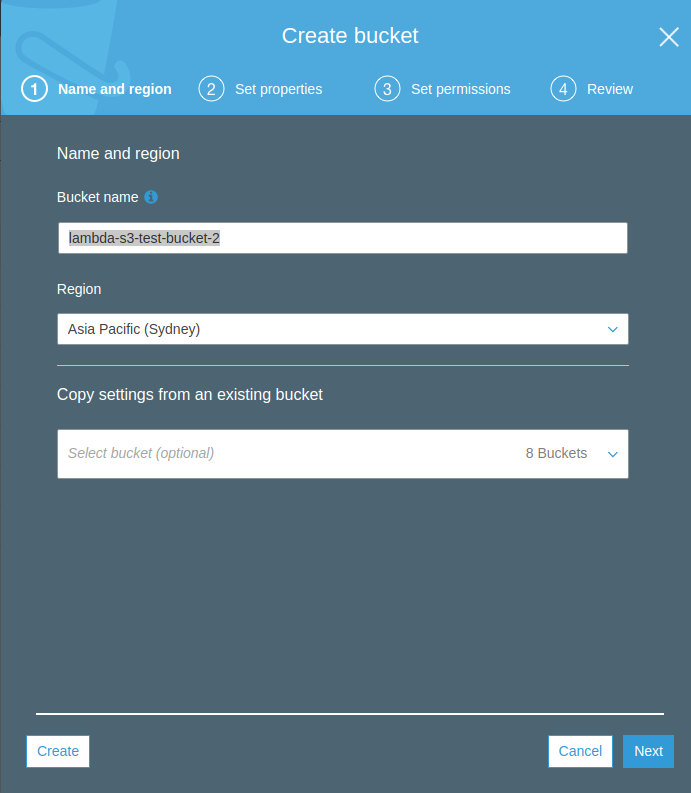
1- Select Region Sydney from top left

# Create S3 bucket

1- From services open S3

2- Click on “Create Bucket” called lambda-s3-test-bucket-2, make sure that the buket name is different since it needs to be unique in all regions. Select region to Syndey and click “Create”

3- Make sure it is not public.

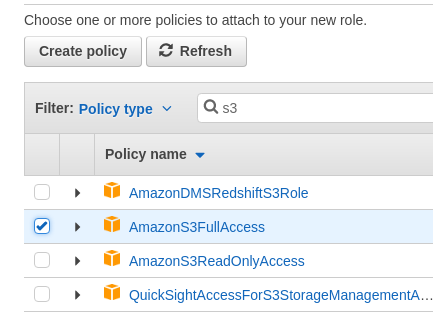
# Create Role for Lambda to access S3 bucket

1- From services select IAM

2- Click roles

3- Click create role

4- Select AmazonS3FullAccess and [CloudWatchLogsFullAccess](https://console.aws.amazon.com/iam/home?region=ap-southeast-2" \l "/policies/arn%3Aaws%3Aiam%3A%3Aaws%3Apolicy%2FCloudWatchLogsFullAccess)



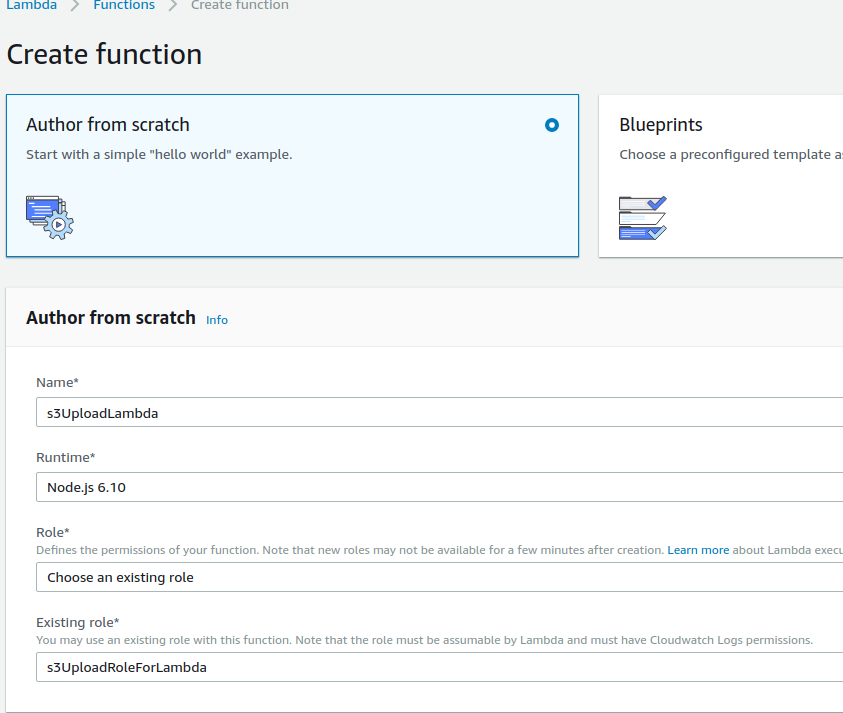
5- Click next and set the role name to “s3UploadRoleForLambda”

# Create Lambda Function

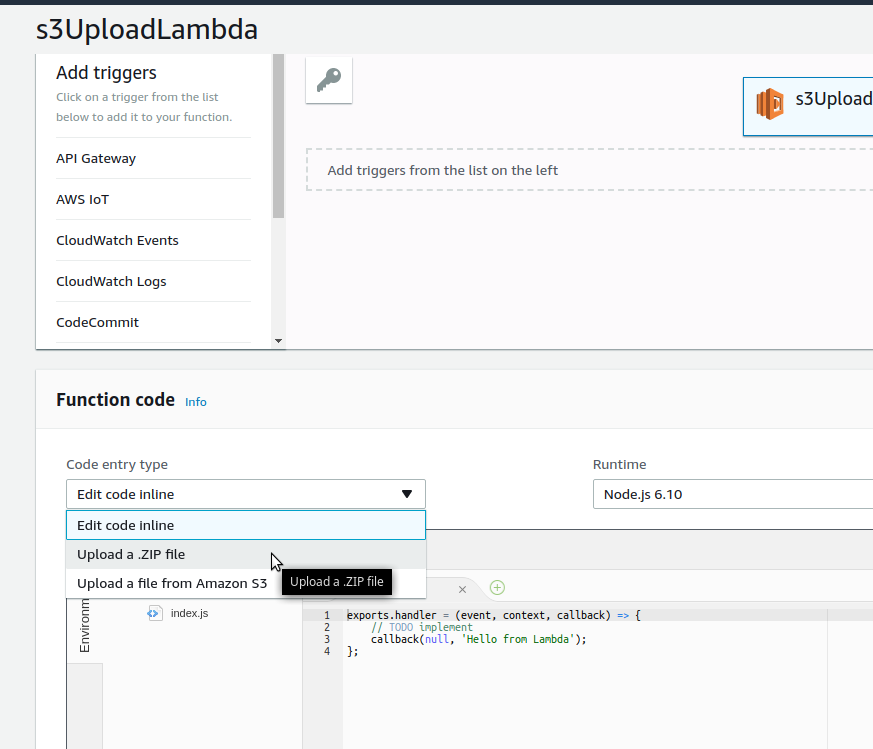
1- From services open Lambda

2- On the right hand click on “Create Function”

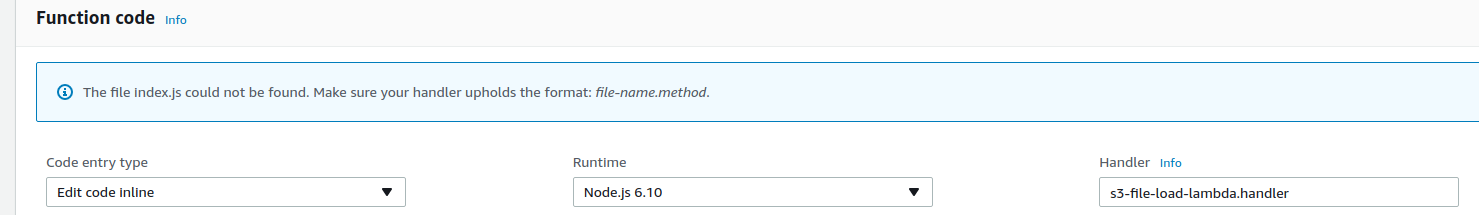
3- name it “s3UploadLambda” and choose the existing role from the list “s3UploadRoleForLambda” and click create function



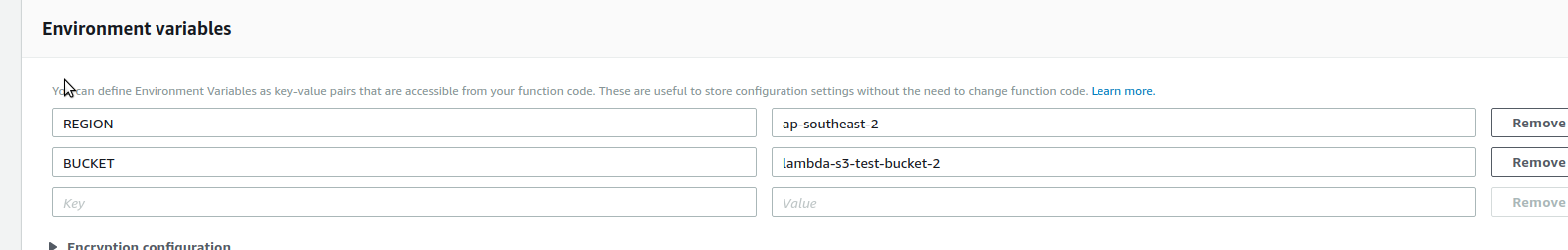
4- In the function code select upload an upload the zip file you have created.



5- In the handler write “s3-file-load-lambda.handler” and click save

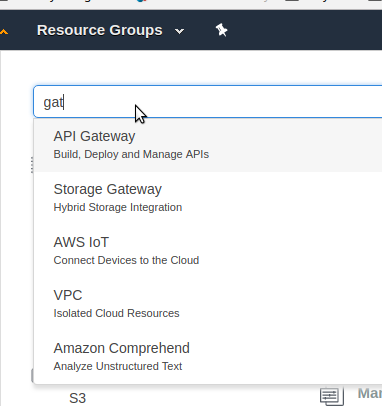


6- Set up the environment variables as REGION=ap-southeast-2 and BUCKET=lambda-s3-test-bucket-2

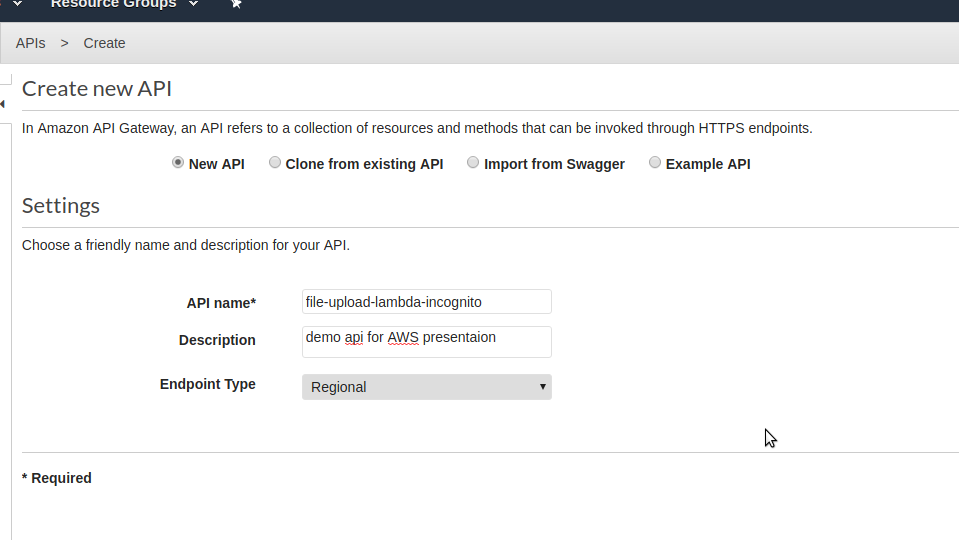
7-

# Create API gateway

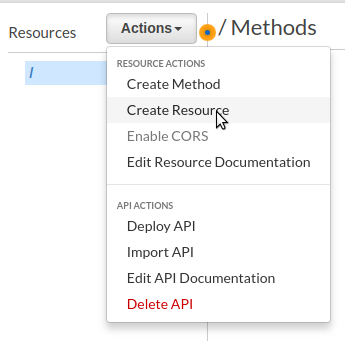
1- Select API gateway from services



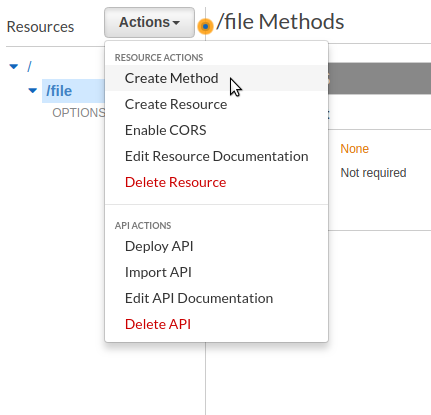
2- Click on “Create API” fill in the details and click on the “Create” button

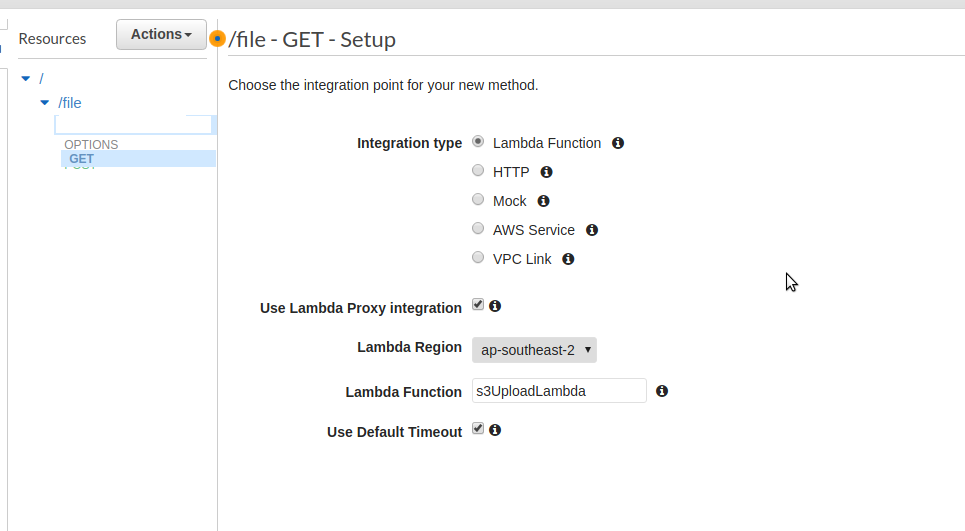


3- In the Actions drop-down select “Create Resource”

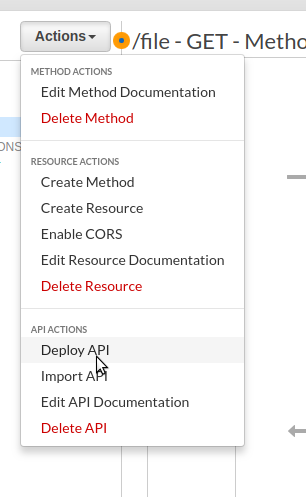
4- Create a resource called “file” and enable CORS

5- Select /file resource on the right and in the actions menu select create method and choose it to be POST. Since we are going to upload file using this endpoint.

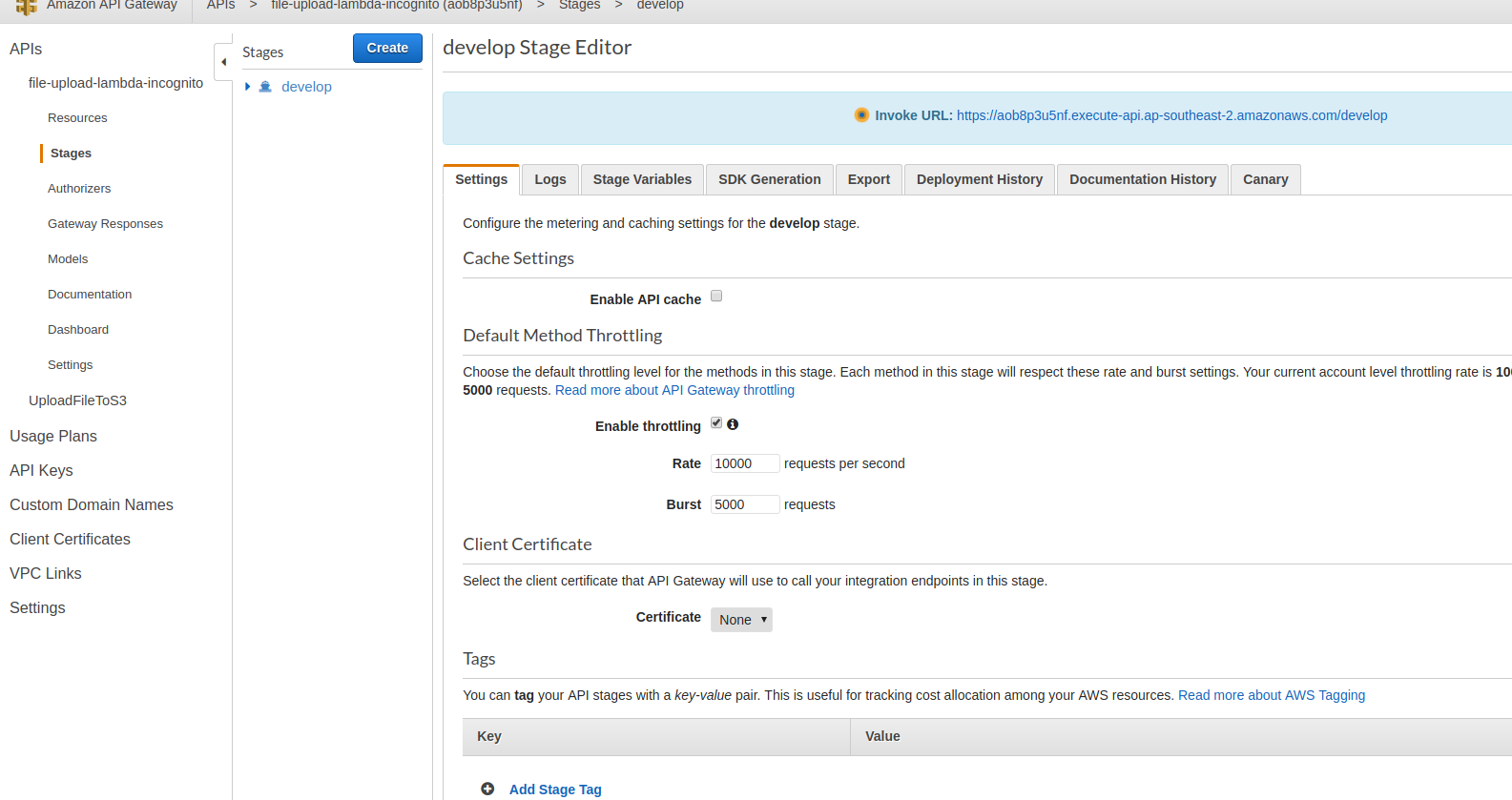
6- Click on the POST request and select the lambda function that has been created. Click OK to approve the permission to invoke Lambda



7- next stage the API by clicking on resource link and from the actions select “deploy API”. You can call it anything you like in this example I call it “develop”



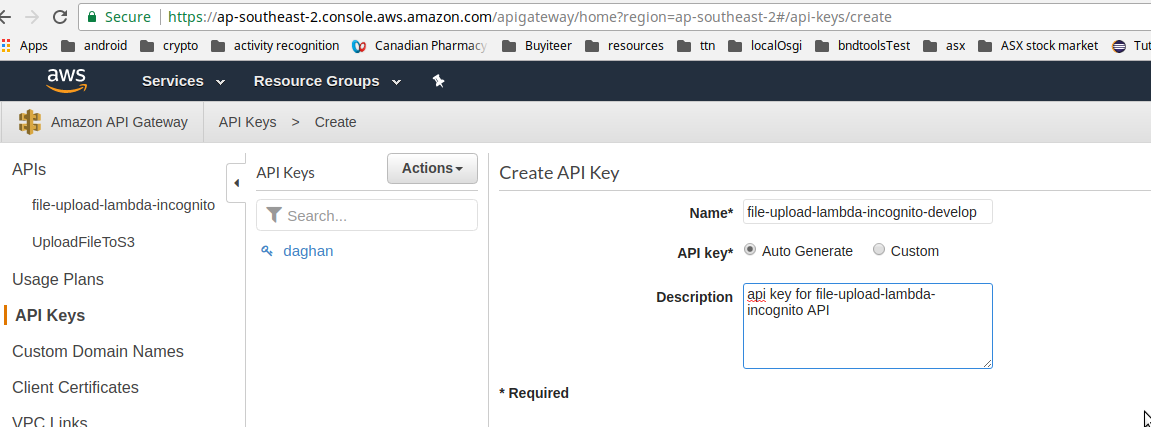
8- Now click on the “Stages” and click on develop and note the URL

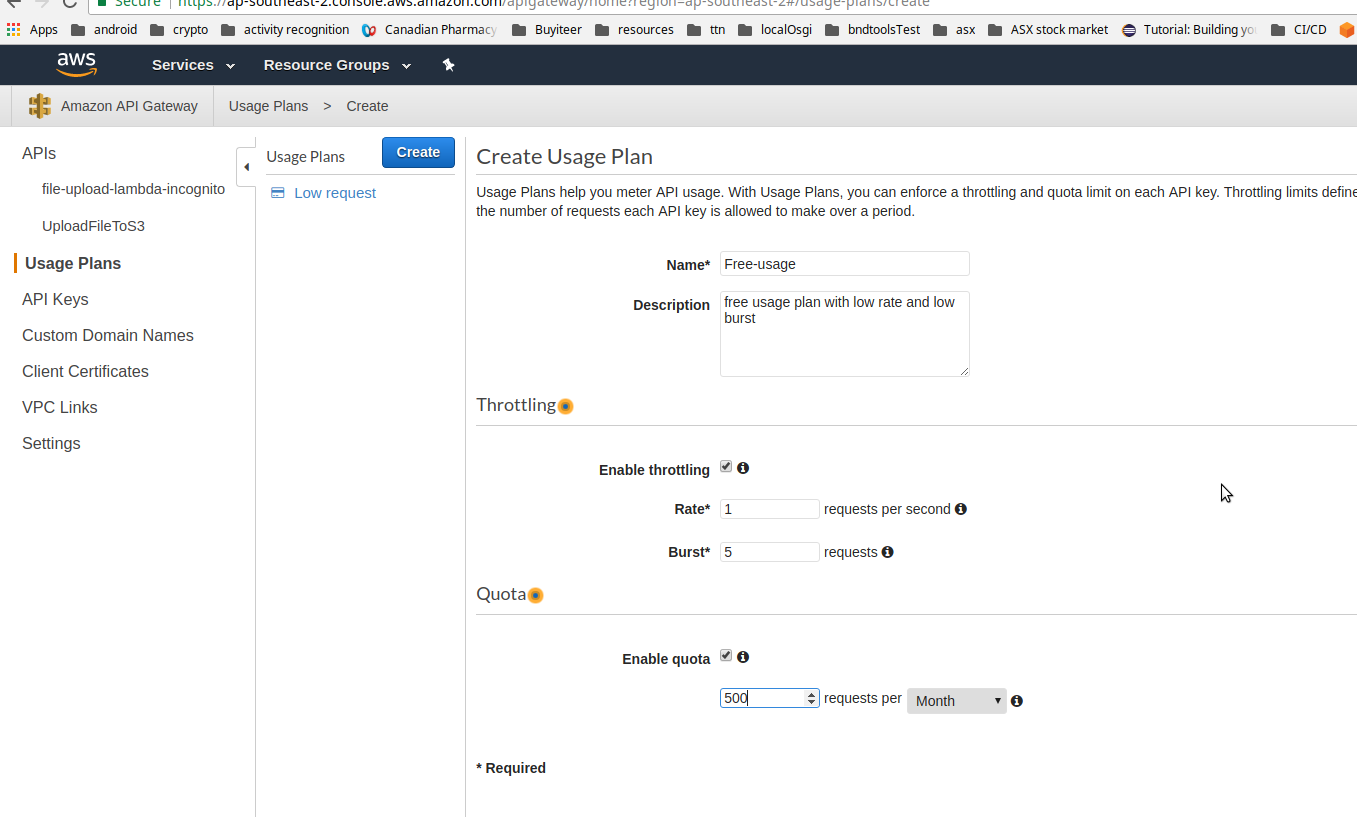


# Adding security

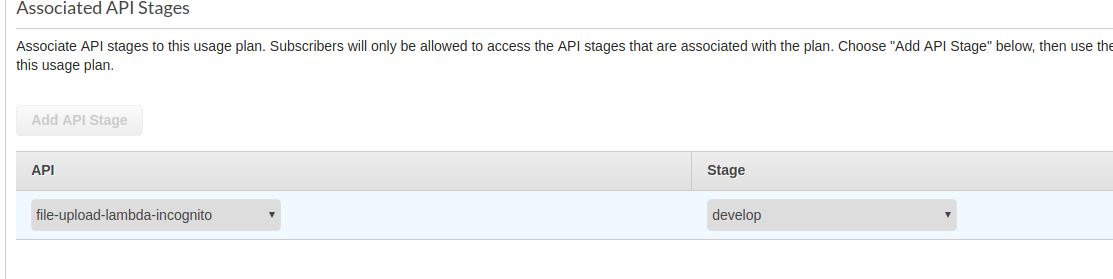
## Application key

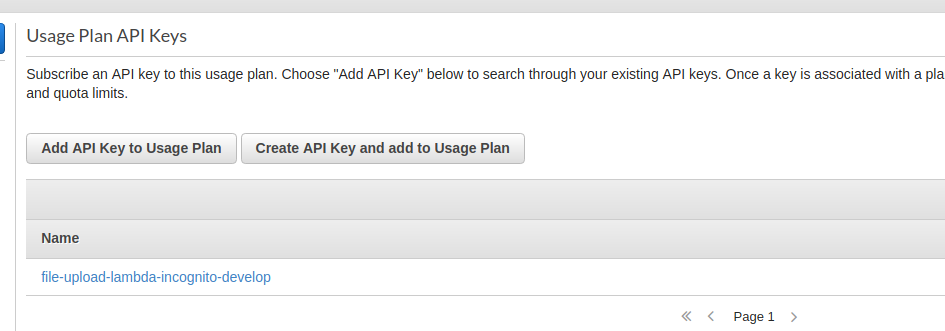
1. 1- Create an API key from the left menu by selecting API key. Call it file-upload-lambda-incognito-develop and click save.



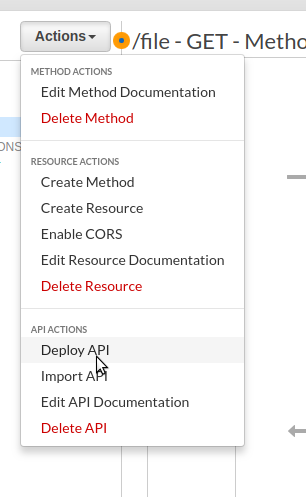
2- Now click on Usage plans click Create. Name it Free-usage for providing the usage plan for free users.

3- Click and select the API and the corresponding stage to associate this usage plan. You can add as many API and stage to your usage plan

4- Click next and associate the API key to your usage plan. Use file-upload-lambda-incognito-develop and click done.



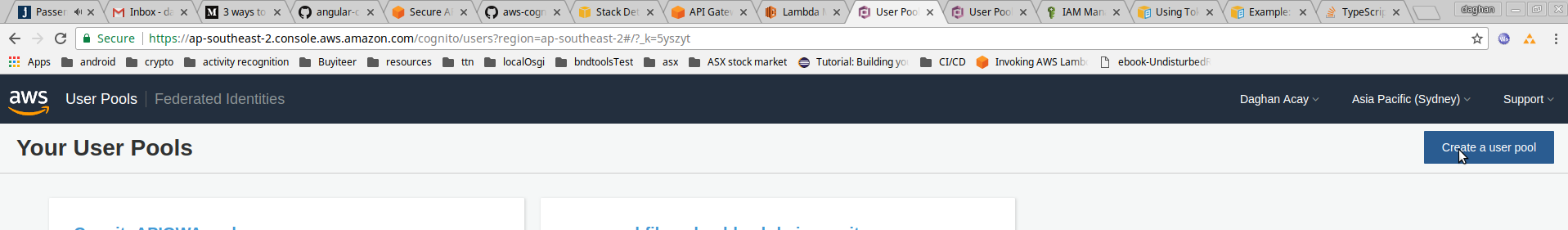
5- WARNING: You need redeploy the API



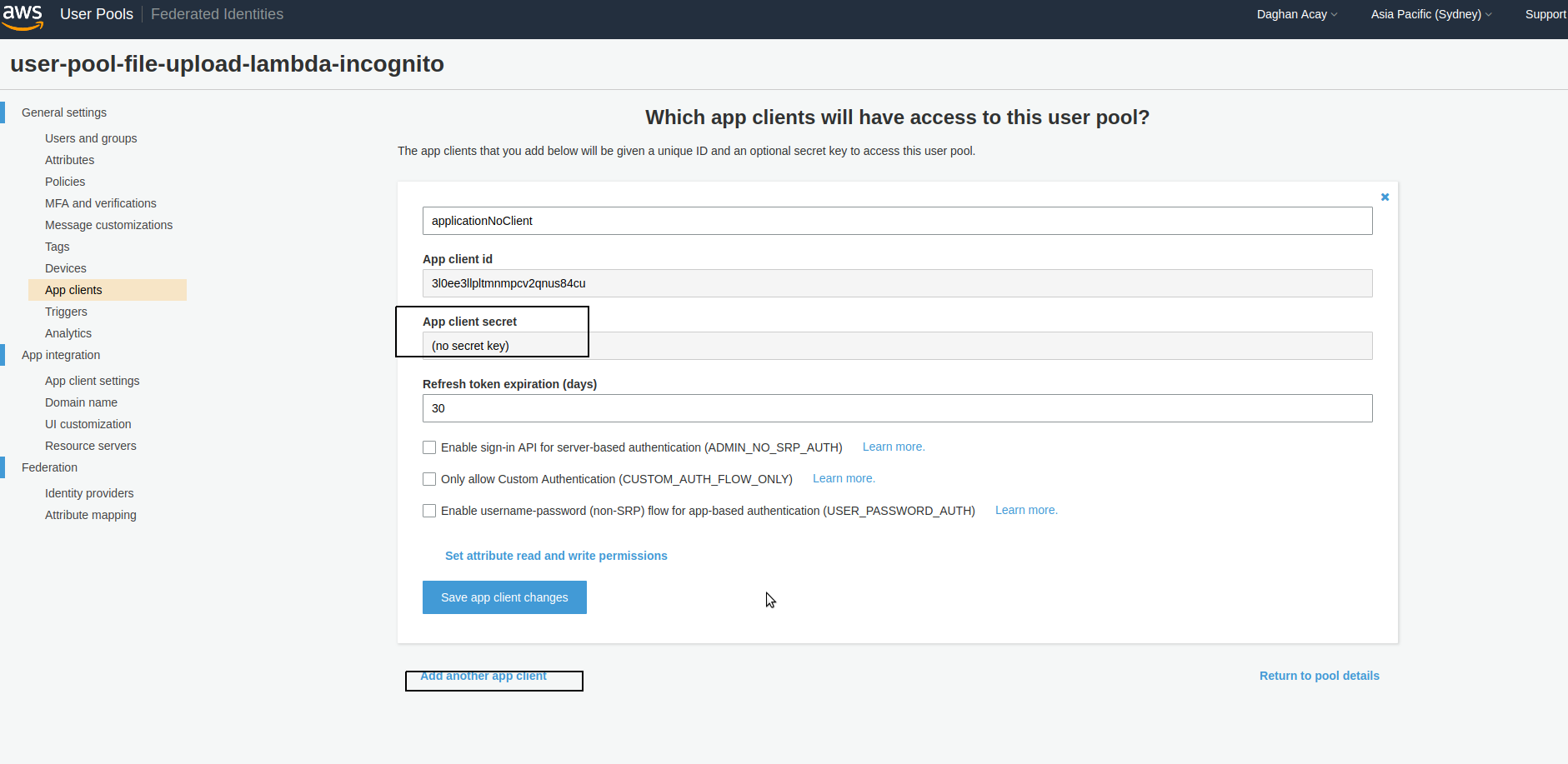
6- You should see the application will receive 403 when you try to load file. Please follow steps from README

## Cognito pools

1- Create a Cognito pool



2- Create client application



3- create users using aws-cli

aws cognito-idp sign-up \

--client-id <Application client ID> \

--username <userName> \

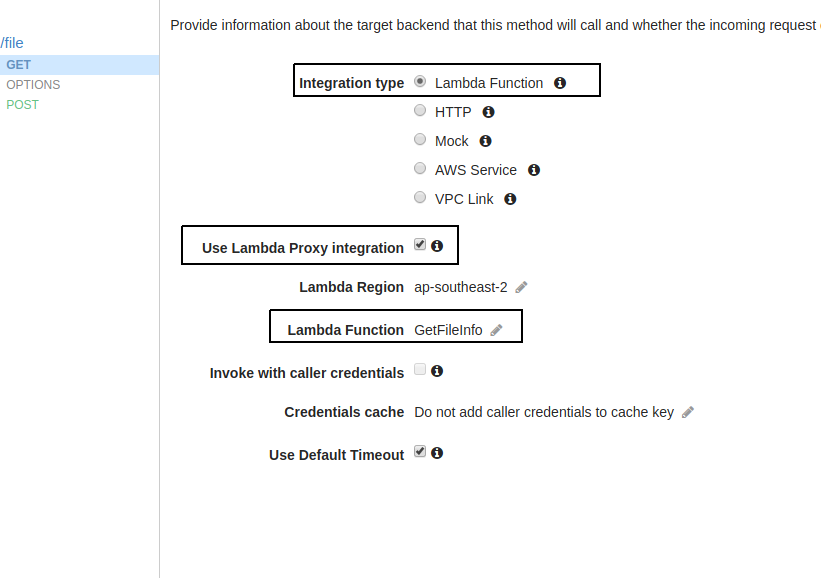
--password <password> \

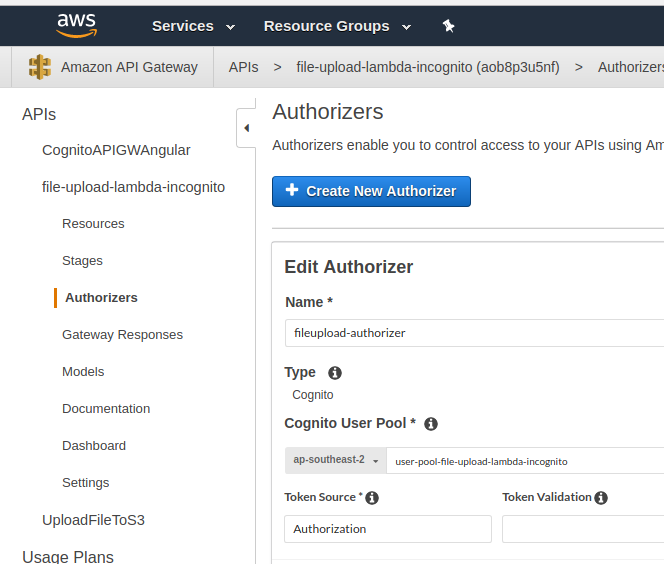
--region ap-southeast-2 \

--user-attributes '[{"Name":"given\_name","Value":"John"},{"Name":"family\_name","Value":"Doe"},{"Name":"email","Value":"[jdoe@myemail.com](mailto:jdoe@myemail.com)"},{"Name":"gender","Value":"Male"},{"Name":"phone\_number","Value":"+61XXXXXXXXXX"}]'

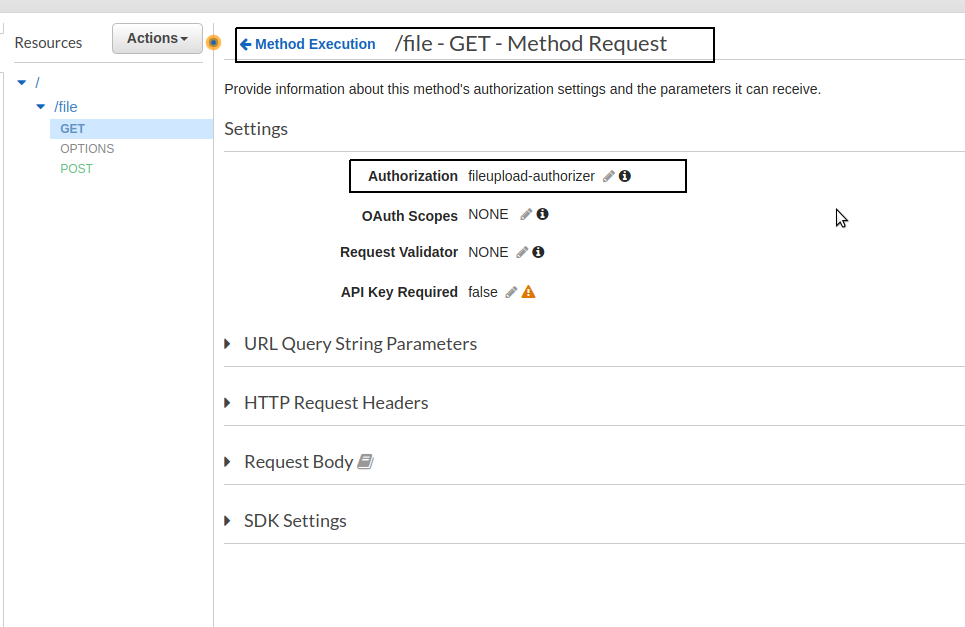
4- Create a new Lambda function by following the instructions above, this time just return a fixed data based on the input parameters. (See lambda/simple-response-lambda.js)

5- Go to resource create a GET method using the newly created Lambda function.

6- Create an authoriser using the pool name



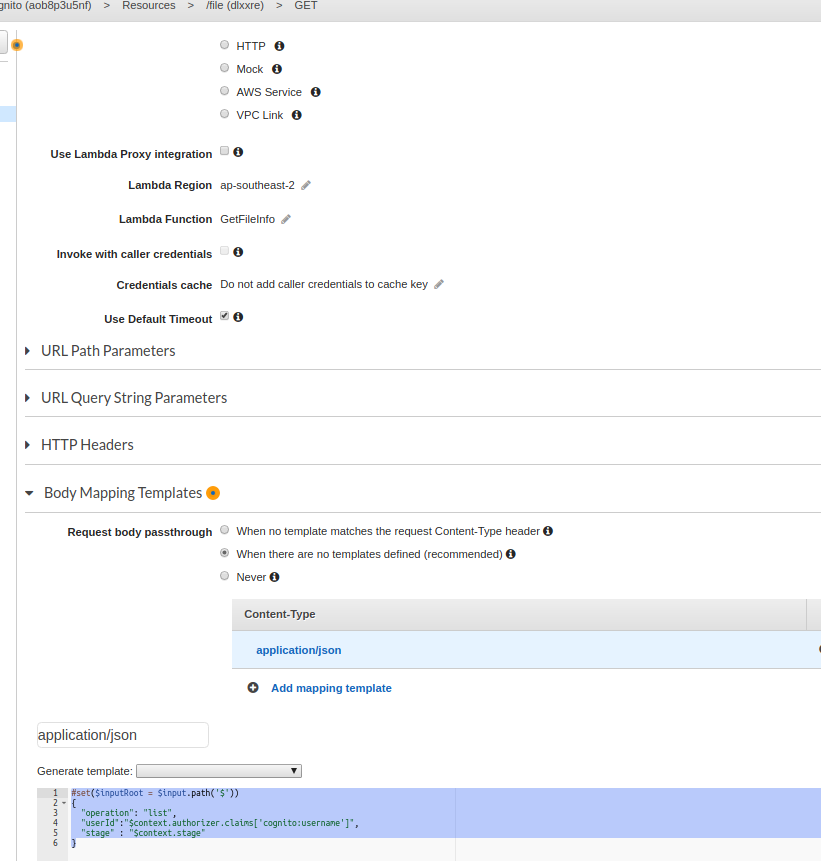
7- go back to Get method and click on the ”Method Execution” and set the Authorisation to file-upload-authorizer.



8- Now you can use the Cognito JWT token for accessing the Api gateway. Use it as a Header value with name **Authorization e.g.**

**Authorization:eyJraWQiOiJ4YWFFTXBJajVQWFBxQm5iQVdZdkIyd20yRTRBSDdJRHJmbFQxWXFxdWxvPSIsImFsZyI6IlJTMjU2In0..Q0Rn60m7iu2igjBI5CcDxk8ILuyZCbuZE4if6dgQiOA4F6NIBc\_nYyulI3z\_fDgWHdHP\_61voTKkfqb3Ai3seaJbiBPx6Qdp-Lmr2Gb5sbtdIbxtZMuxb\_ZTOAvyaFVY1DeFr5A2nzLILjHdsaWlWsUDQd6Kitct97tqbejf20I6K236lgNF\_UZsK3dxQxPqBpfST2BBMR3kCqattz3DJJmkRNLoxk-ET3hUePSepEZs6Paodjf9ZOUVk2RQ\_iZKERdrSC6btKp1QugDq3z3uh\_9NPimsrLo1gAq3ERi7l9h7Q54PSbrOEzuMtd0TJa-B9DkjroABl9t0GL19JqFJw**

9- Now change the API gateway configuration from the integration point And change the lambda to pass data from the security context and use it.

Do not forget to change the integration response to set the Origin headers

