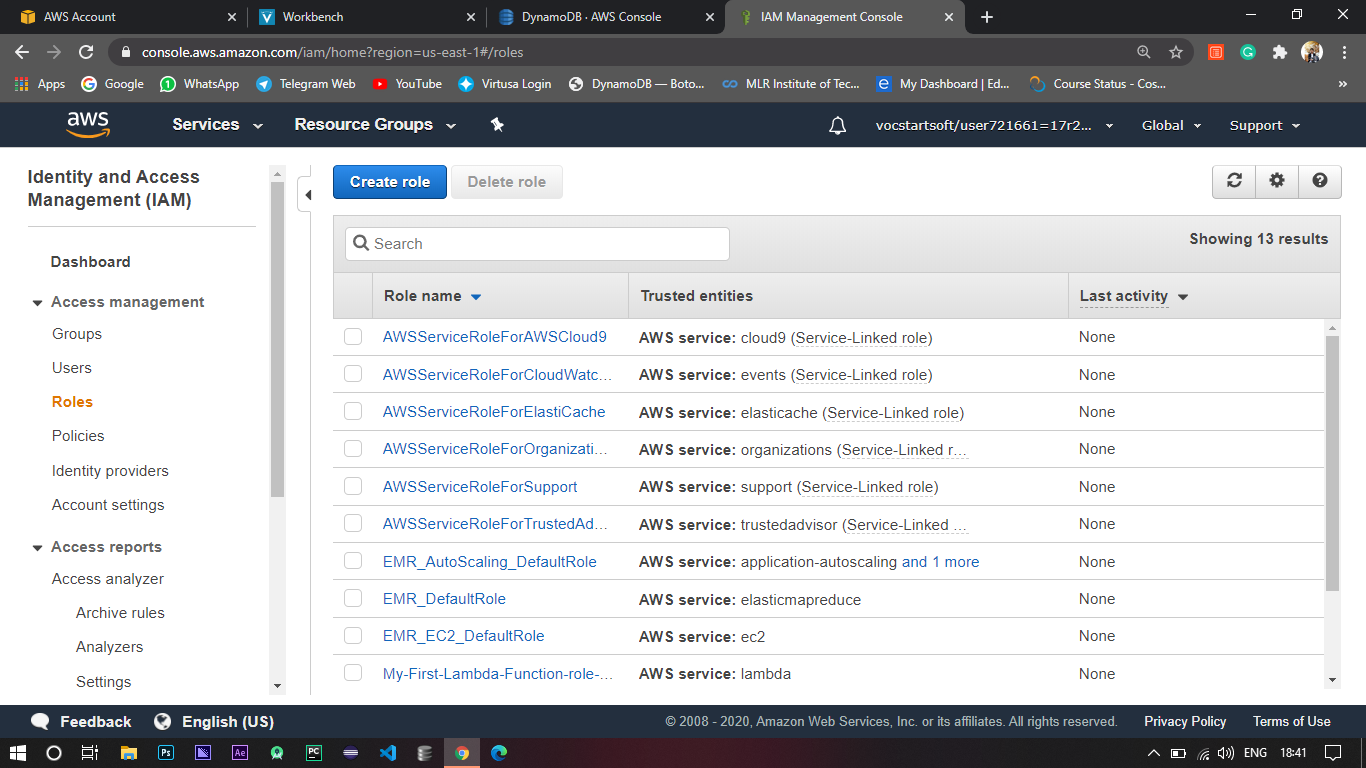
1. Login to AWS console.
2. Go to Dynamo DB under storage services.



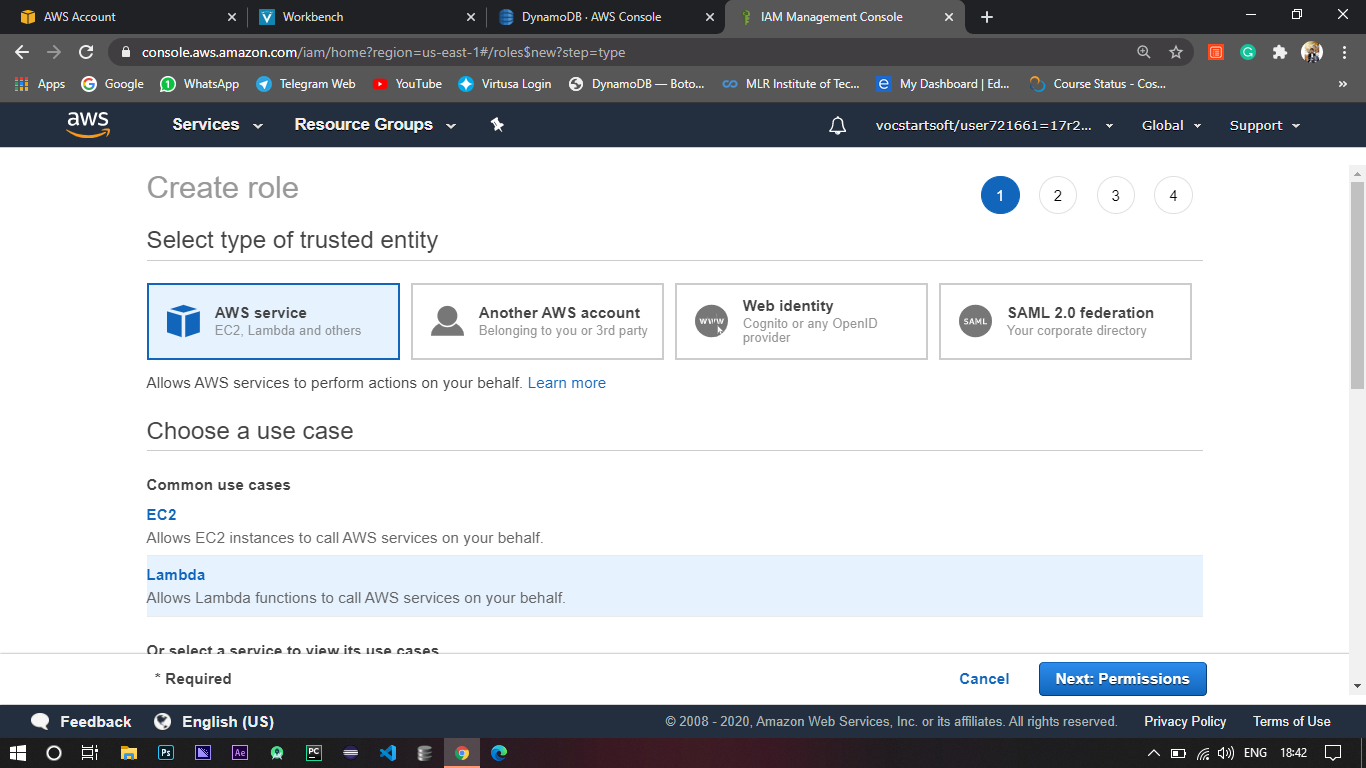
1. Click on create table.
2. Create two tables.
3. Configure the table upon your need.
4. For now I’m giving the default settings and created two tables with Table names as Employee and EmployeeDataBase and userId as Primary Key.



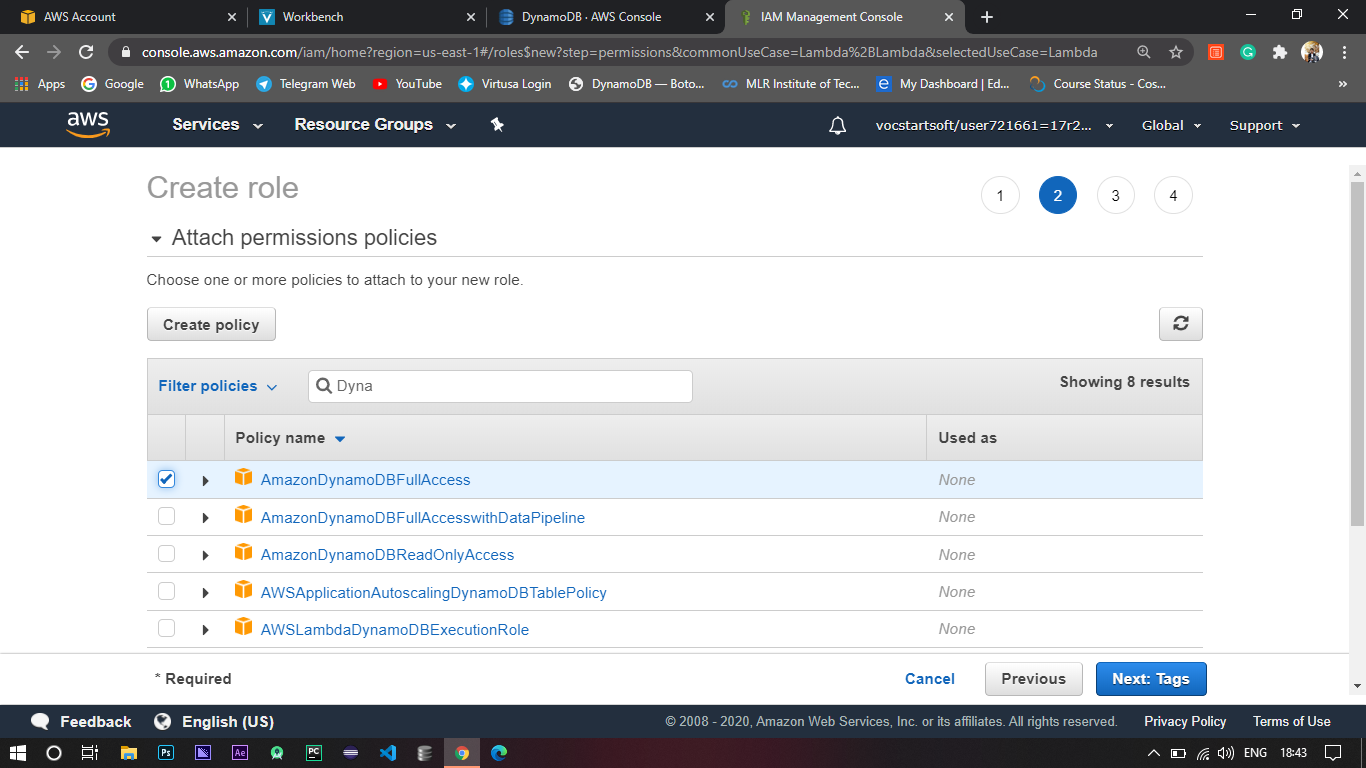
1. Now go to IAM and select on Roles.



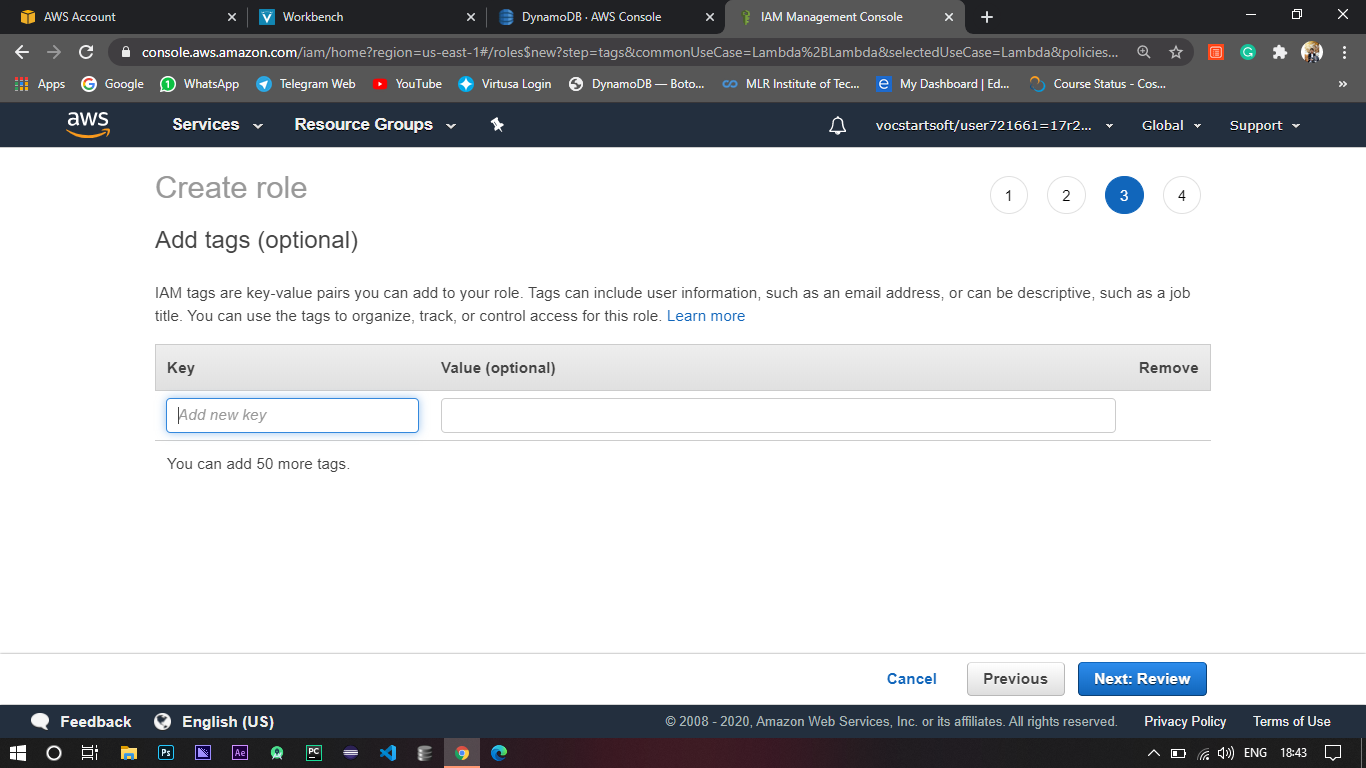
1. Click on Create role.
2. Select Lambda and go to permissions.



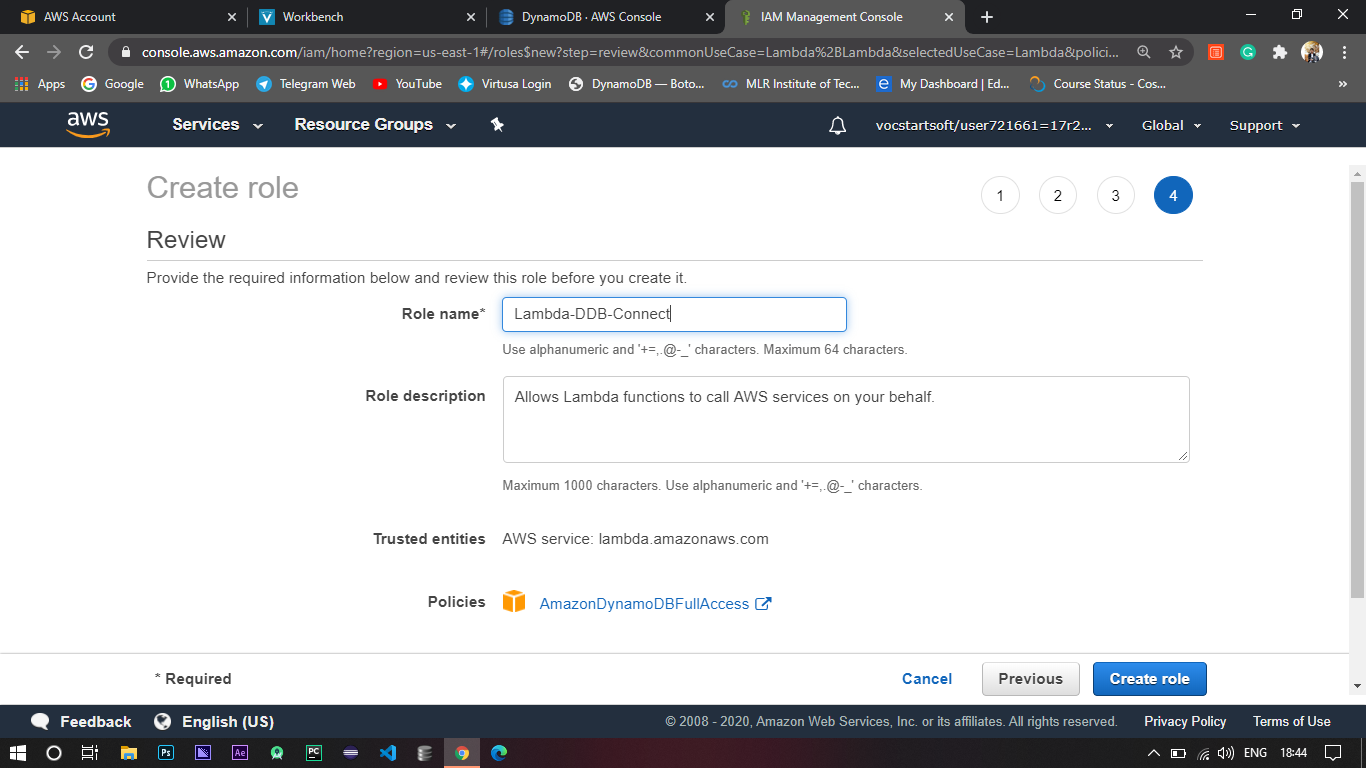
1. To be more secure you can give particular role to upon your need.



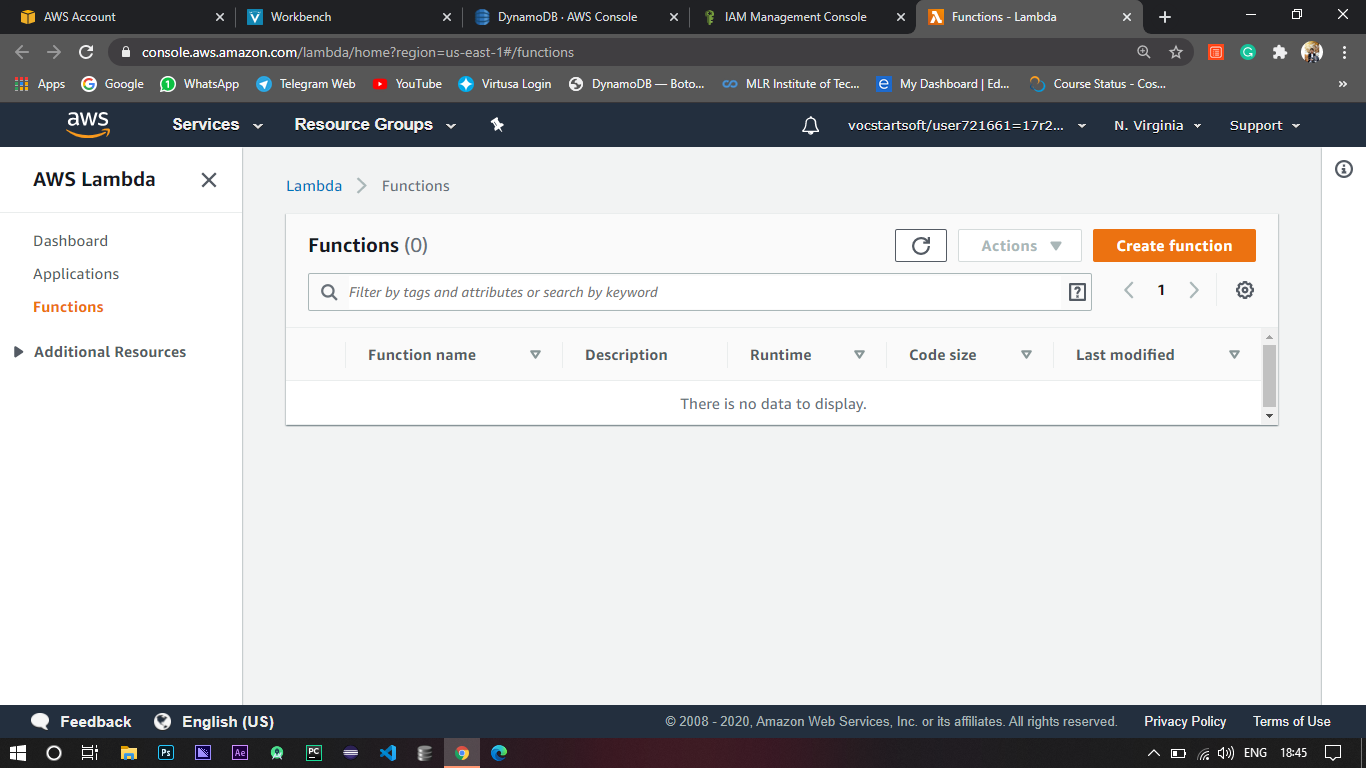
1. For now I’m giving DynamoDBFullAccess so that we can perform all the CRUD operations.



1. Add tags to filter in Future.



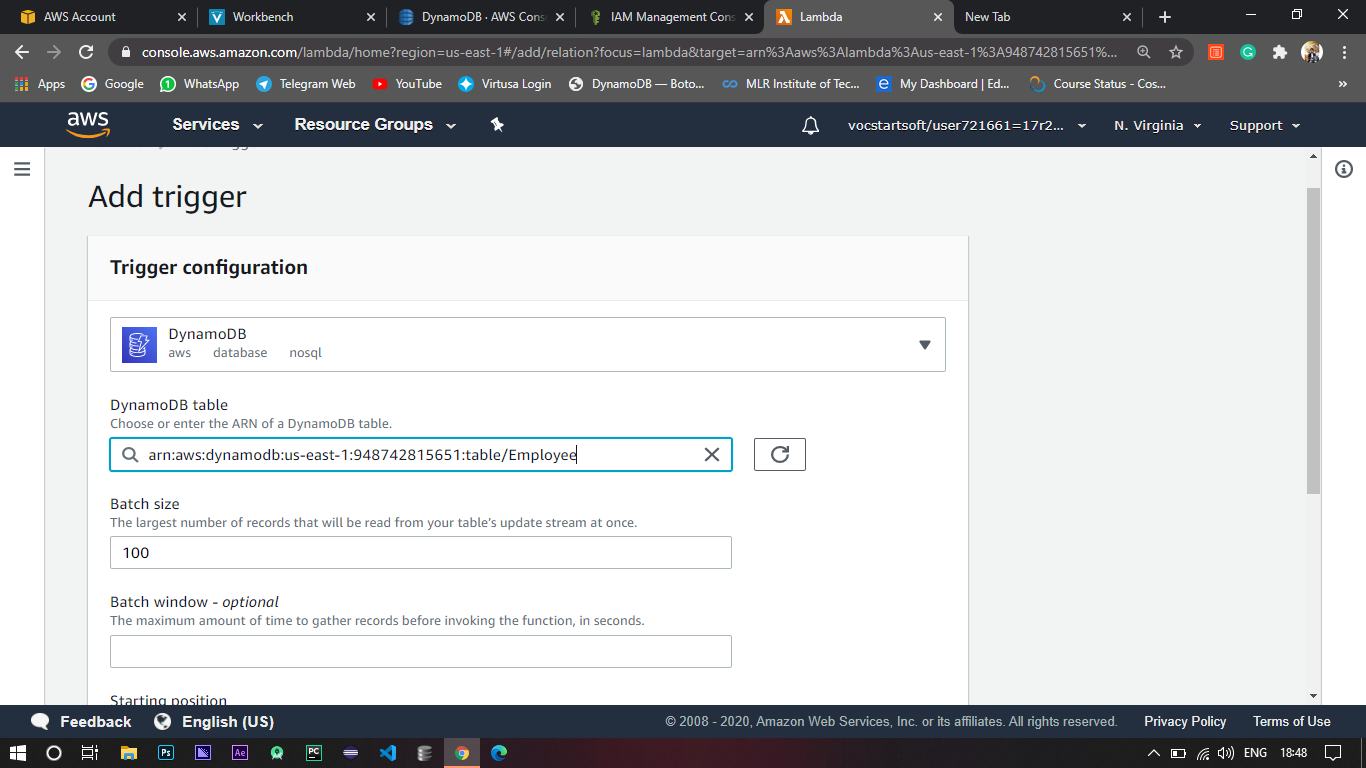
1. Give it a name and Review and create.
2. Lastly, go to Lambda and click on create function.



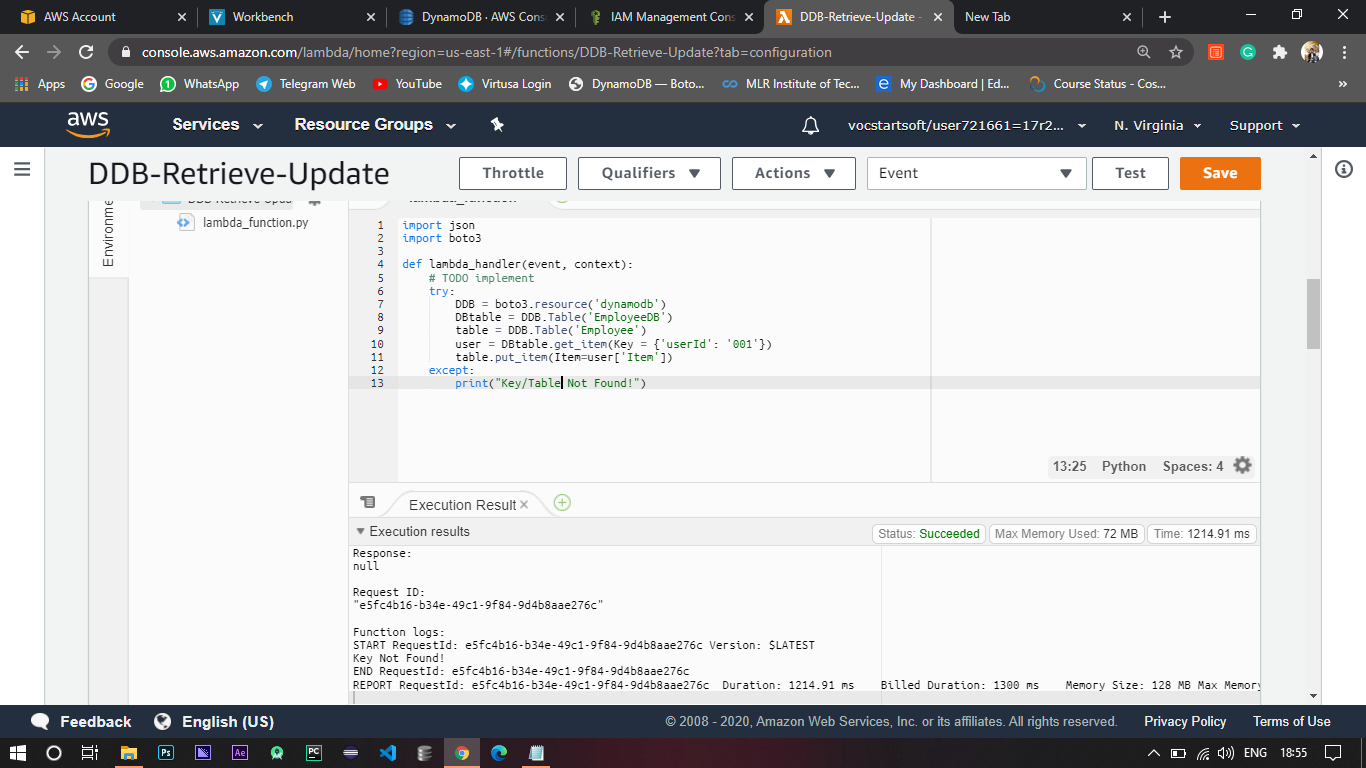
1. Configure the lambda function and give it a role and that has been created earlier and create the function.



1. Add trigger to the function by giving the Employee DynamoDB table.



1. Coming to coding we are using the boto3 sdk to access the resources in AWS.



import boto3

def lambda\_handler(event, context):

try:

DDB = boto3.resource('dynamodb') # Used to access the dynamodb

DBtable = DDB.Table('EmployeeDB') # Accessing tables

table = DDB.Table('Employee')

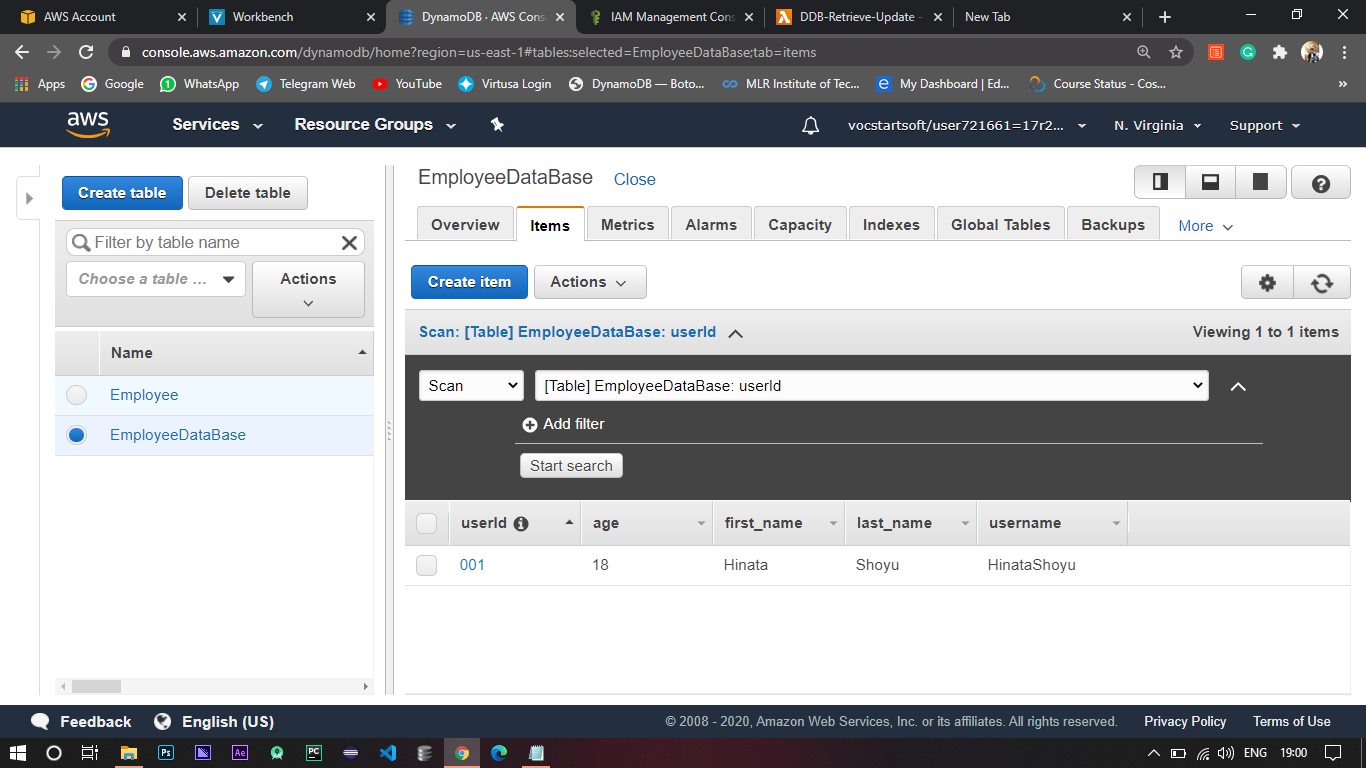
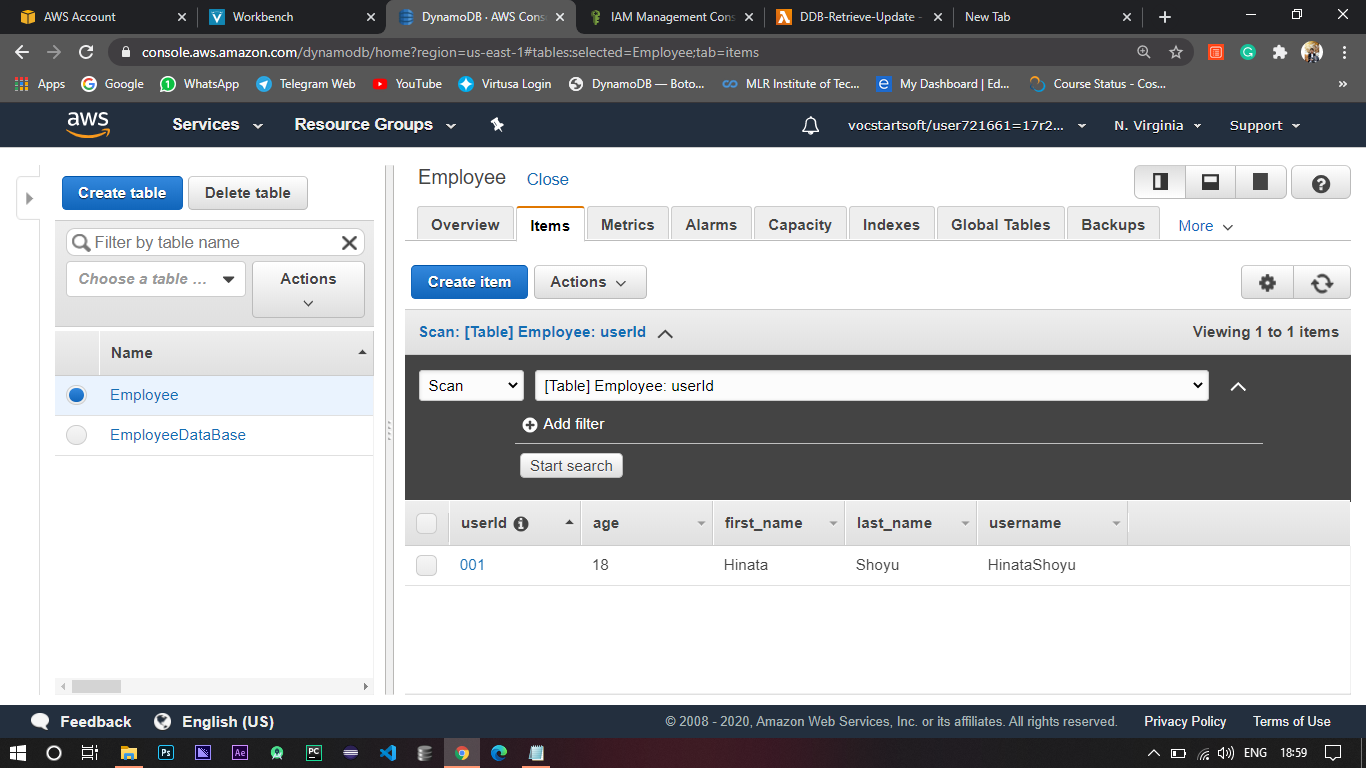
user = DBtable.get\_item(Key = {'userId': '004'}) # Getting the record by using the key

table.put\_item(Item=user['Item']) # Inserting the Record

except:

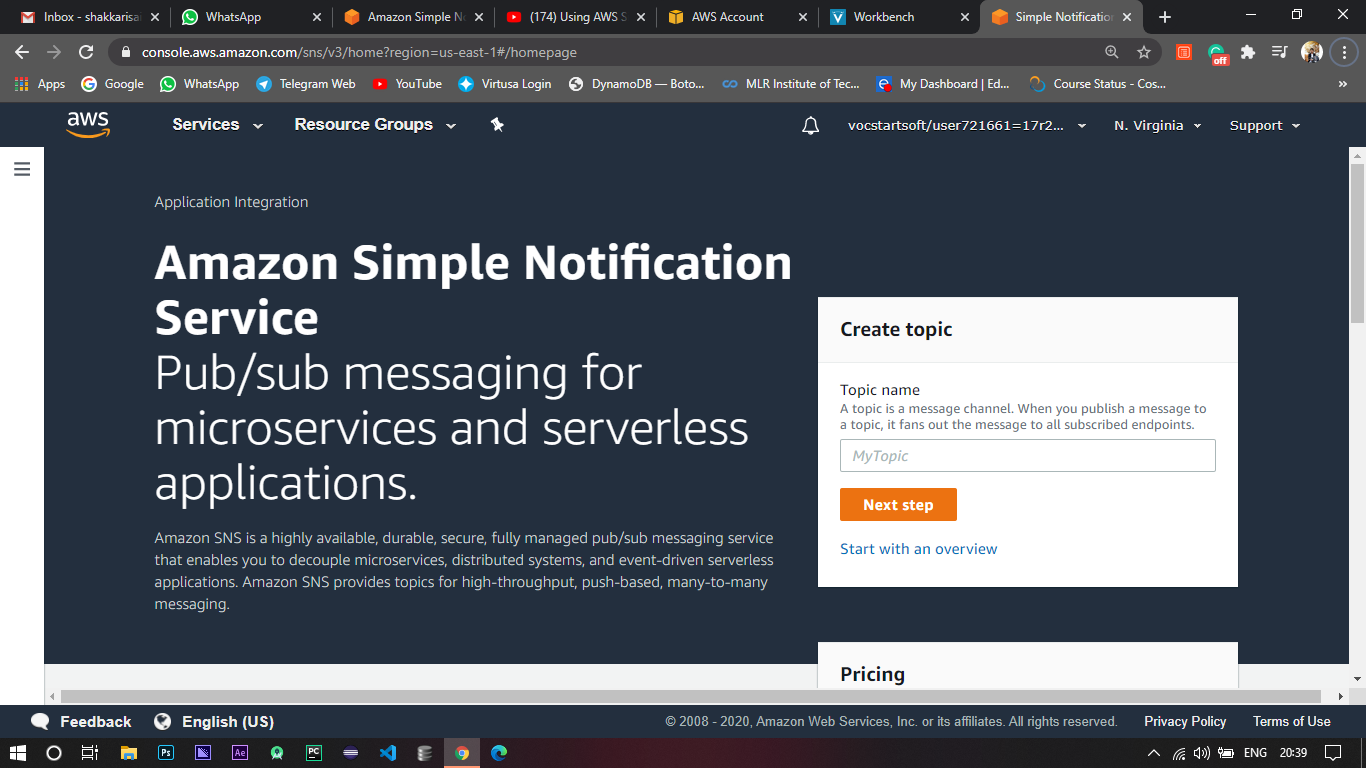
print(“Key / Table Not Found”) # If any Occurs this code will be executed

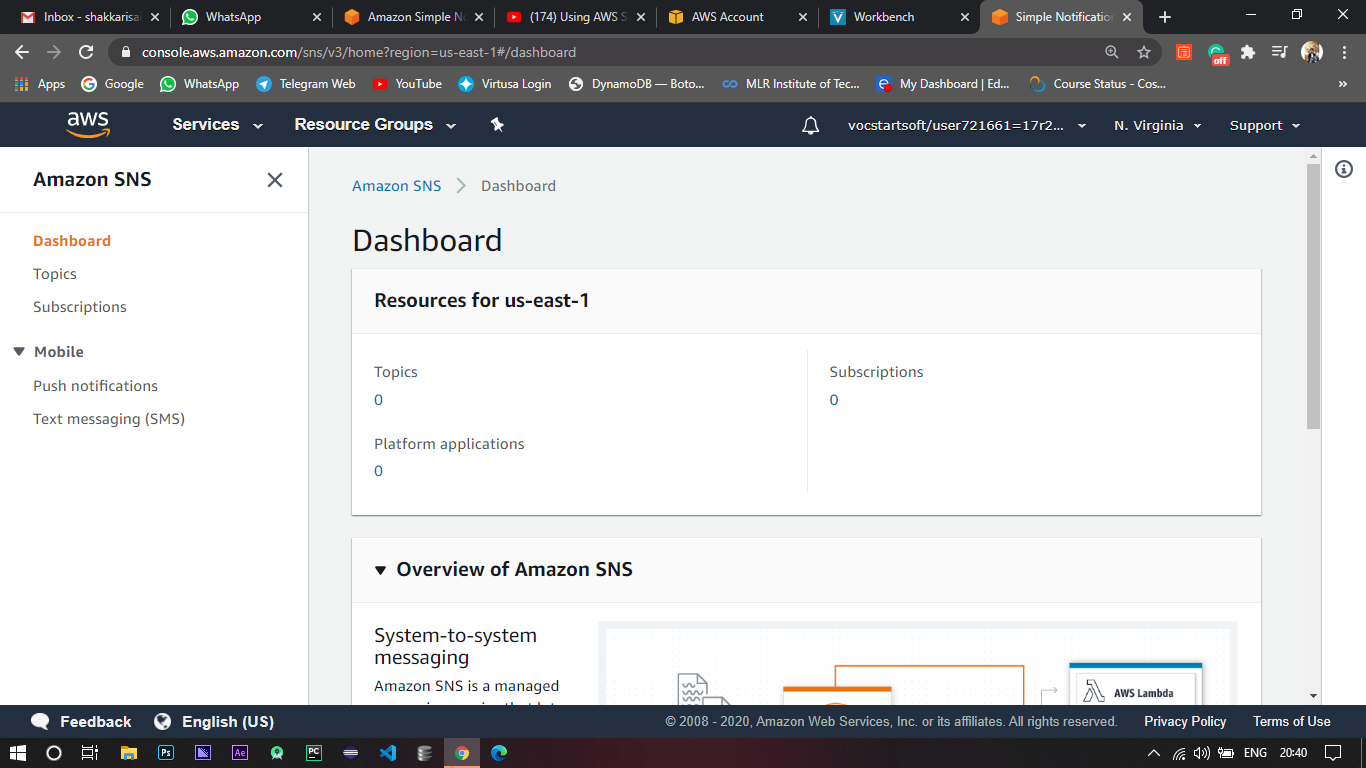
1. Save and test the code.
2. Hence whenever an entry is added to the Employee table with the required key then that entry is retrieved and added to the EmployeeDataBase table.

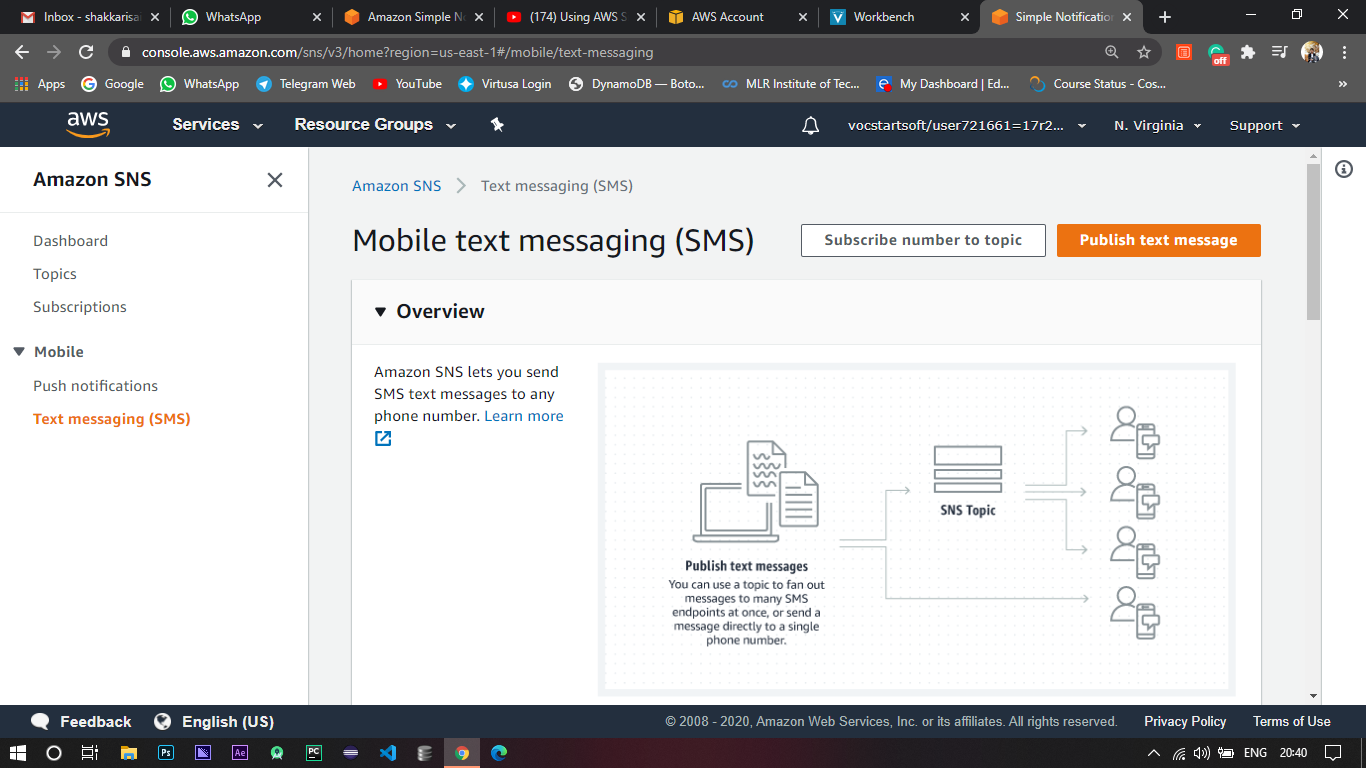


# 5. Create SNS/SQS tasks (Sending text message to mobile).

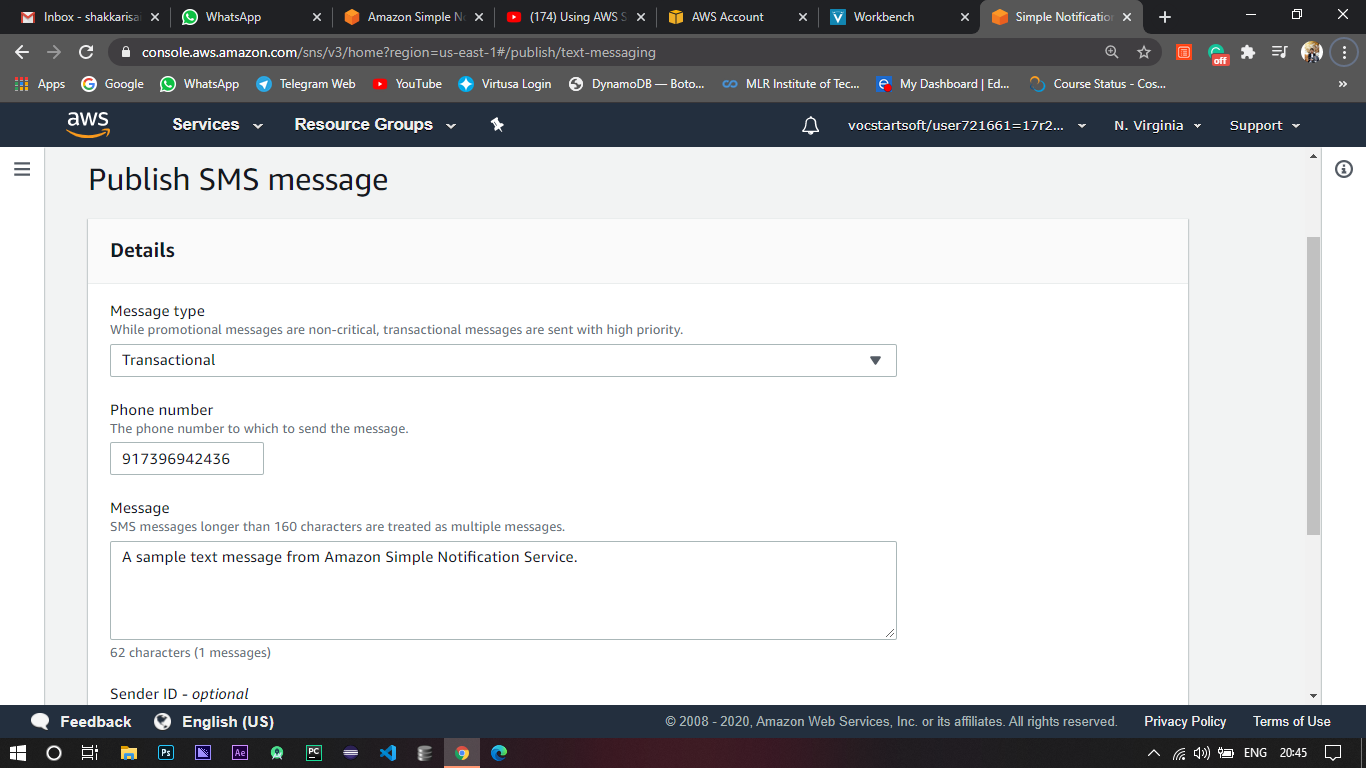
1. Log in to AWS Console.
2. Select SNS from AWS services which is under application integration.



1. Click on Start with an overview which will redirect to the Dashboard of the SNS.
2. Go to text messaging (SMS) where you can see all the information of the messages you have sent till now.



1. Click publish text message to send an SMS to a phone number.



1. Configure by selecting promotional / transactional messages where promotional are non-critical, transactional messages are sent with high priority.
2. Give the phone number to which you want to send the message and publish it.

