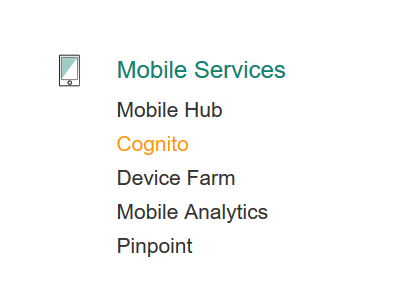
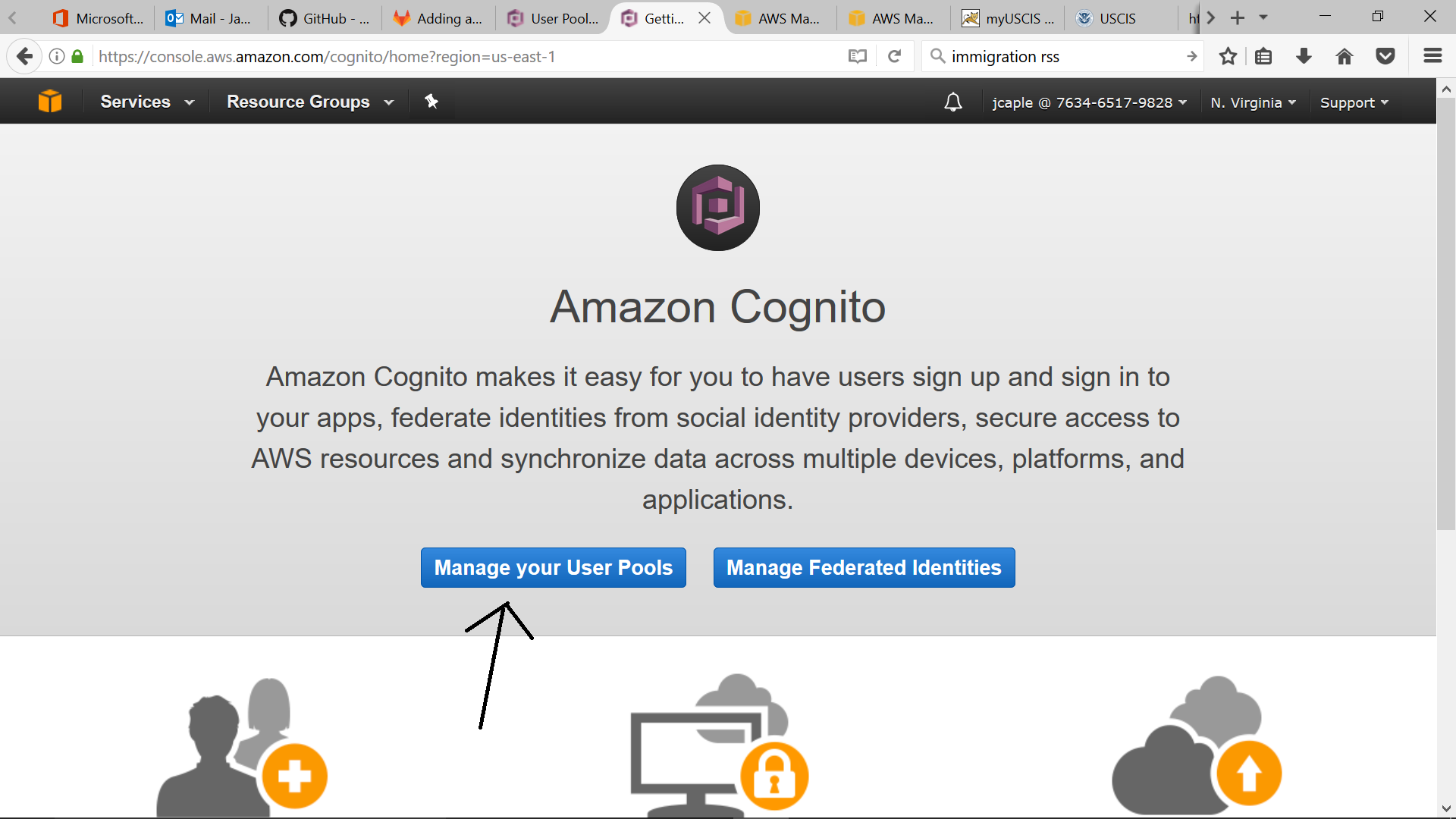
# AWS TICS II Tech Demo Infrastructure

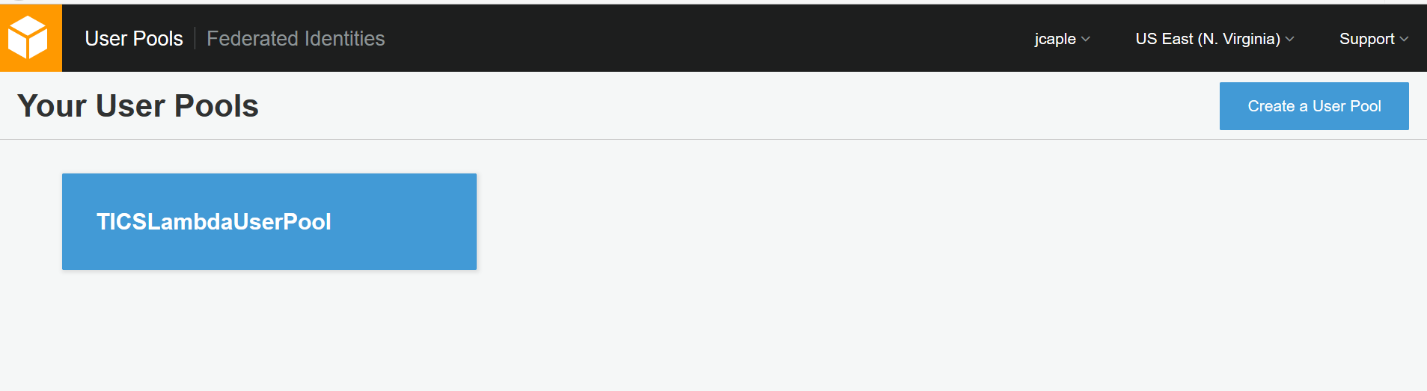
## Cognito User Pools

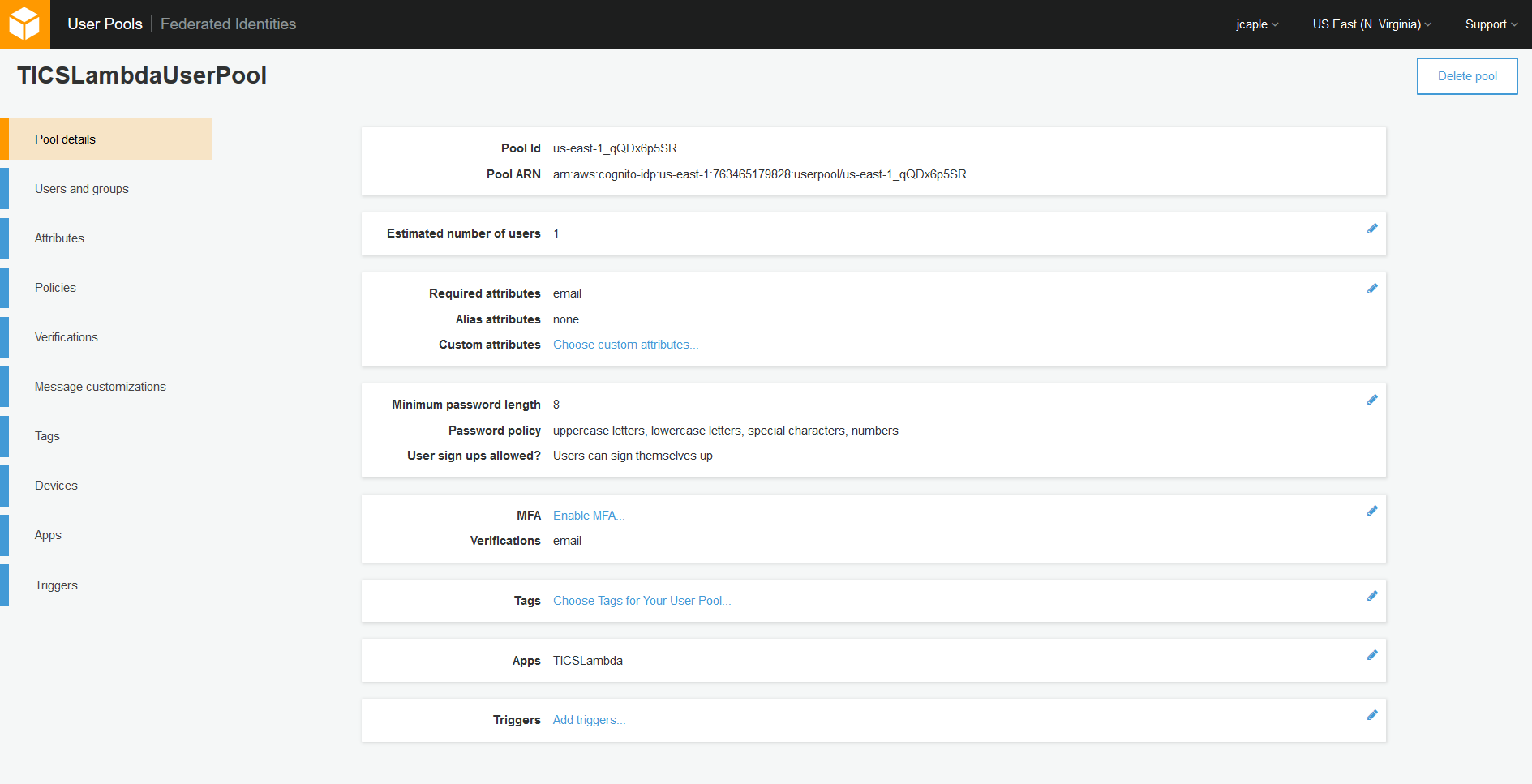
Create a Cognito User Pool named ‘TICSLambdaUserPool’. Key Identifiers generated in this process will be used in our application security configuration, namely:

1. Region (e.g. ‘us-east-1’)
2. UserPoolId (e.g., ‘us-east-1\_qQDx6p5SR’)
3. ClientId (e.g., ‘584a9pau7pmmie35d3t9mv1c4s’)

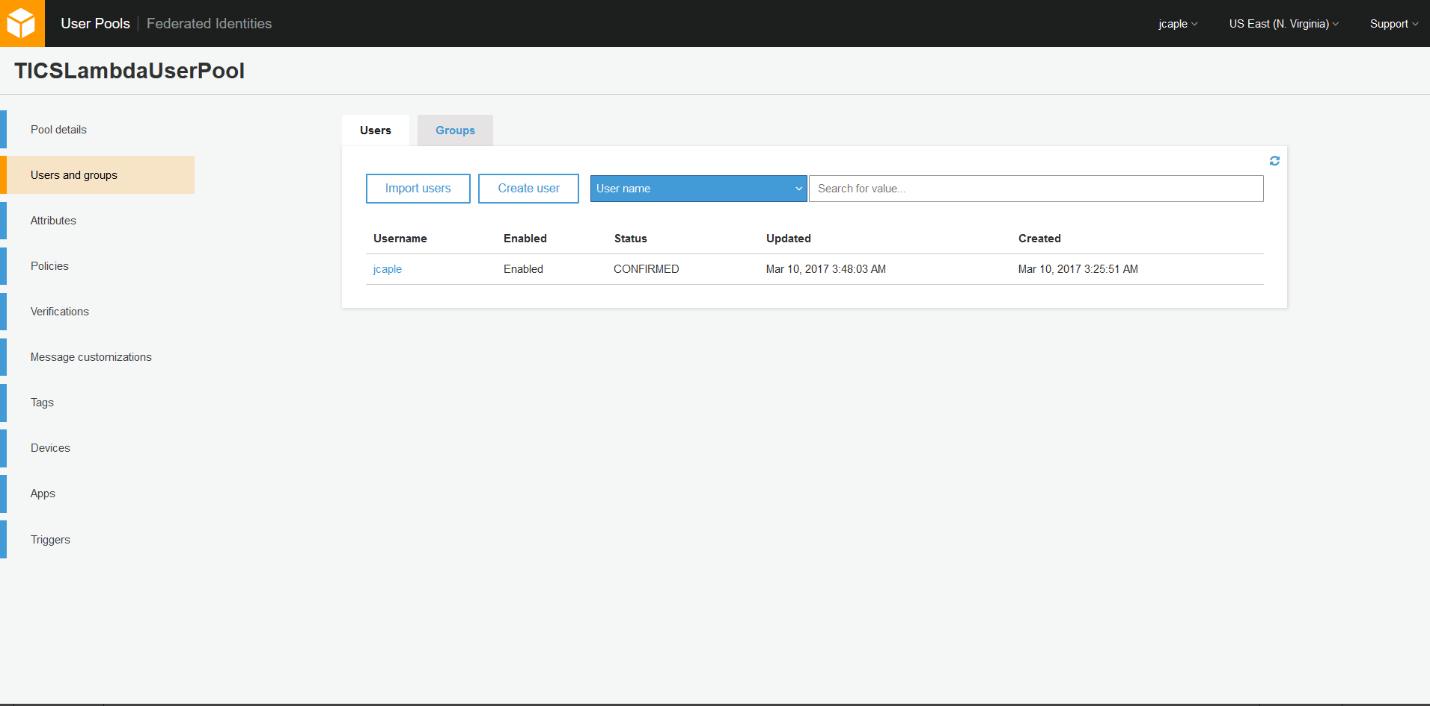




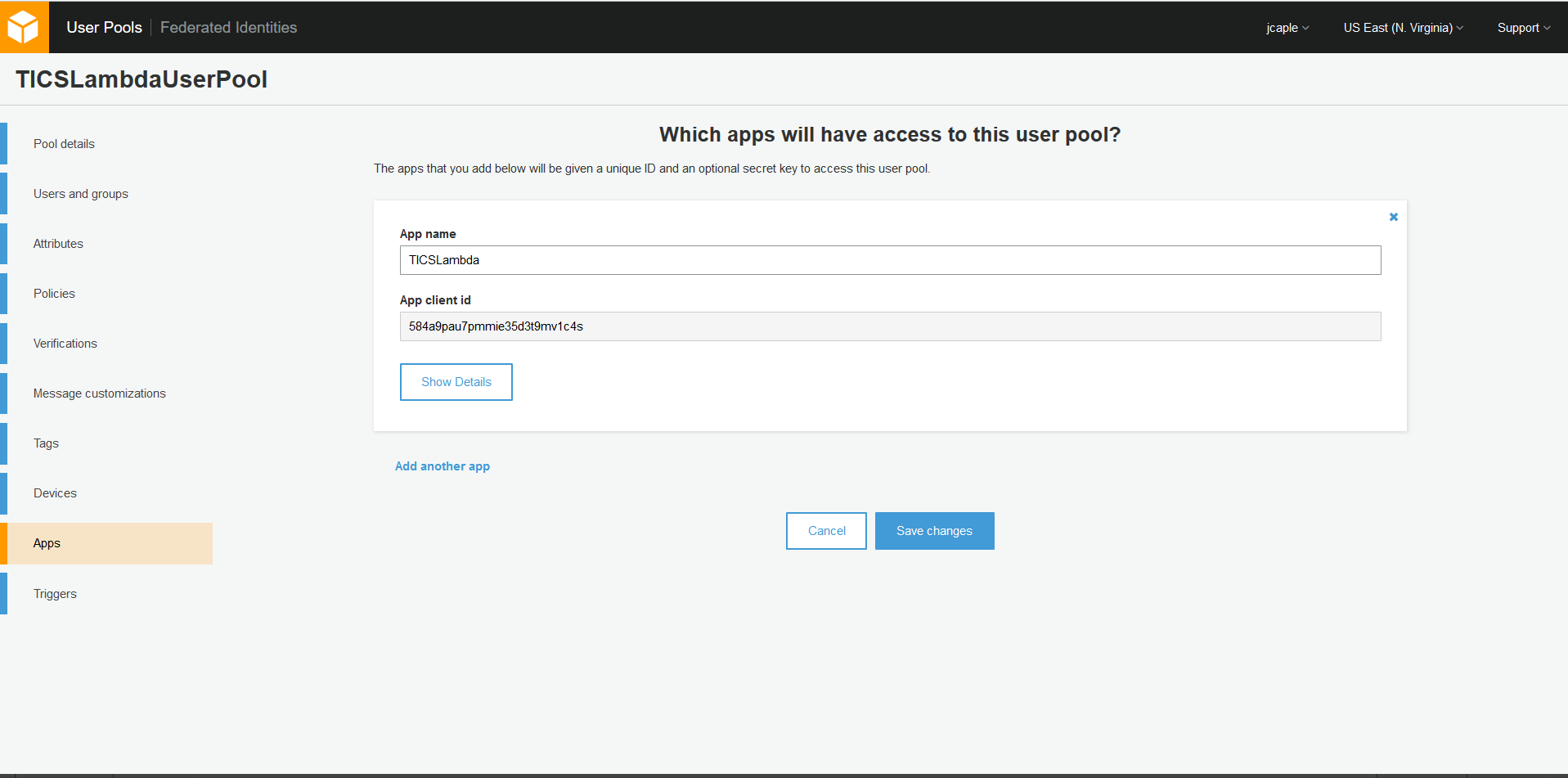




Users we want to have access to the application will be added in the ‘Users and Groups’ section.



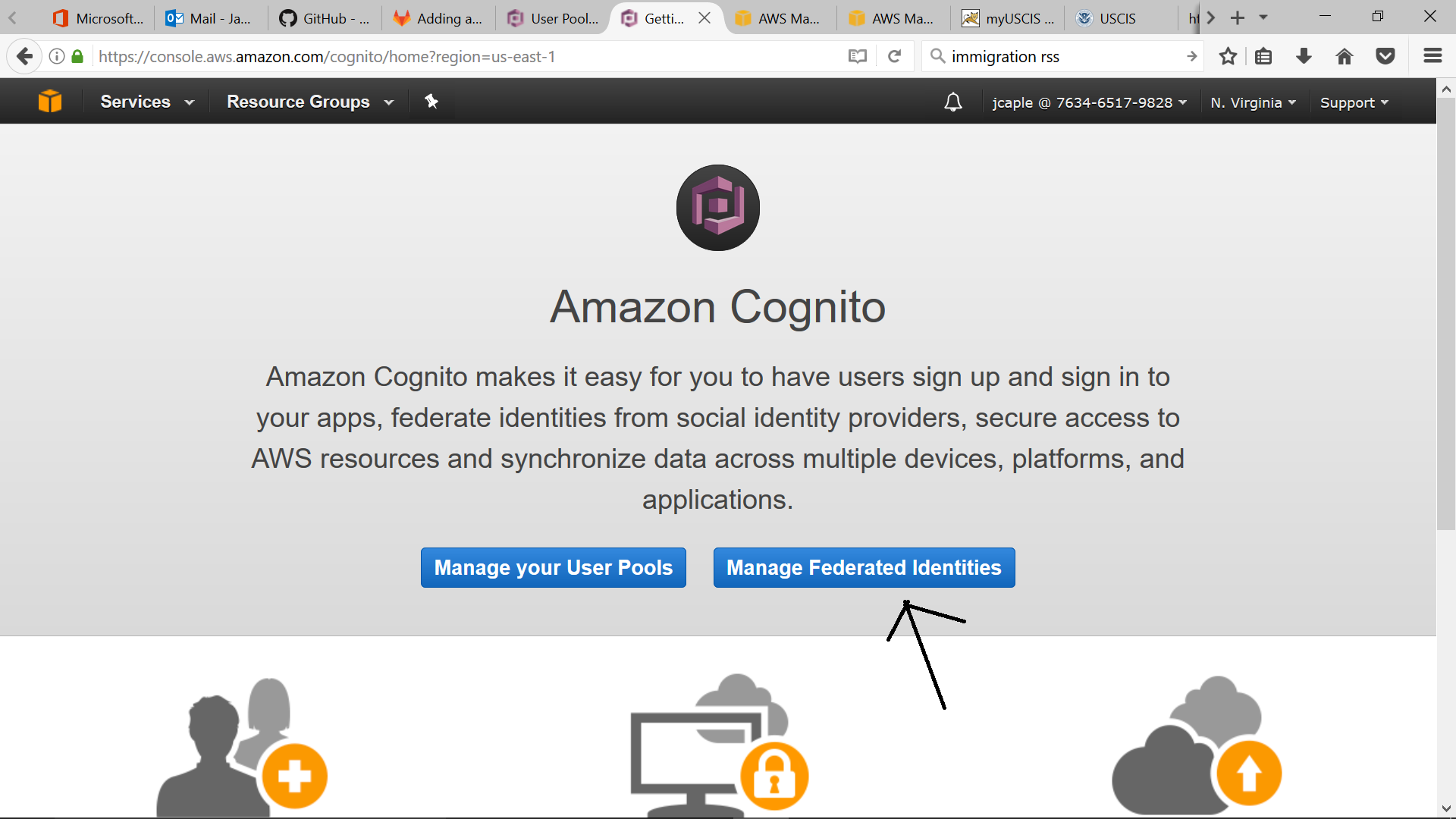
Create a new App to associate with the User Pool and note the generated App client id:



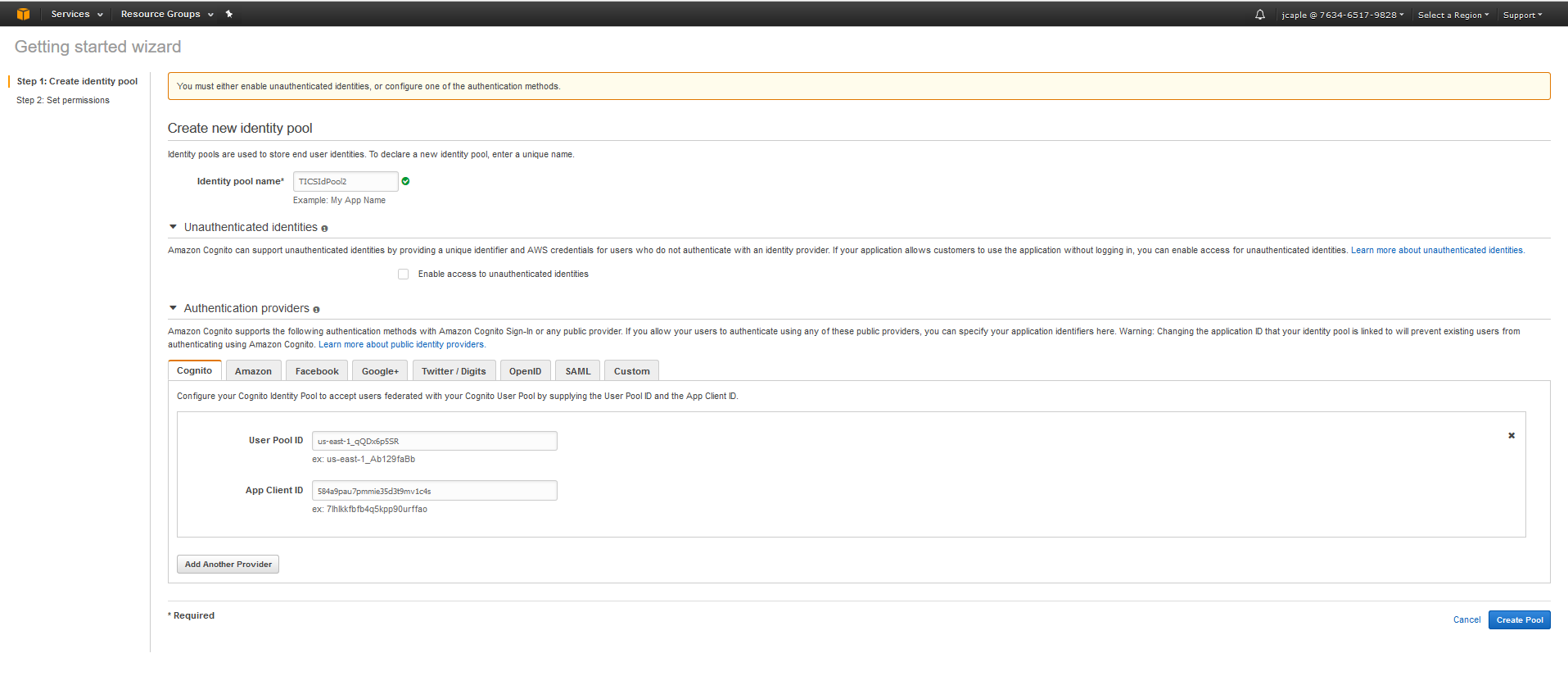
## Cognito Federated Identities

Next, create a new Identity Pool. At the end of this step, you want to note the IdentityPoolId value:

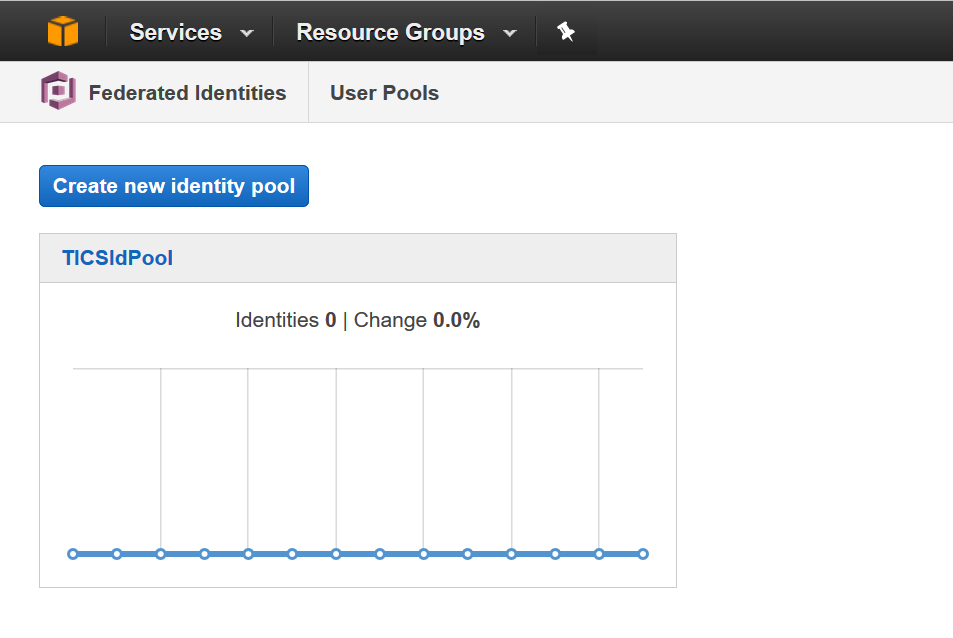
1. IdentityPoolId (e.g., ‘us-east-1:0ad49232-e0db-4717-bf08-3e2ca35507c1’)

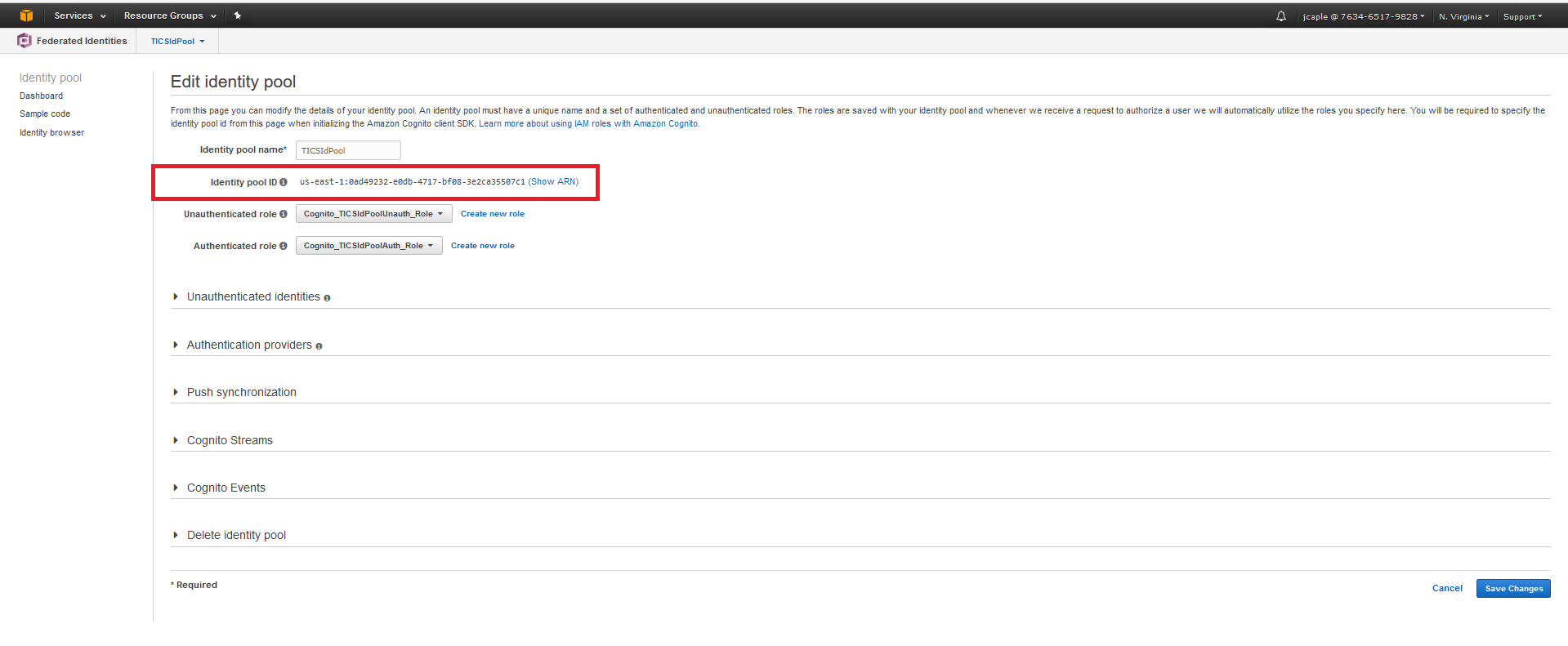


Please call it ‘TICSIdPool’:



Do not check ‘Enable access to unauthorized entities’. Provide the Cognito User Pool ID and App Client ID noted from the previous steps creating the User Pool.

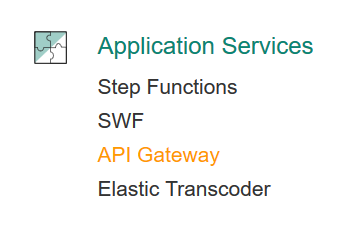




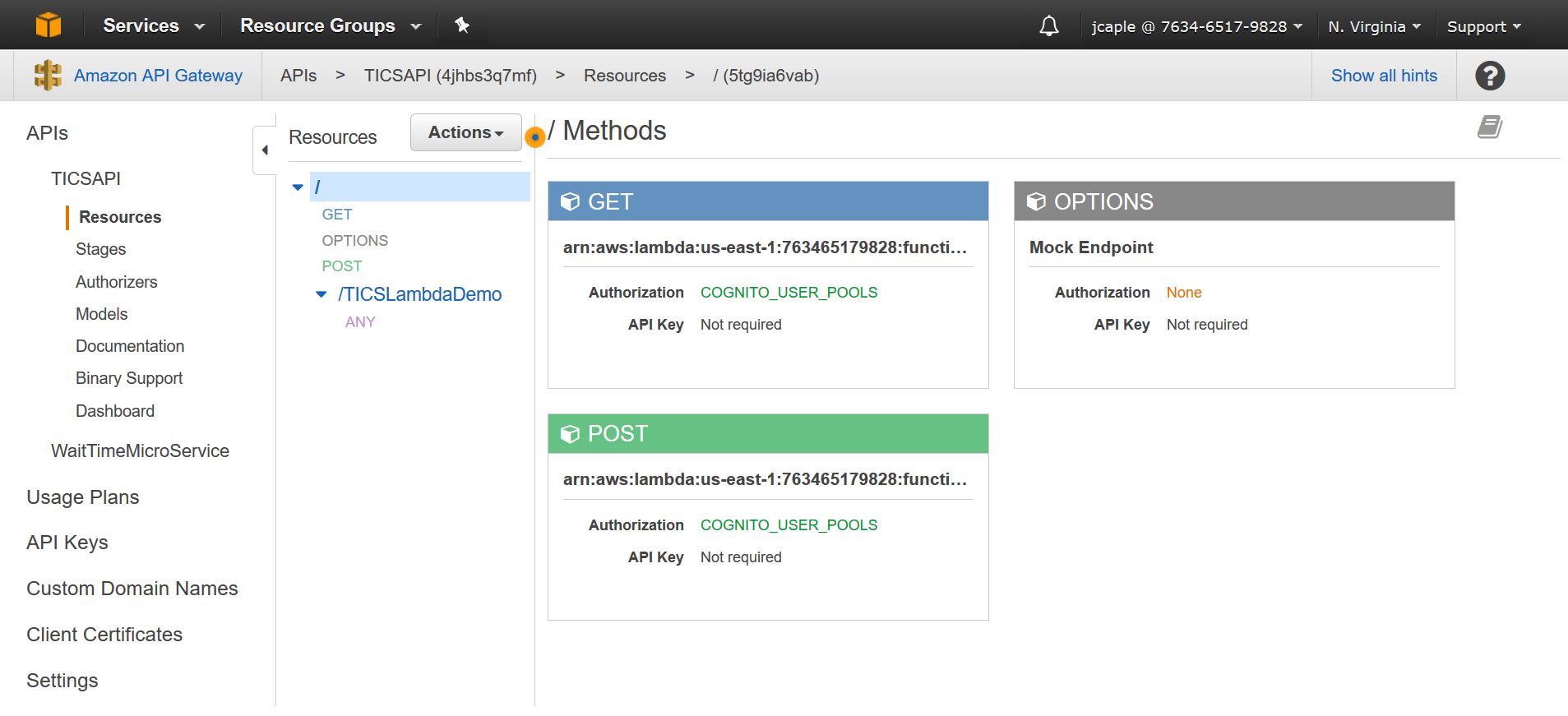
And that should be it for the User Authentication piece.

## API Gateway

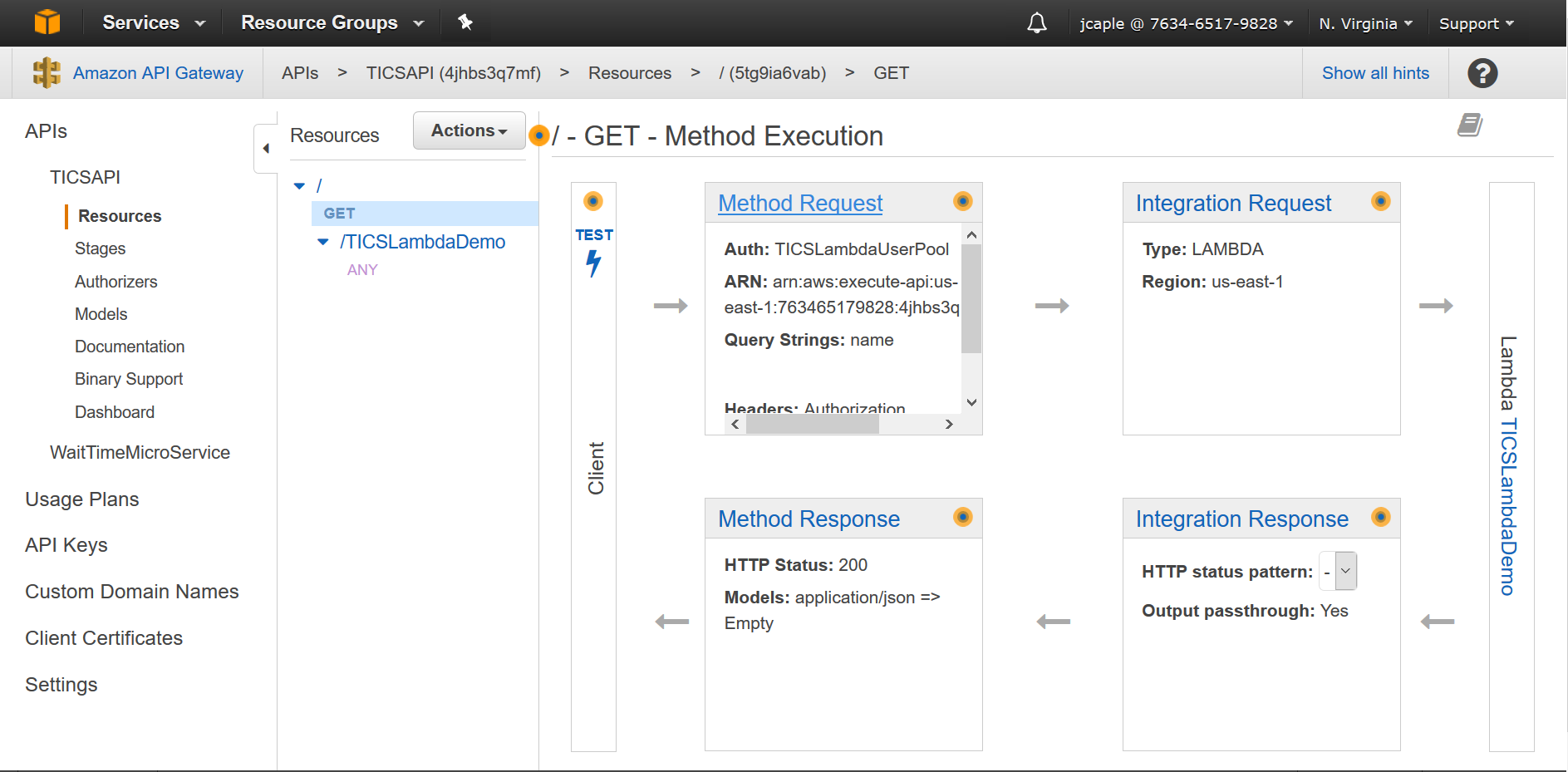
I think we only need one of these. The API Gateway provides https routes to our Lambda Gateway Function. I envision having multiple deployments for the one API Gateway configuration (Dev, Test and Prod).



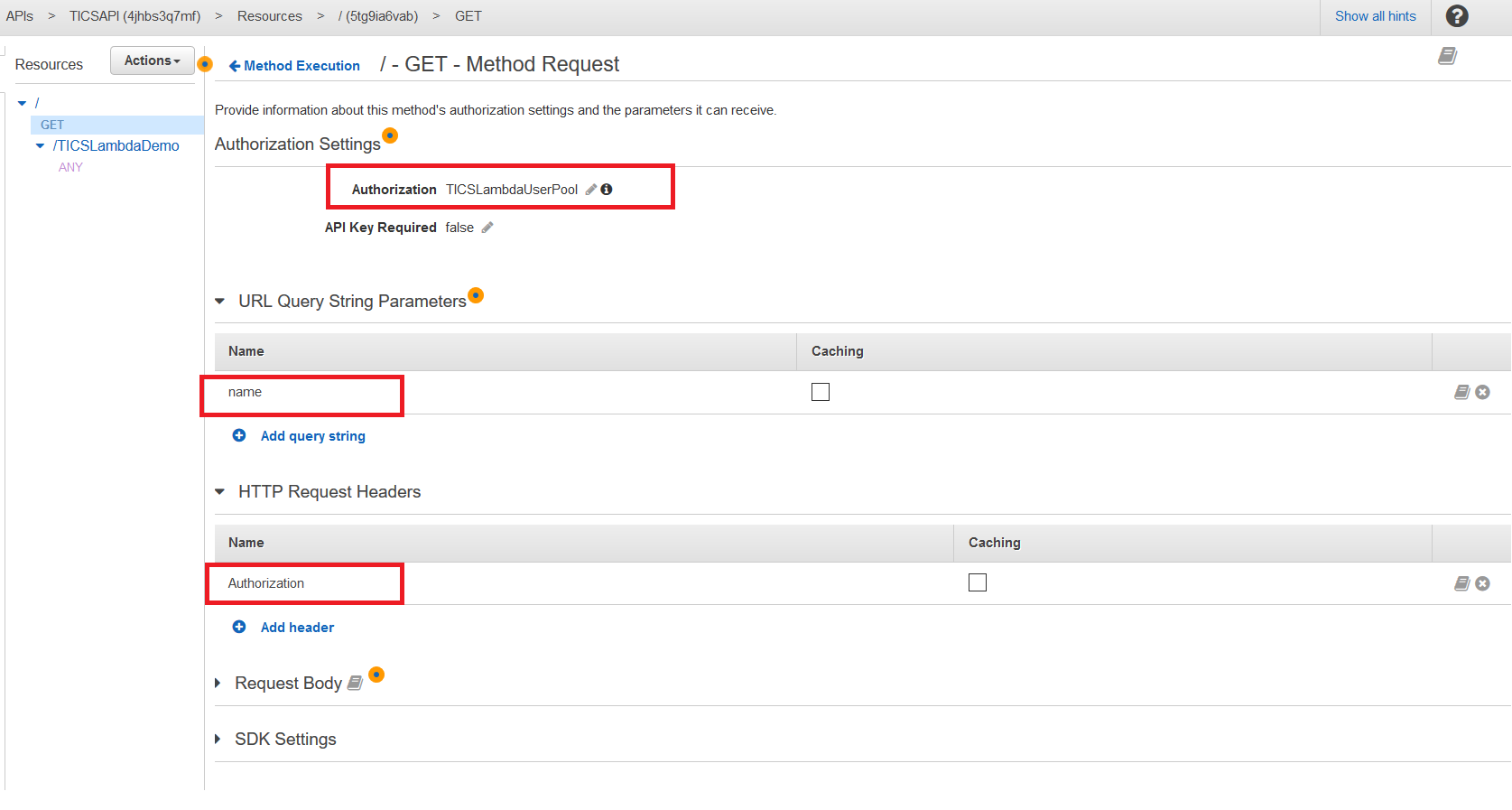
Create a new API call ‘TICSAPI’, which supports ‘GET’ and ‘POST’ operations:



Click the ‘GET’ method in the Resources column, and then click on the ‘Method Request’ link:



You should now see this screen:

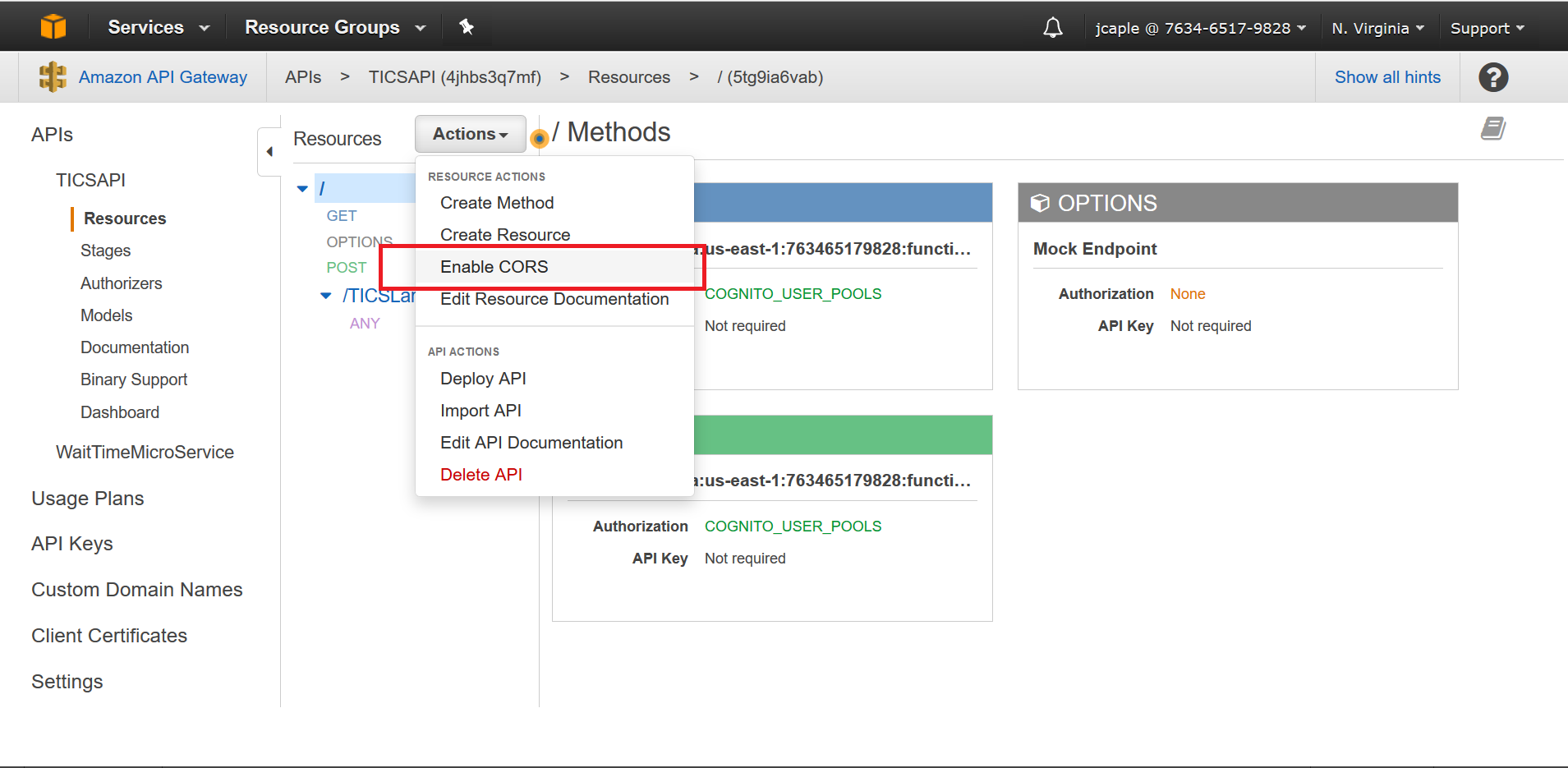


Make sure that the API Authorization is set to the Cognito User Pool name, ‘TICSLambdaUserPool’.

Add ‘name’ as a URL Query String Parameter.

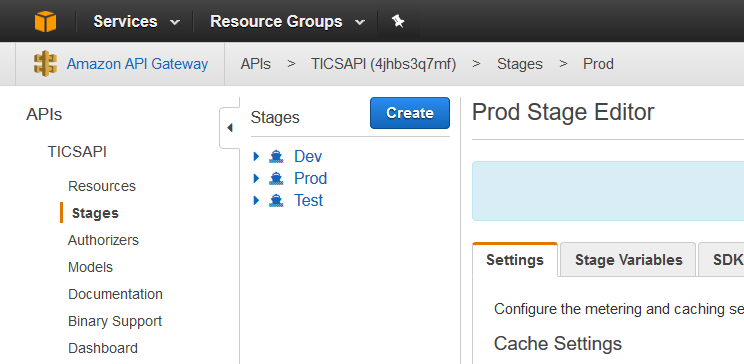
Add ‘Authorization’ as an HTTP Request Header.

Make sure to enable CORS on the API Gateway Endpoint:

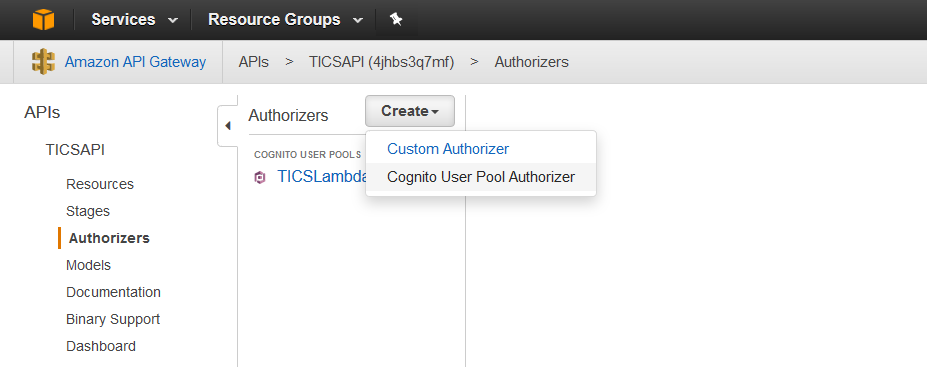


Accept all defaults.

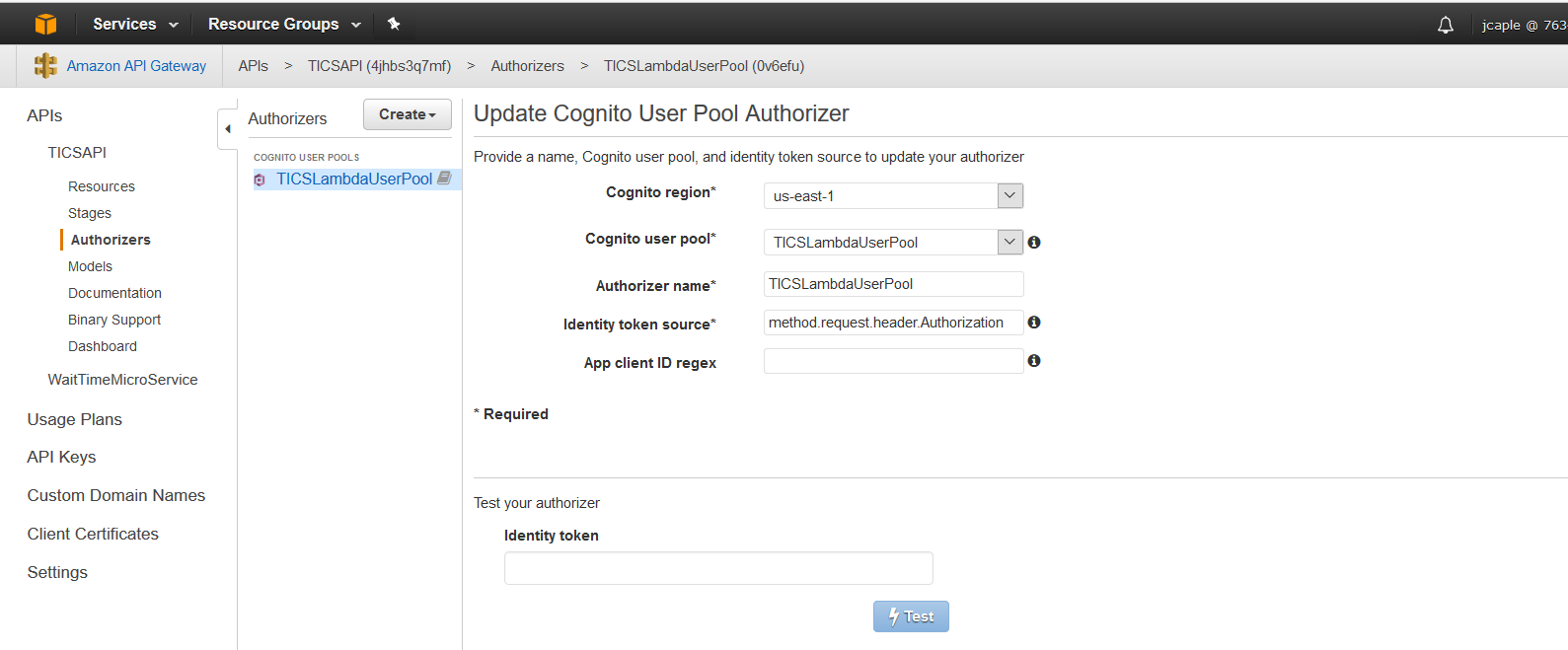
My recommendation is to create three separate deployment environments for everything, but you can create how many ever you think is best:



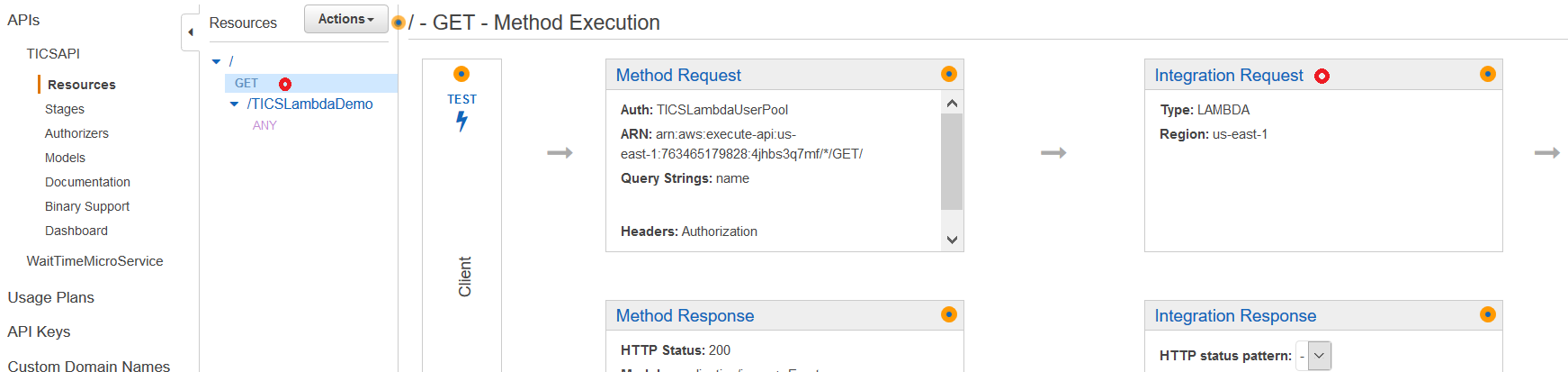
Click on Authorizers and create a new Cognito User Pool Authorizer called ‘TICSLambdaUserPool’:



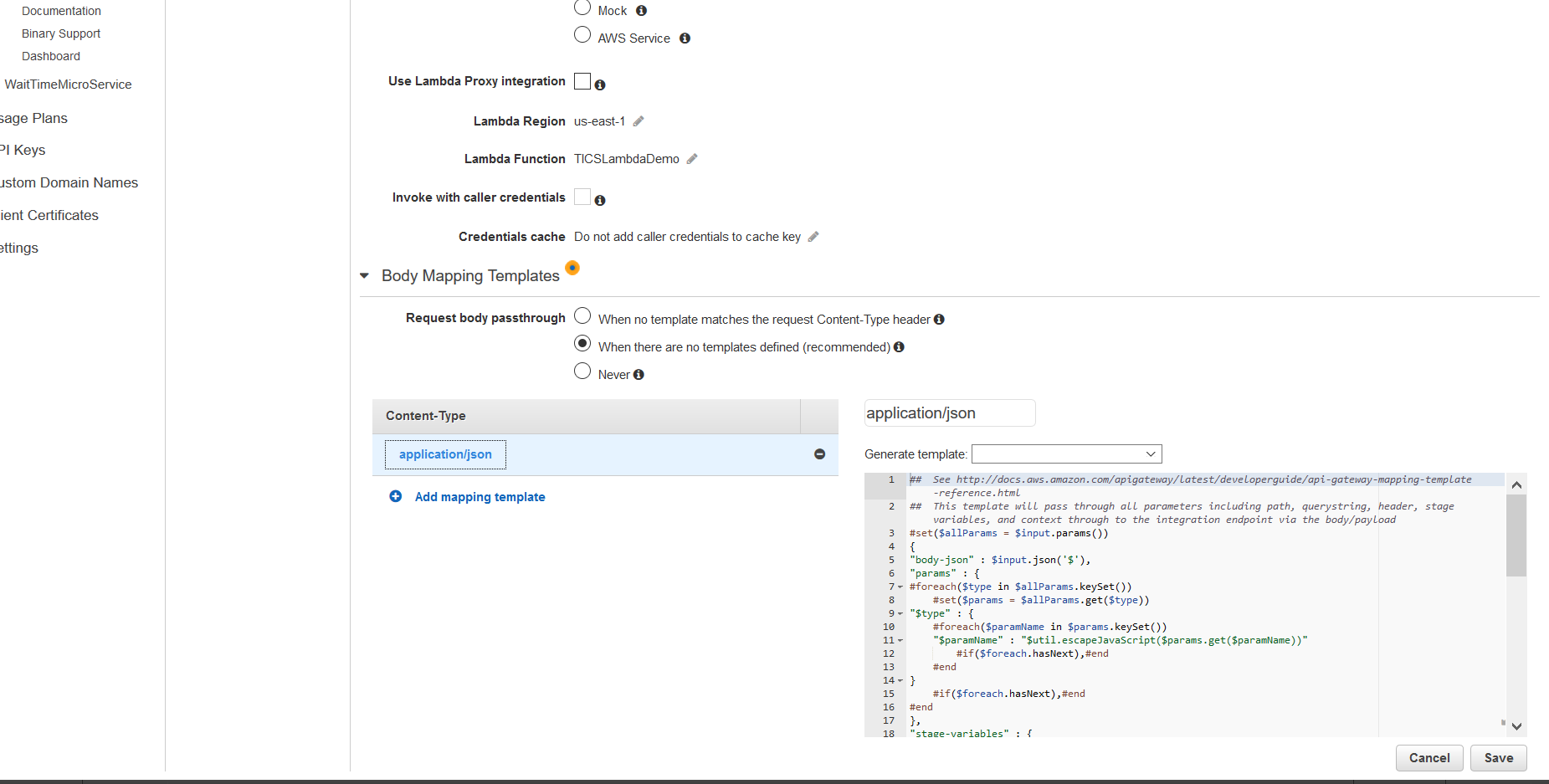
It should look like this once created:



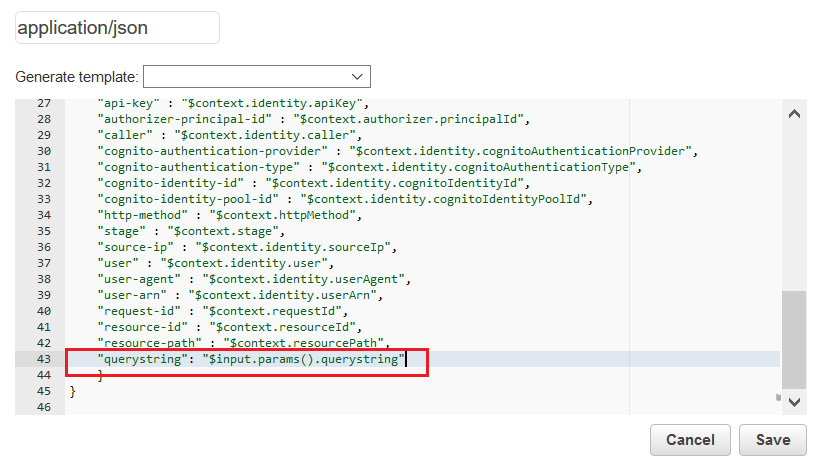
Click on ‘Resources’ again. Then click the ‘GET’ method and then the ‘Integration Test’ link on the right:



Expand the ‘Body Mapping Templates’ section:



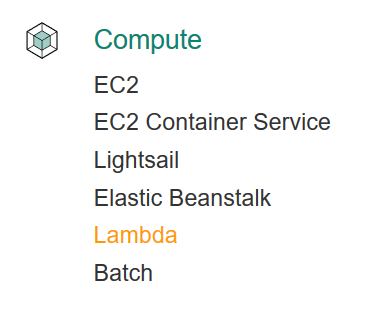
At the bottom of the application/json mapping template on the right, add the following line and then save it:



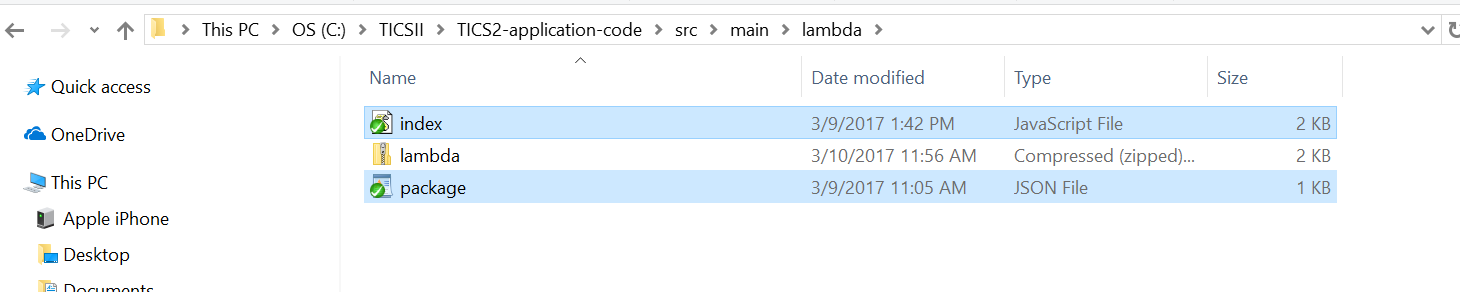
Create a POST Method similar to the GET Method like the above, but you do not need to add the ‘querystring’ key to the JSON Template for POST requests. For POST, just make sure of the following:

1. Add the Authorization Header to both GET and POST
2. Ensure that method authorization is protected by the TICSLambdaUserPool.
3. Ensure CORS is enabled.
4. Deploy the API to the target environment.

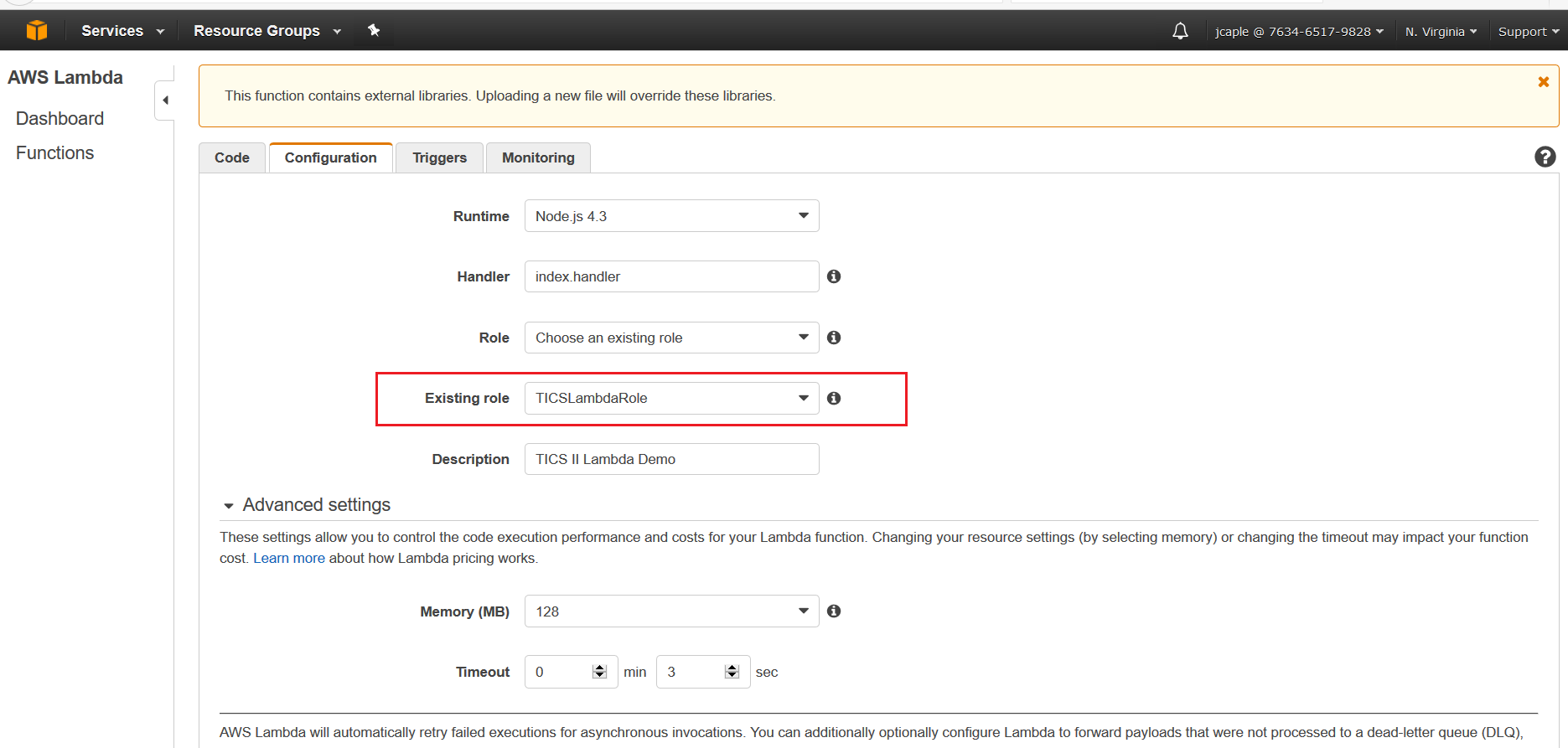
## Lambda Function



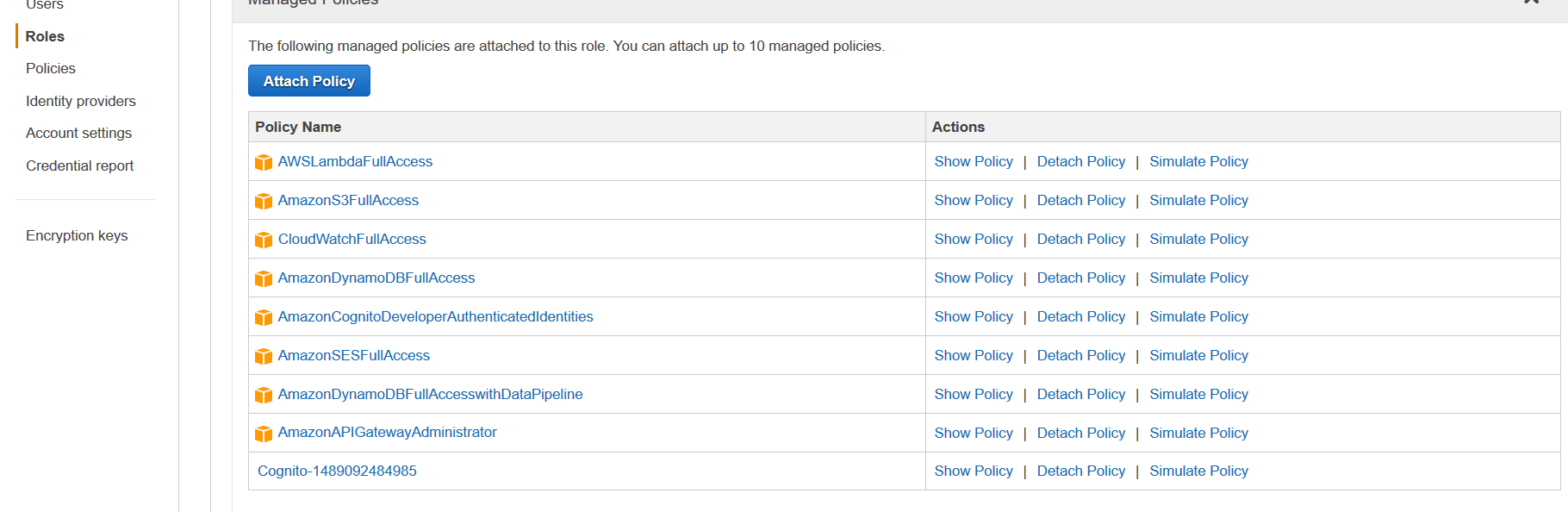
I’m not sure how to automate the creation of Lambda Functions, but my manual process was to create one called ‘TICSLambdaDemo’ using NodeJS 4.3. I typically upload the code as ZIP file which simply contains the index.js (and other other associated custom JS files) and the package.json file:



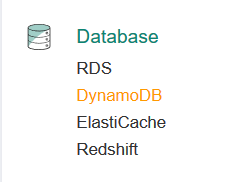
In the example above, I’d just upload the lambda.zip file to the ‘TICSLambdaDemo’ function I just created.



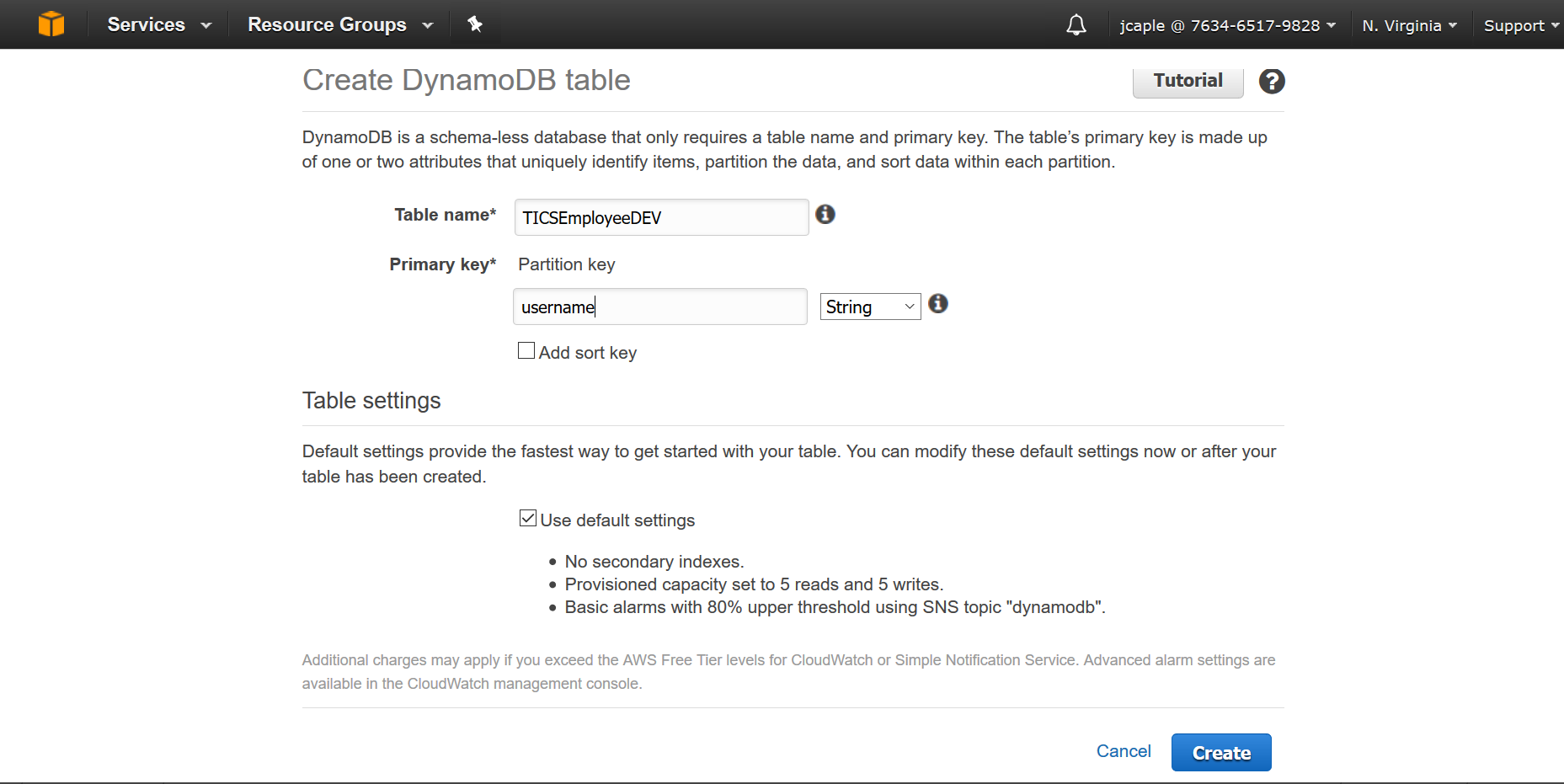
The ‘TICSLambdaRole’ is important. This role must have the following capabilities, or at least these are the permissions I gave it to make sure everything worked:



## DynamoDB Configuration



An DynamoDB Table per environment should be created using table names: TICSEmployeeDEV, TICSEmployeeTEST, TICSEmployeePROD. I figured we could just start with managing employee related data initially just to get the data infrastructure in place. The Partition Key should be ‘username’.



And I think that should do it for now. Of course as we get some code working, we may need to tweak the infrastructure a bit, but I think this is most of the big pieces.